New Frontiers of Attribute-Based Encryption via a General Paradigm and More

罗辑 (Ji Luo)

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2025

Reading Committee:
Huijia Lin, Chair
Stefano Tessaro, Chair
Paul Beame

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Wahsington

Abstract

New Frontiers of Attribute-Based Encryption
via a General Paradigm and More

罗辑 (Ji Luo)

Chairs of the Supervisory Committee:
Huijia Lin and Stefano Tessaro
Computer Science & Engineering

Attribute-based encryption (ABE) is an advanced form of public-key encryption incorporating fine-grained access control. In such a system, keys and ciphertexts are associated with attributes x and policies f, respectively, and decryption is conditioned on x satisfying f. Designing ABE schemes is challenging and its objectives include expressiveness, succinctness, efficiency, achieving strong security, and relying on minimal assumptions. This dissertation pushes the frontiers of ABE in terms of these objectives separately and jointly and studies the interaction among them.

In the first part, we propose a general paradigm that greatly simplifies the task of constructing ABE schemes. It reasonably distributes the complexities into constituents, making each ingredient and the overall scheme easier to understand, reason about, and potentially improve. It is also versatile and powerful. The benefits are demonstrated by four different instantiations, which achieve various ABE schemes with improved objectives and are related to each other by replacements of ingredients.

In the second part, we push the frontiers of ABE outside the paradigm. In one chapter, we resolve a long-standing open problem of constructing depth-unbounded ABE from lattices. In the other, we present the first systematic study of the upper/lower bounds of ABE succinctness and efficiency, showing inherent trade-offs among the objectives and constructing a few Pareto-optimal schemes.

To mom and dad.

Acknowledgments

Words are too pale to convey my gratitude towards Rachel Lin and Stefano Tessaro, my advisors, without whom this dissertation would not be conceivable. I am deeply grateful to Rachel for being patient with me, academically and personally, for teaching me the why's, what's, and how's of research, academia, and beyond, for supporting me to pursue my aspirations, and for constantly encouraging me. Although I did not work with Stefano on research projects, I always appreciate his feedback about my talks and advice about career.

Thank you to Gaku Liu, Paul Beame, Anup Rao, Cynthia Vinzant for agreeing to serve as my doctoral supervisory committee members, although some were unable to sit through completion due to my chaotic scheduling practices. Thank you to Hoeteck Wee and Daniel Wichs, who hosted and mentored me during my internship at NTT Research and visit to Northeastern University. Aside from research collaboration, the inspiration from them has been helpful to my first single-author paper.

I would like to thank John Steinberger, who taught a wonderful course in cryptography, inducting me into its fascinating world when I was an undergraduate at Tsinghua University. I would like to thank Ivan Damgård, who guided me through my first research experience in cryptography when I was an exchange student at Aarhus University.

Thank you to all my coauthors: Ivan Damgård, Sabine Oechsner, Peter Scholl, Mark Simkin, Shengyu Zhao, Tingfung Lau, Eric I-Chao Chang, Yan Xu, Rachel Lin, Hanjun Li, Junqing Gong, Hoeteck Wee, Aayush Jain, Daniel Wichs, Yao-Ching Hsieh, Yevgeniy Dodis. This dissertation is based on joint work with many of them.

I would like to thank Aayush Jain, who is kind to offer me advice from his recent experience of transitioning from a PhD student to a faculty member. I would like to thank Ashrujit Ghoshal, who never hesitates to share with me his experience of navigating life after graduation.

Thanks to Kaifeng Lyu, Hanwen Zhang, Hongxun Wu for inviting me to speak at Yao Class Seminar (IIIS, Tsinghua) — for that, I made my first attempt of delivering academic talks in Chinese to audience in China. Thanks to Junqing Gong, Yu Yu, Yilei Chen, Yanbin Pan for inviting me to speak at East China Normal University, Shanghai Jiaotong University, Forbidden City Crypto Day (IIIS, Tsinghua), AMSS Seminar (Chinese Academy of Sciences), offering me further opportunities to deliver talks in Chinese. Thanks to Aayush Jain, Ke Wu, Ran Canetti, Henry Corrigan-Gibbs, Yael Kalai, Eran Tromer, Vinod Vaikuntanathan, Daniel Wichs for inviting me to speak at numerous seminars. I would also like to thank Vinod for the week I stayed in Boston after a talk. It was a lot of fun to discuss research in coffee shops.

Thanks to the other members of the cryptography group during my time at UW: Rachel Lin, Stefano Tessaro, Binyi Chen, Pratik Soni, Ben Terner, Xihu Zhang, Ashrujit Ghoshal, Anna Kornfeld Simpson, Wei Dai, Joseph Jaeger, Tianren Liu, Hanjun Li, Chenzhi Zhu, Marshall Ball, Champ Chairattana-Apirom, Marian Dietz, Yao-Ching Hsieh, Andrea Coladangelo, Min Jae Song, Er-Cheng Tang, plus all the visitors. I enjoyed all the random conversations, about cryptography or not, as well as the reading group talks, which were fun and intellectually stimulating. Thank you to Xihu and Ashrujit, who were my cohorts for large parts of my PhD journey.

I would like to thank all of my friends. They are always amazing, and there are several times, difficult or celebratory, when I am even more grateful to have them around physically or virtually. Refraining from the bias of recency and to avoid the constant anxiety of wanting to add names after I will have submitted my dissertation, I choose to not list them.

The most important always have to come first or last. Thank you, dad, mom, Zhiwei Luo and Lijuan Li, to whom this dissertation is dedicated. Thank you for everything.

Table of Contents

Title Page **a**

Copyright Page **b**

Abstract **c**

Dedication Page **d**

Acknowledgments **i**

Table of Contents **iii**

List of Figures **viii**

List of Tables **ix**

Chapter 1: Introduction **1**

Chapter 2: Preliminaries **7**

2.1 Partially Hiding Functional Encryption 9

2.2 Groups and Pairing Groups 12

2.3 Lattices 14

Part I: General Paradigm

Chapter 3: Introduction **19**

3.1 ABE $\Leftarrow$ IPFE $\circ$ Garbling 20

3.2 Bird's-Eye View of Instantiations 22

Chapter 4: Compact and Adaptively Secure ABE from k-Lin: Beyond NC^1 and Towards NL (Pairing-Based IPFE, Noiseless Linear Garbling) **25**

4.1 Introduction 25

4.2 Technical Overview 30

4.2.1 1-ABE from Arithmetic Key Garbling and IPFE Schemes 32

4.2.2 Full-Fledged ABE via Slotted IPFE 35

4.2.3 1-ABE for Logspace Turing Machines 38

4.3 Preliminaries 49

4.3.1 Computation Models 51

4.3.2 Pairing 55

4.3.3 Function-Hiding Slotted IPFE 56

4.3.4 1-ABE . . . 58
4.4 Arithmetic Key Garbling Scheme . . . 59
4.4.1 Security Notions . . . 61
4.5 KP-ABE for Arithmetic Branching Programs . . . 64
4.5.1 AKGS for ABP . . . 64
4.5.2 1-ABE for ABP . . . 68
4.5.3 KP-ABE for ABP . . . 77
4.6 KP-ABE for Logspace Turing Machines (Uniform) . . . 87
4.6.1 AKGS for Turing Machines with Time/Space Bounds . . . 87
4.6.2 1-ABE for L . . . 93
4.6.3 KP-ABE for L . . . 114
4.6.4 Extension to NL . . . 126
4.6.5 ABE for DFA/NFA . . . 131
4.7 Further Discussions . . . 134

Chapter 5: Succinct and Adaptively Secure ABE for ABP from k-Lin (Pairing-Based IPFE, Noiseless Linear Garbling) 137

5.1 Introduction . . . 137
5.2 Technical Overview . . . 142
5.2.1 1-ABE Using Function-Hiding IPFE (Review of Chapter 4) . . . 142
5.2.2 Lightweight Alternative to Function-Hiding . . . 145
5.2.3 Dual System Encryption for Full ABE . . . 149
5.3 Preliminaries . . . 153
5.4 IPFE with Gradual Simulation Security . . . 157
5.4.1 Direct Construction . . . 162
5.4.2 One-Time Pad IPFE Scheme . . . 163
5.4.3 Security of Direct Construction . . . 166
5.5 Ciphertext-Policy 1-ABE for ABP . . . 172
5.6 Key-Policy ABE for ABP . . . 176
5.7 Ciphertext-Policy ABE for ABP . . . 184
5.7.1 KP-1-ABE . . . 184
5.7.2 CP-ABE . . . 185
5.8 Further Discussions . . . 190

Chapter 6: Succinct KP-ABE for Circuits and Doubly Succinct CP-ABE for Formulae (Pairing-Based IPFE, Noisy Linear Garbling) 195

6.1 Introduction . . . 195
6.2 Technical Overview . . . 199
6.2.1 KP-ABE with Constant-Size Keys . . . 204
6.2.2 CP-ABE with Double Succinctness . . . 209

6.2.3 Adaptive Security . . . 211
6.3 Preliminaries . . . 212
6.3.1 Lattice Tools . . . 213
6.3.2 Generic Asymmetric Pairing Group Model . . . 217
6.3.3 Inner-Product Functional Encryption . . . 219
6.4 Computational Secret Sharing . . . 220
6.5 KP-ABE for Circuits with Constant-Size Keys . . . 222
6.5.1 More Definitions for Secret Sharing . . . 222
6.5.2 Secret Sharing for Bounded-Depth Circuits from Adaptive LWE 225
6.5.3 Security Proof of Secret Sharing Scheme . . . 228
6.5.4 Construction of KP-ABE Scheme . . . 232
6.5.5 Security of KP-ABE Scheme . . . 234
6.6 IPFE in GGM with Stronger Security . . . 238
6.6.1 The ABDP Scheme . . . 240
6.6.2 Stronger Security of the ABDP Scheme . . . 240
6.7 Doubly Succinct CP-ABE for Boolean Formulae . . . 243
6.7.1 More Definitions for Secret Sharing . . . 243
6.7.2 Secret Sharing for Boolean Formulae from Adaptive LWE . . . 244
6.7.3 Security Proof of Secret Sharing Scheme . . . 247
6.7.4 Construction of CP-ABE Scheme . . . 248
6.7.5 Security of CP-ABE Scheme . . . 251
6.8 Further Discussions . . . 254

Chapter 7: Succinct CP-ABE for Circuits and ABE for DFA from Lattices via Evasive IPFE (Lattice-Based IPFE, Noisy Linear Garbling) 255

7.1 Introduction . . . 255
7.2 Technical Overview . . . 265
7.2.1 Review and Renew . . . 265
7.2.2 ABE from Noisy Linear Secret Sharing and Evasive IPFE . . . 267
7.2.3 Instantiation of Evasive IPFE . . . 272
7.3 Preliminaries . . . 273
7.4 Evasive Inner-Product Functional Encryption . . . 277
7.4.1 Basic Construction — Single-Identity, No Structured Noises . . . 280
7.4.2 Security as an Assumption . . . 282
7.4.3 Security for Restricted Samplers from Evasive LWE . . . 288
7.4.4 Identity-Based Scheme . . . 291
7.4.5 Scheme with Structured Noises . . . 299
7.5 Noisy Linear Garbling . . . 302
7.6 Noisy Linear Garbling for Bounded-Arithmetic Circuits . . . 304

7.6.1 Correctness and Shortness . . . 308
7.6.2 Security with Gaussian Noise . . . 311
7.7 Ciphertext-Policy ABE from Short Noisy Linear Garbling . . . 318
7.7.1 Correctness . . . 320
7.7.2 Security . . . 321
7.7.3 Unbounded Attribute and Summary . . . 325
7.8 ABE for DFA . . . 326
7.8.1 Noisy Linear Garbling for DFA . . . 327
7.8.2 Construction of KP-ABE for DFA . . . 329
7.8.3 Security of KP-ABE for DFA . . . 332
7.8.4 CP-ABE for DFA and Summary . . . 336
7.9 CP-ABE with Succinct Ciphertexts . . . 338
7.9.1 Succinct Noisy Linear Garbling for Circuits . . . 338
7.9.2 CP-ABE from General Noisy Linear Garbling . . . 340
7.9.3 Security and Summary . . . 343
7.10 Further Discussions . . . 349

Part II: More

Chapter 8: ABE for Circuits of Unbounded Depth from Lattices 353

8.1 Introduction . . . 353
8.1.1 Our Results . . . 355
8.1.2 Related Works . . . 363
8.1.3 Technical Overview . . . 366
8.2 Preliminaries . . . 373
8.2.1 Attribute-Based Laconic Function Evaluation . . . 373
8.2.2 Attribute-Based Encryption (Lesser Correctness) . . . 377
8.2.3 Lattices . . . 378
8.2.4 Homomorphic Encryption and Evaluation à la [GSW13] . . . 383
8.2.5 Attribute Encoding and Homomorphic Evaluation aux [BGG+14,BTVW17] . . . 386
8.3 Bootstrapping Homomorphic Evaluation . . . 388
8.3.1 Noise Removal . . . 391
8.3.2 Bootstrapping . . . 393
8.3.3 Unbounded Homomorphic Evaluation . . . 395
8.3.4 Stronger Correctness . . . 401
8.4 Attribute-Based Laconic Function Evaluation . . . 403
8.5 KP-ABE for Circuits of Unbounded Depth . . . 408
8.5.1 Construction of KP-ABE . . . 408
8.5.2 Security of KP-ABE . . . 411
8.5.3 Attribute-Unbounded Depth-Unbounded KP-ABE . . . 415

8.5.4 Scheme with Perfect Correctness . . . 416
8.6 Further Discussions . . . 421

Chapter 9: On the Optimal Succinctness and Efficiency of ABE **423**

9.1 Introduction . . . 423
9.1.1 Our Results . . . 426
9.1.2 What's Next? . . . 428
9.1.3 Technical Overview . . . 430
9.1.4 Related Works . . . 434
9.2 Preliminaries . . . 437
9.2.1 Multi-Tape Random-Access Machine . . . 437
9.2.2 Laconic Garbled RAM . . . 441
9.2.3 PHFE Efficiency and FE for Circuits . . . 444
9.2.4 Universal RAM and ABE for RAM . . . 446
9.2.5 Indistinguishability Obfuscation . . . 447
9.2.6 Laconic Oblivious Transfer . . . 448
9.2.7 Garbled Circuits . . . 451
9.2.8 Puncturable Pseudorandom Function . . . 452
9.2.9 Secret-Key Encryption . . . 453
9.2.10 Primitive Related to Lower Bound . . . 453
9.3 Lower Bound on Ciphertext Size and Decryption Time . . . 454
9.4 Bounded LGRAM with Fixed-Memory Security . . . 458
9.4.1 Construction . . . 458
9.4.2 Security . . . 462
9.5 From Bounded LGRAM to Unbounded LGRAM . . . 475
9.6 ABE for RAM . . . 479
9.7 Trading Short Components for Fast Decryption . . . 490
9.8 Further Discussions . . . 491

Bibliography **495**

The complete set of pages in this dissertation is a–d, i–x, 1–530.
Page numbers are not shown on pages a–d.
Pages x, 16, 18, 24, 350, 352, 494 are intentionally fully blank.

List of Figures

5.1 Roadmap of ABE constructions in this chapter. 141

6.1 Efficiency comparison among KP-ABE schemes. 196
6.2 Efficiency comparison among CP-ABE schemes. 196
6.3 Sketch of secret sharing scheme for KP-ABE. 206
6.4 Sketch of KP-ABE. 209
6.5 Modified secret sharing scheme for CP-ABE. 210
6.6 Sketch of CP-ABE. 211

9.1 Trade-offs between ABE component sizes and decryption time. 428
9.2 An overview of the notion of laconic GRAM. 433
9.3 The circuit GenAugCPU in Construction 9.1. 459
9.4 The circuit AugCPU in Construction 9.1. 461
9.5 The circuit GenAugCPU′ in the proof of Theorem 9.6. 465
9.6 Typical hybrid transitions in the proof of Theorem 9.6. 467
9.7 The machine M' in Construction 9.2. 476
9.8 The circuit GenLGRAM in Construction 9.3. 480

List of Tables

3.1 Summary of instantiations of the general paradigm. 23

4.1 Select single-letter symbols in this chapter. 50
4.2 Hybrids, 1-ABE for ABP, Case 1 ($\mathbf{x}$ then f). 72
4.3 Hybrids, 1-ABE for ABP, Case 2 (f then $\mathbf{x}$), first and final. 73
4.4 Hybrids, 1-ABE for ABP, Case 2 (f then $\mathbf{x}$), loop (middle). 75
4.5 Hybrids, KP-ABE for ABP, first and final. 82
4.6 Hybrids, KP-ABE for ABP, loop (middle). 85
4.7 Hybrids, 1-ABE for L, first and final. 102
4.8 Hybrids, 1-ABE for L, Case 1 ($\mathbf{x}$ then M), loop (middle). 104
4.9 Hybrids, 1-ABE for L, Case 2 (M then $\mathbf{x}$), outer loop (indexed by t). . . . 109
4.10 Hybrids, 1-ABE for L, Case 2 (M then $\mathbf{x}$), inner loop (indexed by t, q). . 110
4.11 Hybrids, KP-ABE for L, first and final. 119
4.12 Hybrids, KP-ABE for L, loop (middle). 123

5.1 Comparison among select KP-ABE schemes with succinct ciphertexts. 138
5.2 Comparison among select CP-ABE schemes with succinct secret keys. . 138
5.3 Select single-letter symbols in this chapter. 153

6.1 Select single-letter symbols in this chapter. 213

7.1 Select single-letter symbols in this chapter. 274

8.1 Comparison among select KP-ABE schemes for circuits. 364
8.2 Select single-letter symbols in this chapter. 374

9.1 Comparison among select KP-ABE schemes. 427
9.2 Select single-letter symbols in this chapter. 438

Chapter 1

Introduction

Attribute-based encryption (ABE) [SW05,GPSW06] is an advanced form of public-key encryption incorporating fine-grained access control. In such a system, using the (master) public key mpk, anyone can encrypt message μ associated with an attribute x, producing a ciphertext $\mathsf{ct}_x(\mu)$; using the master secret key msk, the data owner can generate a restricted secret key sk_f associated with a policy f (each key is generated for and distributed to a user, whose decryption capability is constrained by its f); decrypting $\mathsf{ct}_x(\mu)$ using sk_f succeeds if the attribute x satisfies the policy f. The security requirement of ABE is collusion resistance — any group of users holding multiple keys for different policies learn nothing about the message μ as long as none of them is individually authorized to decrypt the ciphertext.

Other than the direct usage scenario, ABE has found applications in achieving other cryptographic goals such as publicly verifiable delegated computation [PRV12]. Beyond the original notion, ABE has been generalized to functional encryption and multi-authority, multi-user, multi-input, and registered variants. Despite being defined almost two decades ago and having such great utility, there is still room to improve our understanding of ABE, with respect to both *means* and *ends*.

Pursuit of Means. Since its conception, there have been numerous constructions of ABE, relying on a multitude of assumptions (roughly categorized into two kinds, pairing-based and lattice-based) and achieving various properties. Many constructions are described with heavy details of algebra. There are similarities among the different schemes (even when they use different kinds of assumptions). Despite the complexity of each construction and the similarity across multiple schemes, there is a lack of systematic paradigms[1] that simplifies the task of constructing ABE, is general enough

[1]The only notable exception is dual system encryption [Wat09], which was proposed with a focus on proving adaptive security (instead of constructing ABE in general), and its refinements [Wee14,Att14].

to capture a wide array of ABE constructions, and helps in advancing the *ends* of ABE (discussed later).

One goal of this dissertation is to develop a paradigm for constructing ABE that

- reasonably distributes complexities of ABE constructions into the constituents of the paradigm, making each ingredient and the overall scheme easier to understand, reason about, and potentially improve;
- is versatile enough to be instantiated with different constituents, potentially based on different kinds of assumptions, to achieve different results;
- helps obtaining new ABE schemes not known to the literature.

Pursuit of Ends. Once constructed, an ABE scheme is ultimately judged by its properties, not the methodology used. The following properties are desired in general and have been important research topics.

- Expressiveness. The scheme should support as rich a class of policies as possible.
- Succinctness and Efficiency. The keys and the ciphertexts should be as short as possible. Its algorithms should run as fast as possible.
- Strong Security. Its security should hold even when the adversary is allowed to choose the non-authorizing attributes and policies adaptively.
- Minimal Assumptions. Its security should be reduced to as weak assumptions as possible.

The design of ABE is hence a *multi-objective* optimization problem. To make it more interesting, these desiderata might not be independent of each other, e.g., it is often possible to prove strong security (better) based on strong assumptions (worse).

The other goal of this dissertation is to push the frontiers of ABE constructions in terms of these objectives separately and jointly, and to understand how they interact with each other.

However, the schemes are still often presented with delicate algebraic manipulations, and the known instantiations are pairing-based.

Contributions. The contribution of my dissertation work is treated in two parts.

IN PART I, we study the *means* and use it to push the *ends*.

We explore the paradigm of building ABE from two simpler primitives, inner-product functional encryption (IPFE) and linear garbling. A linear garbling encodes a complex computation (of determining whether the message can be decrypted based on f, x) into affine functions, which can then be securely computed using IPFE. The paradigm is instantiated with four combinations of IPFE and linear garbling, leading to many novel ABE schemes.

- Chapter 4 contains the first proposal of this paradigm. By studying IPFE and noiseless linear garbling separately, we were able to discover new structural properties of noiseless linear garbling that enable an easy proof of adaptive security when complemented with existing pairing-based IPFE schemes, which had been notoriously difficult. Results-wise, we achieve

  - the first *compact* (component sizes not dependent on the number of times the attribute is read by a program) and *adaptively secure* ABE for arithmetic branching programs (ABP) based on *standard* assumptions;
  - the first adaptively secure ABE for NFA, DFA based on standard assumptions (also compact, and most previous schemes are also compact);
  - the first ABE for NL, L based on standard assumptions (both also adaptively secure, but they are the first regardless of adaptive security);

  To this day, the results for NL, L, NFA, DFA remain the state of the art among ABE from pairing.

- In Chapter 5, we replace the IPFE in Chapter 4 by a lightweight alternative still sufficient for ABE for ABP, and incorporate dual system encryption methodology into our scheme. It was inspired by studying the roles of IPFE in Chapter 4 more carefully and comparing them with the existing literature. As a result, we improve (still adaptively secure) ABE for ABP from *compact* to *succinct*, meaning that the ciphertext size is independent of the attribute length. We also obtain

a ciphertext-policy ABE (where the key is associated with an attribute and the ciphertext, a policy) whose key is succinct.

- In Chapter 6, we replace the noiseless linear garbling in Chapter 5 by a noisy one based on lattices, and consider stronger security of IPFE in the generic group model. The schemes in Chapter 5 provide succinctness relative to x, the attribute, which is due to the succinctness of IPFE keys. Lattice-based noisy linear garbling is known to provide succinctness relative to f, the policy. By combining the two, we achieve *doubly succinct* KP-ABE for circuits and *doubly succinct* CP-ABE for Boolean formulae.

- In Chapter 7, we replace the pairing-based IPFE by a lattice-based one based on a new kind of assumption called *evasive LWE*. The benefit over using pairing-based IPFE is that the schemes become plausibly post-quantum secure. Results-wise, we obtain CP-ABE for circuits (removing an additional assumption, tensor LWE, from [Wee22]), as well as the first ABE for L, DFA from lattices.

The results of these chapters demonstrate the benefit of studying simpler constituents separately, as each has led to significant improvements in ABE constructions. They also attest to the versatility of our paradigm.

In Part II, we present additional improvements outside the paradigm.

- In Chapter 8, we present the first ABE for circuits of *unbounded depth* based on lattices. In prior lattice-based ABE schemes, once the scheme is set-up, a polynomial upper bound on the maximum depth of circuits supported by the scheme is determined, essentially by the length of mpk. In contrast, it is known how to achieve fully homomorphic encryption (FHE) without a depth bound from lattices. Despite the structural similarity between lattice-based FHE and ABE, depth-unbounded ABE from lattices has remained unknown for almost fifteen years.

- In Chapter 9, we study the limits of succinctness and efficiency of ABE. Based on functional encryption (of arbitrary bad efficiency as long as it is sufficient for

program obfuscation), we construct *optimally succinct* (in terms of component sizes) or *optimally efficient* (in terms of decryption overhead compared to verifying authorization in the clear) ABE schemes. Moreover, we state and prove the first unconditional *time-space trade-off* for ABE, which shows that the schemes we achieve are Pareto-optimal. This is the first systematic study of tight lower/upper bounds of ABE, and it urges us to reconsider the existing accounting model of component sizes.

The results push the frontier of ABE at the more implementable (lattice-based) and the more theoretical (obfuscation-based) extremes in the spectrum of cryptography.

In each chapter, the text is mostly adapted from the underlying paper, so the novelty statements should be understood as "at the time of the original publication".

Outlook. The following questions remain interesting for future research.

- Closing the gap between best security and best functionality of ABE from pairing. The maximum class of policies known from pairing is arithmetic span programs (ASP), which is more general than arithmetic branching programs (ABP). However, there is no known proof of adaptive security of ABE for ASP from pairing with static assumptions. Moreover, we have shown that our method of proving adaptive security cannot work beyond ABP. Closing this gap would require new insights into proofs of adaptive security or alternative characterizations of ASP.

- Achieving various lattice-based ABE from standard assumptions. Generic group model suffers from uninstantiability results and evasive lattice assumptions are of knowledge type, both of which are too strong and unsatisfactory. While they provide a first step towards feasibility, it is highly desirable to be able to prove security from much weaker assumptions. Some recent works have created new lattice-based tools for ABE and research along this line might eventually remove the strong assumptions used in our schemes.

- Resolving the Pareto front of ABE succinctness and efficiency. Currently, only two points are known on the optimal succinctness-efficiency trade-off curve of

ABE, one for optimal component size and one for optimal decryption time. The rest of the recurve remains a mystery and is worth careful investigation.

- Understanding the correct accounting model of succinctness and efficiency. When an n-bit attribute is provided verbatim for free (the current standard formulation), known constructions require additional n bits to achieve optimal decryption efficiency. If the attribute is not provided for free, then the total component size becomes $2n$ bits. It might be possible to reduce the rate 2 if the attribute is cleverly encoded in the components in a way that supports optimal decryption time and does not entail a verbatim copy. In that case, the correct accounting model is not to consider the attribute as provided verbatim for free, contrary to the conventional wisdom of counting "cryptographic overhead".

Chapter 2

Preliminaries

We only consider asymptotic security definitions in this dissertation, which are about the behavior of cryptographic schemes as the security parameter approaches infinity. Let $\lambda \in \mathbb{N}$ be the (computational) security parameter and $\kappa \in \mathbb{N}$ the statistical security parameter such that $\kappa \in \mathrm{poly}(\lambda) \cap \omega(\log \lambda)$. Unless necessary, they are omitted for brevity, except in definitions. We write $\mathrm{poly}(\cdot)$ for either an unspecified polynomially bounded function or the set of such functions. A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ of λ is *negligible*, denoted $\varepsilon = \mathrm{negl}(\lambda)$, if $\varepsilon = \mathrm{O}(\lambda^{-k})$ for all $k \in \mathbb{N}$. By the relation between λ and κ, we have $2^{-\kappa} = \mathrm{negl}(\lambda)$.

Efficient algorithms in cryptographic schemes are probabilistic polynomial-time machines. We stress that the running time is polynomial in the total input length, not just the security parameter. Efficient adversaries can be modeled as probabilistic polynomial-time machines with or without polynomially long advice. (Proofs only use uniform reduction algorithms.) The input to an adversary is 1^λ, potentially with another $\mathrm{poly}(\lambda)$-bit string, so its total running time (as well as advice size) is always $\mathrm{poly}(\lambda)$.

An experiment is a randomized process that given λ and an adversary, produces a one-bit output. By a *hybrid*, we mean an experiment used in the proofs, often obtained by modifying some other experiment. Given an adversary $\mathcal{A}$ and two experiments $\mathsf{Exp}^0, \mathsf{Exp}^1$, its *advantage* (in distinguishing them) is a function mapping λ to

$$\Pr\left[\mathsf{Exp}^0(1^\lambda, \mathcal{A}) \to 1\right] - \Pr\left[\mathsf{Exp}^1(1^\lambda, \mathcal{A}) \to 1\right].$$

We say that they are *(computationally) indistinguishable*, denoted $\mathsf{Exp}^0 \approx \mathsf{Exp}^1$, if the advantage of every efficient adversary is negligible. We write $\mathsf{Exp}^0 \approx_{\mathrm{s}} \mathsf{Exp}^1$ if they are *statistically indistinguishable*, i.e., the advantage of every (potentially inefficient) adversary is negligible, and $\mathsf{Exp}^0 \equiv \mathsf{Exp}^1$ if they are *identical*, i.e., the advantage of every

adversary is zero. The same notations are used for sequences of distributions — the experiments give the adversary a sample from the corresponding distribution and output whatever the adversary outputs.

Vectors are written in lower-case boldfaced letters, and matrices, upper-case. Strings might be boldfaced. They are always indexed using brackets and never subscripts, e.g., $\mathbf{a}_i$ is a vector and $\mathbf{A}[i,j]$ is a scalar. The inner product of $\mathbf{u},\mathbf{v}$ is $\langle \mathbf{u},\mathbf{v}\rangle = \mathbf{u}^\intercal\mathbf{v}$. We write $\mathbf{I}_n$ (or simply $\mathbf{I}$) for the $n\times n$ identity matrix, $\mathbf{0}_{n\times m}$ (or $\mathbf{0}_n$ or $\mathbf{0}$) for the $n\times m$ (or $n\times 1$ or appropriately shaped) zero matrix. When the dimension is clear, $\boldsymbol{\iota}_1,\boldsymbol{\iota}_2,\ldots$ represent the standard basis vectors. We consider the infinity norm and its operator norm:

$$\|\mathbf{a}\| = \max_i |\mathbf{a}[i]|, \qquad\qquad \|\mathbf{A}\| = \max_i \sum_j |\mathbf{A}[i,j]|.$$

We strictly follow the convention of vectors being columns. If $\mathbf{a}\in\mathbb{Z}^z$ is a vector, then $\|\mathbf{a}^\intercal\|$ is an operator norm and $\|\mathbf{a}^\intercal\| \le z\|\mathbf{a}\|$. For two matrices $\mathbf{A},\mathbf{B}$ of shapes $n_1\times m_1$ and $n_2\times m_2$, their Kronecker product is an $n_1n_2\times m_1m_2$ matrix,

$$\mathbf{A}\otimes\mathbf{B} = \begin{pmatrix} \mathbf{A}[1,1]\mathbf{B} & \cdots & \mathbf{A}[1,m_1]\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}[n_1,1]\mathbf{B} & \cdots & \mathbf{A}[n_1,m_1]\mathbf{B} \end{pmatrix}.$$

A useful property is $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = \mathbf{AC}\otimes\mathbf{BD}$ whenever all the multiplications are compatible. The by-column flattening operator $\mathrm{col}(\cdot)$ concatenates all the columns of its input matrix. A convenient fact is $\mathrm{col}(\mathbf{ABC}) = (\mathbf{C}^\intercal\otimes\mathbf{A})\mathrm{col}(\mathbf{B})$ whenever $\mathbf{ABC}$ is compatible. We write $\star$ for the Kleene star, extend its usage to rows and columns of matrices, and overload it also for unnamed suitable value. For example, $\mathbf{E}\in\mathbb{Z}^{2\times\star}$ could mean that $\mathbf{E}$ is a 2-row integer matrix of some suitable number of columns so that its operations with other vectors and matrices are compatible, which should be clear from the context.

Intervals are intersected with integers, e.g., $[a,b] = \{\, x\in\mathbb{Z} \mid a\le x\le b \,\}$. We write $[n]$ for $[1,n]$. Let $q\in\mathbb{N}_{\ge 2}$, we denote by $\mathbb{Z}_q$ the integers modulo q. Matrices over $\mathbb{Z}$ are naturally and implicitly mapped to those over $\mathbb{Z}_q$ so that they can be arbitrarily mixed for various operations. When $z\in\mathbb{Z}_q$ and $S\subseteq\mathbb{Z}$, saying $z\in S$ means that a

representative of z is in S. Given an object, we write $\mathsf{bits}(\cdots)$ for its fixed-length bit representation, arranged in a row, i.e., a matrix in $\{0,1\}^{1\times L}$ for some L. For $x \in \mathbb{R}$, we define $\lfloor x \rceil$ to be $\left\lfloor x + \frac{1}{2} \right\rfloor$. Bits and rounding extend to matrices entry-wise. The operator "$\|$" denotes string concatenation, which might also be denoted by juxtaposition. We denote by $C[x_1]$ the circuit C with x_1 hardwired as the first portion of input so that $C[x_1](x_2) = C(x_1, x_2)$.

2.1 Partially Hiding Functional Encryption

We define partially hiding functional encryption (PHFE) with respect to functionality

$$\varphi : F \times X \times Y \to \{\bot\} \cup (\mathbb{N}_+ \times Z),$$

where F, X, Y, Z are the sets of function description, public input, private input, and output, respectively. Each key is associated with some $f \in F$, and each ciphertext encrypts some private $y \in Y$ and is tied to some public $x \in X$. The decryptor should be able to recover z if $\varphi(f, x, y) = (T, z)$, in which case T is often the time to compute z from f, x, y in the clear and can serve as a baseline for decryption efficiency. For security, we only consider f, x, y for which T is polynomially bounded. On the other hand, when $\varphi(f, x, y) = \bot$, we require neither correctness nor security. The primitive is hence a *promise* one. It allows us to disregard non-halting or overflowing computations. The formulation was developed during the research of Chapter 9.

Definition 2.1 (PHFE). Let $\Phi = \{\varphi_{\lambda,\mathsf{param}}\}_{\lambda\in\mathbb{N},\mathsf{param}\in\mathsf{Params}_\lambda}$ be a family of functionalities, where $\varphi_{\lambda,\mathsf{param}}$ is

$$F_{\lambda,\mathsf{param}} \times X_{\lambda,\mathsf{param}} \times Y_{\lambda,\mathsf{param}} \to \{\bot\} \cup (\mathbb{N}_+ \times Z_{\lambda,\mathsf{param}}) \quad \text{for all } \lambda \in \mathbb{N},\ \mathsf{param} \in \mathsf{Params}_\lambda.$$

A *partially hiding functional encryption scheme* for Φ consists of four algorithms.

- $\mathsf{Setup}(1^\lambda, \mathsf{param})$ takes a functionality description $\mathsf{param} \in \mathsf{Params}_\lambda$ as input, and it outputs a pair of master public/secret keys $(\mathsf{mpk}, \mathsf{msk})$. The algorithm must be efficient.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, f)$ takes as input msk and a function description $f \in F_{\lambda,\mathsf{param}}$, and it outputs a secret key sk_f for f. The algorithm must be efficient.

- $\mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, y)$ takes as input mpk, a public input $x \in X_{\lambda,\mathsf{param}}$, and a private input $y \in Y_{\lambda,\mathsf{param}}$. It outputs a ciphertext ct_x of y tied to x. The algorithm must be efficient.

- $\mathsf{Dec}^{\mathsf{mpk},f,\mathsf{sk}_f,x,\mathsf{ct}_x}(1^\lambda)$ is given random access to $\mathsf{mpk}, f, \mathsf{sk}_f, x, \mathsf{ct}_x$. It is supposed to compute z efficiently if $\varphi_{\lambda,\mathsf{param}}(f,x,y) = (T,z) \neq \bot$, in which case it must halt in time $\mathrm{poly}(\lambda, |\mathsf{param}|, |f|, |x|, |y|, T)$.

The scheme is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, $\mathsf{param} \in \mathsf{Params}_\lambda$, and $f \in F_{\lambda,\mathsf{param}}$, $x \in X_{\lambda,\mathsf{param}}$, $y \in Y_{\lambda,\mathsf{param}}$ such that $\varphi_{\lambda,\mathsf{param}}(f,x,y) = (T,z) \neq \bot$, it holds that

$$\Pr\left[\begin{array}{lll}(\mathsf{mpk},\mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda,\mathsf{param}) & & \\ \mathsf{sk}_f \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda,\mathsf{msk},f) & : & \mathsf{Dec}^{\mathsf{mpk},f,\mathsf{sk}_f,x,\mathsf{ct}_x}(1^\lambda) = z \\ \mathsf{ct}_x \xleftarrow{\$} \mathsf{Enc}(1^\lambda,\mathsf{mpk},x,y) & & \end{array}\right] = 1.$$

If $T \leq \mathrm{poly}(\lambda, |\mathsf{param}|, |f|, |x|, |y|)$ holds whenever $\varphi_{\lambda,\mathsf{param}}(f,x,y) = (T,z) \neq \bot$ and the efficiency is characterized without T,[2] it is not important hence can be omitted and the codomain of $\varphi_{\lambda,\mathsf{param}}$ can be regarded as $\{\bot\} \cup Z_{\lambda,\mathsf{param}}$. The components $\mathsf{mpk}, f, \mathsf{sk}_f, x, \mathsf{ct}_x$ are put at superscript position to emphasize random access (Dec might run in time shorter than their total length as it might not query every bit in them). If the precise efficiency is not important, those components can be written in the parentheses to improve readability. In group- or pairing-based schemes, the functionalities depend on gp (Definitions 2.4 and 2.5), Setup additionally takes gp as input, and correctness is quantified over all gp in a support of the output of $\mathsf{GroupGen}(1^\lambda)$.[3] For brevity, gp is implicit in $\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}$ in constructions.

We consider perfect correctness by default. Weaker variants are defined as needed.

Security. We consider the standard ciphertext indistinguishability (IND-CPA) security for polynomially bounded T. Intuitively, it says that the only information revealed by ciphertexts and secret keys is from the functionality of the scheme.

[2] This is the case except for PHFE for TM/RAM.

[3] See Definition 4.4 and Construction 4.3 for examples.

Definition 2.2 (PHFE security). A PHFE scheme (Definition 2.1) is *(adaptively IND-CPA) secure* if $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$, where $\mathsf{Exp}^\beta_{\mathrm{PHFE}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive from it some $\mathsf{param} \in \mathsf{Params}_\lambda$ and $1^{\overline{T}}$. Run

$$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{param})$$

  and send mpk to $\mathcal{A}$.

- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ submits some $f_j \in F_{\lambda,\mathsf{param}}$. Upon receiving such query, run

$$\mathsf{sk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, f_j)$$

  and send sk_j to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ submits $x \in X_{\lambda,\mathsf{param}}$ and $y_0, y_1 \in Y_{\lambda,\mathsf{param}}$. Upon the challenge, run

$$\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, y_\beta)$$

  and send ct to $\mathcal{A}$.

- **Query II.** Same as Query I.

- **Guess.** $\mathcal{A}$ outputs a bit β'. The outcome of the experiment is β' if

$$\begin{aligned} & |y_0| = |y_1|, \\ \text{and} \quad & \varphi_{\lambda,\mathsf{param}}(f_j, x, y_0) = \varphi_{\lambda,\mathsf{param}}(f_j, x, y_1) = (T_j, z_j) \neq \perp \quad \text{for all } j, \\ \text{and} \quad & T_j \leq \overline{T} \quad \text{for all } j. \end{aligned}$$

  Otherwise, the outcome is set to 0.

For *selective (IND-CPA) security*, the adversary $\mathcal{A}$ must choose x, y_0, y_1 together with $\mathsf{param}, 1^{\overline{T}}$ at the beginning. For *very selective (IND-CPA) security* [AWY20] (also known as *static* [RW15]), it must choose $x, y_0, y_1, \{f_j\}_j$ (everything) together with $\mathsf{param}, 1^{\overline{T}}$ at the beginning. In group- or pairing-based schemes, gp (Definitions 2.4 and 2.5) is freshly sampled by $\mathsf{GroupGen}(1^\lambda)$, and the adversary's choice can depend on gp even for (very) selective security. When T is not important (see Definition 2.1), the adversary does not have to provide $1^{\overline{T}}$ and the checks of $T_j \leq \overline{T}$ pass by default.

Instantiations. Definition 2.1 can be instantiated into ABE.

Definition 2.3 (ABE). Let $P = \{P_{\lambda,\mathsf{param}}\}_{\lambda\in\mathbb{N},\mathsf{param}\in\mathsf{Params}_\lambda}$ be a family of predicates, where

$$P_{\lambda,\mathsf{param}} : F_{\lambda,\mathsf{param}} \times X_{\lambda,\mathsf{param}} \to \{\bot\} \cup (\mathbb{N}_+ \times \{0,1\}) \quad \text{for all } \lambda \in \mathbb{N},\ \mathsf{param} \in \mathsf{Params}_\lambda.$$

An *attribute-based encryption scheme* for P is a PHFE scheme (Definition 2.1) with

$$\mu \in Y_{\lambda,\mathsf{param}} = \{0,1\}, \qquad Z_{\lambda,\mathsf{param}} = \{(1,0),(1,1),(0,\bot)\},$$

$$\varphi_{\lambda,\mathsf{param}}(f,x,\mu) = \begin{cases} (T,(1,\mu)), & \text{if } P_{\lambda,\mathsf{param}}(f,x) = (T,1); \\ (T,(0,\bot)), & \text{if } P_{\lambda,\mathsf{param}}(f,x) = (T,0); \\ \bot, & \text{if } P_{\lambda,\mathsf{param}}(f,x) = \bot. \end{cases}$$

Again, T can be omitted if not important (see Definition 2.1). The message space $Y_{\lambda,\mathsf{param}}$ can be other sets with at least two elements. For ABE, it is customary to output μ in place of $(1,\mu)$ and $\bot$ in place of $(0,\bot)$.

When $Y_{\lambda,\mathsf{param}} = \{0,1\}$, the experiment $\mathsf{Exp}^{\beta}_{\mathrm{ABE}}$ is a modified version of $\mathsf{Exp}^{\beta}_{\mathrm{PHFE}}$, where μ is prescribed to be β instead of being chosen by the adversary.

Note that in the customary format of Dec, outputting $\bot$ is not the same as $P(f,x) = \bot$ — in the latter case, there is no guarantee of the behavior of Dec. Also, when $Y \neq \{0,1\}$, we stick to $\mathsf{Exp}^{\beta}_{\mathrm{PHFE}}$, leaving $\mathsf{Exp}^{\beta}_{\mathrm{ABE}}$ undefined.

Symbols. We use symbols more meaningful for the input values in PHFE and ABE. In ABE, the private input y (per Definition 2.1) becomes μ (or another symbol for the message). In CP-ABE, f (per PHFE) is replaced by $\mathbf{x}$ (or x), and x (per PHFE) is replaced by M (machine) or C (circuit) or f (function or formulae).

2.2 Groups and Pairing Groups

In some chapters, we use prime-order (pairing) groups in which the matrix decisional Diffie–Hellman (MDDH) assumption holds.

Definition 2.4 (group). A *(prime-order) group sampler* is an efficient algorithm $\mathsf{GroupGen}(1^\lambda)$ that outputs a group parameter $\mathsf{gp} = (p,\mathbb{G},g,\dots)$, where p is a prime,

$\mathbb{G}$ is a group of order p (written additively), and g generates $\mathbb{G}$. The group operations (subtraction and zero-testing) are efficiently computable from gp and the operand(s).

Definition 2.5 (pairing group). A *(prime-order) pairing group sampler* is an efficient algorithm $\mathsf{GroupGen}(1^\lambda)$ that outputs a pairing group parameter $\mathsf{gp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, g_\mathrm{T}, e, \dots)$, where p is a prime, $\mathbb{G}_i$ is a group of order p (written additively) generated by g_i for all $i \in \{1, 2, \mathrm{T}\}$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathrm{T}$ satisfies $e(ag_1, bg_2) = abg_\mathrm{T}$ for all $a, b \in \mathbb{Z}_p$. The group operations and e are efficiently computable given gp and the operand(s).

We assume that group elements are uniquely encoded. A pairing group naturally induces three groups, $(p, \mathbb{G}_1, g_1, \mathsf{gp})$ and so on.

Bracket Notation. For a group, when gp is clear from the context (hence omitted), we write $[\![a]\!]$ for ag and allow operations with numbers, so $a[\![b]\!] = [\![a]\!]b = [\![ab]\!]$. The notation is extended entry-wise to matrices and $\mathbf{A}[\![\mathbf{B}]\!]^\intercal\mathbf{C} = [\![\mathbf{A}\mathbf{B}^\intercal\mathbf{C}]\!]$ can be efficiently computed from $\mathbf{A}, [\![\mathbf{B}]\!], \mathbf{C}$. For a pairing group, $[\![a]\!]_i$ means ag_i for $i \in \{1, 2, \mathrm{T}\}$, and $[\![\mathbf{A}]\!]_2\mathbf{B}[\![\mathbf{C}]\!]_1 - \mathbf{D}[\![\mathbf{E}]\!]_\mathrm{T} = [\![\mathbf{A}\mathbf{B}\mathbf{C} - \mathbf{D}\mathbf{E}]\!]_\mathrm{T}$ can be efficiently computed given $[\![\mathbf{A}]\!]_2, \mathbf{B}, [\![\mathbf{C}]\!]_1$, $\mathbf{D}, [\![\mathbf{E}]\!]_\mathrm{T}$, thanks to e.

Matrix Decisional Diffie–Hellman Assumption. We consider the MDDH assumption over a group (on its own) or a group in a pairing group.

Assumption 2.1 (MDDH [EHK+13]). Let GroupGen be a group sampler (Definition 2.4) and k, ℓ, q polynomially bounded functions of λ. The $\mathsf{MDDH}^q_{k,\ell}$ assumption over $\mathbb{G}$ is

$$\{(1^\lambda, \mathsf{gp}, [\![\mathbf{A}]\!], [\![\mathbf{S}^\intercal\mathbf{A}]\!])\}_{\lambda\in\mathbb{N}} \approx \{(1^\lambda, \mathsf{gp}, [\![\mathbf{A}]\!], [\![\mathbf{C}^\intercal]\!])\}_{\lambda\in\mathbb{N}},$$

where $(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k(\lambda)\times\ell(\lambda)}$, $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{k(\lambda)\times q(\lambda)}$, $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_p^{\ell(\lambda)\times q(\lambda)}$. For a pairing group sampler, $\mathsf{MDDH}^q_{k,\ell}$ over $\mathbb{G}_i$ is similarly defined for $i \in \{1, 2, \mathrm{T}\}$ (the distribution contains gp of the pairing). By default, $\ell(\lambda) = k(\lambda) + 1$ and $q(\lambda) = 1$.

It is known [EHK+13] that k-Lin (an assumption not defined here) implies MDDH_k, which further implies $\mathsf{MDDH}^q_{k,\ell}$ for all $\ell, q \le \mathrm{poly}(\lambda)$.

2.3 Lattices

Let $n, m \geq 1$ and $q \geq 2$ be integers such that $\log_2 q \leq m/n \in \mathbb{Z}$. Let

$$\mathbf{g} = (2^0, \ldots, 2^{m/n-1})^\intercal \qquad \text{and} \qquad \mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\intercal$$

be the gadget vector and the gadget matrix. Given $\mathbf{p} \in \mathbb{Z}_q^n$, we denote by $\mathbf{G}^{-1}(\mathbf{p})$ the vector $\mathbf{k} \in \{0,1\}^m$, where $\mathbf{k}[(i-1)\cdot m/n + 1], \ldots, \mathbf{k}[i \cdot m/n]$ are the bits of (the non-negative representative less than q of) $\mathbf{p}[i]$, low to high. It holds that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{p}) = \mathbf{p}$. The notation $\mathbf{G}^{-1}(\cdot)$ extends to matrices column-wise.

Given $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{p} \in \mathbb{Z}_q^n$ such that $\mathbf{B}\mathbf{k} = \mathbf{p}$ has a solution $\mathbf{k}^* \in \mathbb{Z}^m$, write

$$\Lambda_{\mathbf{p}}^{\perp}(\mathbf{B}) = \{\, \mathbf{k} \in \mathbb{Z}^m \mid \mathbf{B}\mathbf{k} = \mathbf{p} \,\} = \mathbf{k}^* + \Lambda_{\mathbf{0}}^{\perp}(\mathbf{B}),$$

which is a lattice coset. Let Λ' be any lattice coset and $\sigma \geq 0$, we denote by $\mathcal{D}_{\Lambda',\sigma}$ the discrete Gaussian distribution [MP11] over Λ' with width σ. For perfect correctness, it is necessary to truncate unbounded noises in algorithms.

Definition 2.6 (truncated $\mathcal{D}_{\mathbb{Z},\sigma}$). Let $B \geq 0$, the distribution $\mathcal{D}_{\mathbb{Z},\sigma,\leq B}$ is sampled by first sampling $x \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma}$, then returning x if $|x| \leq B$, and 0 otherwise.

Lemma 2.1 (truncation of $\mathcal{D}_{\mathbb{Z},\sigma}$). *For all $\sigma \geq 1$ and $\kappa \geq 2$,*

$$\Pr\left[\, x \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma} \;:\; |x| > \sigma\sqrt{\kappa} \,\right] \leq 2^{-\kappa},$$

so $\mathcal{D}_{\mathbb{Z},\sigma,\leq\sigma\sqrt{\kappa}}$ is $2^{-\kappa}$-close to $\mathcal{D}_{\mathbb{Z},\sigma}$.

We will also need noise flooding with discrete Gaussian.

Lemma 2.2 (noise flooding using $\mathcal{D}_{\mathbb{Z},\sigma}$). *For all $y \in \mathbb{Z}$ and $\sigma \geq 2^{\kappa+6}y$, it holds that $(y + \mathcal{D}_{\mathbb{Z},\sigma})$ is $2^{-\kappa}$-close to $\mathcal{D}_{\mathbb{Z},\sigma}$. As a corollary, $(y + \mathcal{D}_{\mathbb{Z},\sigma,\leq\sigma\sqrt{\kappa}})$ is $2^{-\Omega(\kappa)}$-close to $\mathcal{D}_{\mathbb{Z},\sigma,\leq\sigma\sqrt{\kappa}}$.*

Trapdoor Generation and Gaussian Sampling. We need the following lemma.

Lemma 2.3 ([MP12; Theorem 2]). *There are efficient algorithms* TrapGen *and* SampD*, and functions $m_0 \in \Theta(n \log q)$ and $\sigma_0 \in \omega(\sqrt{m \log m}) \cap O(m)$ satisfying these conditions.*

- $\mathsf{TrapGen}(1^n, 1^m, q)$ *takes as input $n \geq 1$, $q \geq 2$, and $m \geq m_0(n,q)$. It outputs $(\mathbf{B}, \tau)$ such that $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{B}$ is $\mathrm{negl}(n)$-close to uniform over $\mathbb{Z}_q^{n\times m}$.*
- $\mathsf{SampD}(\mathbf{B}, \tau, \mathbf{p}, \sigma)$ *takes as input $\mathbf{B}, \tau$ from* TrapGen*, some $\mathbf{p} \in \mathbb{Z}_q^n$, and $\sigma \geq \sigma_0(n,m)$. It outputs $\mathbf{k} \in \mathbb{Z}^m$ such that $\mathbf{B}\mathbf{k} = \mathbf{p}$, $\|\mathbf{k}\| \leq \sigma\sqrt{m}$, and $\mathbf{k}$ is $\mathrm{negl}(n)$-close to $\mathcal{D}_{\Lambda_{\mathbf{p}}^{\perp}(\mathbf{B}),\sigma}$.*

Batch Notation. It is convenient to extend SampD to process multiple $\mathbf{p}$'s in one shot. Let $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_{m'})$ be a matrix or a batch of vectors, then $\mathsf{SampD}(\mathbf{B}, \tau, \mathbf{P}, \sigma)$ is

$$\mathbf{K} \xleftarrow{\$} \big(\mathsf{SampD}(\mathbf{B}, \tau, \mathbf{p}_1, \sigma), \ \ldots, \ \mathsf{SampD}(\mathbf{B}, \tau, \mathbf{p}_{m'}, \sigma)\big),$$

with fresh randomness for each call to SampD on the right-hand side. We also write $\mathbf{B}^{-1}(\mathbf{P})$ for an output of $\mathsf{SampD}(\mathbf{B}, \tau, \mathbf{P}, \sigma)$.

Learning with Errors Assumption. We rely on the LWE assumption.

Assumption 2.2 (LWE [Reg05]). Let $n, m \le \mathrm{poly}(\lambda)$, $q \le 2^{\mathrm{poly}(\lambda)}$, $\sigma = \sigma(\lambda)$, and

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_{q(\lambda)}^{n(\lambda) \times m(\lambda)}, \qquad \mathbf{s} \xleftarrow{\$} \mathbb{Z}_{q(\lambda)}^{n(\lambda)}, \qquad \mathbf{e} \xleftarrow{\$} \mathbb{Z}_{q(\lambda)}^{m(\lambda)}, \qquad \boldsymbol{\delta} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma(\lambda)}.$$

The *LWE assumption* $\mathsf{LWE}_{n,m,q,\sigma}$ states that

$$\{(1^\lambda, \mathbf{A}, \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal)\}_{\lambda \in \mathbb{N}} \approx \{(1^\lambda, \mathbf{A}, \boldsymbol{\delta}^\intercal)\}_{\lambda \in \mathbb{N}}.$$

Assumption 2.3 (flipped LWE). Let $N, m \le \mathrm{poly}(\lambda)$, $q \le 2^{\mathrm{poly}(\lambda)}$, $\sigma = \sigma(\lambda)$, and

$$\mathbf{A} \xleftarrow{\$} \{0,1\}^{N(\lambda) \times m(\lambda)}, \qquad \mathbf{s} \xleftarrow{\$} \mathbb{Z}_{q(\lambda)}^{N(\lambda)}, \qquad \mathbf{e} \xleftarrow{\$} \mathbb{Z}_{q(\lambda)}^{m(\lambda)}, \qquad \boldsymbol{\delta} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma(\lambda)}.$$

The *flipped LWE assumption* $\mathsf{FlipLWE}_{N,m,q,\sigma}$ states that

$$\{(1^\lambda, \mathbf{A}, \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal)\}_{\lambda \in \mathbb{N}} \approx \{(1^\lambda, \mathbf{A}, \boldsymbol{\delta}^\intercal)\}_{\lambda \in \mathbb{N}}.$$

Lemma 2.4 (flipped LWE). *Suppose* $\mathsf{LWE}_{n,m,q,\sigma}$ *holds and* $N \le \mathrm{poly}(\lambda)$ *satisfies*

$$0 \le N/n - \log_2 q \in \omega(\log \lambda),$$

then $\mathsf{FlipLWE}_{N,m,q,\sigma}$ *holds.*

Parameters. In this dissertation, we rely on LWE with subexponential modulus-to-noise ratio, i.e., $n = \lambda^{\Theta(1)}$, $m \le \mathrm{poly}(n)$, $q \le 2^{\mathrm{poly}(n)}$, $\frac{q}{\sigma\sqrt{n}} = \Theta(2^{n^\rho})$, where $0 < \rho < \frac{1}{2}$ is a constant. Strictly speaking, those parameters should be chosen by the adversary subject to those constraints — many algorithms require a prime q, it is currently unknown [TCH12] how to pick $\mathrm{poly}(\lambda)$-bit primes in deterministic polynomial time, and the adversary's choice of scheme parameters affects the range of LWE parameters.

Part I

General Paradigm

Chapter 3

Introduction

In ABE, revelation or hiding of the message is dictated by a *joint* computation on policy and attribute, which are *separately* chosen by different parties in the cryptosystem. In addition, the hiding case should prevail even when there are *multiple* unauthorized keys. Either of tying independent data and remaining secure against collusion is non-trivial on their own, and ABE combines both — constructing ABE schemes is no easy task. To tackle it, we would like to find a general paradigm, or framework, that is modular (to compartmentalize complexity), powerful (to imply new results), and versatile (to be instantiable under multiple assumptions).

In this chapter, we present our general paradigm for constructing ABE schemes, which is instantiated in the other chapters in this part. We also give a bird's-eye view of the instantiations.

Existing Methods. There are three major existing methods of constructing ABE. None is flexible to be instantiated under different kinds of assumptions.

Dual system encryption was introduced by Waters [Wat09] for obtaining adaptively secure identity-based encryption, a special kind of attribute-based encryption. It later found applications in constructing and proving (adaptive) security of ABE — it has become the most used method of constructing ABE from pairing-based assumptions. It has been refined into pair encoding [Att14] and predicate encryption [Wee14]. One downside is that dual system encryption is only known to be instantiable under pairing-based assumptions. In addition, it is not exactly easy to learn or use, and works using dual system encryption often involves heavy algebraic manipulation in its security proofs (in fact, see Chapter 5 for an example).

For lattice-based ABE, the known methods are two-to-one recoding [GVW13] and key-homomorphic encryption [BGG+14]. They only imply new results under lattice-based assumptions, and do not have many instantiations.

3.1 ABE ⇐ IPFE ∘ Garbling

One way to construct ABE is to reduce it to functional encryption (FE). In FE, a ciphertext encrypts some input y and a key is associated with a function f'. Decryption recovers $f'(y)$ but nothing else about y. Let the ABE ciphertext of μ with attribute x be an FE ciphertext of $y = (x, \mu)$ and the ABE key with policy f an FE key with $f'(y) = \mu \cdot f(x)$, then upon decryption, we learn μ if $f(\mathbf{x}) \neq 0$. The security of such an ABE scheme follows from that of the underlying FE.

Of course, FE is harder to construct than ABE (if they have *on par* functionalities). If we look more closely, the computation $f'(y) = \mu \cdot f(x)$ is high-degree in x while being linear in μ. Using FE is an overkill — it hides x (by extension, a very complex computation on x), but in ABE, x does not have to be hidden, only to be *binding*, and its computation only needs to be *authenticated*. If we could decompose $f(x)$ to lower the degree in x, it might help, as the FE might only need to hide a low-degree (in fact, linear) computation. For degree reduction, we use linear partially hiding garbling schemes (for conditional secret disclosure; i.e., secret sharing schemes). FE for linear functions are known as inner-product functional encryption (IPFE).

Linear Garbling. A garbling scheme [Yao82,Yao86,AIK11] decomposes a computation $f(\mathbf{x})$ into three steps. First, it transforms a (high-degree) function f of $\mathbf{x}$ into label functions $L_1, \ldots, L_m$ that are affine in $\mathbf{x}$ — linear in $(1, \mathbf{x}^\intercal)$. Pragmatically, the label functions are represented by their coefficients $\mathbf{L}_1, \ldots, \mathbf{L}_m$. Next, L's are evaluated at $\mathbf{x}$, producing the labels $\ell_1 = L_1(\mathbf{x}) = (1, \mathbf{x}^\intercal)\mathbf{L}_1, \ldots, \ell_m = L_m(\mathbf{x}) = (1, \mathbf{x}^\intercal)\mathbf{L}_m$. Lastly, the labels are used to reconstruct $f(\mathbf{x})$. The second step is a simple computation by definition, since L's are affine in $\mathbf{x}$ — linear in $(1, \mathbf{x})$. The last step is as high-degree as f is in $\mathbf{x}$. The standard security of garbling guarantees that the labels reveal nothing about $\mathbf{x}$ beyond $f(\mathbf{x})$.

When applied to ABE, the message μ is part of the computation and needs to be hidden, but $\mathbf{x}$ can be public, which lessens the security requirement. This primitive is known as partially hiding garbling [IW14]. Refined to our application, its basic syntax is as follows.

- The garbling procedure $\mathsf{Garble}(f, \mu)$ transforms f and μ into label functions, outputting their coefficients $\mathbf{L}_1, \ldots, \mathbf{L}_m$.
- The labels are $\ell_1 = (1, \mathbf{x}^\intercal)\mathbf{L}_1, \ldots, \ell_m = (1, \mathbf{x}^\intercal)\mathbf{L}_m$.
- The evaluation procedure $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ outputs $\mu \cdot f(\mathbf{x})$.

There are usually additional linearity properties required for the garbling scheme to be compatible with IPFE to obtain ABE. They will be specified in each instantiation. The security requirement is that $\ell_1, \ldots, \ell_m$ does not reveal any information about μ if $f(\mathbf{x}) = 0$ — it might well reveal the entirety of $\mathbf{x}$, i.e., partially hiding. Its security can be information-theoretic or just computational.

Linear garbling converts a complex computation into a simple one (computing the labels) followed by a complex one (running Eval). The benefit is that the security requirement is contained inside the simple step — the Eval step does not have to be protected, since its input, namely the labels, already has all the information about μ (when $f(\mathbf{x}) = 0$) removed.

Inner-Product Functional Encryption. IPFE is functional encryption for linear functions. Each key $\mathsf{isk}(\mathbf{v})$ is associated with a (key) vector $\mathbf{v}$ and each ciphertext $\mathsf{ict}(\mathbf{u})$ encrypts a (plaintext) vector $\mathbf{u}$. Upon decryption, the decryptor learns $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^\intercal \mathbf{v}$ but nothing else. For this overview, the easiest mental model is to regard $\mathbf{v}, \mathbf{u}$ both as hidden. The functionality and security of IPFE lends itself well to *securely* computing the labels, as explained below.

Composing IPFE and Garbling. Given IPFE and linear garbling, the straight-forward way to construct ABE is

$$\left.\begin{aligned} \mathsf{sk}_f(\mu) &= \big(\mathsf{isk}(\mathbf{L}_1), \ldots, \mathsf{isk}(\mathbf{L}_m)\big) \\ \mathsf{ct}_{\mathbf{x}} &= \mathsf{ict}(1, \mathbf{x}^\intercal) \end{aligned}\right\} \xrightarrow[\text{decryption}]{\text{IPFE}} (\ell_1, \ldots, \ell_m) \xrightarrow[\text{evaluation}]{\text{garbling}} \mu \cdot f(\mathbf{x}).$$

(Ignore for the moment the issue of message being associated with keys.) Functionality works by composition. Some IPFE schemes only outputs the inner products in an encoded form (namely, inside the exponent of a group with hard discrete logarithm), so the computation doable to the labels are limited. Others might output approximate

inner products. Depending on the IPFE scheme used, the Eval algorithm of the linear garbling scheme is required to have certain specific format.

Intuitively, security also follows by composition. The IPFE keys has information about μ because it encodes the coefficients of label functions, which are created dependent on f, μ. But, since the IPFE is secure, the only information revealed to the adversary is the labels. Now that the linear garbling scheme is secure, the labels convey no information about μ if $f(\mathbf{x}) = 0$. This concludes the security of the proposed ABE scheme.

However, composition of security turns out to be *non-trivial*. One consistent theme in the line of research using this paradigm is to formulate the security definitions of IPFE and garbling in a compatible way to achieve ABE security.

Aside from that, the aforementioned proposal is only one-time secure. When there are two ciphertexts, the same set of label functions are evaluated twice, on potentially different inputs (attributes) — the security of garbling no longer applies. For multi-time security, each pair of ABE keys and ciphertexts should use a *fresh* set of label functions. This can be achieved in a computational manner, i.e., each pair uses pseudorandomness to generate the label functions. To implement it using IPFE, we rely on linear rerandomization such as DDH-based or LWE-based techniques.

3.2 Bird's-Eye View of Instantiations

To summarize, we propose the general paradigm of constructing ABE by composing IPFE and linear garbling. Under appropriate formulations, functionality follows from those of the underlying primitives, and security is a three-wise composition:

- IPFE security ensures that only rerandomized labels are revealed;
- a "raw" assumption (such as DDH or LWE) ensures that the rerandomized labels indeed use pseudorandomness;
- garbling security ensures that the message is hidden (in every set of labels) if decryption is unauthorized.

The paradigm is **modular** as it hides most "raw" usages of assumptions into the

security of IPFE and garbling. As shown in Table 3.1, it is **powerful** as it has helped achieving various ABE with better properties than previously known, and it is **versatile** as it can be instantiated with pairing- or lattice-based assumptions, or a combination of both.

Table 3.1. Summary of instantiations of the general paradigm.

	in...	**new ABE results**	**IPFE**	**garbling**
	Chapter 4	compact and adaptive for ABP, DFA for NFA (also compact and adaptive) for L, NL (also adaptive)	pairing	I.T.
change IPFE	Chapter 5	succinct and adaptive for ABP		
change garbling	Chapter 6	constant-size-key KP for circuits doubly succinct CP for formulae		lattice
change IPFE	Chapter 7	CP for circuits from fewer kinds of assumptions for DFA, L from lattices	lattice	I.T.

DFA, NFA are (non-)deterministic finite automata; L, NL are (non-)deterministic logspace; RAM is random-access machine; KP is key-policy; CP is ciphertext-policy; I.T. is information-theoretic.

Chapter 4

Compact and Adaptively Secure ABE from k-Lin: Beyond NC^1 and Towards NL

(Pairing-Based IPFE, Noiseless Linear Garbling)

This chapter is an adaptation of [LL20b,LL20a].[4]

In this chapter, we present the first instantiation of the general paradigm and construct *compact* and *adaptively secure* ABE schemes from k-Lin in asymmetric pairing groups.

4.1 Introduction

A primary goal of research on ABE is designing ABE schemes for expressive classes of policies, usually defined by computation models or complexity classes. A beautiful and fruitful line of works have constructed ABE for many different policy classes. For non-uniform computation, we have ABE for Boolean [GPSW06,KW19] or arithmetic formulae, branching/span programs [LOS+10,OT10,LW11,OT12,LW12,IW14,GV15,Att16, CGKW18a], and circuits [GVW13,BGG+14,Att17]. For uniform computation, we have ABE for deterministic finite automata [Wat12,Att14,Att16,AC17,GWW19,AMY19b], non-deterministic finite automata [AMY19a], and even Turing machines [AM18,AS17b]. These constructions, however, achieve different trade-offs among security, efficiency, and underlying computational assumptions. It is rare to have a construction that simultaneously achieves the following natural desiderata on all fronts:

- *security* — (full) adaptive security (as opposed to selective or semi-adaptive security);

[4] Huijia Lin, Ji Luo: *Compact and Adaptively Secure ABE from k-Lin: Beyond* NC^1 *and Towards* NL, DOI 10.1007/978-3-030-45727-3_9.

- *efficiency* — compact secret key and ciphertext, whose sizes grow linearly with the description size of the policy and the length of the attribute, respectively;
- *assumptions* — relying on standard and simple assumptions, such as LWE and k-Lin or SXDH in pairing groups (in particular, it is preferable to avoid the use of strong primitives such as indistinguishability obfuscation, and instance-dependent assumptions such as q-type assumptions, whose strength can be weakened by adversarially chosen parameters).

All previous constructions of ABE fail to achieve at least one of the desirable properties, except for the recent construction [KW19] of ABE for *Boolean formulae* from the k-Lin assumption. We ask:

Can we construct ABE schemes with all the desirable properties above for more expressive classes of policies than Boolean formulae?

When it comes to uniform computation, the state of affairs is even less satisfactory. All constructions of ABE for general Turing machines are based on strong primitives such as indistinguishability obfuscation and multilinear maps. Without these powerful tools, existing schemes can only handle the weak computation model of finite automata. We ask:

Can we construct ABE schemes based on standard assumptions for more expressive uniform computations than finite automata?

Our Results. Via a unified framework, we construct *compact and adaptively secure* ABE schemes based on the k-Lin assumption in asymmetric prime-order pairing groups for the following classes of policies.

Arithmetic Branching Programs. ABP capture many functions of interest, including arithmetic computations like sparse polynomials, mean, and variance, as well as combinatorial computations like string-matching, finite automata, and decision trees. It is also known that Boolean/arithmetic formulae and Boolean branching programs can all be converted into ABP with polynomial blow-up in description size. Therefore, ABP can be viewed as a more powerful computational model than them.

Previous ABE schemes for ABP only provide selective security [GVW13,IW14] or do not have compact ciphertexts [CGKW18a].[5] In addition to achieving both adaptive security and compactness, our scheme is the first one that handles ABP directly without converting it to circuits or arithmetic span programs, which leads to an efficiency improvement in the size of the secret keys from up to quadratic to linear in the size of the ABP.[6]

(Non-)Deterministic Logspace Turing Machines (L and NL). Here, a secret key is associated with a Turing machine M, and the attribute in a ciphertext specifies an input $\mathbf{x}$, a polynomial time bound T, and a logarithmic space bound S. Decryption succeeds if and only if M accepts $\mathbf{x}$ within time T and space S. Our scheme is *unbounded* in the sense that the public parameters do not restrict the sizes of the Turing machine M and input $\mathbf{x}$, nor the time/space bounds T, S. Furthermore, it enjoys the advantage of ABE for uniform computation that a secret key for M can decrypt ciphertexts with arbitrarily long inputs and arbitrary time/space bounds. This stands in contrast with ABE for non-uniform computation (like ABP), where a program or circuit f takes inputs of a specific length n, and a secret key for f decrypts only ciphertext of length-n inputs. Achieving this feature is precisely the challenge in constructing ABE for uniform models of computation.

Our scheme is the first ABE for large classes of Turing machine computation, captured by the complexity classes L and NL, *without using the heavy machineries* of multilinear maps, extractable witness encryption, or indistinguishability obfuscation as in previous works [GKP+13a,AS16,AM18,KNTY19]. In addition, our scheme is adaptively secure and *half-compact*. The secret keys are compact, of size $O(|M|)$ linear in the description size of M, while the ciphertext size depends linearly in $|\mathbf{x}|TS2^S$ (both ignoring fixed polynomial factors in the security parameter). Note that the same secret key decrypt ciphertexts with different parameters $|\mathbf{x}|, T, S$.

---

[5] More precisely, they construct ABE for read-once branching programs. For general branching programs, one can duplicate each attribute component for the number of times it is accessed [KLMM19]. As such, the ciphertext size grows linearly with the size of the branching program.

[6] An ABP is specified by a directed graph, with edges weighted by affine functions of the input. The size of an ABP is measured by the number of vertices (instead of edges) in the graph.

Removing the dependency on 2^S or T is an interesting open problem that requires technical breakthrough. In particular, removing the dependency on 2^S would give an ABE for polynomial-time Turing machine computation from pairing, a long sought-after goal that has remained elusive for more than a decade. Removing the dependency of encryption time on T even only in the 1-key 1-ciphertext setting implies a succinct message-hiding encoding [KLW15],[7] which is only known from strong primitives like indistinguishability obfuscation or functional encryption [KLW15,CHJV15,BGL+15, KNTY19]. Removing the dependency of ciphertext size on T might be an easier task, but would need new techniques different from ours.

FINITE AUTOMATA. As a special case of ABE for L and NL, we obtain ABE for deterministic finite automata (DFA) and non-deterministic finite automata (NFA).[8] This simply follows from the fact that DFA and NFA can be represented as simple deterministic and non-deterministic Turing machines with space complexity 1 and time complexity N that always move the input tape pointer to the right and never use the working tape.

Previous schemes for DFA based on pairing either achieve only selective security [Wat12,GWW19,AMY19b] or rely on q-type assumptions [Att14,Att16,AC17]. The only direct construction of ABE for NFA [AMY19a] based on LWE, however, is symmetric-key and only selectively secure. We settle the open problem of constructing adaptively secure ABE for DFA from static assumptions [GWW19] and that of constructing ABE for NFA that is public-key, adaptively secure, or based on assumptions other than LWE [AMY19a].

---

[7] Message-hiding encodings [KLW15] are a weaker variant of randomized encodings that allow encoding a public computation f, x with a secret message m such that the encoding reveals m if and only if $f(x) = 1$. Such encodings are *succinct* if the time to encode is much smaller than the running time of the computation. A pair of ABE secret key for predicate f and ciphertext for attribute x and message m is a message-hiding encoding.

[8] DFA and NFA both characterize regular languages, yet a DFA recognizing a language could have exponentially more states than an NFA recognizing the same language. In this dissertation, by ABE for DFA/NFA, we mean ABE schemes that run in time polynomial in the description size of the finite automata.

New Techniques for Constructing Adaptively Secure ABE. Constructing adaptively secure ABE is a challenging task. Roughly speaking, previous constructions proceed in two steps. First, a secure core secret-key ABE component for a single ciphertext and a single secret key — termed 1-ABE — is designed. Then, dual system encryption framework, originally proposed in [Wat09] and refined in [Att14,Wee14,CGW15,Att16, AC17], provides guidance on how to lift 1-ABE to the public-key and multi-secret-key setting. The main technical challenge lies in the first step: Adaptively secure schemes prior to that of Kowalczyk and Wee [KW19] either impose a read-once restriction on the attribute[9] [LOS+10,Wee14] or rely on q-type assumptions [LW12,BSW07,Att14, AC17]. Kowalczyk and Wee [KW19] elegantly applied the "partial selectivization" framework [ACC+16,JKK+17] for achieving adaptive security in general to constructing 1-ABE. In particular, they used a variant of the secret sharing scheme for Boolean formulae in [JKK+17], whose selective simulation security can be proven via a sequence of hybrids, each only requiring *partial* information of the input to be chosen selectively. Then, to show adaptive security, the reduction can *guess* this partial information while incurring only a polynomial security loss.

However, secret sharing schemes as needed in [KW19] are only known for Boolean formulae. When dealing with computation over arithmetic domains of potentially exponential size, we have the additional challenge that it is hard to guess even a single component of the input, except with exponentially small probability, rendering the partial selectivization framework ineffective. When dealing with uniform computation, we further encounter the challenge that neither the secret key nor the ciphertext is as large as the secret sharing, making it impossible to directly use information-theoretically secure secret sharing schemes. We develop new techniques to overcome these challenges.

1. We present a generic framework for constructing adaptively secure 1-ABE from *i)* an information theoretic primitive called *arithmetic key garbling*, and *ii)* a computational primitive called *function-hiding inner-product functional*

[9] As mentioned in Footnote 5, read-once restriction can be circumvented by duplicating attribute components at the cost of losing ciphertext compactness.

encryption (IPFE) [BJK15,TAO16,Lin17,KLM+18]. Our arithmetic key garbling schemes are partial garbling schemes [IW14] with special structures, which act as the counterpart of secret sharing schemes for arithmetic computation. Our framework is modular: It decomposes the task of constructing 1-ABE to *first* designing an arithmetic key garbling scheme for the computation class of interest, and *second* applying a generic transformation depending solely on structural properties of the garbling and agnostic of the underlying computation. In particular, the security proof of the transformation does not attempt to trace the computation, unlike [KW19,GWW19].

2. We formulate structural properties of arithmetic key garbling schemes — called piecewise security — sufficient for achieving adaptive security. The properties are natural and satisfied by the garbling scheme for ABP in [IW14]. For logspace Turing machine computation, we present a simple arithmetic key garbling scheme for L and NL, inspired by the garbling schemes in [AIK11,BGG+14].

3. We present a new method of lifting 1-ABE to full-fledged ABE using function-hiding IPFE. Our method can be cast into the dual system encryption framework, but is natural on its own, without seeing through the lens of dual system encryption. IPFE provides a conceptually simple abstraction, which allows moving information between ABE keys and ciphertexts easily, and hides away lower-level details on how to guarantee security. This feature makes it a convenient tool in many other parts of the security proof as well.

4. To overcome the unique challenge related to ABE for uniform computation, we further enhance our generic method to be able to use partial garbling generated with *pseudorandomness* so that the total size of the secret keys and ciphertexts can be smaller than the garbling.

4.2 Technical Overview

We now give an overview of our technique, starting with introducing the two key tools, arithmetic key garbling schemes and IPFE.

Arithmetic Key Garbling Scheme. We use a refinement of the notion of partial garbling schemes [IW14] (which in turn is based on the notion of garbling and randomized encoding [Yao86,IK02,AIK04]). An *arithmetic key garbling scheme* (AKGS) is an information-theoretic partial garbling scheme for computing $(\alpha f(\mathbf{x}) + \beta)$ that hides the *secrets* $\alpha, \beta \in \mathbb{Z}_p$, but not $f, \mathbf{x}$.

- A garbling procedure $(\mathbf{L}_1, \ldots, \mathbf{L}_m) \leftarrow \mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ turns f and two secrets α, β (using randomness $\mathbf{r}$) into m affine *label functions* $L_1, \ldots, L_m$, described by their coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ over $\mathbb{Z}_p$. The label functions specify how to encode an input $\mathbf{x}$ to produce the *labels* for computing $f(\mathbf{x})$ with secrets α, β:

$$\widehat{f(\mathbf{x})}_{\alpha,\beta} = (\ell_1, \ldots, \ell_m), \quad \text{where } \ell_j = L_j(\mathbf{x}) = \langle \mathbf{L}_j, (1, \mathbf{x}^\intercal) \rangle \text{ over } \mathbb{Z}_p. \tag{4.1}$$

- A *linear* evaluation procedure $\gamma \leftarrow \mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$. It recovers the sum $\gamma = \alpha f(\mathbf{x}) + \beta$ weighted by the function value $f(\mathbf{x})$.

AKGS is a *partial* garbling as it only hides information of the secrets α and β beyond the weighted sum $(\alpha f(\mathbf{x}) + \beta)$, and does not hide f or $\mathbf{x}$, captured by a simulation procedure $(\ell'_1, \ldots, \ell'_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$ that produces the same distribution as the honest labels.

Ishai and Wee [IW14] proposed a partial garbling scheme for ABP, which directly implies an AKGS for ABP. It is also easy to observe that the (fully secure) garbling scheme for arithmetic formulae in [AIK11] can be weakened [BGG+14] to an AKGS. Later, we will introduce additional structural and security properties of AKGS needed for our 1-ABE construction. These properties are natural and satisfied by both schemes [IW14,AIK11].

Inner-Product Functional Encryption. A *function-hiding* (secret-key) inner-product functional encryption (IPFE)[10] supports generating many secret keys $\mathsf{isk}(\mathbf{v}_j)$ and ciphertexts $\mathsf{ict}(\mathbf{u}_i)$ associated with vectors $\mathbf{v}_j$ and $\mathbf{u}_i$ such that decryption yields all the inner products $\{\langle \mathbf{u}_i, \mathbf{v}_j \rangle\}_{i,j}$ (modulo p) and nothing else. In this chapter, we need an

[10] Some works use "inner-product encryption" (IPE) to refer to IPFE [BJK15,DDM16,LV16,Lin17] and some others [KSW13,SSW09,OT09,OT10,OT12,CGW18] use it for inner-product *predicate* encryption.

adaptively secure IPFE, whose security holds even against adversaries choosing all the vectors adaptively. Such an IPFE scheme can be constructed [Wee17,LV16] based on the k-Lin assumption in pairing groups. The known scheme also has nice structural properties that will be instrumental to our construction of ABE.

- $\mathsf{isk}(\mathbf{v}) \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}]\!]_2)$ only uses the vector $\mathbf{v}$ in the exponent of $\mathbb{G}_2$. The size of the secret key $\mathsf{isk}(\mathbf{v})$ grows linearly with the dimension of $\mathbf{v}$.
- $\mathsf{ict}(\mathbf{u}) \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{imsk}, [\![\mathbf{u}]\!]_1)$ only uses the plaintext vector $\mathbf{u}$ in the exponent of $\mathbb{G}_1$. The size of the ciphertext $\mathsf{ict}(\mathbf{u})$ grows linearly with the dimension of $\mathbf{u}$.
- $\mathsf{IPFE.Dec}(\mathsf{isk}(\mathbf{v}), \mathsf{ict}(\mathbf{u}))$ computes the inner product $[\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_T$ in the exponent of the target group $\mathbb{G}_T$.

4.2.1 1-ABE from Arithmetic Key Garbling and IPFE Schemes

1-ABE is the technical heart of our ABE construction. It works in the setting where a single ciphertext $\mathsf{ct}_{\mathbf{x}}$ for an input $\mathbf{x}$ and a single secret key $\mathsf{sk}(f', \mu)$ for a policy $f' = f_{\neq 0}$ and a secret μ are published. Decryption reveals μ if $f(\mathbf{x}) \neq 0$; otherwise, μ is hidden.[11]

1-ABE. To hide μ conditioned on $f(\mathbf{x}) = 0$, our key idea is using IPFE to compute an AKGS garbling $\widehat{f(\mathbf{x})}_{\mu,0}$ of $f(\mathbf{x})$ with secrets $\alpha = \mu$ and $\beta = 0$. The security of AKGS guarantees that only $\mu f(\mathbf{x})$ is revealed, which information-theoretically hides μ when $f(\mathbf{x}) = 0$.

The reason that it is possible to use IPFE to compute the garbling is attributed to the *affine input-encoding* property of AKGS — the labels $\ell_1, \ldots, \ell_m$ are the output of affine functions $L_1, \ldots, L_j$ of $\mathbf{x}$ as described in Equation (4.1). Since f, α, β are known at key generation time, the ABE key can be a collection of IPFE secret keys, each encoding the coefficient vector $\mathbf{L}_j$ of one label function L_j. On the other hand, the ABE ciphertext can be an IPFE ciphertext encrypting $(1, \mathbf{x}^\intercal)$. When put together for

[11] We can also handle policies of the form $f_{=0}$ so that μ is revealed if and only if $f(\mathbf{x}) = 0$. For simplicity, we focus on one case in this overview.

decryption, they reveal exactly the labels $L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})$, as described below on the left.

		Honest Algorithms	Hybrid for Selective Security
$\mathsf{ct}_{\mathbf{x}}$:		$\mathsf{ict}\ (\ (1, \mathbf{x}^\intercal)\ ,\ \mathbf{0}\ )$	$\mathsf{ict}\ (\ (1, \mathbf{x}^\intercal)\ ,\ 1\ ,\ \mathbf{0}\ )$
$\mathsf{sk}(f_{\neq 0}, \mu)$:	$(j \in [m])$	$\mathsf{isk}_j(\quad \mathbf{L}_j \quad ,\ \mathbf{0}\ )$	$\mathsf{isk}_j(\quad \mathbf{0} \quad ,\ \ell_j\ ,\ \mathbf{0}\ )$

We note that the positions or slots at the right end of the vectors encoded in isk and ict are set to zero by the honest algorithms — $\mathbf{0}$ is a vector (of unspecified length) of zeros. These slots provide programming space in the security proof.

It is extremely simple to prove selective (or semi-adaptive) security, where the input $\mathbf{x}$ is chosen before seeing the sk. By the function-hiding property of IPFE, it is indistinguishable to switch the secret keys and the ciphertext to encode any vectors that preserve the inner products. This allows us to hardwire honestly generated labels $\widehat{f(\mathbf{x})}_{\mu,0} = \{\ell_j \leftarrow \langle \mathbf{L}_j, (1, \mathbf{x}^\intercal)\rangle\}_{j\in[m]}$ in the secret keys as described above on the right. The simulation security of AKGS then implies that only $\mu f(\mathbf{x})$ is revealed, i.e., nothing about μ is revealed.

Achieving Adaptive Security. When it comes to adaptive security, where the input $\mathbf{x}$ is chosen *after* seeing the sk, we can no longer hardwire the honest labels $\widehat{f(\mathbf{x})}_{\mu,0}$ in the secret key, as $\mathbf{x}$ is undefined when sk is generated, and hence cannot invoke the simulation security of AKGS. Our second key idea is relying on a stronger security property of AKGS, named *piecewise security*, to hardwire simulated labels into the secret key in a piecemeal fashion.

Piecewise Security of AKGS. An AKGS is piecewise secure if it satisfies the following two properties: *i)* reverse sampleability — there is an efficient procedure RevSamp that can *perfectly* reversely sample the first label ℓ_1 given the output $(\alpha f(\mathbf{x}) + \beta)$ and all the other labels $\ell_2, \ldots, \ell_m$; and *ii)* marginal randomness[12] — each non-first ℓ_j (i.e., $j > 1$) is uniformly distributed over $\mathbb{Z}_p$ even given all the subsequent *label functions*

[12]This is an unfortunate misnomer since [LL20a,LL20b]. The correct wording is "conditionally random". We keep the original version for consistency.

$\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$. More formally,

$$\begin{aligned}
\left(\ell_1 \leftarrow \langle \mathbf{L}_1, (1, \mathbf{x}^\top)\rangle, \quad \mathbf{L}_2, \ldots, \mathbf{L}_m\right) &\equiv \left(\ell_1' \xleftarrow{\$} \mathsf{RevSamp}(\cdots), \quad \mathbf{L}_2, \ldots, \mathbf{L}_m\right), \qquad (4.2)\\
\left(\ell_j \leftarrow \langle \mathbf{L}_j, (1, \mathbf{x}^\top)\rangle, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\right) &\equiv \left(\ell_j' \xleftarrow{\$} \mathbb{Z}_p \qquad\quad , \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\right).
\end{aligned}$$

In Equation (4.2), $\ell_1' \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$. These properties are natural and satisfied by the existing AKGS [IW14,AIK11] for ABP and arithmetic formulae.

ADAPTIVE SECURITY VIA PIECEWISE SECURITY. We are now ready to prove the adaptive security of our 1-ABE. The proof strategy is to first hardwire ℓ_1 in the ciphertext and sample it reversely as $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, 0, \ell_2, \ldots, \ell_m)$, where $\ell_j = \langle \mathbf{L}_j, (1, \mathbf{x}^\top)\rangle$ for $j > 1$ and $\mu f(\mathbf{x}) = 0$ by the constraint, as described in hybrid $k = 1$ below. The indistinguishability follows immediately from the function-hiding property of IPFE and the reverse sampleability of AKGS. Then, we gradually replace each remaining label function $\mathbf{L}_j$ for $j > 1$ by a randomly sampled label $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ in the secret key, as described in hybrids $1 \le k \le m + 1$. It is easy to observe that in the final hybrid $k = m + 1$, where all labels $\ell_2, \ldots, \ell_m$ are random and ℓ_1 is reversely sampled *without* μ, the secret μ is information-theoretically hidden.

	HYBRID $1 \le k \le m+1$	HYBRID $k : 1$ OR $k : 2$
$\mathsf{sk}(f_{\neq 0}, \mu)$:	$\mathsf{isk}_1($ $\mathbf{0}$, 1 , 0 , 0 $)$	
$(1 < j < k)$	$\mathsf{isk}_j($ $\mathbf{0}$, 0 , ℓ_j , 0 $)$	
	$\mathsf{isk}_k($ $\mathbf{L}_k$, 0 , 0 , 0 $)$	$\mathsf{isk}_k($ $\mathbf{0}$, 0 , 0 , 1 $)$
$(j > k)$	$\mathsf{isk}_j($ $\mathbf{L}_j$, 0 , 0 , 0 $)$	
$\mathsf{ct}_{\mathbf{x}}$:	$\mathsf{ict}\ (\ (1, \mathbf{x}^\top)\ ,\ \ell_1\ ,\ 1\ ,\ 0\ )$	$\mathsf{ict}\ (\ (1, \mathbf{x}^\top)\ ,\ \ell_1\ ,\ 1\ ,\ \ell_k\ )$
$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$ AND $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ (FOR $1 < j < k$)		$\ell_k \leftarrow \langle \mathbf{L}_k, (1, \mathbf{x}^\top)\rangle$ OR $\ell_k \xleftarrow{\$} \mathbb{Z}_p$

To move from hybrid k to $(k + 1)$, we want to switch the k^{th} IPFE secret key isk_k from encoding the label function $\mathbf{L}_k$ to a simulated label $\ell_k \xleftarrow{\$} \mathbb{Z}_p$. This is possible via two moves. First, by the function-hiding property of IPFE, we can hardwire the honest $\ell_k = \langle \mathbf{L}_k, (1, \mathbf{x}^\top)\rangle$ in the ciphertext as in hybrid $k : 1$ (at encryption time, $\mathbf{x}$ is known). Then, by the marginal randomness property of AKGS, we switch to sample ℓ_k as

random in hybrid $k:2$. Lastly, hybrid $k:2$ is indistinguishable to hybrid $(k+1)$ again by the function-hiding property of IPFE.

1-ABE for ABP. Plugging in the AKGS for ABP due to Ishai and Wee [IW14], we immediately obtain 1-ABE for ABP based on k-Lin. The size of the garbling grows linearly with the number of vertices $|V|$ in the graph describing the ABP, i.e., $m = \mathrm{O}(|V|)$. Combined with the fact that the IPFE has linear-size secret keys and ciphertexts, our 1-ABE scheme for ABP has secret keys of size $\mathrm{O}(m|\mathbf{x}|) = \mathrm{O}(|V| \cdot |\mathbf{x}|)$ and ciphertexts of size $\mathrm{O}(|\mathbf{x}|)$. This gives an efficiency improvement over previous 1-ABE or ABE schemes for ABP [IW14,CGKW18a], where the secret key size grows linearly with the number of edges $|E|$ in the ABP graph, due to that their schemes first convert ABP into an arithmetic span program, which incurs the efficiency loss.

Discussion. Our method for constructing 1-ABE is generic and modular. It has the advantage that the proof of adaptive security is agnostic of the computation being performed and merely carries out the simulation of AKGS in a mechanic way. Indeed, if we plug in an AKGS for arithmetic formulae or any other classes of non-uniform computation, the proof remains the same. (However, see Section 4.7 for further discussion on the applicability of this method.) Our 1-ABE for logspace Turing machines also follows the same blueprint, but needs additional ideas. Furthermore, note that our method departs from the partial selectivization technique used in [KW19], which is not applicable to arithmetic computation as the security reduction cannot afford to guess even one component of the input $\mathbf{x}$.[13] The problem is circumvented by using IPFE to hardwire the labels (i.e., ℓ_1, ℓ_k) that depend on $\mathbf{x}$ in the ciphertext.

4.2.2 Full-Fledged ABE via Slotted IPFE

From 1-ABE in the 1-key 1-ciphertext setting to full-fledged ABE, we need to support publishing multiple keys and make encryption public-key. It turns out that the security of our 1-ABE scheme directly extends to the many-key 1-ciphertext (still secret-key)

[13] Guessing even a single element of $\mathbb{Z}_p$ would incur a loss of $\Theta(p)$ in the advantage, but discrete logarithm, a weakest assumption in groups, can be trivially solved with advantage $\Theta(1/p)$.

setting via a simple hybrid argument. Consider the scenario where one ciphertext ct and multiple keys $\{\mathsf{sk}_q(f_{q,\neq 0}, \mu_q)\}_{q\in[Q]}$ that are unauthorized to decrypt the ciphertext are published. Combining the above security proof for 1-ABE with a hybrid argument, we can gradually switch each secret key sk_q from encoding honest label functions for μ_q to ones for an independent secret $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. Therefore, all the secrets $\{\mu_q\}_{q\in[Q]}$ are hidden.

The security of our 1-ABE breaks down as soon as two ciphertexts are released. Consider publishing just a single secret key $\mathsf{sk}(f_{\neq 0}, \mu)$ and two ciphertexts $\mathsf{ct}_{\mathbf{x}_1}, \mathsf{ct}_{\mathbf{x}_2}$. Since the label functions $L_1, \ldots, L_m$ are encoded in sk, decryption computes two AKGS garblings $\widehat{f(\mathbf{x}_1)}_{\mu,0}$ and $\widehat{f(\mathbf{x}_2)}_{\mu,0}$ from the *same* set of label functions. However, AKGS security does not apply when the label functions are reused.

We want IPFE decryption to compute two garblings $\widehat{f(\mathbf{x}_1)}_{\mu,0} = (L_1(\mathbf{x}_1), \ldots, L_m(\mathbf{x}_1))$ and $\widehat{f(\mathbf{x}_2)}_{\mu,0} = (L'_1(\mathbf{x}_2), \ldots, L'_m(\mathbf{x}_2))$ using *independent* label functions. This can be achieved in a *computational* fashion relying on the fact that the IPFE scheme encodes the vectors and the decryption results in the exponent of pairing groups. We can use computational assumptions such as SXDH or k-Lin, combined with the function-hiding property of IPFE, to argue that the produced garblings are computationally independent. We modify the 1-ABE scheme as follows.

- If SXDH holds over the pairing, we encode in the ciphertext $(1, \mathbf{x}^\intercal)$ multiplied by a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$. As such, decryption computes $(sL_1(\mathbf{x}), \ldots, sL_m(\mathbf{x}))$ *in the exponent*. We argue that the label functions $sL_1, \ldots, sL_m$ are computationally random *in the exponent*. By the function-hiding property of IPFE, it is indistinguishable to multiply s not with the plaintext vector, but with the coefficient vectors in the secret key, as depicted below on the right. By DDH (over $\mathbb{G}_2$) and the linearity of Garble (i.e., the coefficients $\mathbf{L}_j$ depend linearly on the secrets α, β and the randomness $\mathbf{r}$ used by Garble), the $s\mathbf{L}_j$'s are coefficients of pseudorandom label functions.

	Algorithms based on SXDH		Hybrid	
				$\approx \mathbf{L}'_j$ (fresh)
$\mathsf{sk}(f_{\neq 0}, \mu)$:	$(j \in [m])$	$\mathsf{isk}_j(\ \mathbf{L}_j\ , \mathbf{0})$	$\mathsf{isk}_j(\mathbf{L}_j,$	$s\mathbf{L}_j\ , \mathbf{0})$
$\mathsf{ct}_{\mathbf{x}}$:		$\mathsf{ict}\ (s(1, \mathbf{x}^\intercal), \mathbf{0})$	$\mathsf{ict}\ (\mathbf{0}\ , (1, \mathbf{x}^\intercal), \mathbf{0})$	

- If k-Lin holds over the pairing, we encode in the secret key k independent sets of label functions $L_1^t, \ldots, L_m^t$ for $t \in [k]$, and in the ciphertexts k copies of $(1, \mathbf{x}^\intercal)$ multiplied by independent random scalars $\mathbf{s}[t]$ for $t \in [k]$. This way, decryption computes, in the exponent, a random linear combination of the garblings

$$\left(\sum_{t\in[k]} \mathbf{s}[t] L_1^t(\mathbf{x}), \ \ldots, \ \sum_{t\in[k]} \mathbf{s}[t] L_m^t(\mathbf{x}) \right),$$

  which corresponds to pseudorandom label functions in the exponent.

ALGORITHMS BASED ON k-LIN

$$\begin{array}{rlll} \mathsf{sk}(f_{\neq 0}, \mu): & (j \in [m]) & \mathsf{isk}_j(& \mathbf{L}_j^1 \ , \ \ldots \ , \ \mathbf{L}_j^k \ , \ \mathbf{0} \ , \ \mathbf{0} \ ) \\ \mathsf{ct}_{\mathbf{x}}: & & \mathsf{ict}\ (& \mathbf{s}[1](1, \mathbf{x}^\intercal) \ , \ \ldots \ , \ \mathbf{s}[k](1, \mathbf{x}^\intercal) \ , \ \mathbf{0} \ , \ \mathbf{0} \ ) \end{array}$$

HYBRID

$$\begin{array}{rlll} \mathsf{sk}(f_{\neq 0}, \mu): & (j \in [m]) & \mathsf{isk}_j(& \mathbf{L}_j^1 \ , \ \ldots \ , \ \mathbf{L}_j^k \ , \ \sum_{t\in[k]} \mathbf{s}[t] \mathbf{L}_j^t \ , \ \mathbf{0} \ ) \\ \mathsf{ct}_{\mathbf{x}}: & & \mathsf{ict}\ (& \mathbf{0} \ , \ \ldots \ , \ \mathbf{0} \ , \ (1, \mathbf{x}^\intercal) \ , \ \mathbf{0} \ ) \end{array}$$

$\approx \mathbf{L}'_j$ (fresh)

The above modification yields a secret-key ABE secure in the many-ciphertext many-key setting. The final hurdle is how to make the scheme public-key, which we resolve using slotted IPFE.

Slotted IPFE. Proposed in [LV16], *slotted* IPFE is a hybrid between a secret-key function-hiding IPFE and a public-key IPFE. Here, a vector $\mathbf{u} \in \mathbb{Z}_p^n$ is divided into two parts $(\mathbf{u}_{\text{pub}}, \mathbf{u}_{\text{priv}})$ with $\mathbf{u}_{\text{pub}} \in \mathbb{Z}_p^{n_{\text{pub}}}$ in the public slot and $\mathbf{u}_{\text{priv}} \in \mathbb{Z}_p^{n_{\text{priv}}}$ in the private slot $(n_{\text{pub}} + n_{\text{priv}} = n)$. Like a usual secret-key IPFE, the encryption algorithm $\mathsf{IPFE.Enc}$ using the master secret key imsk can encrypt to both the public and private slots, i.e., encrypting any vector $\mathbf{u}$. In addition, there is an $\mathsf{IPFE.SlotEnc}$ algorithm that uses the master public key impk, but can only encrypt to the public slot, i.e., encrypting vectors such that $\mathbf{u}_{\text{priv}} = \mathbf{0}$. Since anyone can encrypt to the public slot, it is impossible to hide the public slot part $\mathbf{v}_{\text{pub}}$ of a secret-key vector $\mathbf{v}$. As a result, slotted IPFE guarantees function-hiding only with respect to the private slot, and the weaker (ciphertext) indistinguishability with respect to the public slot. Based on the construction of slotted IPFE in [Lin17], we obtain adaptively secure slotted IPFE based on k-Lin.

The aforementioned secret-key ABE scheme can be easily turned into a public-key one with slotted IPFE. The ABE encryption algorithm simply uses $\mathsf{IPFE.SlotEnc}$ and impk to encrypt to the public slots. In the security proof, we move vectors encrypted in the public slot of the challenge ciphertext to the private slot, where function-hiding holds and the same security arguments outlined above can be carried out.

Discussion. Our method can be viewed as using IPFE to implement dual system encryption [Wat09]. We believe that IPFE provides a valuable abstraction, making it conceptually simpler to design strategies for moving information between the secret key and the ciphertext, as done in the proof of 1-ABE, and for generating independent randomness, as done in the proof of full ABE. The benefit of this abstraction is even more prominent when it comes to ABE for logspace Turing machines.

4.2.3 1-ABE for Logspace Turing Machines

We now present the ideas for constructing 1-ABE for L, then its extension to NL and how to handle DFA and NFA as special cases for better efficiency. Moving to full-fledged ABE follows the same ideas in the previous subsection, though slightly more complicated, which we omit in this overview.

1-ABE for L enables generating a single secret key $\mathsf{sk}(M, \mu)$ for a logspace Turing machine M and a secret μ, and a single ciphertext $\mathsf{ct}_{\mathbf{x},T,S}$ specifying an input $\mathbf{x}$ of length N, a polynomial time bound $T = \mathrm{poly}(N)$, and a logarithmic space bound $S = \mathrm{O}(\log N)$ such that decryption reveals $\mu M|_{N,T,S}(\mathbf{x})$, where $M|_{N,T,S}(\mathbf{x})$ represents the computation of running $M(\mathbf{x})$ for T steps with a working tape of size S, which outputs 1 if and only if the computation lands in an accepting state after T steps and has *never* exceeded the space bound S. A key feature of ABE for uniform computation is that a secret key $\mathsf{sk}(M, \mu)$ can decrypt ciphertexts with inputs of unbounded lengths and unbounded time / (logarithmic) space bounds. (In contrast, for non-uniform computation, the secret key decides the input length and the time/space bounds.) Our 1-ABE for L follows the same blueprint of combining AKGS with IPFE, but uses new ideas in order to implement the unique feature of ABE for uniform computation.

Notations for Turing Machines. We start by introducing notations for logspace Turing machines (TM) over the binary alphabet. A Turing machine $M = (Q, q_{\text{acc}}, \delta)$ has Q states, with the initial state being 1, the accepting state[14] being $q_{\text{acc}} \in [Q]$, and the transition function being δ. The computation of $M|_{N,T,S}(\mathbf{x})$ goes through a sequence of $(T+1)$ *configurations* $(\mathbf{x}, (i, j, \mathbf{W}, q))$, where $i \in [N]$ is the input tape pointer, $j \in [S]$ is the working tape pointer, $\mathbf{W} \in \{0,1\}^S$ is the content of the working tape, and $q \in [Q]$ is the current state. The initial *internal* configuration is thus $(i, j, \mathbf{W}, q) = (1, 1, \mathbf{0}_S, 1)$, and the transition from one internal configuration $(i, j, \mathbf{W}, q)$ to the next $(i', j', \mathbf{W}', q')$ is governed by the transition function δ and the input $\mathbf{x}$. Namely, if $\delta(q, \mathbf{x}[i], \mathbf{W}[j]) = (q', w', \Delta i, \Delta j)$, then

$$(i, j, \mathbf{W}, q) \rightsquigarrow (i', j', \mathbf{W}', q') = (i + \Delta i,\ j + \Delta j,\ \mathsf{overwrite}(\mathbf{W}, j, w'),\ q').$$

In other words, the transition function δ, given the current state q and the bits $\mathbf{x}[i]$, $\mathbf{W}[j]$ on the input and working tapes under scan, outputs the next state q', the new bit $w' \in \{0,1\}$ to be written to the working tape, and the directions $\Delta i, \Delta j \in \{0, \pm 1\}$ to move the input and working tape pointers. The next internal configuration is then derived by updating the current configuration accordingly, where $\mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w')$ is a vector obtained by overwriting the j^{th} cell of $\mathbf{W}$ with w' and keeping the other cells unchanged.

AKGS for Logspace Turing Machines. To obtain an AKGS for L, we represent the TM computation algebraically as a sequence of matrix multiplications over $\mathbb{Z}_p$, for which we design an AKGS. To do so, we represent each internal configuration as a standard basis vector $\boldsymbol{\iota}_{(i,j,\mathbf{W},q)}$ of dimension $NS2^S Q$ with a single 1 at position $(i, j, \mathbf{W}, q)$. We want to find a *transition matrix* $\mathbf{M}(\mathbf{x})$, dependent on $\delta, \mathbf{x}$, such that moving to the next state $\boldsymbol{\iota}_{(i',j',\mathbf{W}',q')}$ simply involves (right) multiplying $\mathbf{M}(\mathbf{x})$, i.e., $\boldsymbol{\iota}^{\mathsf{T}}_{(i',j',\mathbf{W}',q')} = \boldsymbol{\iota}^{\mathsf{T}}_{(i,j,\mathbf{W},q)}\mathbf{M}(\mathbf{x})$. It is easy to verify that the correct transition matrix is

$$\begin{aligned}\mathbf{M}(\mathbf{x})[(i, j, \mathbf{W}, q), (i', j', \mathbf{W}', q')] = \mathsf{CanTransit}[(i, j, \mathbf{W}), (i', j', \mathbf{W}')] \\ \cdot \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q, q'],\end{aligned} \tag{4.3}$$

[14] For simplicity, in this overview, we assume there is exactly one accepting state.

$$\mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] = \begin{cases} 1, & \text{if } \mathbf{W}'[\neq j] = \mathbf{W}[\neq j] \text{ and } i'-i, j'-j \in \{0,\pm 1\}; \\ 0, & \text{otherwise;} \end{cases}$$

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = \begin{cases} 1, & \text{if } \delta(q,x,w') = (q',w',\Delta i,\Delta j); \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Here, $\mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')]$ indicates whether it is possible, regardless of δ, to move from an internal configuration with $(i,j,\mathbf{W})$ to one with $(i',j',\mathbf{W}')$. When possible, $\mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],\Delta i,\Delta j}[q,q']$ indicates whether δ permits moving from state q with bits $x = \mathbf{x}[i]$, $w = \mathbf{W}[j]$ under scan to state q' with overwriting bit $w' = \mathbf{W}'[j]$ and moving directions $\Delta i = i' - i$, $\Delta j = j' - j$. As such, the TM computation can be done by right-multiplying the matrix $\mathbf{M}(\mathbf{x})$ for T times with the initial configuration $\boldsymbol{\iota}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)}$, reaching the final configuration $\boldsymbol{\iota}^{\mathsf{T}}_{(i_T,j_T,\mathbf{W}_T,q_T)}$, and testing whether $q_T = q_{\mathrm{acc}}$. More precisely,

$$M|_{N,T,S}(\mathbf{x}) = \boldsymbol{\iota}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \big(\mathbf{M}(\mathbf{x})\big)^T \mathbf{t} \qquad \text{for} \qquad \mathbf{t} = \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{\iota}_{q_{\mathrm{acc}}}.$$

To construct AKGS for L, it boils down to construct AKGS for matrix multiplication. Our construction is inspired by the randomized encoding for arithmetic NC^1 due to [AIK11] and the garbling mechanism for multiplication gates in [BGG+14]. Let's focus on garbling the computation $M|_{N,T,S}(\mathbf{x})$ with secrets $\alpha = \mu$ and $\beta = 0$ (the case needed in our 1-ABE). The garbling algorithm Garble produces the following affine label functions of $\mathbf{x}$:

$$\begin{aligned} \ell_{\mathrm{init}} &= L_{\mathrm{init}}(\mathbf{x}) = \boldsymbol{\iota}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \mathbf{r}_0, \\ (\text{for } t \in [T]) \qquad \boldsymbol{\ell}_t &= (\ell_{t,z})_z = \big(L_{t,z}(\mathbf{x})\big)_z = -\boxed{\mathbf{r}_{t-1}} + \mathbf{M}(\mathbf{x}) \cdot \mathbf{r}_t, \\ \boldsymbol{\ell}_{T+1} &= (\ell_{T+1,z})_z = \big(L_{T+1,z}(\mathbf{x})\big)_z = -\boxed{\mathbf{r}_T} + \mu \cdot \mathbf{t}. \end{aligned}$$

Here, $z = (i,j,\mathbf{W},q)$ runs through all $NS2^S Q$ possible internal configurations and $\mathbf{r}_t \xleftarrow{\$} \mathbb{Z}_p^{[N]\times[S]\times\{0,1\}^S\times[Q]}$ for all $t \in [0,T]$. The evaluation proceeds inductively, starting with $\ell_{\mathrm{init}} = \boldsymbol{\iota}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)} \mathbf{r}_0$, going through $\boldsymbol{\iota}^{\mathsf{T}}_{(i_t,j_t,\mathbf{W}_t,q_t)} \mathbf{r}_t$ for every $t \in [T]$ using the identity below, and completing after T steps by combining $\boldsymbol{\iota}^{\mathsf{T}}_{(i_T,j_T,\mathbf{W}_T,q_T)} \mathbf{r}_T$ with $\boldsymbol{\ell}_{T+1}$ to get

$\boldsymbol{\iota}^{\mathsf{T}}_{(i_T,j_T,\mathbf{W}_T,q_T)} \cdot \mu \cdot \mathbf{t} = \mu M|_{N,T,S}(\mathbf{x})$ as desired:

$$\boldsymbol{\iota}^{\mathsf{T}}_{(i_{t+1},j_{t+1},\mathbf{W}_{t+1},q_{t+1})}\mathbf{r}_{t+1} = \boldsymbol{\iota}^{\mathsf{T}}_{(i_t,j_t,\mathbf{W}_t,q_t)}\mathbf{r}_t + \boldsymbol{\iota}^{\mathsf{T}}_{(i_t,j_t,\mathbf{W}_t,q_t)}\underbrace{(-\mathbf{r}_t + \mathbf{M}(\mathbf{x}) \cdot \mathbf{r}_{t+1})}_{\boldsymbol{\ell}_{t+1}}.$$

The above AKGS is *piecewise secure*. First, ℓ_{init} is reversely sampleable. Since Eval is linear in the labels and ℓ_{init} has coefficient 1, given all but the first label ℓ_{init}, one can reversely *compute* ℓ_{init} — the value uniquely determined by the linear equation[15] imposed by the correctness of Eval. Second, the marginal randomness property holds because every label $\boldsymbol{\ell}_t$ (in fact, group of labels) is random due to the random additive term $\mathbf{r}_{t-1}$ not used in subsequent label functions $L_{t',z}$ for all $t' > t$ and z, nor in the non-constant terms of $L_{t,z}$'s — we call $\mathbf{r}_{t-1}$ the *randomizers* of $\boldsymbol{\ell}_t$ (highlighted in the boxes). Lastly, we observe that the garbling size is $(1 + N(T+1)S2^S Q)$.

1-ABE for L. We now try constructing 1-ABE for L from AKGS for L, following the same blueprint of using IPFE. Yet, applying the exact same method for non-uniform computation fails for multiple reasons. In 1-ABE for non-uniform computation, the ciphertext ct contains a single IPFE ciphertext ict encoding $(1, \mathbf{x}^{\mathsf{T}})$, and the secret key sk contains a set of IPFE secret keys isk_j encoding all the label functions. However, in the uniform setting, the secret key $\mathsf{sk}(M, \mu)$ depends only on the machine M and the secret μ, and is supposed to work with ciphertexts $\mathsf{ct}_{\mathbf{x},T,S}$ of arbitrary (*a priori* unbounded) $N = |\mathbf{x}|$, T, S. Therefore, at key generation time, the size of the AKGS garbling, $1 + N(T+1)S2^S Q$, is *unknown*, let alone generating and encoding all the label functions. Moreover, we want our 1-ABE to be compact, with secret key size $|\mathsf{sk}| = \mathrm{O}(Q)$ linear in the number Q of states and ciphertext size $|\mathsf{ct}| = \mathrm{O}(NTS2^S)$, ignoring $\mathrm{poly}(\lambda)$ factors. The total size of secret key and ciphertext is much smaller than the total number of label functions, i.e., $|\mathsf{sk}| + |\mathsf{ct}| = \mathrm{o}(1 + N(T+1)S2^S Q)$.

To overcome these challenges, our idea is that instead of encoding the label functions in either the secret key or the ciphertext (for which there is not enough space), we let the secret key and the ciphertext jointly generate the label functions.

[15] This means RevSamp is deterministic, and we can reversely compute ℓ_{init} *in the exponent* and when the randomness is not uniform, which is important for our (1-)ABE construction.

For this idea to work, the label functions cannot be generated with true randomness, which cannot be "compressed", and must use pseudorandomness instead. More specifically, our 1-ABE secret key $\mathsf{sk}(M,\mu)$ contains $\Theta(Q)$ IPFE secret keys $\{\mathsf{isk}(\mathbf{v}_q)\}_q$, while the ciphertext $\mathsf{ct}_{\mathbf{x},T,S}$ contains $\Theta(NTS2^S)$ IPFE ciphertexts $\{\mathsf{ict}(\mathbf{u}_{t,i,j,\mathbf{W}})\}_{t,i,j,\mathbf{W}}$ — decryption computes in the exponent $\Theta(NTS2^SQ)$ cross inner products $\langle \mathbf{u}_{t,i,j,\mathbf{W}}, \mathbf{v}_q\rangle$ corresponding to a garbling of $M|_{N,T,S}(\mathbf{x})$ with secret μ. To achieve this, we rely crucially on the special *block structure* of the transition matrix $\mathbf{M}$ (which in turn stems from the local structure of TM computation, where the same transition function is applied in every step). Furthermore, as discussed above, we replace every truly random value $\mathbf{r}_t[(i,j,\mathbf{W},q)]$ by a product $\mathbf{r}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\cdot\mathbf{r}_{\mathrm{f}}[q]$, which can be shown pseudorandom in the exponent based on SXDH.[16]

Block Structure of the Transition Matrix. Let's examine the transition matrix again (see Equations (4.3, 4.4)):

$$\mathbf{M}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] = \mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] \cdot \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q,q'].$$

We see that every block $\mathbf{M}(\mathbf{x})[(i,j,\mathbf{W},\sqcup),(i',j',\mathbf{W}',\sqcup)]$ either is the $Q\times Q$ zero matrix or belongs to a small set $\mathcal{T}$ of constantly many *transition blocks*:

$$\mathcal{T} = \left\{ \mathbf{M}_{x,w,w',\Delta i,\Delta j} \,\middle|\, x,w,w'\in\{0,1\},\ \Delta i,\Delta j\in\{0,\pm1\} \right\}.$$

Moreover, in the $\mathfrak{i}=(i,j,\mathbf{W})^{\text{th}}$ "block row", $\mathbf{M}(\mathbf{x})[(\mathfrak{i},\sqcup),(\sqcup,\sqcup,\sqcup,\sqcup)]$, each transition block $\mathbf{M}_{x,w,w',\Delta i,\Delta j}$ either does not appear at all if $x\neq\mathbf{x}[i]$ or $w'\neq\mathbf{W}[j]$, or appears once as the block $\mathbf{M}(\mathbf{x})[(\mathfrak{i},\sqcup),(\mathfrak{i}',\sqcup)]$, where $\mathfrak{i}'$ is the triplet obtained by updating $\mathfrak{i}$ appropriately according to $(w',\Delta i,\Delta j)$, i.e.,

$$\mathfrak{i}' \stackrel{\text{def}}{=} \mathsf{writemove}(\mathfrak{i},(w',\Delta i,\Delta j)) = (i+\Delta i,\ j+\Delta j,\ \mathsf{overwrite}(\mathbf{W},j,w')),$$
$$\mathbf{M}(\mathbf{x})[(\mathfrak{i},\sqcup),(\mathfrak{i}',\sqcup)] = \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],w',\Delta i,\Delta j}.$$

[16] Our scheme readily extends to be based on k-Lin. However, that makes the scheme more complex to present. We choose to present this scheme using SXDH in this chapter.

Therefore, we can "decompose" every label $\boldsymbol{\ell}_t[(\mathsf{i}, q)]$ as an inner product $\langle \mathbf{u}_{t,\mathsf{i}}, \mathbf{v}_q \rangle$ as

$$\begin{aligned}
\boldsymbol{\ell}_t[(\mathsf{i}, q)] &= -\mathbf{r}_{t-1}[(\mathsf{i}, q)] + \mathbf{M}(\mathbf{x})[(\mathsf{i}, q), (\textvisiblespace, \textvisiblespace, \textvisiblespace, \textvisiblespace)] \cdot \mathbf{r}_t \\
&= -\mathbf{r}_{t-1}[(\mathsf{i}, q)] + \sum_{w', \Delta i, \Delta j} \left(\mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \cdot \mathbf{r}_t[(\mathsf{i}', \textvisiblespace)]\right)[q] \qquad (\mathsf{i}' \text{ as above}) \\
&= -\mathbf{r}_{\mathrm{x}}[(t-1, \mathsf{i})] \cdot \mathbf{r}_{\mathrm{f}}[q] + \sum_{w', \Delta i, \Delta j} \mathbf{r}_{\mathrm{x}}[(t, \mathsf{i}')] \cdot (\mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \cdot \mathbf{r}_{\mathrm{f}})[q] \\
&= \langle \mathbf{u}_{t,\mathsf{i}}, \mathbf{v}_q \rangle, \qquad \left(\mathbf{r}_{t''}[(\mathsf{i}'', q'')] = \mathbf{r}_{\mathrm{x}}[(t'', \mathsf{i}'')] \cdot \mathbf{r}_{\mathrm{f}}[q'']\right)
\end{aligned}$$

where vectors $\mathbf{u}_{t,\mathsf{i}}$ and $\mathbf{v}_q$ are as follows, with $\mathbb{1}\{\cdots\}$ indicating where a condition (its argument) is true:

$$\begin{aligned}
\mathbf{u}_{t,\mathsf{i}} &= \big(\quad \mathbf{r}_{\mathrm{x}}[(t-1, \mathsf{i})] \quad , \quad \cdots \quad , \quad \mathbf{r}_{\mathrm{x}}[(t, \mathsf{i}')] \cdot \mathbb{1}\{x = \mathbf{x}[i] \text{ and } w = \mathbf{W}[j]\} \quad , \quad \cdots \quad , \quad \mathbf{0} \quad \big), \\
\mathbf{v}_q &= \big(\quad -\mathbf{r}_{\mathrm{f}}[q] \quad , \quad \cdots \quad , \quad (\mathbf{M}_{x,w,w',\Delta i,\Delta j} \mathbf{r}_{\mathrm{f}})[q] \quad , \quad \cdots \quad , \quad \mathbf{0} \quad \big).
\end{aligned}$$

Similarly, we can "decompose" $\ell_{\mathsf{init}} = \boldsymbol{\iota}_{1,1,\mathbf{0}_S,1}^{\intercal} \mathbf{r}_0$ as $\langle \mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)], \mathbf{r}_{\mathrm{f}}[1] \rangle$. (We omit the details on how to handle $\boldsymbol{\ell}_{T+1}$ for simplicity.) Given such decomposition, our semi-compact 1-ABE scheme follows immediately by using IPFE to compute the garbling.

HONEST ALGORITHMS

$$\begin{aligned}
\mathsf{sk}(M, \mu): &\quad \mathsf{isk}_{\mathsf{init}}(\quad \mathbf{r}_{\mathrm{f}}[1] \quad , \mathbf{0}), \quad (\forall q) \quad \mathsf{isk}_q(\ \mathbf{v}_q \ , \mathbf{0}) \\
\mathsf{ct}_{\mathbf{x},T,S}: &\quad \mathsf{ict}_{\mathsf{init}}(\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)] \ , \mathbf{0}), \quad (\forall t, \mathsf{i}) \quad \mathsf{ict}_{t,\mathsf{i}}(\mathbf{u}_{t,\mathsf{i}} \ , \mathbf{0})
\end{aligned}$$

Decrypting the pair $\mathsf{isk}_{\mathsf{init}}, \mathsf{ict}_{\mathsf{init}}$ (generated using one master secret key) gives exactly the first label ℓ_{init}, while decrypting $\mathsf{isk}_q, \mathsf{ict}_{t,\mathsf{i}}$ (generated using another master secret key) gives the label $\boldsymbol{\ell}_t[(\mathsf{i}, q)]$ in the exponent, generated using pseudorandomness $\mathbf{r}_t[(\mathsf{i}, q)] = \mathbf{r}_{\mathrm{x}}[(t, \mathsf{i})] \cdot \mathbf{r}_{\mathrm{f}}[q]$. Note that the honest algorithms encode $\mathbf{r}_{\mathrm{f}}[q]$ (in $\mathbf{v}_q$) and $\mathbf{r}_{\mathrm{x}}[(t, \mathsf{i})]$ (in $\mathbf{u}_{t,\mathsf{i}}$) in IPFE secret keys and ciphertexts that use the *two* source groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. As such, we cannot directly use SXDH to argue the pseudorandomness of $\mathbf{r}_t[(\mathsf{i}, q)]$. In the security proof, we will use the function-hiding property of IPFE to move both $\mathbf{r}_{\mathrm{x}}[(t, \mathsf{i})]$ and $\mathbf{r}_{\mathrm{f}}[q]$ into the same source group before invoking SXDH.

Adaptive Security. To show adaptive security, we follow the same blueprint of going through a sequence of hybrids, where we first hardwire ℓ_{init} and compute it reversely using RevSamp, and next simulate the other labels $\boldsymbol{\ell}_t[(\mathsf{i}, q)]$ one by

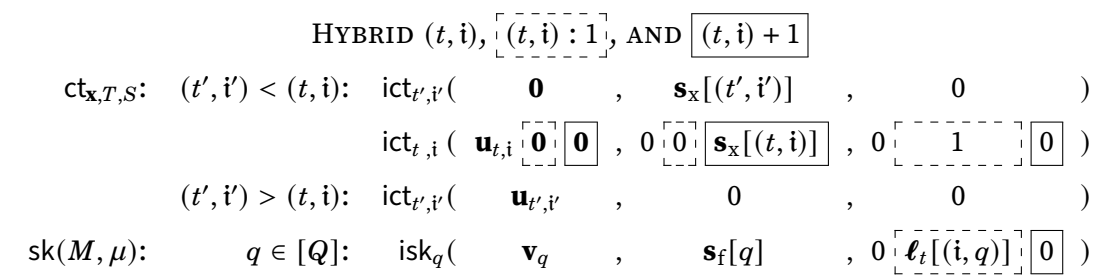
one. Hardwiring ℓ_{init} is easy by relying on the function-hiding property of IPFE. However, it is now more difficult to simulate $\boldsymbol{\ell}_t[(\mathrm{i}, q)]$ because *i)* before simulating $\boldsymbol{\ell}_t[(\mathrm{i}, q)]$, we need to switch its *randomizer* $\mathbf{r}_{t-1}[(\mathrm{i}, q)] = \mathbf{r}_{\mathrm{x}}[(t-1, \mathrm{i})] \cdot \mathbf{r}_{\mathrm{f}}[q]$ to truly random $\mathbf{r}_{t-1}[(\mathrm{i}, q)] \xleftarrow{\$} \mathbb{Z}_p$, which enables us to simulate the label $\boldsymbol{\ell}_t[(\mathrm{i}, q)]$ as random, and *ii)* to keep simulation progressing, we need to switch the random $\boldsymbol{\ell}_t[(\mathrm{i}, q)]$ back to a pseudorandom value $\boldsymbol{\ell}_t[(\mathrm{i}, q)] = \mathbf{s}_{\mathrm{x}}[(t, \mathrm{i})] \cdot \mathbf{s}_{\mathrm{f}}[q]$, as otherwise, there is not enough space to store all $\Theta(NTS2^S Q)$ random labels $\boldsymbol{\ell}_t[(\mathrm{i}, q)]$.

We illustrate how to carry out above proof steps in the simpler case where the adversary queries for the ciphertext first and the secret key second. The other case where the secret key is queried first is handled using similar ideas, but the technicality becomes much more delicate.

In hybrid (t, i),

- the first label ℓ_{init} is reversely computed and hardwired in the secret key $\mathsf{isk}_{\text{init}}$, i.e., $\mathsf{ict}_{\text{init}}$ encrypts $(1, \mathbf{0})$ and $\mathsf{isk}_{\text{init}}$ encrypts $(\ell_{\text{init}}, \mathbf{0})$ with $\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$,
- all the labels $\boldsymbol{\ell}_{t'}[(\mathrm{i}', q)]$ with $(t', \mathrm{i}') < (t, \mathrm{i})$ are simulated as $\mathbf{s}_{\mathrm{x}}[(t', \mathrm{i}')] \cdot \mathbf{s}_{\mathrm{f}}[q]$ — observe that the ciphertext $\mathsf{ict}_{t',\mathrm{i}'}$ encodes only $\mathbf{s}_{\mathrm{x}}[(t', \mathrm{i}')]$ in the second slot, which is multiplied by $\mathbf{s}_{\mathrm{f}}[q]$ in the second slot of isk_q,
- all the labels $\boldsymbol{\ell}_{t'}[(\mathrm{i}', q)]$ with $(t', \mathrm{i}') \geq (t, \mathrm{i})$ are generated honestly as the honest algorithms do.

The hybrid transition is depicted as follows.



Moving from hybrid (t, i) to its successor $((t, \mathrm{i}) + 1)$, the only difference is that the labels $\boldsymbol{\ell}_t[(\mathrm{i}, q)]$ (for all $q \in [Q]$) are switched from being honestly generated $\langle \mathbf{u}_{t,\mathrm{i}}, \mathbf{v}_q \rangle$ to pseudorandom $\mathbf{s}_{\mathrm{x}}[(t, \mathrm{i})] \cdot \mathbf{s}_{\mathrm{f}}[q]$, as depicted above with values in the solid box (the rest of the hybrid is identical to hybrid (t, i)). The transition can be done via an

intermediate hybrid $(t, \mathsf{i}) : 1$ with values in the dashed box. In this hybrid, all the labels $\boldsymbol{\ell}_t[(\mathsf{i}, q)]$ produced as the inner products of all $\mathbf{v}_q$'s and $\mathbf{u}_{t,\mathsf{i}}$ are temporarily hardwired in the secret keys isk_q, using the third slot (which is zeroed out in all the other $\mathbf{u}_{(t',\mathsf{i}')\neq(t,\mathsf{i})}$'s). Furthermore, $\mathbf{u}_{t,\mathsf{i}}$ is removed from $\mathsf{ict}_{t,\mathsf{i}}$. As such, the random scalar $\mathbf{r}_{\mathsf{x}}[(t-1, \mathsf{i})]$ (formerly embedded in $\mathbf{u}_{t,\mathsf{i}}$) no longer appears in the exponent of $\mathbb{G}_1$, and $\ell_{\mathsf{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ can be performed using $\mathbf{r}_{\mathsf{x}}[(t-1, \mathsf{i})]$, $\mathbf{r}_{\mathsf{f}}[q]$, $\mathbf{r}_{t-1}[(\mathsf{i}, q)]$ in the exponent of $\mathbb{G}_2$. Therefore, we can invoke SXDH over $\mathbb{G}_2$ to switch the randomizers $\mathbf{r}_{t-1}[(\mathsf{i}, q)] = \mathbf{r}_{\mathsf{x}}[(t-1, \mathsf{i})] \cdot \mathbf{r}_{\mathsf{f}}[q]$ to be truly random, and hence so are the labels $\boldsymbol{\ell}_t[(\mathsf{i}, q)] \xleftarrow{\$} \mathbb{Z}_p$ (due to marginal randomness). By a similar argument, this intermediate hybrid $(t, \mathsf{i}) : 1$ is also indistinguishable to $((t, \mathsf{i}) + 1)$, as the random $\boldsymbol{\ell}_t[(\mathsf{i}, q)]$ can be switched to $\mathbf{s}_{\mathsf{x}}[(t, \mathsf{i})] \cdot \mathbf{s}_{\mathsf{f}}[q]$ in hybrid $((t, \mathsf{i}) + 1)$, relying again on SXDH and the function-hiding property of IPFE. This concludes our argument of security in the simpler case where the ciphertext is queried first.

AKGS and 1-ABE for NL. Our construction of AKGS and 1-ABE essentially works for NL without modification, because the computation of a non-deterministic logspace Turing machine $M = (Q, q_{\mathsf{acc}}, \delta)$ on an input $\mathbf{x}$ can also be represented as a sequence of matrix multiplications. We briefly describe how by pointing out the difference from L. The transition function δ of an NTM does not instruct a unique transition, but rather specifies a set of legitimate transitions. Following one internal configuration $(i, j, \mathbf{W}, q)$, there are potentially many legitimate successors:

$$(i, j, \mathbf{W}, q) \rightsquigarrow \Big\{ (i + \Delta i,\ j + \Delta j,\ \mathsf{overwrite}(\mathbf{W}, j, w'),\ q') \\ \Big| \ (q', w', \Delta i, \Delta j) \in \delta(q, \mathbf{x}[i], \mathbf{W}[j]) \Big\}.$$

The computation is accepting if and only if *there exists* a path with T legitimate transitions starting from $(1, 1, \mathbf{0}_S, 1)$, going through $(i_t, j_t, \mathbf{W}_t, q_t)$ for $t \in [T]$, and landing at $q_T = q_{\mathsf{acc}}$.

Naturally, we modify the transition matrix as follows to reflect all legitimate transitions. The only difference is that each transition block determined by δ may map a state q to multiple states q', as highlighted in the solid box (see Equations (4.3, 4.4)):

$$\mathbf{M}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] = \mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] \cdot \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q,q'],$$

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = \begin{cases} 1, & \text{if } (q',w',\Delta i,\Delta j) \boxed{\in} \delta(q,x,w'); \\ 0, & \text{otherwise.} \end{cases}$$

Let's observe the effect of right-multiplying $\mathbf{M}(\mathbf{x})$ to an $\boldsymbol{\iota}_{\mathfrak{i},q}^{\intercal}$ indicating configuration $(\mathfrak{i},q)$. The product $\boldsymbol{\iota}_{\mathfrak{i},q}^{\intercal}\mathbf{M}(\mathbf{x})$ is a vector $\mathbf{c}_1$ such that $\mathbf{c}_1[(\mathfrak{i}',q')] = 1$ if and only if $(\mathfrak{i}',q')$ is a legitimate next configuration. Multiplying $\mathbf{M}(\mathbf{x})$ one more time, $\boldsymbol{\iota}_{\mathfrak{i},q}^{\intercal}\big(\mathbf{M}(\mathbf{x})\big)^2$ gives $\mathbf{c}_2$ where $\mathbf{c}_2[(\mathfrak{i}',q')]$ is the number of length-2 paths of legitimate transitions from $(\mathfrak{i},q)$ to $(\mathfrak{i}',q')$. Inductively, $\boldsymbol{\iota}_{\mathfrak{i},q}^{\intercal}\big(\mathbf{M}(\mathbf{x})\big)^t$ yields $\mathbf{c}_t$ that counts the number of length-t paths from $(\mathfrak{i},q)$ to any other internal configuration $(\mathfrak{i}',q')$. Therefore, we can *arithmetize* the computation of M on $\mathbf{x}$ as

$$M|_{N,T,S}(\mathbf{x}) = \boldsymbol{\iota}_{(1,1,\mathbf{0},1)}^{\intercal}\big(\mathbf{M}(\mathbf{x})\big)^T\mathbf{t} \qquad \text{for} \qquad \mathbf{t} = \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \boldsymbol{\iota}_{q_{\mathrm{acc}}}. \tag{4.5}$$

Right-multiplying $\mathbf{t}$ at the end sums up the number of paths to $(\mathfrak{i},q_{\mathrm{acc}})$ for all $\mathfrak{i}$ in $\mathbf{c}_T$ (i.e., accepting paths).

If the computation is not accepting — there is no path to any $(\mathfrak{i},q_{\mathrm{acc}})$ — the final sum would be 0 as desired. If the computation is accepting — there is a path to some $(\mathfrak{i},q_{\mathrm{acc}})$ — then the sum should be non-zero (up to the following technicality). Now that we have represented NL computation as matrix multiplication, we immediately obtain AKGS and 1-ABE for NL using the same constructions for L.

A Technicality in the Correctness for NL. The correctness of our scheme relies on the fact that when the computation is accepting, the matrix multiplication formula (Equation (4.5)) counts correctly the total number of length-T accepting paths. However, a subtle issue is that in our 1-ABE, the matrix multiplications are carried out over $\mathbb{Z}_p$, where p is the order of pairing groups. This means that if the total number of accepting paths happens to be a multiple of p, the sequence of matrix multiplications modulo p carried out in 1-ABE would return 0, while the correct output should be non-zero. This technicality can be circumvented if p is entropic (with $\omega(\log\lambda)$ bits of min-entropy) and the computation $(M,\mathbf{x},T,S)$ is independent of p. In that case,

the probability that the number of accepting paths is a multiple of p is negligible. If the 1-ABE is based on pairing groups of entropic order, we have statistical correctness for computations $(M, \mathbf{x}, T, S)$ chosen statically ahead of time (independent of p). We believe such *static* correctness is sufficient for most applications where correctness is meant for non-adversarial behaviors. However, if the computation $(M, \mathbf{x}, T, S)$ is chosen *adaptively* to make the number of accepting paths a multiple of p, then an accepting computation will be mistakenly rejected. We stress that security is unaffected since if an adversary chooses M and $(\mathbf{x}, T, S)$ as such, it only learns less information.

The Special Cases of DFA and NFA. DFA and NFA are special cases of L and NL, respectively, as they can be represented as Turing machines with a working tape of size $S = 1$ that always runs in time $T = N$, and the transition function δ always moves the input tape pointer to the right. Therefore, the internal configuration of a finite automaton contains only the state q, and the transition matrix $\mathbf{M}(x)$ is determined by δ and the current input bit x under scan. Different from the case of L and NL, here the transition matrix no longer keeps track of the input tape pointer since its movement is fixed — the i^{th} step uses the transition matrix $\mathbf{M}(\mathbf{x}[i])$ depending only on $\mathbf{x}[i]$. Thus, the computation can be represented as follows:

$$M(\mathbf{x}) = \boldsymbol{\iota}_1^\intercal \prod_{i=1}^{N} \mathbf{M}(\mathbf{x}[i]) \cdot \boldsymbol{\iota}_{q_{\text{acc}}} = \boldsymbol{\iota}_1^\intercal \prod_{i=1}^{N} \big(\mathbf{M}_0(1-\mathbf{x}[i]) + \mathbf{M}_1\mathbf{x}[i]\big) \cdot \boldsymbol{\iota}_{q_{\text{acc}}},$$

where $\mathbf{M}_b[q, q'] = \mathbb{1}\{q' = \delta(q, b)\}$ (for DFA) or $\mathbf{M}_b[q, q'] = \mathbb{1}\{q' \in \delta(q, b)\}$ (for NFA). Our construction of AKGS directly applies:

$$\begin{aligned} \ell_{\text{init}} &= L_{\text{init}}(\mathbf{x}) = \boldsymbol{\iota}_1^\intercal \mathbf{r}_0, \\ \text{for } i \in [N], \qquad \boldsymbol{\ell}_i &= \big(L_{i,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_{i-1} + \boxed{\mathbf{M}(\mathbf{x}[i])}\,\mathbf{r}_i, \qquad (\mathbf{r}_{i-1}, \mathbf{r}_i \xleftarrow{\$} \mathbb{Z}_p^Q) \\ \boldsymbol{\ell}_{N+1} &= \big(L_{N+1,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_N + \mu \cdot \boldsymbol{\iota}_{q_{\text{acc}}}. \end{aligned}$$

When using pseudorandomness $\mathbf{r}_i[q] = \mathbf{r}_{\text{x}}[i] \cdot \mathbf{r}_{\text{f}}[q]$, the labels $\boldsymbol{\ell}_i[q]$ can be computed as $\langle \mathbf{u}_i, \mathbf{v}_q \rangle$ with

$$\begin{aligned} \mathbf{v}_q &= (\ -\mathbf{r}_{\text{f}}[q]\ ,\ \ (\mathbf{M}_0\mathbf{r}_{\text{f}})[q]\ \ ,\ (\mathbf{M}_1\mathbf{r}_{\text{f}})[q]\ ,\ \mathbf{0}), \\ \mathbf{u}_i &= (\ \mathbf{r}_{\text{x}}[i-1]\ ,\ (1-\mathbf{x}[i]) \cdot \mathbf{r}_{\text{x}}[i]\ ,\ \mathbf{x}[i] \cdot \mathbf{r}_{\text{x}}[i]\ ,\ \mathbf{0}). \end{aligned}$$

Applying our 1-ABE construction with respect to such "decomposition" gives *compact* 1-ABE for DFA and NFA with secret keys of size $O(Q)$ and ciphertexts of size $O(N)$.

Discussion. Previously, there have been constructions of ABE for DFA based on pairing [Wat12,Att14,Att16,AC17,GWW19,AMY19b] and ABE for NFA based on LWE [AMY19a]. However, no previous scheme achieves adaptive security unless based on q-type assumptions [Att14,Att16]. The work of [BL15] constructed ABE for DFA, and that of [AFS19] for random-access machines, both based on LWE, but they only support inputs of bounded length, giving up the important advantage of uniform computation of handling unbounded-length inputs. There are also constructions of ABE (and even the stronger generalization, functional encryption) for Turing machines [GKP+13b, AS16,AM18,KNTY19] based on strong primitives such as multilinear maps, extractable witness encryption, and indistinguishability obfuscation. However, these primitives are non-standard and currently not well-understood.

In terms of techniques, this chapter is most related to previous pairing-based ABE for DFA, in particular, the recent construction based on k-Lin [GWW19]. These ABE schemes for DFA use a linear secret sharing scheme for DFA first proposed in [Wat12], and combining the secret key and the ciphertext produces a secret sharing in the exponent, which reveals the secret if and only if the DFA computation is accepting. Proving (even selective) security is complicated. Roughly speaking, the work of [GWW19] relies on an *entropy propagation technique* to trace the DFA computation and propagate a few random masks "down" the computation path, with which they can argue that secret information related to the states that are *backward reachable* from the final accepting states is hidden. The technique is implemented using "nested two-slot" dual system encryption [CGKW18a,HKS15,LW10,LW11,OT12, Wat09] combined with a combinatorial mechanism for propagation.

Our AKGS is a generalization of Waters' secret sharing scheme to L and NL, and the optimized version for DFA is identical to Waters' secret sharing scheme. Furthermore, our 1-ABE scheme from AKGS and IPFE is more modular. In particular, our proof (similar to our 1-ABE for non-uniform computation) does not reason about or trace the computation, and simply relies on the structure of AKGS. Using IPFE enables us to

design sophisticated sequences of hybrids without getting lost in the algebra, as IPFE helps separating the logic of changes in different hybrids from how to implement the changes. For instance, we can easily manage multiple slots in the vectors encoded in IPFE for holding temporary values and generating pseudorandomness.

The concurrent work of [GW20a] achieves adaptively secure ABE for DFA using the partial selectivization techniques. See Section 4.7 for more discussion on this.

4.3 Preliminaries

Table 4.1 explains select single-letter symbols used in this chapter.

Indexing. Throughout this chapter, we use $\mathcal{I}, \mathcal{s}$ to denote finite sets of indices from some (contextually specified) index universe.

For ease of exposition, we extensively use non-integer indices for vectors. Let S be any set, we write $S^{\mathcal{I}}$ for the set of vectors whose entries are in S and indexed by $\mathcal{I}$, i.e., $S^{\mathcal{I}} = \{ (\mathbf{v}[i])_{i\in\mathcal{I}} \mid \mathbf{v}[i] \in S \}$. For example, $\mathbb{Z}_p^{[n]}$ is just $\mathbb{Z}_p^n$. Suppose $\mathcal{s}_1, \mathcal{s}_2$ are two index sets with $\mathcal{s}_1 \subseteq \mathcal{s}_2$, for any vector $\mathbf{v} \in \mathbb{Z}_p^{\mathcal{s}_2}$, we denote by $\mathbf{v}|_{\mathcal{s}_1}$ its canonical projection onto $\mathbb{Z}_p^{\mathcal{s}_1}$, i.e., $\mathbf{v}|_{\mathcal{s}_1} \in \mathbb{Z}_p^{\mathcal{s}_1}$ and $\mathbf{v}|_{\mathcal{s}_1}[i] = \mathbf{v}[i]$ for all $i \in \mathcal{s}_1$.

Similarly, non-integer indices can be used for matrices. Let S be a set and $\mathcal{I}_1, \mathcal{I}_2$ two index sets, we write $S^{\mathcal{I}_1\times\mathcal{I}_2}$ for the set of matrices whose entries are in S and indexed by $\mathcal{I}_1$ in the row and $\mathcal{I}_2$ in the column. Matrix multiplication and multiplication between matrices and vectors take their natural meaning so that the inner (common) index set is summed over. For example, a matrix in $\mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_2}$ can multiply a vector in $\mathbb{Z}_p^{\mathcal{I}_2}$ to the left yielding a vector in $\mathbb{Z}_p^{\mathcal{I}_1}$, and it can also multiply any matrix in $\mathbb{Z}_p^{\mathcal{I}_2\times\mathcal{I}_3}$ to the left yielding a matrix in $\mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_3}$.

To make transposition of vectors meaningful, we equate $S^{\mathcal{I}}$ with $S^{\mathcal{I}\times\{0\}}$ and S with $S^{\{0\}}$ (here, $\{0\}$ is just any singleton set). For example, suppose $\mathbf{u} \in \mathbb{Z}_p^{\mathcal{I}_1}$, $\mathbf{v} \in \mathbb{Z}_p^{\mathcal{I}_2}$, $\mathbf{M} \in \mathbb{Z}_p^{\mathcal{I}_1\times\mathcal{I}_2}$, then $\mathbf{u}$, $\mathbf{v}$ are also matrices of shape $\mathcal{I}_1 \times \{0\}$, $\mathcal{I}_2 \times \{0\}$, respectively. The expression $\mathbf{u}^\intercal\mathbf{M}\mathbf{v}$ is a chained multiplication of matrices of shapes $\{0\} \times \mathcal{I}_1$, $\mathcal{I}_1 \times \mathcal{I}_2$, $\mathcal{I}_2 \times \{0\}$, so its value is a matrix in $\mathbb{Z}_p^{\{0\}\times\{0\}}$, which is just an element of $\mathbb{Z}_p$, the same as in the case where $\mathbf{u}, \mathbf{v}, \mathbf{M}$ are vectors and matrices indexed by integers.

Table 4.1. Select single-letter symbols in this chapter.

symbol	meaning
b	challenge bit
$f, \mathbf{x}, \mathbf{M}$	function, input, ABP matrix
α, β, γ	linear coefficient, constant, evaluation result
$L, \mathbf{L}, \ell, m, j$	label function, coefficients, label, count, index
M, Q, δ	machine, state count, transition function
N, T, S	input length, time bound, space bound
t, i	time step, input tape pointer
$j, \mathbf{W}$	working tape pointer, working tape content
$\mathbf{M}_\tau, \mathbf{M}_0, \mathbf{M}_1$	transition matrices
$\mathbf{s}, \mathbf{r}$	MDDH_k randomness, garbling randomness (ABE for ABP)
Q, q	query count, query index (ABE for ABP)
$\mathbf{r}, \mathbf{R}$ / $\mathbf{s}, \mathbf{S}$	garbling/simulation randomness (both MDDH_k; ABE for NL)
Φ, φ	query count, query index (ABE for NL)

The notation of $\mathbf{0}$ is extended to arbitrary index sets. We write $\mathbf{1}_{\mathcal{I}}$ (when clear in context, $\mathcal{I}$ can be omitted) for the vector whose entries are indexed by $\mathcal{I}$ and are all 1.

The Kronecker product uses the Cartesian product of the index sets. Let $\mathbf{v}_1 \in \mathbb{Z}_p^{\mathfrak{s}_1}$ and $\mathbf{v}_2 \in \mathbb{Z}_p^{\mathfrak{s}_2}$ be two vectors, $\mathbf{v} = \mathbf{v}_1 \otimes \mathbf{v}_2$ is a vector in $\mathbb{Z}_p^{\mathfrak{s}_1 \times \mathfrak{s}_2}$ (not a matrix) with entries defined by $\mathbf{v}[(s_1, s_2)] = \mathbf{v}_1[s_1]\mathbf{v}_2[s_2]$. For two matrices $\mathbf{M}_1 \in \mathbb{Z}_p^{\mathcal{I}_1 \times \mathcal{I}_1'}$ and $\mathbf{M}_2 \in \mathbb{Z}_p^{\mathcal{I}_2 \times \mathcal{I}_2'}$, their Kronecker product $\mathbf{M} = \mathbf{M}_1 \otimes \mathbf{M}_2$ is a matrix in $\mathbb{Z}_p^{(\mathcal{I}_1 \times \mathcal{I}_2) \times (\mathcal{I}_1' \times \mathcal{I}_2')}$ with entries defined by $\mathbf{M}[(i_1, i_2), (j_1, j_2)] = \mathbf{M}_1[i_1, j_1]\mathbf{M}_2[i_2, j_2]$.

Partial Application. Partially applying a function yields another function with fewer parameters, and specifying part of the indices of a vector/matrix yields another vector/matrix with fewer indices. We use the usual syntax for evaluating a function or indexing into a vector/matrix, except that unspecified variables are represented by "$\sqcup$". For example, let $\mathbf{M} \in \mathbb{Z}_p^{(A \times B) \times (C \times D)}$, $a \in A$, $d \in D$, then $\mathbf{M}[(a, \sqcup), (\sqcup, d)]$ is the matrix $\mathbf{N} \in \mathbb{Z}_p^{B \times C}$ such that $\mathbf{N}[b, c] = \mathbf{M}[(a, b), (c, d)]$ for all $b \in B$, $c \in C$.

Coefficient Vector. An affine function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ is conveniently associated with its coefficient vector $\mathbf{f} \in \mathbb{Z}_p^{\mathfrak{s}}$ (the same letter in boldface) for $\mathfrak{s} = \{\mathsf{const}\} \cup \{\, \mathsf{coef}_i \mid i \in \mathcal{I} \,\}$ such that

$$f(\mathbf{x}) = \mathbf{f}[\mathsf{const}] + \sum_{i \in \mathcal{I}} \mathbf{f}[\mathsf{coef}_i]\mathbf{x}[i].$$

4.3.1 Computation Models

In this chapter, we consider arithmetic computations, i.e., functions with domain $\mathbb{Z}_p^{\mathcal{I}}$ for some prime p and index set $\mathcal{I}$ and with codomain $\mathbb{Z}_p$. For them to work with ABE, we first associate Boolean-output functions (predicates) with them. There are two natural ways of associating arithmetic functions with predicates — whether or not the output is zero and whether or not the output is non-zero.

Definition 4.1 (zero-test predicates). Let $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$. Define

$$f_{\neq 0}(\mathbf{x}) = \begin{cases} 0, & \text{if } f(\mathbf{x}) = 0; \\ 1, & \text{if } f(\mathbf{x}) \neq 0; \end{cases} \qquad f_{=0}(\mathbf{x}) = \neg f_{\neq 0}(\mathbf{x}) \qquad \text{for } \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}.$$

Arithmetic Branching Programs.[17] ABP is a computation model introduced in [Nis91] and later studied in [IK97,BG99,IK00,IK02,IW14]. We focus on the variant over $\mathbb{Z}_p$.

Definition 4.2 (ABP). An *arithmetic branching program* over $\mathbb{Z}_p^{\mathcal{I}}$ is a weighted directed acyclic graph (V, E, w, s, t) with two distinguished vertices, where V is the set of vertices, E is the set of edges, $w : E \to (\mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p)$ specifies an affine weight function for each edge, and $s, t \in V$ are the distinguished vertices. The in-degree of s and the out-degree of t are 0. Such a program *computes* a function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ defined by

$$f(\mathbf{x}) = \sum_{\substack{s\text{-}t \text{ path} \\ e_1 \cdots e_i}} \prod_{j=1}^{i} w(e_j)(\mathbf{x}).$$

The *size* of the ABP is $|V|$, the number of vertices. The class of all ABP is

$$\mathsf{ABP} = \left\{ f \,\middle|\, f \text{ is an ABP over } \mathbb{Z}_p^{\mathcal{I}} \text{ for some prime } p \text{ and index set } \mathcal{I} \right\}.$$

When there is no ambiguity, we use f to represent both the ABP and the function it computes, and ABP both the class of all ABP and the class of all functions computed by some ABP.

ABP as Restricted Multiplication. Alternatively, one can think of an ABP as a straight-line program with addition and restricted multiplication, where one multiplicand must be affine in the input. Let $f = (V, E, w, s, t)$ be some ABP of size m and $s = v_0, v_1, \ldots, v_{m-1} = t$ the vertices sorted topologically. The function $f(\mathbf{x})$ can be computed by the following recurrence:

$$y_0 = 1, \qquad y_j = \sum_{(v_i, v_j) \in E} w(v_i, v_j)(\mathbf{x}) \cdot y_i, \qquad f(\mathbf{x}) = y_{m-1}.$$

Note that in the recurrence relation, we have $i < j$ by the sorting.

ABP as Determinant. It is known [IK97,IK02] that one can compute an ABP by computing the determinant of a special matrix.

[17] It is also known as algebraic branching programs or mod-p counting branching programs, though the precise definitions vary.

Lemma 4.1 (Lemma 1 in [IK02]). *Let $f = (V, E, w, s, t)$ be an ABP (Definition 4.2) of size m and $s = v_0, v_1, \ldots, v_{m-1} = t$ sorted topologically. Let $\mathbf{M}$ be the $(m-1) \times (m-1)$ matrix whose entries are*

$$\mathbf{M}[i+1, j] = \begin{cases} 0, & \text{if } i > j; \\ -1, & \text{if } i = j; \\ 0, & \text{if } i < j \text{ and } (v_i, v_j) \notin E; \\ w(v_i, v_j)(\mathbf{x}), & \text{if } i < j \text{ and } (v_i, v_j) \in E. \end{cases}$$

The entries of $\mathbf{M}$ are affine in $\mathbf{x}$ and $f(\mathbf{x}) = \det \mathbf{M}$.

Turing Machines. The other computation model we consider in this chapter is Turing machines with a read-only input tape and a read-write working tape. This type of Turing machine is used to define space-bounded complexity classes of decision problems, and in this chapter, we construct an ABE scheme for uniform logspace predicates. One tricky issue with handling Turing machines is that the time/space complexities of a Turing machine are non-computable. However, our ABE construction can only handle computation with polynomial time complexity and logarithmic space complexity. Below we define time/space bounded computation of a Turing machine. Correspondingly, in our ABE scheme, the attribute specifies, in addition to an input to the Turing machine, upper bounds on the concrete time complexity T and space complexity S, and decryption succeeds if and only if the machine accepts the input within this time/space bound. We also confine ourselves to the binary alphabet. The definition below captures these features explicitly.

Definition 4.3 (Turing machine, time/space bounded computation). A (deterministic) *Turing machine* (over $\{0,1\}$) is a triplet $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$, where $Q \geq 1$ is the number of states (the set of states is $[Q]$ and the initial state is 1), $\mathbf{y}_{\text{acc}} \in \{0,1\}^Q$ indicates whether each state is accepting, and

$$\delta : [Q] \times \{0,1\} \times \{0,1\} \to [Q] \times \{0,1\} \times \{0, \pm 1\} \times \{0, \pm 1\},\ (q, x, w) \mapsto (q', w', \Delta i, \Delta j)$$

is the state transition function, which, given the current state q, the symbol x on the input tape under scan, and the symbol w on the working tape under scan, specifies

the new state q', the symbol w' overwriting w, the direction Δi to which the input tape pointer moves, and the direction Δj to which the working tape pointer moves. The machine must *hang (instead of halting)* once it reaches an accepting state, i.e., for all $q \in [Q]$ such that $\mathbf{y}_{\text{acc}}[q] = 1$ and all $x, w \in \{0,1\}$, it holds that $\delta(q, x, w) = (q, w, 0, 0)$.

For input length $N \geq 1$ and space complexity bound $S \geq 1$, the set of *internal configurations* of M is $C_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]$, where $(i, j, \mathbf{W}, q) \in C_{M,N,S}$ specifies the input tape pointer $i \in [N]$, the working tape pointer $j \in [S]$, the content of the working tape $\mathbf{W} \in \{0,1\}^S$ and the machine state $q \in [Q]$.

For bit-string $\mathbf{x} \in \{0,1\}^N$ (with $N \geq 1$) and time/space complexity bounds $T, S \geq 1$, the machine M *accepts* $\mathbf{x}$ *within time* T *and space* S if there exists a sequence of internal configurations (a *computation path* of T steps) $c_0, \ldots, c_T \in C_{M,N,S}$ with $c_t = (i_t, j_t, \mathbf{W}_t, q_t)$ such that

$$
\begin{aligned}
& i_0 = 1,\ j_0 = 1,\ \mathbf{W}_0 = \mathbf{0}_S,\ q_0 = 1 && \text{(initial configuration);} \\
& \text{(for } 0 \leq t < T\text{)} \quad \begin{cases} \delta(q_t, \mathbf{x}[i_t], \mathbf{W}_t[j_t]) = (q_{t+1}, \mathbf{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t), \\ \mathbf{W}_{t+1}[j] = \mathbf{W}_t[j] \text{ for all } j \neq j_t \end{cases} && \text{(valid transitions);} \\
& \mathbf{y}_{\text{acc}}[q_T] = 1 && \text{(accepting).}
\end{aligned}
$$

Denote by $M|_{N,T,S}$ the function $\{0,1\}^N \to \{0,1\}$ mapping $\mathbf{x}$ to whether M accepts $\mathbf{x}$ in time T and space S. Define $\mathsf{TM} = \{\, M \mid M \text{ is a Turing machine} \,\}$ to be the set of all Turing machines.

Remarks. The above definition disallows moving off the input/working tape. For example, if δ specifies moving the input tape pointer to the left when it is already at the leftmost position, there is no valid next internal configuration.

We can avoid the problem of moving off the input tape by encoding the input string $ab\cdots cd$ as $1a0b0\cdots0c0d1$. Here, the 1's at the ends indicate moving to one direction (the last direction to which the input tape pointer moved, or left initially) should be avoided, and the interleaving 0's indicate that both directions are allowed. The machine uses two additional bits in its state to remember

- whether it is reading an indicator bit or an input bit (initially "indicator"), and

- to which direction the input tape pointer last moved (initially "left").

The machine can maintain those bits and avoid moving off the input tape accordingly.

We can also eliminate the problem of moving off the working tape *to the left* by encoding $ab\cdots cd$ on the working tape as $1a0b0\cdots 0c0d000\cdots$, where the leading 1 indicates moving to the left should be avoided, and the interleaving 0's indicate both directions are allowed. The machine uses one additional bit in its state and avoids moving off the working tape to the left accordingly.

The problem of moving off the working tape *to the right* is undetectable by the machine, and this is intended. When the space bound is violated, the input is *silently* rejected.

Relations Among Computation Models. A Boolean (resp. arithmetic) formula can be converted to a Boolean (resp. arithmetic) branching program in linear time, and a Boolean branching program (BP) can be trivially converted into an ABP. Therefore, ABP is a model that generalizes all the three without degrade in efficiency. Moreover, if a non-deterministic BP is unambiguous (i.e., there is at most one accepting path), it can be trivially converted to an ABP; otherwise, it can be approximated by an ABP of the same size with random weights. Lastly, a family of polynomial-size ABP can decide languages in $\mathsf{NL}/\mathrm{poly}$ [BG99]. However, this does not mean that ABE for ABP trivially implies ABE for NL. In the former, each secret key is associated with a specific ABP instance hence only works for a specific input length, whereas in the latter, each secret key is associated with a Turing machine and can work for any input length.

4.3.2 Pairing

We choose to present the more complex schemes using the (symmetric external) decisional Diffie–Hellman assumption.

Assumption 4.1 (DDH, SXDH). Let $\mathsf{GroupGen}$ be a group sampler (Definition 2.4), the *DDH assumption* over $\mathbb{G}$ is

$$\{(1^\lambda, \mathsf{gp}, [\![a, b, ab]\!])\}_{\lambda\in\mathbb{N}} \approx \{(1^\lambda, \mathsf{gp}, [\![a, b, c]\!])\}_{\lambda\in\mathbb{N}},$$

where $(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda)$ and $a, b, c \xleftarrow{\$} \mathbb{Z}_p$. For a pairing group sampler, DDH over $\mathbb{G}_i$ is similarly defined for $i \in \{1, 2, \mathrm{T}\}$ (the distribution contains gp of the pairing), and the *SXDH assumption* over the pairing group stipulates that DDH holds over $\mathbb{G}_1$ and $\mathbb{G}_2$ separately.

Remarks. Assumption 4.1 is DDH with respect to the generator chosen by GroupGen, whereas MDDH_1 (Assumption 2.1) is DDH with respect to a random generator. The latter is implied by the former [EHK+13], and the work of [BMZ19] separates the two in the generic group model. In fact, the constructions in this chapter are black-box in the group, so they might as well be instantiated with a modified GroupGen that randomizes the generator. Abandoning the minute technical difference between DDH and MDDH_1 helps focusing on the high-level ideas.

4.3.3 Function-Hiding Slotted IPFE

In this chapter, we rely on a hybrid variant between secret-key and public-key inner-product functional encryption (IPFE) schemes, called *slotted IPFE* [LV16,Lin17]. Here, the index set $\mathfrak{s}$ of vectors is partitioned into two subsets, $\mathfrak{s}_{\mathrm{pub}}$ and $\mathfrak{s}_{\mathrm{priv}}$, referred to as the public/private slot, respectively. Like in a secret-key IPFE, using the master *secret* key, one can generate ciphertexts and keys encoding any vectors. In addition, similar to a public-key IPFE, using the master *public* key, one can generate ciphertexts, however, only for vectors $\mathfrak{s}$ with values set to zero in the private slot (i.e., $\mathbf{u}|_{\mathfrak{s}_{\mathrm{priv}}} = \mathbf{0}$). In other words, master public key allows for encrypting into the public slot only.

As we shall see later, pairing-based slotted IPFE allows us to employ techniques akin to dual system encryption [Wat09,LW10,Att16].

Definition 4.4 (slotted IPFE). Let GroupGen be a pairing group sampler (Definition 2.5). A *slotted inner-product functional encryption scheme* based on GroupGen consists of five efficient algorithms.

- $\mathsf{Setup}(1^\lambda, \mathsf{gp}, \mathfrak{s}_{\mathrm{pub}}, \mathfrak{s}_{\mathrm{priv}})$ takes as input gp (from $\mathsf{GroupGen}(1^\lambda)$) and two disjoint index sets (the public slot $\mathfrak{s}_{\mathrm{pub}}$ and the private slot $\mathfrak{s}_{\mathrm{priv}}$). It outputs a pair of master public/secret keys $(\mathsf{impk}, \mathsf{imsk})$. The overall index set is $\mathfrak{s} = \mathfrak{s}_{\mathrm{pub}} \cup \mathfrak{s}_{\mathrm{priv}}$. Let the group order be p.

- $\mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, [\![\mathbf{v}]\!]_2)$ takes as input imsk (*secret*) and a vector $\mathbf{v} \in \mathbb{Z}_p^{\mathbb{s}}$ encoded in $\mathbb{G}_2$. It outputs a secret key isk for $\mathbf{v}$.
- $\mathsf{Enc}(1^\lambda, \mathsf{imsk}, [\![\mathbf{u}]\!]_1)$ takes as input imsk (*secret*) and a vector $\mathbf{u} \in \mathbb{Z}_p^{\mathbb{s}}$ encoded in $\mathbb{G}_1$. It outputs a ciphertext ict of $\mathbf{u}$.
- $\mathsf{Dec}(1^\lambda, \mathsf{isk}, \mathsf{ict})$ takes as input isk and ict. It is supposed to compute $[\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_T$.
- $\mathsf{SlotEnc}(1^\lambda, \mathsf{impk}, [\![\mathbf{u}]\!]_1)$ takes as input impk (*public*) and a vector $\mathbf{u} \in \mathbb{Z}_p^{\mathbb{s}_{\mathrm{pub}}}$ encoded in $\mathbb{G}_1$. It outputs a ciphertext ict of $\mathbf{u}$.

The scheme must be *decryption-correct*, i.e., for all $\lambda \in \mathbb{N}$, gp in the support of the output of $\mathsf{GroupGen}(1^\lambda)$, disjoint $\mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}}$, and $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^{\mathbb{s}}$,

$$\Pr\left[\begin{array}{lll} (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, \mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}}) & & \\ \mathsf{isk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, [\![\mathbf{v}]\!]_2) & : & \mathsf{Dec}(1^\lambda, \mathsf{isk}, \mathsf{ict}) = [\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_T \\ \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{imsk}, [\![\mathbf{u}]\!]_1) & & \end{array}\right] = 1.$$

The scheme must be *slot-mode-correct*, i.e., for all $\lambda \in \mathbb{N}$, gp in the support of the output of $\mathsf{GroupGen}(1^\lambda)$, disjoint $\mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}}$, and $\mathbf{u} \in \mathbb{Z}_p^{\mathbb{s}_{\mathrm{pub}}}$, let

$$\mathbf{u}' \in \mathbb{Z}_p^{\mathbb{s}}, \qquad \mathbf{u}'[s] = \begin{cases} \mathbf{u}[s], & \text{if } s \in \mathbb{s}_{\mathrm{pub}}; \\ 0, & \text{if } s \in \mathbb{s}_{\mathrm{priv}}; \end{cases}$$

then the following distributions are identical:

$$\left\{\begin{array}{lll} (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, \mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}}) & & \\ \mathsf{ict} \xleftarrow{\$} \mathsf{SlotEnc}(1^\lambda, \mathsf{impk}, [\![\mathbf{u}]\!]_1) & : & (\mathsf{impk}, \mathsf{imsk}, \mathsf{ict}) \end{array}\right\},$$

$$\left\{\begin{array}{lll} (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, \mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}}) & & \\ \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{imsk}, [\![\mathbf{u}']\!]_1) & : & (\mathsf{impk}, \mathsf{imsk}, \mathsf{ict}) \end{array}\right\}.$$

Slotted IPFE generalizes both secret-key and public-key IPFE. A secret-key IPFE can be obtained by setting $\mathbb{s}_{\mathrm{pub}} = \emptyset$, $\mathbb{s}_{\mathrm{priv}} = \mathbb{s}$; a public-key IPFE, $\mathbb{s}_{\mathrm{pub}} = \mathbb{s}$, $\mathbb{s}_{\mathrm{priv}} = \emptyset$. If we drop Enc and rename SlotEnc to Enc in a public-key IPFE, it formally is a pairing-based PHFE per Definition 2.1.

Security. We consider (adaptive) *function-hiding* property.

Definition 4.5 (function-hiding slotted IPFE). A slotted IPFE scheme (Definition 4.4) is *function-hiding* if $\mathsf{Exp}^0_{\mathrm{FH}} \approx \mathsf{Exp}^1_{\mathrm{FH}}$, where $\mathsf{Exp}^b_{\mathrm{FH}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Run $\mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda)$, then run $\mathcal{A}(1^\lambda, \mathsf{gp})$ and receive two disjoint index sets $\mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}}$ from it. Run $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, \mathbb{s}_{\mathrm{pub}}, \mathbb{s}_{\mathrm{priv}})$ and send impk to $\mathcal{A}$. Let $\mathbb{s} = \mathbb{s}_{\mathrm{pub}} \cup \mathbb{s}_{\mathrm{priv}}$.
- **Challenge.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ has two options.
  - $\mathcal{A}$ can submit $[\![\mathbf{v}^0_j]\!]_2, [\![\mathbf{v}^1_j]\!]_2$ for a secret key, where $\mathbf{v}^0_j, \mathbf{v}^1_j \in \mathbb{Z}_p^{\mathbb{s}}$. Upon this challenge, run $\mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, [\![\mathbf{v}^b_j]\!]_2)$ and send isk_j to $\mathcal{A}$.
  - $\mathcal{A}$ can submit $[\![\mathbf{u}^0_i]\!]_1, [\![\mathbf{u}^1_i]\!]_1$ for a ciphertext, where $\mathbf{u}^0_i, \mathbf{u}^1_i \in \mathbb{Z}_p^{\mathbb{s}}$. Upon this challenge, run $\mathsf{ict}_i \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{imsk}, [\![\mathbf{u}^b_i]\!]_1)$ and send ict_i to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit b'. The outcome of the experiment is b' if $\mathbf{v}^0_j\big|_{\mathbb{s}_{\mathrm{pub}}} = \mathbf{v}^1_j\big|_{\mathbb{s}_{\mathrm{pub}}}$ for all j and $\langle \mathbf{u}^0_i, \mathbf{v}^0_j \rangle = \langle \mathbf{u}^1_i, \mathbf{v}^1_j \rangle$ for all i, j. Otherwise, the outcome is set to 0.

Lemma 4.2 ([ALS16,Wee17,LV16,Lin17]). *Let* GroupGen *be a pairing group sampler (Definition 2.5) and $k \geq 1$ an integer constant. If* MDDH_k *(Assumption 2.1) holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$, then there exists a* function-hiding slotted *IPFE scheme (Definitions 4.4 and 4.5) based on* GroupGen.

See Section 4.7 for further discussion of Lemma 4.2.

4.3.4 1-ABE

At the core of our adaptively secure ABE is a construction for the simpler case of 1-key 1-ciphertext secure secret-key ABE — termed 1-ABE. For technical reasons, it is more convenient to put the message into the key.[18] It captures our key ideas for achieving adaptive security using arithmetic key garbling (Definition 4.7) and function-hiding IPFE (Definitions 4.4 and 4.5) while keeping the ciphertext compact.

[18] Jumping ahead, we will fully embrace the idea of putting the message and the function together in Chapter 5.

Definition 4.6 (1-ABE). Let GroupGen be a pairing group sampler (Definition 2.5). A *(secret-key) 1-ABE scheme* based on GroupGen is an ABE (Definition 2.3) with the following modifications.

- Setup outputs msk without mpk.
- KeyGen additionally takes a message $\mu \in \mathbb{Z}_p$ as input.
- Enc uses msk in place of mpk, and does not take a message as input.
- Dec recovers $[\![\mu]\!]_T$ if decryption is authorized.

Such a scheme is *1-key 1-ciphertext secure* (or simply *secure*) if $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$, where $\mathsf{Exp}^b_{\text{1-ABE}}$ is $\mathsf{Exp}^b_{\text{PHFE}}$ (Definition 2.2) with the following modifications.

- In **Setup**, $\mathcal{A}$ does not receive mpk (it still receives gp prior to choosing param).
- In **Query I/II**, when $\mathcal{A}$ submits a query f, sample random pads $\mu^0, \mu^1 \xleftarrow{\$} \mathbb{Z}_p$, run $\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, f, \mu^0)$, and send (sk, μ^b) to $\mathcal{A}$.
- In **Challenge**, $\mathcal{A}$ submits a challenge x without μ. Upon the challenge, run $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, x)$, and send ct to $\mathcal{A}$.
- In **Guess**, $\mathcal{A}$ outputs a bit b'. The outcome of the experiment is b' if $\mathcal{A}$ makes only one key query for some f and $P_{\lambda,\mathsf{gp},\mathsf{param}}(f, x) = 0$. Otherwise, the outcome is set to 0.

4.4 Arithmetic Key Garbling Scheme

Arithmetic key garbling scheme (AKGS) is an information-theoretic primitive refined from randomized encodings [AIK11] and partial garbling schemes [IW14]. It is the information-theoretic core in our construction of 1-ABE (Definition 4.6). Given a function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$, an AKGS produces label functions $L_1, \ldots, L_m : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ that are affine in $\mathbf{x}$. For any $\mathbf{x}$, one can compute $(\alpha f(\mathbf{x}) + \beta)$ from $L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})$ together with f and $\mathbf{x}$, while all other information about α, β are hidden.

Definition 4.7 (AKGS, adopted from Definition 1 in [IW14]). An *arithmetic key garbling scheme* for a function class $\mathcal{F} = \{f\}$, where each $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ for some $p, \mathcal{I}$ specified by f, consists of two efficient algorithms.

- $\mathsf{Garble}(f, \alpha, \beta)$ takes as input $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}$ and $\alpha, \beta \in \mathbb{Z}_p$. It is randomized and outputs m affine functions $L_1, \ldots, L_m : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ (called the *label functions*, specifying how an input is encoded into labels). Pragmatically, it outputs the coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$.

- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ takes as input $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}$, $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, and $\ell_1, \ldots, \ell_m \in \mathbb{Z}_p$. It is deterministic and outputs a value in $\mathbb{Z}_p$. The input $\ell_1, \ldots, \ell_m$ are called the *labels*, which are supposed to be the values of the label functions at $\mathbf{x}$.

The scheme must be *correct*, i.e., for all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$,

$$\Pr\left[\begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_j \leftarrow L_j(\mathbf{x}) \text{ for all } j \in [m] & : \quad \mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \alpha f(\mathbf{x}) + \beta \end{array}\right] = 1.$$

The scheme must also have *deterministic shape*, i.e., m is determined solely by f, independent of α, β and the randomness in Garble. The number of label functions, m, is called the *garbling size* of f under this scheme.

Definition 4.8 (linear AKGS). An AKGS (Definition 4.7) is *linear* if the following conditions hold.

- $\mathsf{Garble}(f, \alpha, \beta)$ uses a *uniformly random* vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m'}$ as its randomness, where m' is determined solely by f, independent of α, β.

- The coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ produced by $\mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ are linear in $(\alpha, \beta, \mathbf{r})$.

- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ is linear in $(\ell_1, \ldots, \ell_m)$.

In this dissertation, AKGS always refers to linear AKGS.

4.4.1 Security Notions

The basic security notion of AKGS requires the existence of an efficient simulator that draws a sample from the real labels' distribution given f, $\mathbf{x}$, and $(\alpha f(\mathbf{x}) + \beta)$. We emphasize, as it is the same case in [IW14], that AKGS does *not* hide $\mathbf{x}$ and hides all other information about α, β except the value $(\alpha f(\mathbf{x}) + \beta)$.

Definition 4.9 ((usual) simulation security, Definition 1 in [IW14]). An AKGS (Definition 4.7) is *secure* if there exists an efficient algorithm Sim such that for all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, the following distributions are identical:

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \dots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_j \leftarrow L_j(\mathbf{x}) \text{ for all } j \in [m] & : \quad (\ell_1, \dots, \ell_m) \end{array} \right\},$$

$$\left\{ \ (\ell_1, \dots, \ell_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta) \quad : \quad (\ell_1, \dots, \ell_m) \right\}.$$

As discussed in Section 4.2.1, the usual simulation security suffices for selective (or semi-adaptive) security. To achieve adaptive security, we need the following stronger property.

Definition 4.10 (piecewise security). An AKGS (Definition 4.7) is *piecewise secure* if the following conditions hold.

- The first label is *reversely sampleable* from the other *labels* together with f and $\mathbf{x}$. The sampling must be perfect even given all the other *label functions*. Formally, there exists an efficient algorithm RevSamp such that for all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, the following distributions are identical:

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \dots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_1 \leftarrow L_1(\mathbf{x}) & : \quad (\ell_1, \mathbf{L}_2, \dots, \mathbf{L}_m) \end{array} \right\},$$

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \dots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_j \leftarrow L_j(\mathbf{x}) \text{ for all } j \in [m],\ j > 1 & : \quad (\ell_1, \mathbf{L}_2, \dots, \mathbf{L}_m) \\ \qquad\qquad \ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \dots, \ell_m) & \end{array} \right\}.$$

- For the other *labels*, each is *marginally random*[19] even given all the *label functions* after it. Formally, for all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, and all $j \in [m]$ such that $j > 1$, the following distributions are identical:

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \leftarrow L_j(\mathbf{x}) & : \quad (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \end{array} \right\},$$

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \xleftarrow{\$} \mathbb{Z}_p & : \quad (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \end{array} \right\}.$$

Lemma 4.3 (¶). *A* piecewise secure *AKGS (Definitions 4.7 and 4.10) is also* secure *(Definition 4.9).*

Proof (Lemma 4.3). Assuming the AKGS is piecewise secure, we let the simulator $\mathsf{Sim}(f, \mathbf{x}, \gamma)$

1. sample $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ for $1 < j \leq m$,
2. reversely sample $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \gamma, \ell_2, \ldots, \ell_m)$, and
3. output $(\ell_1, \ldots, \ell_m)$.

It is readily verified that Sim efficiently and perfectly simulates the labels. Therefore, the AKGS is also secure. □

Special Piecewise Security. We identify a special structural form of AKGS, referred to as special piecewise security. It is equivalent to piecewise security, and has the advantage of being easier to verify. In particular, all AKGS considered in this dissertation have this form.

Definition 4.11 (special piecewise security). An AKGS (Definition 4.7) is *special piecewise secure* if it has the following special form.

- For all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ in $\mathcal{F}$ and all $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, it holds that $\mathsf{Eval}(f, \mathbf{x}, \overset{\ell_1}{\downarrow} 1, \overset{\ell_2}{\downarrow} 0, \ldots, \overset{\ell_m}{\downarrow} 0) \neq 0$.

[19] This is an unfortunate misnomer since [LL20a,LL20b]. The correct wording is "conditionally random". We keep the original version for consistency.

- For all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ in $\mathcal{F}$ and all $\alpha, \beta \in \mathbb{Z}_p$, let $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m'}$ be the randomness $\mathsf{Garble}(f, \alpha, \beta)$ uses. For all $j \in [m]$ such that $j > 1$, the label function L_j produced by $\mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ can be written as

$$L_j(\mathbf{x}) = k_j \mathbf{r}[j-1] + L'_j(\mathbf{x}; \alpha, \beta, \mathbf{r}[\geq j]),$$

where $k_j \in \mathbb{Z}_p$ is a *non-zero constant* (not depending on $\mathbf{x}, \alpha, \beta, \mathbf{r}$) and L'_j is an affine function of $\mathbf{x}$ whose coefficient vector is linear in $(\alpha, \beta, \mathbf{r}[j], \mathbf{r}[j+1], \dots)$. The component $\mathbf{r}[j-1]$ is called the *randomizer* of L_j and ℓ_j.

We note that the first requirement of special piecewise security, on top of Eval being linear in the labels (always required as a linear AKGS), simply says that the coefficient of ℓ_1 is always non-zero. An AKGS with the special form is clearly piecewise secure.

Lemma 4.4 (¶). *A* special piecewise secure *AKGS (Definitions 4.7 and 4.11) is also* piecewise secure *(Definition 4.10). Moreover, the* $\mathsf{RevSamp}$ *algorithm (required in piecewise security) obtained for a special piecewise secure AKGS is linear in* $\gamma, \ell_2, \dots, \ell_m$ *and perfectly recovers* ℓ_1 *even if the randomness of* Garble *is not uniformly sampled.*

Proof (Lemma 4.4). Assuming that the AKGS is special piecewise secure, we first show that the first label is reversely sampleable. Since Eval is linear in the labels, we have

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ell_2, \dots, \ell_m) = \ell_1 \mathsf{Eval}(f, \mathbf{x}, 1, 0, \dots, 0) + \mathsf{Eval}(f, \mathbf{x}, 0, \ell_2, \dots, \ell_m).$$

Define the reverse sampling algorithm as

$$\mathsf{RevSamp}(f, \mathbf{x}, \gamma, \ell_2, \dots, \ell_m) = \frac{\gamma - \mathsf{Eval}(f, \mathbf{x}, 0, \ell_2, \dots, \ell_m)}{\mathsf{Eval}(f, \mathbf{x}, 1, 0, \dots, 0)},$$

which is clearly efficient and linear in $\gamma, \ell_2, \dots, \ell_m$. Note that given $f, \mathbf{x}, \gamma, \ell_2, \dots, \ell_m$ with $\gamma = \alpha f(\mathbf{x}) + \beta$, the first label ℓ_1 is uniquely determined due to the requirement of Eval in special piecewise security, and the algorithm $\mathsf{RevSamp}$ defined above computes this value. Since ℓ_1 is uniquely determined, it does not matter if the label functions $\mathbf{L}_2, \dots, \mathbf{L}_m$ are revealed, nor if the garbling is not generated using uniform randomness.

For the marginal randomness property, for all $1 < j \leq m$, given the label functions $\mathbf{L}_{j+1}, \dots, \mathbf{L}_m$, the label $\ell_j = k_j \mathbf{r}[j-1] + L'_j(\mathbf{x})$ is uniformly random since $\mathbf{r}[j-1]$ is a uniformly random value independent of $\mathbf{L}'_j, \mathbf{L}_{j+1}, \dots, \mathbf{L}_m$. □

A piecewise secure AKGS can be converted into a special piecewise secure one.

Theorem 4.5. *A* piecewise secure *AKGS (Definitions 4.7 and 4.10) is also* special piecewise secure *(Definition 4.11) after an appropriate change of variable for the randomness used by* Garble.

See Section 4.7 for further discussion of Theorem 4.5.

4.5 KP-ABE for Arithmetic Branching Programs

In this section, we show how to construct a KP-ABE for ABP in three steps: *i)* construct a piecewise secure AKGS for ABP; *ii)* build a 1-ABE using a piecewise secure AKGS and a function-hiding secret-key IPFE; *iii)* lift the 1-ABE into a full-fledged ABE using slotted IPFE.

The first step is specific to the class of predicates for which we want to construct an ABE. The second and third steps are independent of the predicates as long as all the predicates share the same domain.

4.5.1 AKGS for ABP

Our piecewise secure AKGS for ABP is a special case of the partial garbling scheme for ABP in [IW14], which is in turn built upon randomizing polynomials [IK00,IK02]. In a partial garbling scheme (PGS), the input of a function $f : \mathbb{Z}_p^{\mathcal{I}_x} \times \mathbb{Z}_p^{\mathcal{I}_z} \to \mathbb{Z}_p$ is divided into two parts, the public part $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}_x}$ and the private part $\mathbf{z} \in \mathbb{Z}_p^{\mathcal{I}_z}$. The labels for f with input $\mathbf{x}, \mathbf{z}$ in a secure PGS contain no other information about $\mathbf{z}$ except the value $f(\mathbf{x}, \mathbf{z})$, but might otherwise completely reveal $\mathbf{x}$.

More specifically, our AKGS for computation $f(\mathbf{x})$ with secrets α, β is essentially a PGS for the computation $(\alpha f(\mathbf{x}) + \beta)$, where $\mathbf{x}$ is the public input and α, β are the private inputs. In this special case (or slightly more generally, when the computation has degree 1 in the private inputs), the PGS of [IW14] satisfies the linearity structure and the (usual) simulation security of AKGS. We further show that it also satisfies the stronger *piecewise security*.

AKGS and PGS have slightly different syntax. In an AKGS, the label functions L_j's are regarded as affine functions of $\mathbf{x}$ with $(\alpha, \beta, \mathbf{r})$ hardwired, and the coefficient vectors $\mathbf{L}_j$ are linear in $(\alpha, \beta, \mathbf{r})$. In a PGS, L_j's are regarded as affine functions of $(\mathbf{x}, \alpha, \beta)$ with $\mathbf{r}$ hardwired, and the coefficient vectors $\mathbf{L}_j$ are affine in $\mathbf{r}$. Below, we recall the PGS for ABP [IW14] in the syntax of an AKGS.

Construction 4.1 (AKGS for ABP [IW14]). Let $\mathcal{F} = \mathsf{ABP}$ be the class of all ABP. The AKGS $(\mathsf{Garble}, \mathsf{Eval})$ for $\mathcal{F}$ operates as follows.

- $\mathsf{Garble}(f, \alpha, \beta)$ takes as input an ABP $f \in \mathcal{F} = \mathsf{ABP}$ and two secrets α, β. Suppose f is over $\mathbb{Z}_p^{\mathcal{I}}$ and of size m, and $\alpha, \beta \in \mathbb{Z}_p$. The algorithm computes the matrix $\mathbf{M}$ in Lemma 4.1 (whose determinant is the output of the function) and augments it into an $m \times m$ matrix

$$\mathbf{M}' = \begin{pmatrix} \overset{\mathbf{M}}{\overbrace{\begin{matrix} * & * & \cdots & * & * \\ -1 & * & \cdots & * & * \\ & -1 & \cdots & * & * \\ & & \ddots & \vdots & \vdots \\ 0 & & & -1 & * \end{matrix}}} & \begin{matrix} \beta \\ 0 \\ 0 \\ \vdots \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & \cdots & 0 & -1 \end{matrix} & \alpha \end{pmatrix} = \begin{pmatrix} \mathbf{M} & \mathbf{m}_1 \\ \mathbf{m}_2^\intercal & \alpha \end{pmatrix}.$$

  Here, the augmented matrix $\mathbf{M}'$ is the matrix for an ABP computing $(\alpha f(\mathbf{x}) + \beta)$. It then samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m-1}$ as its randomness, sets

$$\mathbf{N} = \begin{pmatrix} \mathbf{I}_{m-1} & \mathbf{r} \\ \mathbf{0} & 1 \end{pmatrix},$$

  and defines the label functions by

$$\widehat{\mathbf{M}} = \mathbf{M}'\mathbf{N} = \begin{pmatrix} \mathbf{M} & \mathbf{M}\mathbf{r} + \mathbf{m}_1 \\ \mathbf{m}_2^\intercal & \mathbf{m}_2^\intercal \mathbf{r} + \alpha \end{pmatrix} = \begin{pmatrix} \begin{matrix} & & & \\ & \mathbf{M} & & \\ & & & \\ & & & \end{matrix} & \begin{matrix} L_1(\mathbf{x}) \\ L_2(\mathbf{x}) \\ \vdots \\ L_{m-1}(\mathbf{x}) \end{matrix} \\ \begin{matrix} 0 & 0 & \cdots & 0 & -1 \end{matrix} & L_m(\mathbf{x}) \end{pmatrix}.$$

  The algorithm outputs the coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ of $L_1, \ldots, L_m$.

- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ takes as input $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F} = \mathsf{ABP}$ of size m, an input $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$, and m labels $\ell_1, \ldots, \ell_m$. It computes $\mathbf{M}$ (with the value of $\mathbf{x}$ substituted) in Lemma 4.1 and sets

$$\widehat{\mathbf{M}} = \begin{pmatrix} & & & \ell_1 \\ & \mathbf{M} & & \ell_2 \\ & & & \vdots \\ & & & \ell_{m-1} \\ 0 & 0 & \cdots\ 0 & -1 & \ell_m \end{pmatrix}.$$

  The algorithm outputs $\det \widehat{\mathbf{M}}$.

Syntax and Correctness. We verify that Construction 4.1 is indeed a linear AKGS (Definitions 4.7 and 4.8).

- $L_1, \ldots, L_m$ are affine functions of $\mathbf{x}$. The entries of $\mathbf{M}'$ are affine in $\mathbf{x}$ and those of $\mathbf{N}$ are constant with respect to $\mathbf{x}$. Therefore, the label functions $L_1, \ldots, L_m$ (the last column of $\mathbf{M}'\mathbf{N}$) are affine in $\mathbf{x}$.

- Shape determinism holds. The garbling size of f is its size as an ABP.

- Garble is linear in $(\alpha, \beta, \mathbf{r})$. The matrices $\mathbf{M}, \mathbf{m}_2$ are constant with respect to $(\alpha, \beta, \mathbf{r})$, and $\mathbf{r}, \mathbf{m}_1, \alpha$ are linear in $(\alpha, \beta, \mathbf{r})$, so $(\mathbf{M}\mathbf{r} + \mathbf{m}_1)$ and $(\mathbf{m}_2^\intercal \mathbf{r} + \alpha)$, hence the coefficients of $L_1, \ldots, L_{m-1}$ and L_m, are linear in $(\alpha, \beta, \mathbf{r})$.

- Correctness. Since $\ell_j = L_j(\mathbf{x})$ for all $j \in [m]$, the matrix $\widehat{\mathbf{M}}$ in Eval is the matrix $\widehat{\mathbf{M}}$ in Garble with $\mathbf{x}$ plugged into it, and we have

$$\begin{aligned} \det \widehat{\mathbf{M}} = \det(\mathbf{M}'\mathbf{N}) &= \det \mathbf{M}' \det \mathbf{N} = \det \mathbf{M}' \\ \left(\begin{smallmatrix}\text{Laplace expansion in} \\ \text{the last column of } \mathbf{M}'\end{smallmatrix}\right) &= \alpha \det \mathbf{M} + \beta \\ (\text{Lemma 4.1}) &= \alpha f(\mathbf{x}) + \beta. \end{aligned}$$

- Eval is linear in $\ell_1, \ldots, \ell_m$. This follows from the Laplace expansion of $\det \widehat{\mathbf{M}}$ in the last column.

Security. Ishai and Wee [IW14] observed that in Garble of Construction 4.1, all the possible $\mathbf{N}$'s form a matrix subgroup

$$H = \left\{ \begin{pmatrix} \mathbf{I}_{m-1} & \mathbf{r} \\ \mathbf{0} & 1 \end{pmatrix} \middle| \ \mathbf{r} \in \mathbb{Z}_p^{m-1} \right\},$$

and that the labels are the last column of a uniformly random element from the left coset $\mathbf{M}'H = \{\, \mathbf{M}'\mathbf{N} \mid \mathbf{N} \in H \,\}$. Moreover, each such left coset has a canonical representative $\mathbf{M}''$ determined by f, $\mathbf{x}$, and $(\alpha f(\mathbf{x}) + \beta)$:

$$\mathbf{M}'' = \begin{pmatrix} \boxed{\mathbf{M}} & \begin{matrix} \alpha f(\mathbf{x}) + \beta \\ 0 \\ \vdots \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & \cdots & 0 & -1 \end{matrix} & 0 \end{pmatrix} \in \mathbf{M}'H.$$

By the properties of cosets, $\mathbf{M}''H = \mathbf{M}'H$. Therefore, given f, $\mathbf{x}$, and $(\alpha f(\mathbf{x}) + \beta)$, one can sample a uniformly random element from $\mathbf{M}''H = \mathbf{M}'H$ to simulate the labels. This proves the (usual) simulation security.

We now proceed to the stronger security notions introduced in this chapter.

Theorem 4.6 (¶). *Construction 4.1 is* special piecewise secure *(Definition 4.11).*

Proof (Theorem 4.6). We first show that ℓ_1 has non-zero coefficient in Eval. Recall that

$$\widehat{\mathbf{M}} = \begin{pmatrix} & & & \overset{\mathbf{M}}{\downarrow} & & \\ \boxed{\begin{matrix} * & * & \cdots & * & * \\ -1 & * & \cdots & * & * \\ & -1 & \cdots & * & * \\ & & \ddots & \vdots & \vdots \\ 0 & & & -1 & * \end{matrix}} & & & & & \begin{matrix} \ell_1 \\ \ell_2 \\ \ell_3 \\ \vdots \\ \ell_{m-1} \end{matrix} \\ \begin{matrix} 0 & 0 & \cdots & 0 & -1 \end{matrix} & & & & & \ell_m \end{pmatrix}.$$

Consider the Laplace expansion of $\det \widehat{\mathbf{M}}$ in the last column:

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \det \widehat{\mathbf{M}} = A_1\ell_1 + A_2\ell_2 + \cdots + A_m\ell_m,$$

where A_j is the (j, m)-cofactor of $\widehat{\mathbf{M}}$. Observe that by definition, each A_j is solely

determined by f and $\mathbf{x}$, so the above expansion expresses Eval as a linear function of the labels, and in particular, the coefficient of ℓ_1 is $A_1 = (-1)^{1+m}(-1)^{m-1} = 1 \neq 0$.

Now we show that the label functions are in the special form. Recall that the label functions are defined by the last column of

$$\widehat{\mathbf{M}} = \mathbf{M}'\mathbf{N} = \begin{pmatrix} \mathbf{M}[1,1] & \mathbf{M}[1,2] & \cdots & \mathbf{M}[1,m-2] & \mathbf{M}[1,m-1] & \beta \\ -1 & \mathbf{M}[2,2] & \cdots & \mathbf{M}[2,m-2] & \mathbf{M}[2,m-1] & 0 \\ & -1 & \cdots & \mathbf{M}[3,m-2] & \mathbf{M}[3,m-1] & 0 \\ & & \ddots & \vdots & \vdots & \vdots \\ & & & -1 & \mathbf{M}[m-1,m-1] & 0 \\ & & & & -1 & \alpha \end{pmatrix} \begin{pmatrix} 1 & & & & \mathbf{r}[1] \\ & 1 & & & \mathbf{r}[2] \\ & & 1 & & \mathbf{r}[3] \\ & & & \ddots & \vdots \\ & & & & 1 \quad \mathbf{r}[m-1] \\ & & & & 1 \end{pmatrix}.$$

Computing the label functions $L_2, \ldots, L_m$, we get

$$\begin{array}{lrcl} & & \overset{k_j=-1}{\downarrow} & \overset{L'_j(\mathbf{x};\alpha,\beta,\mathbf{r}_{\geq j})}{\downarrow} \\ (1 < j < m) \qquad L_j(\mathbf{x}) & = & -\mathbf{r}[j-1] & + \sum_{j'=j}^{m-1} \mathbf{r}[j']\mathbf{M}[j,j'], \\ L_m(\mathbf{x}) & = & -\mathbf{r}[m-1] & + \alpha. \\ & & \underset{k_m=-1}{\uparrow} & \underset{L'_m(\mathbf{x};\alpha,\beta)}{\uparrow} \end{array}$$

They are in the required special form (the randomizer of L_j is $\mathbf{r}[j-1]$).

We conclude that Construction 4.1 is special piecewise secure. □

4.5.2 1-ABE for ABP

For any function class $\mathcal{F}$ (e.g., arithmetic branching programs), we show how to construct a 1-ABE (Definition 4.6) for zero-test predicates (Definition 4.1) of $\mathcal{F}$ using a piecewise secure AKGS for $\mathcal{F}$ and a function-hiding secret-key IPFE scheme.

Ingredients of Construction 4.2. Let

- (Garble, Eval) be an AKGS (Definitions 4.7 and 4.8) for $\mathcal{F}$,
- GroupGen a pairing group sampler (Definition 2.5), and
- (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec) a secret-key IPFE (Definition 4.4) based on GroupGen.

Construction 4.2 (1-ABE). Define (see Definitions 2.3 and 4.6)

$$\mathsf{Params}_{\mathsf{gp}} = \{\, 1^n \mid n \in \mathbb{N} \,\}, \qquad F_{\mathsf{gp},1^n} = \{\, f_{\neq 0}, f_{=0} \mid f \in \mathcal{F},\ f : \mathbb{Z}_p^n \to \mathbb{Z}_p \,\},$$
$$X_{\mathsf{gp},1^n} = \mathbb{Z}_p^n, \qquad P_{\mathsf{gp},1^n}(f', \mathbf{x}) = f'(\mathbf{x}),$$

where p is the group order in gp. Our 1-ABE scheme based on GroupGen operates as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^n)$ takes as input the group description gp and the attribute length n in unary. It sets up the IPFE $\mathsf{imsk} \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, \mathfrak{s}_{\text{1-ABE}})$ for the index set $\mathfrak{s}_{\text{1-ABE}} = \{\mathsf{const}, \mathsf{coef}_1, \ldots, \mathsf{coef}_n, \mathsf{sim}_1, \mathsf{sim}_\star\}$. The algorithm outputs $\mathsf{msk} = \mathsf{imsk}$.

  The positions indexed by $\mathsf{const}, \mathsf{coef}_1, \ldots, \mathsf{coef}_n$ in the secret key encode the coefficient vectors of the label functions produced by garbling f with secrets α, β. They encode 1 and $\mathbf{x}$ in the ciphertext. The positions indexed by $\mathsf{sim}_1, \mathsf{sim}_\star$ are set to zero by the honest algorithms, and are only used in the security proof.

- $\mathsf{KeyGen}(\mathsf{msk}, f', \mu)$ takes as input msk, some $f' \in F_{\mathsf{gp},1^n}$, and $\mu \in \mathbb{Z}_p$. It samples $\eta \xleftarrow{\$} \mathbb{Z}_p$ and garbles the function f underlying f' by

$$\begin{cases} \alpha \leftarrow \mu, \quad \beta \leftarrow 0, & \text{if } f' = f_{\neq 0}; \\ \alpha \leftarrow \eta, \quad \beta \leftarrow \mu, & \text{if } f' = f_{=0}; \end{cases} \qquad (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta).$$

  The algorithm generates an IPFE key $\mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2)$ for the following vector $\mathbf{v}_j$ encoding each label function L_j:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0

  It outputs $\mathsf{sk} = (\mathsf{isk}_1, \ldots, \mathsf{isk}_m)$.

- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x})$ takes as input msk and $\mathbf{x} \in \mathbb{Z}_p^n$. It generates an IPFE ciphertext $\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{imsk}, [\![\mathbf{u}]\!]_1)$ encrypting the vector $\mathbf{u}$ containing 1 and $\mathbf{x}$:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0

  The algorithm outputs $\mathsf{ct} = \mathsf{ict}$.

- $\mathsf{Dec}(f', \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ outputs $\perp$ if $f'(\mathbf{x}) = 0$. Otherwise, it parses sk, ct as in KeyGen, Enc and computes

$$(\text{for } j \in [m]) \qquad [\![\ell_j]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}),$$

$$[\![\mu']\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } f' = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } f' = f_{=0}. \end{cases}$$

  The algorithm outputs $[\![\mu']\!]_{\mathrm{T}}$.

Correctness. We verify that the scheme is correct. First, by the correctness of IPFE and the definition of $\{\mathbf{v}_j\}_j, \mathbf{u}$, we have $\ell_j = \langle \mathbf{u}, \mathbf{v}_j \rangle = L_j(\mathbf{x})$ for all $j \in [m]$. Next, by the linearity of Eval in $\ell_1, \ldots, \ell_m$, we can evaluate the garbling in the exponent of the target group and obtain $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \alpha f(\mathbf{x}) + \beta$ in the exponent. In the two cases where decryption should succeed, we have

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} \mu f(\mathbf{x}), & \text{if } f' = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0; \\ \mu, & \text{if } f' = f_{=0} \text{ and } f(\mathbf{x}) = 0. \end{cases}$$

In both cases, the μ' above is equal to μ. Therefore, correctness holds.

Security. We state and prove the security of our 1-ABE.

Theorem 4.7 (¶). *Suppose in Construction 4.2, the AKGS is* piecewise secure *(Definition 4.10) and the IPFE scheme is* function-hiding *(Definition 4.5), then the resultant 1-ABE scheme is* 1-key 1-ciphertext secure *(Definition 4.6).*

Proof (Theorem 4.7). The statement is $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$. Recall that in $\mathsf{Exp}^b_{\text{1-ABE}}$, the adversary is allowed to obtain a single key sk for a policy f' encrypting μ^0 and a single ciphertext ct for an attribute $\mathbf{x}$, where $f', \mathbf{x}$ are chosen adaptively by the adversary and satisfy either $f' = f_{=0}$ and $f(\mathbf{x}) \neq 0$, or $f' = f_{\neq 0}$ and $f(\mathbf{x}) = 0$. (If those constraints are not satisfied, the outcome of the experiment is set to 0.) In addition, the adversary receives μ^b along with sk.

Since $\mathsf{Exp}^0_{\text{1-ABE}}$ and $\mathsf{Exp}^1_{\text{1-ABE}}$ are identical until the adversary receives (sk, μ^b), we analyze the indistinguishability of these two experiments in two cases.

- Case 1: The ciphertext challenge $\mathbf{x}$ appears before the secret key query f'. This is the simpler case, where adaptive security collapses down to selective security, because by the time sk needs to be generated, the challenge attribute $\mathbf{x}$ is already known. The proof proceeds in two simple steps: *i)* by the function-hiding property of IPFE, we can remove the coefficient vectors $\mathbf{L}_j$ of the label functions L_j encoded in isk_j's, and hardwire the labels $\ell_j = L_j(\mathbf{x})$ in isk_j's (such hardwiring is possible thanks to the fact that sk contains many IPFE secret keys, which provide sufficient space for embedding ℓ_j's); and *ii)* after replacing $\mathbf{L}_j$'s by ℓ_j's, by the (usual) simulation security of AKGS, the hardwired labels $\{\ell_j\}_{j\in[m]}$ are independent of μ^0 whenever decryption is unauthorized. Since μ^0 and μ^1 are identically distributed, the two hybrids are identical.

- Case 2: The ciphertext challenge $\mathbf{x}$ appears after the secret key query f'. Now, at the time sk needs to be generated, the adversary has not chosen the challenge attribute $\mathbf{x}$. Consequently, the previous argument breaks down at the step of hardwiring — we can no longer hardwire the labels $\{\ell_j = L_j(\mathbf{x})\}_{j\in[m]}$ in sk, as $\mathbf{x}$ is not yet known, nor can we hardwire the labels in ct, as ct contains only a single IPFE ciphertext ict and does not have enough space for embedding all the ℓ_j's. We resolve this by relying on the piecewise security of AKGS. Roughly speaking, it allows us to gradually switch the coefficient vectors $\mathbf{L}_j$ of the label functions L_j to simulated labels $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ via a series of hybrids. Between neighboring hybrids, only two label functions/labels are changed, which can be hardwired in ict.

Below, we first prove security in the simpler Case 1 then move to Case 2.

Case 1. Consider the following hybrids.

- H_0^b is $\mathsf{Exp}^b_{\text{1-ABE}}$, where the vector $\mathbf{u}$ encrypted in the IPFE ciphertext ict (in the 1-ABE ciphertext ct) and the vectors $\mathbf{v}_j$ encoded in the IPFE secret keys isk_j (in the 1-ABE secret key sk) are described in Table 4.2. Note that $\mathbf{u}$ appears before $\mathbf{v}_j$ in the experiment.

- H_1^b proceeds identically to H_0^b, except that it sets $\mathbf{v}_j[\mathsf{const}]$ to the desired inner product of $\mathbf{v}_j$ with $\mathbf{u}$ — the honest labels $\ell_j = L_j(\mathbf{x})$ — and the other positions

to 0, as described in Table 4.2. Since the inner products $\langle \mathbf{u}, \mathbf{v}_j \rangle$ are the same in H_1^b and H_0^b, it follows directly from the function-hiding property of IPFE that $\mathsf{H}_0^b \approx \mathsf{H}_1^b$.

- H_2^b proceeds identically to H_1^b, except that the hardwired labels $(\ell_1, \ldots, \ell_m)$ are now simulated using $\mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$. By the (usual) simulation security of AKGS, $\mathsf{H}_1^b \equiv \mathsf{H}_2^b$.

We further argue $\mathsf{H}_2^0 \equiv \mathsf{H}_2^1$. In both hybrids, the labels are simulated as $\mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$. In the two cases where decryption is not authorized, we have

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} 0, & \text{if } f' = f_{\neq 0} \text{ and } f(\mathbf{x}) = 0; \\ \eta f(\mathbf{x}) + \mu^0 \quad (\eta \xleftarrow{\$} \mathbb{Z}_p), & \text{if } f' = f_{=0} \text{ and } f(\mathbf{x}) \neq 0. \end{cases}$$

Either way, $(\alpha f(\mathbf{x}) + \beta)$ is independent of μ^0, hence μ^0 is a uniform random value independent of everything else in H_2^0 — so is μ^1 in H_2^1. In addition, everything except μ^0, μ^1 are identically distributed in H_2^0 and H_2^1. Therefore, the two hybrids are identical.

By hybrid argument, we conclude $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$ conditioned on Case 1.

Table 4.2. Hybrids, 1-ABE for ABP, Case 1 ($\mathbf{x}$ then f).

hybrid	vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathsf{H}_0^b \equiv \mathsf{Exp}^b_{\text{1-ABE}}$	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
	$\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
H_1^b	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
	$\mathbf{v}_j$	$\ell_j = L_j(\mathbf{x})$	0	0	0
H_2^b	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
	$\mathbf{v}_j$	simulated ℓ_j	0	0	0
Simulation in H_2^b:		$(\ell_1, \ldots, \ell_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$.			

CASE 2. We prove $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$ conditioned on Case 2 via a series of hybrids. The hybrids are logically divided into three groups: *i)* the first few hybrids remove the first label function, and hardwire the first label that is reversely sampled; *ii)* the middle hybrids are a loop, and in each iteration one label function is replaced by a simulated label; and *iii)* in the final hybrid, all the labels are simulated, and the message μ^0 is information-theoretically hidden.

Table 4.3. Hybrids, 1-ABE for ABP, Case 2 (f then $\mathbf{x}$), first and final.

hybrid	vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathsf{H}^b_0 \equiv \mathsf{Exp}^b_{\text{1-ABE}}$	$\mathbf{v}_1$	$\mathbf{L}_1[\mathsf{const}]$	$\mathbf{L}_1[\mathsf{coef}_i]$	0	0
	$j>1$: $\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0
H^b_1	$\mathbf{v}_1$	0	0	1	0
	$j>1$: $\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	$\ell_1 = L_1(\mathbf{x})$	0
$\mathsf{H}^b_2 \equiv \mathsf{H}^b_{3,2,1}$	$\mathbf{v}_1$	0	0	1	0
	$j>1$: $\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$	0
$\mathsf{H}^b_{3,2\sim m,1\sim 4}$			.....		
$\mathsf{H}^b_4 \equiv \mathsf{H}^b_{3,m,4}$	$\mathbf{v}_1$	0	0	1	0
	$j>1$: $\mathbf{v}_j$	$\ell_j \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
	$\mathbf{u}$	1	$\mathbf{x}[i]$	$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$	0

Reverse sampling in $\mathsf{H}^b_2, \mathsf{H}^b_4$: $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$.

In H^b_2, the labels ℓ_j ($j > 1$) are computed honestly as $L_j(\mathbf{x})$ using IPFE.

In H^b_4, they are simulated as random and hardwired in sk.

We start with the first few hybrids.

- H_0^b is $\mathsf{Exp}_{\text{1-ABE}}^b$, where the vector $\mathbf{u}$ encrypted in the IPFE ciphertext ict (in the 1-ABE ciphertext ct) and the vectors $\mathbf{v}_j$ encoded in the IPFE secret keys isk_j (in the 1-ABE secret key sk) are described in Table 4.3. Different from Case 1, the vector $\mathbf{u}$ appears *after* $\mathbf{v}_j$ in the experiment.

- H_1^b proceeds identically to H_0^b, except that it removes the coefficients of the first label function $\mathbf{L}_1$ from $\mathbf{v}_1[\mathsf{const}], \mathbf{v}_1[\mathsf{coef}_i]$ and hardwires the honest first label $\ell_1 = L_1(\mathbf{x})$ into $\mathbf{u}[\mathsf{sim}_1]$, as described in Table 4.3. Since the inner product remains the same as in H_0^b, by the function-hiding property of IPFE, we have $\mathsf{H}_0^b \approx \mathsf{H}_1^b$.

- H_2^b proceeds identically to H_1^b, except that the hardwired first label is now reversely sampled as

$$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m),$$

  where $\ell_j = L_j(\mathbf{x})$ for $j > 1$ are the honest labels. By the reverse sampleability of AKGS (Definition 4.10), we have $\mathsf{H}_1^b \equiv \mathsf{H}_2^b$.

The final hybrid is H_4^b, which proceeds identically to H_2^b, except that for all $j > 1$, the coefficient vector $\mathbf{L}_j$ of the label function L_j is removed from $\mathbf{v}_j$ and a simulated label $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ is hardwired in $\mathbf{v}_j[\mathsf{const}]$, as depicted in Table 4.3. Correspondingly, the first label ℓ_1 is reversely sampled using these simulated labels. The hybrid is similar to H_2^b in Case 1, except for the location where ℓ_1 is embedded. However, we cannot move from H_2^b to H_4^b in one shot. Instead, we go through a loop consisting of $\mathsf{H}_{3,j,1}^b, \ldots, \mathsf{H}_{3,j,4}^b$ for $1 < j \le m$.

- $\mathsf{H}_{3,j,1}^b$ proceeds identically to H_2^b, except that for all j' such that $1 < j' < j$, the coefficient vector $\mathbf{L}_{j'}$ of the label function $L_{j'}$ is removed from $\mathbf{v}_{j'}$ and a simulated label $\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$ is hardwired in $\mathbf{v}_{j'}[\mathsf{const}]$, as described in Table 4.4. In this hybrid, ℓ_1 is reversely sampled using simulated $\ell_2, \ldots, \ell_{j-1}$ and honest $\ell_j, \ldots, \ell_m$.

- $\mathsf{H}_{3,j,2}^b$ proceeds identically to $\mathsf{H}_{3,j,1}^b$, except that it removes the coefficient vector of the j^{th} label function L_j from $\mathbf{v}_j$ and hardwires the honest j^{th} label $\ell_j = L_j(\mathbf{x})$

Table 4.4. Hybrids, 1-ABE for ABP, Case 2 (f then $\mathbf{x}$), loop (middle).

hybrid	vector		const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathsf{H}^b_{3,j,1}$		$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$:	$\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
		$\mathbf{v}_j$	$\boxed{\mathbf{L}_j[\mathsf{const}]}$	$\boxed{\mathbf{L}_j[\mathsf{coef}_i]}$	0	$\boxed{0}$
	$j' > j$:	$\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
		$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	$\boxed{0}$
$\mathsf{H}^b_{3,j,2}$		$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$:	$\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
		$\mathbf{v}_j$	$\boxed{0}$	$\boxed{0}$	0	$\boxed{1}$
	$j' > j$:	$\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
		$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	$\boxed{\ell_j = L_j(\mathbf{x})}$
$\mathsf{H}^b_{3,j,3}$		$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$:	$\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
		$\mathbf{v}_j$	$\boxed{0}$	0	0	$\boxed{1}$
	$j' > j$:	$\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
		$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	$\boxed{\ell_j \xleftarrow{\$} \mathbb{Z}_p}$
$\mathsf{H}^b_{3,j,4} \equiv \mathsf{H}^b_{3,j+1,1}$		$\mathbf{v}_1$	0	0	1	0
	$1 < j' < j$:	$\mathbf{v}_{j'}$	$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$	0	0	0
		$\mathbf{v}_j$	$\boxed{\ell_j \xleftarrow{\$} \mathbb{Z}_p}$	0	0	$\boxed{0}$
	$j' > j$:	$\mathbf{v}_{j'}$	$\mathbf{L}_{j'}[\mathsf{const}]$	$\mathbf{L}_{j'}[\mathsf{coef}_i]$	0	0
		$\mathbf{u}$	1	$\mathbf{x}[i]$	ℓ_1	$\boxed{0}$

In this iteration, the labels $\ell_{j'}$ are...

- $(j' = 1)$ simulated as $\mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$ and hardwired in ct;
- $(1 < j' < j)$ simulated as random and hardwired in sk;
- $(j' = j)$ computed as $L_j(\mathbf{x})$ using IPFE
  - → computed as $L_j(\mathbf{x})$ and hardwired in ct
  - → simulated as random and hardwired in ct
  - → simulated as random and hardwired in sk;
- $(j' > j)$ computed as $L_{j'}(\mathbf{x})$ using IPFE.

in $\mathbf{u}[\mathsf{sim}_\star]$, i.e., $\mathbf{L}_j$ is gone in $\mathsf{H}^b_{3,j,2}$ and the honest label ℓ_j is left. The modified vectors $\mathbf{v}_j$ and $\mathbf{u}$ are described in Table 4.4. Since the inner products remain the same, by the function-hiding property of IPFE, we have $\mathsf{H}^b_{3,j,1} \approx \mathsf{H}^b_{3,j,2}$.

- $\mathsf{H}^b_{3,j,3}$ proceeds identically to $\mathsf{H}^b_{3,j,2}$, except that the j^{th} label is simulated as $\ell_j \xleftarrow{\$} \mathbb{Z}_p$. In $\mathsf{H}^b_{3,j,2}$ and $\mathsf{H}^b_{3,j,3}$, only the coefficient vectors $\mathbf{L}_{j'}$ of the label functions $L_{j'}$ for $j' > j$ appear, whereas for $1 < j' < j$, a simulated label $\ell_{j'}$ is hardwired in $\mathbf{v}_{j'}[\mathsf{const}]$. In both hybrids, the first label ℓ_1 can be efficiently reversely sampled from $\ell_2, \ldots, \ell_{j-1}$ (simulated), ℓ_j (honest or simulated) and $\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$ (producing honest labels). Therefore, by the marginal randomness property of AKGS (Definition 4.10), we have $\mathsf{H}^b_{3,j,2} \equiv \mathsf{H}^b_{3,j,3}$.

- $\mathsf{H}^b_{3,j,4}$ proceeds identically to $\mathsf{H}^b_{3,j,3}$, except that the simulated j^{th} label ℓ_j is moved from $\mathbf{u}[\mathsf{sim}_\star]$ to $\mathbf{v}_j[\mathsf{const}]$ as described in Table 4.4 (so that $\mathbf{u}[\mathsf{sim}_\star]$ is free to be used in the next iteration of hybrids from $\mathsf{H}^b_{3,j+1,1}$ to $\mathsf{H}^b_{3,j+1,4}$). Since the inner products remain the same, by the function-hiding property of IPFE, we have $\mathsf{H}^b_{3,j,3} \approx \mathsf{H}^b_{3,j,4}$. In addition, observe that $\mathsf{H}^b_{3,j,4} \equiv \mathsf{H}^b_{3,j+1,1}$.

Moreover, $\mathsf{H}^b_2 \equiv \mathsf{H}^b_{3,2,1}$ and $\mathsf{H}^b_4 \equiv \mathsf{H}^b_{3,m,4}$. Therefore, $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ by hybrid argument. We further argue $\mathsf{H}^0_4 \equiv \mathsf{H}^1_4$. Observe that in both hybrids, the labels are simulated as

$$\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p \quad \text{for } j' = 2, \ldots, m, \qquad \ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m).$$

In the two cases where decryption is not authorized, we have

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} 0, & \text{if } f' = f_{\neq 0} \text{ and } f(\mathbf{x}) = 0; \\ \eta f(\mathbf{x}) + \mu^0 \quad (\eta \xleftarrow{\$} \mathbb{Z}_p), & \text{if } f' = f_{=0} \text{ and } f(\mathbf{x}) \neq 0. \end{cases}$$

Either way, $(\alpha f(\mathbf{x}) + \beta)$ is independent of μ^0, hence μ^0 is a uniform random value independent of everything else in H^0_4 — so is μ^1 in H^1_4. In addition, everything except μ^0, μ^1 are identically distributed in H^0_4 and H^1_4. Therefore, they are identical.

By hybrid argument, we conclude $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$ conditioned on Case 2. □

4.5.3 KP-ABE for ABP

We now lift the 1-ABE scheme to a full-fledged ABE by implementing the ideas discussed in Section 4.2.2.

Ingredients of Construction 4.3. Let

- (Garble, Eval) be an AKGS (Definitions 4.7 and 4.8) for $\mathcal{F}$,
- GroupGen a pairing group sampler (Definition 2.5) for which MDDH_k (Assumption 2.1) holds over $\mathbb{G}_2$, and
- (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec, IPFE.SlotEnc) a slotted IPFE (Definition 4.4) based on GroupGen.

Construction 4.3 (KP-ABE). Define (see Definition 2.3)

$$\mathsf{Params}_{\mathsf{gp}} = \{\, 1^n \mid n \in \mathbb{N} \,\}, \qquad F_{\mathsf{gp},1^n} = \{\, f_{\neq 0}, f_{=0} \mid f \in \mathcal{F},\ f : \mathbb{Z}_p^n \to \mathbb{Z}_p \,\},$$
$$X_{\mathsf{gp},1^n} = \mathbb{Z}_p^n, \qquad Y_{\mathsf{gp},1^n} = \mathbb{G}_T, \qquad P_{\mathsf{gp},1^n}(f', \mathbf{x}) = f'(\mathbf{x}),$$

where p is the group order in gp. Our KP-ABE scheme based on GroupGen operates as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^n)$ takes as input the group description gp and the attribute length n in unary. It sets up the IPFE $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, \mathfrak{s}_{\mathrm{pub}}, \mathfrak{s}_{\mathrm{priv}})$ for

  $$\mathfrak{s}_{\mathrm{pub}} = \{\, \mathsf{pad}, \mathsf{const}^t, \mathsf{coef}_i^t \mid t \in [k], i \in [n] \,\},$$
  $$\mathfrak{s}_{\mathrm{priv}} = \{\, \mathsf{const}, \mathsf{coef}_i, \mathsf{sim}_1, \mathsf{sim}_\star \mid i \in [n] \,\} \quad (= \mathfrak{s}_{\text{1-ABE}}),$$

  where $\mathfrak{s}_{\text{1-ABE}}$ is defined in Construction 4.2. The algorithm outputs $\mathsf{mpk} = \mathsf{impk}$ and $\mathsf{msk} = \mathsf{imsk}$.

  The names of the indices are as follows. For each $t \in [k]$, the positions indexed by $\mathsf{const}^t, \mathsf{coef}_1^t, \ldots, \mathsf{coef}_n^t$ encode the coefficient vectors $\mathbf{L}_j^t$ of the label functions of one garbling in the secret key, and encrypt a random multiple of $(1, \mathbf{x}^\intercal)$ in the ciphertext. The position indexed by pad encodes 1 in the secret key and encrypts a random pad $h \xleftarrow{\$} \mathbb{Z}_p$ in the ciphertext. The private slot is reserved for the security proof and values at those indices are set to 0 by the honest algorithms.

- $\mathsf{KeyGen}(\mathsf{msk}, f')$ takes as input msk and policy $f' \in F_{\mathsf{gp},1^n}$. It samples $\boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\boldsymbol{\eta} \xleftarrow{\$} \mathbb{Z}_p^k$, and creates k garblings of the function f underlying f' by

$$\begin{cases} \boldsymbol{\alpha} \leftarrow \boldsymbol{\mu}, \quad \boldsymbol{\beta} \leftarrow \mathbf{0}, & \text{if } f' = f_{\neq 0}; \\ \boldsymbol{\alpha} \leftarrow \boldsymbol{\eta}, \quad \boldsymbol{\beta} \leftarrow \boldsymbol{\mu}, & \text{if } f' = f_{=0}; \end{cases}$$
$$(\text{for } t \in [k]) \qquad (\mathbf{L}_1^t, \ldots, \mathbf{L}_m^t) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}[t], \boldsymbol{\beta}[t]; \mathbf{r}_t).$$

  The randomness vectors $\mathbf{r}_1, \ldots, \mathbf{r}_k$ for garbling f are independently sampled and written explicitly. The algorithm sets the vector $\mathbf{v}_{\mathsf{pad}} \in \mathbb{Z}_p^{\mathsf{s}}$ to encode the random pads $\boldsymbol{\mu}$ and the vector $\mathbf{v}_j \in \mathbb{Z}_p^{\mathsf{s}}$ for $j \in [m]$ to encode the j^{th} label function in all the k garblings as follows:

vector	pad	const^t	coef_i^t	in $\mathsf{s}_{\mathsf{priv}}$
$\mathbf{v}_{\mathsf{pad}}$	1	$\boldsymbol{\mu}[t]$	0	0
$\mathbf{v}_j$	0	$\mathbf{L}_j^t[\mathsf{const}]$	$\mathbf{L}_j^t[\mathsf{coef}_i]$	

  It then generates IPFE secret keys for them by

$$\mathsf{isk}_{\mathsf{pad}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_{\mathsf{pad}}]\!]_2),$$
$$(\text{for } j \in [m]) \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2).$$

  The algorithm outputs $\mathsf{sk} = (\mathsf{isk}_{\mathsf{pad}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, g)$ takes as input mpk, attribute $\mathbf{x} \in \mathbb{Z}_p^n$, and message $g \in \mathbb{G}_{\mathrm{T}}$. It samples a random pad $h \xleftarrow{\$} \mathbb{Z}_p$ and random multipliers $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$, and sets the vector $\mathbf{u} \in \mathbb{Z}_p^{\mathsf{s}_{\mathsf{pub}}}$ to encode them as follows:

vector	pad	const^t	coef_i^t
$\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$

  The algorithm generates an IPFE ciphertext $\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{impk}, [\![\mathbf{u}]\!]_1)$ and outputs $\mathsf{ct} = (g + [\![h]\!]_{\mathrm{T}}, \mathsf{ict})$.

- $\mathsf{Dec}(\mathsf{mpk}, f', \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ outputs $\perp$ if $f'(\mathbf{x}) = 0$. Otherwise, it parses sk as in KeyGen and ct as $([\![z]\!]_{\mathrm{T}}, \mathsf{ict})$, and runs

$$\begin{aligned} & [\![z']\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{pad}}, \mathsf{ict}), \\ (\text{for } j \in [m]) \qquad & [\![\ell_j]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}), \\ & [\![\mu']\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } f' = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } f' = f_{=0}. \end{cases} \end{aligned}$$

  The algorithm computes and outputs $([\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}})$.

Correctness. To verify correctness, consider the random linear combination $\overline{L}_j$ of label functions $L_j^1, \ldots, L_j^k$ with weights $\mathbf{s}$,

$$(\text{for } j \in [m]) \qquad \overline{L}_j(\mathbf{x}) = \sum_{t \in [k]} \mathbf{s}[t] L_j^t(\mathbf{x}), \qquad \overline{\mathbf{L}}_j = \sum_{t \in [k]} \mathbf{s}[t] \mathbf{L}_j^t.$$

The coefficient vector of $\overline{L}_j$ is exactly $\overline{\mathbf{L}}_j$ define above. Furthermore, by the linearity of Garble, we have $(\overline{\mathbf{L}}_1, \ldots, \overline{\mathbf{L}}_m) = \mathsf{Garble}(f, \overline{\alpha}, \overline{\beta}; \overline{\mathbf{r}})$, where $\overline{\alpha}, \overline{\beta}, \overline{\mathbf{r}}$ are the random linear combinations (with the same weights $\mathbf{s}$) of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \{\mathbf{r}_t\}_{t \in [k]}$ in the k garblings, i.e.,

$$\overline{\alpha} = \langle \mathbf{s}, \boldsymbol{\alpha} \rangle, \qquad \overline{\beta} = \langle \mathbf{s}, \boldsymbol{\beta} \rangle, \qquad \overline{\mathbf{r}} = \sum_{t \in [k]} \mathbf{s}[t] \mathbf{r}_t.$$

Observe that by the definition of $\mathbf{v}_{\mathrm{pad}}, \mathbf{v}_j, \mathbf{u}$ and the correctness of the IPFE scheme,

$$\begin{aligned} & z' = \langle \mathbf{u}, \mathbf{v}_{\mathrm{pad}} \rangle = h + \langle \mathbf{s}, \boldsymbol{\mu} \rangle = h + \overline{\mu}, \\ (\text{for } j \in [m]) \qquad & \ell_j = \langle \mathbf{u}, \mathbf{v}_j \rangle = \sum_{t \in [k]} \mathbf{s}[t] L_j^t(\mathbf{x}) = \overline{L}_j(\mathbf{x}), \end{aligned}$$

where $\overline{\mu} = \langle \mathbf{s}, \boldsymbol{\mu} \rangle$ is the linear combination of $\boldsymbol{\mu}$ by $\mathbf{s}$. By the linearity of Eval in $\ell_1, \ldots, \ell_m$, we can evaluate the garbling in the exponent of the target group and obtain $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \overline{\alpha} f(\mathbf{x}) + \overline{\beta}$ in the exponent. In the two cases where decryption should succeed, we have

$$\overline{\alpha} f(\mathbf{x}) + \overline{\beta} = \begin{cases} \overline{\alpha} f(\mathbf{x}) = \overline{\mu} f(\mathbf{x}), & \text{if } f' = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0; \\ \overline{\beta} = \overline{\mu}, & \text{if } f' = f_{=0} \text{ and } f(\mathbf{x}) = 0. \end{cases}$$

Either way, $\mu' = \overline{\mu}$, hence $[\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}} = (g + [\![h]\!]_{\mathrm{T}}) + [\![\overline{\mu}]\!]_{\mathrm{T}} - [\![h + \overline{\mu}]\!]_{\mathrm{T}} = g$, i.e., Dec correctly recovers the message.

Security. We state and prove the security of our ABE construction.

Theorem 4.8 (¶). *Suppose in Construction 4.3, the AKGS is* piecewise secure *(Definition 4.10),* MDDH_k *(Assumption 2.1) holds over* $\mathbb{G}_2$*, and the slotted IPFE scheme is* function-hiding *(Definition 4.5), then the resultant ABE scheme is* adaptively secure *(Definition 2.2).*

Proof (Theorem 4.8). The statement is $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$. Recall that in $\mathsf{Exp}^b_{\mathrm{PHFE}}$, the adversary can receive many secret keys $\{\mathsf{sk}_q\}_{q\in[Q]}$ for different policies $\{f'_q\}_q$ and a ciphertext ct for attribute $\mathbf{x}$ and message g_b, where $\{f'_q\}_q, \mathbf{x}, g_0, g_1$ are chosen adaptively by the adversary subject to $f'_q(\mathbf{x}) = 0$ for all $q \in [Q]$. (If the constraint is not satisfied, the outcome of the experiment is set to 0.)

We start with the high-level ideas of the proof. By construction, ct and sk_q contain respectively a slotted IPFE ciphertext ict and many IPFE keys $\mathsf{isk}_{q,\mathrm{pad}}, \{\mathsf{isk}_{q,j}\}_{j\in[m_q]}$ such that IPFE decryption computes in the exponent *i)* the sum $(h + \overline{\mu}_q)$ and *ii)* the labels $\{\overline{L}_{q,j}(\mathbf{x})\}_{j\in[m_q]}$ revealing a random pad $\overline{\mu}_q$ from the evaluation result $(\overline{\alpha}_q f_q(\mathbf{x}) + \overline{\beta}_q)$ if the policy is satisfied. Revealing any $[\![\overline{\mu}_q]\!]_{\mathrm{T}}$ allows one to recover the message g_b from $(g_b + [\![h]\!]_{\mathrm{T}})$ in ct. Therefore, to show $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$, we want to argue that when no policy is satisfied, all the pads $\overline{\mu}_q$ are pseudorandom hence hide h.

To do so, the proof proceeds in three steps via a sequence of hybrids. In the first step (from $\mathsf{H}^b_0 \equiv \mathsf{Exp}^b_{\mathrm{PHFE}}$ to H^b_3), in the *private* slot, we embed the vector $(1, \mathbf{x}^\intercal)$ into ict, *randomly and independently sampled* pads $\widehat{\mu}_q$ into $\mathsf{isk}_{q,\mathrm{pad}}$'s, and *randomly and independently generated* label functions $\widehat{L}_{q,j}$ for $\widehat{\mu}_q$'s into $\mathsf{isk}_{q,j}$'s. In the second step (from $\mathsf{H}^b_{4,1} \equiv \mathsf{H}^b_3$ to $\mathsf{H}^b_{4,Q+1} \equiv \mathsf{H}^b_5$), we observe that the *private* slots of ict and $\mathsf{isk}_{q,j}$'s match exactly the ciphertext and secret keys of Construction 4.2, our 1-ABE scheme (i.e., they encode the same vectors and have the function-hiding property). Therefore, by the same argument as the security proof of 1-ABE, we can switch the random pads $\widehat{\mu}_q$ encoded in $\mathsf{isk}_{q,\mathrm{pad}}$'s to independent random values $\widehat{\mu}'_q$. Lastly, in the third step (from H^b_5 to H^b_6), we remove $h, \{\widehat{\mu}'_q\}_q$ from ict, $\{\mathsf{isk}_{q,\mathrm{pad}}\}_q$, and embed $\{h + \widehat{\mu}'_q\}_q$ in $\mathsf{isk}_{q,\mathrm{pad}}$'s. This allows us to argue that h is hidden by $\{\widehat{\mu}'_q\}_q$ and consequently, that the message g_b is hidden.

We now describe the hybrids formally.

- H_0^b is $\mathsf{Exp}_{\mathrm{PHFE}}^b$, where $\mathbf{u}$ encrypted in the IPFE ciphertext ict (in ct) and $\mathbf{v}_{q,\mathrm{pad}}, \mathbf{v}_{q,j}$ encoded in the IPFE secret keys $\mathsf{isk}_{q,\mathrm{pad}}, \mathsf{isk}_{q,j}$ (in sk_q) are described in Table 4.5. In particular, note that the values in the private slot of $\mathbf{u}$ are implicitly set to 0 and ict is generated using $\mathsf{IPFE.SlotEnc}$ (depicted as $\perp$ in the private slot of $\mathbf{u}$).

- H_1^b proceeds identically to H_0^b, except that the IPFE ciphertext ict is generated using $\mathsf{IPFE.Enc}$ with values in the private slot explicitly set to 0. In Table 4.5, the "$\perp$" values in H_0^b change into 0 in H_1^b. By the *slot-mode* correctness of the slotted IPFE scheme, $\mathsf{H}_0^b \equiv \mathsf{H}_1^b$.

- H_2^b proceeds identically to H_1^b, except that how the pad $\overline{\mu}_q$ and the labels $\overline{L}_{q,j}(\mathbf{x})$ for $q \in [Q]$, $j \in [m_q]$ are computed (as inner products) is changed — instead of combining k independent garblings (in the secret keys) with weights $\mathbf{s}$ (in the ciphertext), we put the k independent garblings as well as their linear combinations in the secret keys, and the ciphertext is set to directly use *that* linear combination.

  In H_1^b (the same as in $\mathsf{Exp}_{\mathrm{PHFE}}^b$), $\overline{\mu}_q$ and $\overline{L}_{q,j}(\mathbf{x})$ are linear combinations of $\{\boldsymbol{\mu}_q[t]\}_{t\in[k]}$ and $\{L_{q,j}^t(\mathbf{x})\}_{t\in[k]}$ with weights $\mathbf{s} \in \mathbb{Z}_p^k$, computed using IPFE as

$$\overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle, \qquad \overline{L}_{q,j}(\mathbf{x}) = \left(\mathbf{s}[1](1, \mathbf{x}^\top), \ldots, \mathbf{s}[k](1, \mathbf{x}^\top)\right) \begin{pmatrix} \mathbf{L}_{q,j}^1 \\ \vdots \\ \mathbf{L}_{q,j}^k \end{pmatrix},$$

  where the random multiples $\left(\mathbf{s}[1](1, \mathbf{x}^\top), \ldots, \mathbf{s}[k](1, \mathbf{x}^\top)\right)$ of the attribute are in the public slot of $\mathbf{u}$, the pads $\boldsymbol{\mu}_q$ in the public slots of $\mathbf{v}_{q,\mathrm{pad}}$'s, and the coefficients of the k label functions in the public slots of $\mathbf{v}_{q,j}$'s. In H_2^b, they are instead computed as

$$\overline{\mu}_q = \langle 1, \overline{\mu}_q \rangle, \qquad \overline{L}_{q,j}(\mathbf{x}) = \langle (1, \mathbf{x}^\top)^\top, \overline{\mathbf{L}}_{q,j} \rangle, \qquad \overline{\mathbf{L}}_{q,j} = \sum_{t\in[k]} \mathbf{s}[t] \mathbf{L}_{q,j}^t,$$

  where $(1, \mathbf{x}^\top)$ is in the *private* slot of $\mathbf{u}$, the combined pad $\overline{\mu}_q$ in the *private* slots of $\mathbf{v}_{q,\mathrm{pad}}$'s, and the combined label functions $\overline{\mathbf{L}}_{q,j}$ in the *private* slots of $\mathbf{v}_{q,j}$'s. Furthermore, to keep the inner products $\langle \mathbf{u}, \mathbf{v}_{q,\mathrm{pad}} \rangle, \langle \mathbf{u}, \mathbf{v}_{q,j} \rangle$ the same as those in H_1^b, the random multiples of the attribute $\left(\mathbf{s}[1](1, \mathbf{x}^\top), \ldots, \mathbf{s}[k](1, \mathbf{x}^\top)\right)$ in the

Table 4.5. Hybrids, KP-ABE for ABP, first and final.

hybrid	vector	pad	const^t	coef$_i^t$	const	coef$_i$	sim$_1$, sim$_\star$
$\mathsf{H}_0^b \equiv \mathsf{Exp}^b_{\mathrm{PHFE}}$	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	0	0	0, 0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}^t_{q,j}[\mathsf{const}]$	$\mathbf{L}^t_{q,j}[\mathsf{coef}_i]$	0	0	0, 0
	ct: $\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$	$\perp$	$\perp$	$\perp, \perp$
H_1^b	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	0	0	0, 0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}^t_{q,j}[\mathsf{const}]$	$\mathbf{L}^t_{q,j}[\mathsf{coef}_i]$	0	0	0, 0
	ct: $\mathbf{u}$	h	$\mathbf{s}[t]$	$\mathbf{s}[t]\mathbf{x}[i]$	0	0	0, 0
H_2^b	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\overline{\mu}_q$	0	0, 0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}^t_{q,j}[\mathsf{const}]$	$\mathbf{L}^t_{q,j}[\mathsf{coef}_i]$	$\overline{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\overline{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0, 0
	ct: $\mathbf{u}$	h	0	0	1	$\mathbf{x}[i]$	0, 0
$\mathsf{H}_3^b \equiv \mathsf{H}_{4,1}^b$	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\widehat{\mu}_q$	0	0, 0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}^t_{q,j}[\mathsf{const}]$	$\mathbf{L}^t_{q,j}[\mathsf{coef}_i]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0, 0
	ct: $\mathbf{u}$	h	0	0	1	$\mathbf{x}[i]$	0, 0
$\mathsf{H}^b_{4,1\sim Q+1}$	.....						
$\mathsf{H}_5^b \equiv \mathsf{H}_{4,Q+1}^b$	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$\widehat{\mu}'_q \xleftarrow{\$} \mathbb{Z}_p$	0	0, 0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}^t_{q,j}[\mathsf{const}]$	$\mathbf{L}^t_{q,j}[\mathsf{coef}_i]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0, 0
	ct: $\mathbf{u}$	h	0	0	1	$\mathbf{x}[i]$	0, 0
H_6^b	sk_q: $\mathbf{v}_{q,\mathrm{pad}}$	1	$\boldsymbol{\mu}_q[t]$	0	$h + \widehat{\mu}'_q$	0	0, 0
	sk_q: $\mathbf{v}_{q,j}$	0	$\mathbf{L}^t_{q,j}[\mathsf{const}]$	$\mathbf{L}^t_{q,j}[\mathsf{coef}_i]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{const}]$	$\widehat{\mathbf{L}}_{q,j}[\mathsf{coef}_i]$	0, 0
	ct: $\mathbf{u}$	0	0	0	1	$\mathbf{x}[i]$	0, 0

The hybrids are illustrated with the ciphertext challenge coming after the key queries (as in the more difficult case for 1-ABE). In reality, the key queries can be made before and after the ciphertext challenge, and the proof is unaffected.

In H_2^b, the linearly combined $\overline{\mathbf{L}}_{q,j} = \sum_{t\in[k]} \mathbf{s}[t]\mathbf{L}^t_{q,j}$ are valid garblings for the linearly combined pads $\overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle$.

In H_3^b, the garblings $\widehat{\mathbf{L}}_{q,j}$ are fresh and are for freshly sampled pads $\widehat{\mu}_q$.

In $\mathsf{H}_5^b, \mathsf{H}_6^b$, the pads $\widehat{\mu}'_q$ in $\mathsf{isk}_{q,\mathrm{pad}}$'s are independent of the pads $\widehat{\mu}_q$ being garbled.

public slot of $\mathbf{u}$ are replaced by zero (the public slots of $\mathbf{v}_{q,\mathrm{pad}}, \mathbf{v}_{q,j}$ remain unchanged). By the function-hiding property of IPFE, $\mathsf{H}_1^b \approx \mathsf{H}_2^b$.

- H_3^b proceeds identically to H_2^b, except that the random linear combinations $\overline{\mu}_q, \overline{\mathbf{L}}_{q,j}$ in the private slots of the secret keys are replaced by randomly and independently generated pads and garblings $\widehat{\mu}_q, \widehat{\mathbf{L}}_{q,j}$, as shown in Table 4.5.

In $\mathsf{H}_2^b, \mathsf{H}_3^b$, the k instances of garblings in the public slot are generated by

$$\boldsymbol{\mu}_q \xleftarrow{\$} \mathbb{Z}_p^k, \qquad \boldsymbol{\eta}_q \xleftarrow{\$} \mathbb{Z}_p^k, \qquad \begin{cases} \boldsymbol{\alpha}_q \leftarrow \boldsymbol{\mu}_q, \quad \boldsymbol{\beta}_q \leftarrow \mathbf{0}, & \text{if } f'_q = f_{q,\neq 0}; \\ \boldsymbol{\alpha}_q \leftarrow \boldsymbol{\eta}_q, \quad \boldsymbol{\beta}_q \leftarrow \boldsymbol{\mu}_q, & \text{if } f'_q = f_{q,=0}; \end{cases}$$

$$(\text{for } t \in [k]) \qquad (\mathbf{L}_{q,1}^t, \ldots, \mathbf{L}_{q,m_q}^t) \leftarrow \mathsf{Garble}(f_q, \boldsymbol{\alpha}_q[t], \boldsymbol{\beta}_q[t]; \mathbf{r}_{q,t}).$$

In H_2^b, the private slots of the keys contain linear combinations of the garblings. By the linearity of AKGS, they can also be generated as

$$\begin{cases} \overline{\alpha}_q = \langle \mathbf{s}, \boldsymbol{\alpha}_q \rangle = \overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle, & \overline{\beta}_q = \langle \mathbf{s}, \boldsymbol{\beta}_q \rangle = \ 0 \ = \langle \mathbf{s}, \mathbf{0} \rangle, & \text{if } f'_q = f_{q,\neq 0}; \\ \overline{\alpha}_q = \langle \mathbf{s}, \boldsymbol{\alpha}_q \rangle = \overline{\eta}_q = \langle \mathbf{s}, \boldsymbol{\eta}_q \rangle, & \overline{\beta}_q = \langle \mathbf{s}, \boldsymbol{\beta}_q \rangle = \overline{\mu}_q = \langle \mathbf{s}, \boldsymbol{\mu}_q \rangle, & \text{if } f'_q = f_{q,=0}; \end{cases}$$

$$\mathbf{r}_q = \sum_{t \in [k]} \mathbf{s}[t] \mathbf{r}_{q,t}, \qquad (\overline{\mathbf{L}}_{q,1}, \ldots, \overline{\mathbf{L}}_{q,m_q}) \leftarrow \mathsf{Garble}(f_q, \overline{\alpha}_q, \overline{\beta}_q; \overline{\mathbf{r}}_q).$$

In H_3^b, the $\widehat{\mathbf{L}}_{q,j}$'s are generated for randomly and independently sampled pad $\widehat{\mu}_q$ with secrets $\widehat{\alpha}_q, \widehat{\beta}_q$ set according to $\widehat{\mu}_q$ and independent randomness $\widehat{\mathbf{r}}_q$:

$$\widehat{\mu}_q \xleftarrow{\$} \mathbb{Z}_p, \qquad \widehat{\eta}_q \xleftarrow{\$} \mathbb{Z}_p, \qquad \begin{cases} \widehat{\alpha}_q \leftarrow \widehat{\mu}_q, \quad \widehat{\beta}_q \leftarrow 0, & \text{if } f'_q = f_{q,\neq 0}; \\ \widehat{\alpha}_q \leftarrow \widehat{\eta}_q, \quad \widehat{\beta}_q \leftarrow \widehat{\mu}_q, & \text{if } f'_q = f_{q,=0}; \end{cases}$$

$$(\widehat{\mathbf{L}}_{q,1}, \ldots, \widehat{\mathbf{L}}_{q,m_q}) \leftarrow \mathsf{Garble}(f_q, \widehat{\alpha}_q, \widehat{\beta}_q; \widehat{\mathbf{r}}_q).$$

Note that the only difference between H_2^b and H_3^b is that the former uses linear combinations $\overline{\mu}_q, \overline{\eta}_q, \overline{\mathbf{r}}_q$ to set the secrets and the randomness for the garbling in the private slot, whereas the latter uses fresh randomness. In H_2^b, the weights $\mathbf{s}$ and $\boldsymbol{\mu}_q, \boldsymbol{\eta}_q, \mathbf{r}_{q,t}$ are only used in the exponent of $\mathbb{G}_2$, since they are only encoded in IPFE secret keys. Therefore, by MDDH_k (Assumption 2.1), $\overline{\mu}_q, \overline{\eta}_q, \overline{\mathbf{r}}_q$ are pseudorandom in the exponent of $\mathbb{G}_2$, i.e.,

$$[\![\underbrace{\{\boldsymbol{\mu}_q, \boldsymbol{\eta}_q, \{\mathbf{r}_{q,t}\}_{t\in[k]}\}_{q\in[Q]}}_{\mathbf{A}}, \underbrace{\{\overline{\mu}_q, \overline{\eta}_q, \overline{\mathbf{r}}_q\}_{q\in[Q]}}_{\mathbf{s}^\top \mathbf{A}}]\!]_2 \approx [\![\mathbf{A}, \underbrace{\{\widehat{\mu}_q, \widehat{\eta}_q, \widehat{\mathbf{r}}_q\}_{q\in[Q]}}_{\mathbf{c}^\top}]\!]_2.$$

Furthermore, by the linearity of Garble, given $\mathbf{A}$ and $\mathbf{s}^\intercal\mathbf{A}$ or $\mathbf{c}^\intercal$ encoded in $\mathbb{G}_2$, the hybrids H_2^b and H_3^b can be efficiently generated. Therefore, $\mathsf{H}_2^b \approx \mathsf{H}_3^b$.

We have completed the first step of the proof. Observe that in H_3^b, for each $q \in [Q]$, the private slots of $\{\mathsf{isk}_{q,j}\}_{j\in[m_q]}$ encode (the coefficients of) label functions $\widehat{L}_{q,j}$ and the private slot of ict encodes $(1, \mathbf{x}^\intercal)$, identical to our 1-ABE scheme (Construction 4.2). The security of 1-ABE (Theorem 4.7) ensures that the adversary cannot distinguish the pad $\widehat{\mu}_q$ encoded in the secret key from another independent pad $\widehat{\mu}'_q$ (inconsistent with $\widehat{L}_{q,j}$'s). Next, in $\mathsf{H}_{4,1}^b, \ldots, \mathsf{H}_{4,Q+1}^b$, we use syntactically the same proof to gradually switch each $\widehat{\mu}_q$ to $\widehat{\mu}'_q$.

- $\mathsf{H}_{4,q}^b$ proceeds identically to H_3^b, except that in the first $(q-1)$ secret keys, the random pads $\widehat{\mu}_{q'}$ (for $q' < q$) in the private slots of $\mathbf{v}_{q',\mathrm{pad}}$ are replaced by an independent random value $\widehat{\mu}'_{q'}$, as illustrated in Table 4.6. The only difference between $\mathsf{H}_{4,q}^b$ and $\mathsf{H}_{4,q+1}^b$ is whether $\mathbf{v}_{q,\mathrm{pad}}$ contains the pad $\widehat{\mu}_q$ consistent with $\widehat{L}_{q,j}$'s or an independent one $\widehat{\mu}'_q$.

Analogously to the proof of 1-ABE security (Theorem 4.7), we have $\mathsf{H}_{4,q}^b \approx \mathsf{H}_{4,q+1}^b$. Recall that in the more difficult case (f_q before $\mathbf{x}$), it goes through a series of hybrids where the coefficients of the label functions $\widehat{L}_{q,j}$ are gradually removed from $\mathbf{v}_{q,j}|_{\mathsf{s}_{1\text{-ABE}}}$ and replaced by a simulated label hardwired in either $\mathbf{v}_{q,j}[\mathsf{const}]$ (using $\mathbf{u}[\mathsf{sim}_\star]$ as a relay) or $\mathbf{u}[\mathsf{sim}_1]$, and that simulated labels are sampled at random $\ell_{q,j} \xleftarrow{\$} \mathbb{Z}_p$ (for $j > 1$) except that the first one is reversely sampled

$$\ell_{q,1} \xleftarrow{\$} \mathsf{RevSamp}(f_q, \mathbf{x}, \widehat{\alpha}_q f_q(\mathbf{x}) + \widehat{\beta}_q, \ell_{q,2}, \ldots, \ell_{q,m_q}).$$

If decryption is not authorized, $(\widehat{\alpha}_q f_q(\mathbf{x}) + \widehat{\beta}_q)$ perfectly hides $\widehat{\mu}_q$, so $\widehat{\mu}_q$ can be switched to $\widehat{\mu}'_q$. The indistinguishability of neighboring hybrids follows from either AKGS piecewise security or the function-hiding property of IPFE.

The only differences between 1-ABE security proof and the transition from $\mathsf{H}_{4,q}^b$ to $\mathsf{H}_{4,q+1}^b$ are that in the latter, IPFE secret keys and ciphertext encode vectors with additional public slots, and that there are many (namely, Q) ABE secret keys. They do not affect the application of AKGS piecewise security nor the function-hiding property of IPFE. First, the piecewise security of AKGS is information-theoretic — given that

Table 4.6. Hybrids, KP-ABE for ABP, loop (middle).

hybrid	vector	in $\mathfrak{s}_{\text{pub}}$	const	coef_i	$\text{sim}_1, \text{sim}_\star$
$\mathsf{H}^b_{4,q}$	$q' < q$ $\begin{cases}\mathbf{v}_{q',\text{pad}}\\ \mathbf{v}_{q',j}\end{cases}$	normal	$\widehat{\mu}'_{q'} \xleftarrow{\$} \mathbb{Z}_p$	0	$0, 0$
			$\widehat{\mathbf{L}}_{q',j}[\text{const}]$	$\widehat{\mathbf{L}}_{q',j}[\text{coef}_i]$	$0, 0$
	sk_q $\begin{cases}\mathbf{v}_{q,\text{pad}}\\ \mathbf{v}_{q,j}\end{cases}$		$\boxed{\widehat{\mu}_q}$	0	$0, 0$
			$\widehat{\mathbf{L}}_{q,j}[\text{const}]$	$\widehat{\mathbf{L}}_{q,j}[\text{coef}_i]$	$0, 0$
	$q' > q$ $\begin{cases}\mathbf{v}_{q',\text{pad}}\\ \mathbf{v}_{q',j}\end{cases}$		$\widehat{\mu}_{q'}$	0	$0, 0$
			$\widehat{\mathbf{L}}_{q',j}[\text{const}]$	$\widehat{\mathbf{L}}_{q',j}[\text{coef}_i]$	$0, 0$
	ct: $\mathbf{u}$	$h, \mathbf{0}, \mathbf{0}$	1	$\mathbf{x}[i]$	$0, 0$
$\mathsf{H}^b_{4,q+1}$	$q' < q$ $\begin{cases}\mathbf{v}_{q',\text{pad}}\\ \mathbf{v}_{q',j}\end{cases}$	normal	$\widehat{\mu}'_{q'} \xleftarrow{\$} \mathbb{Z}_p$	0	$0, 0$
			$\widehat{\mathbf{L}}_{q',j}[\text{const}]$	$\widehat{\mathbf{L}}_{q',j}[\text{coef}_i]$	$0, 0$
	sk_q $\begin{cases}\mathbf{v}_{q,\text{pad}}\\ \mathbf{v}_{q,j}\end{cases}$		$\boxed{\widehat{\mu}'_q \xleftarrow{\$} \mathbb{Z}_p}$	0	$0, 0$
			$\widehat{\mathbf{L}}_{q,j}[\text{const}]$	$\widehat{\mathbf{L}}_{q,j}[\text{coef}_i]$	$0, 0$
	$q' > q$ $\begin{cases}\mathbf{v}_{q',\text{pad}}\\ \mathbf{v}_{q',j}\end{cases}$		$\widehat{\mu}_{q'}$	0	$0, 0$
			$\widehat{\mathbf{L}}_{q',j}[\text{const}]$	$\widehat{\mathbf{L}}_{q',j}[\text{coef}_i]$	$0, 0$
	ct: $\mathbf{u}$	$h, \mathbf{0}, \mathbf{0}$	1	$\mathbf{x}[i]$	$0, 0$

The proof is unaffected by whether each sk appears before/after ct.

The "normal" in $\mathbf{v}|_{\mathfrak{s}_{\text{pub}}}$ represents the values specified by the honest KeyGen (the same as in Table 4.5).

The garblings $\widehat{\mathbf{L}}_{q',j}$ are fresh and are for freshly sampled pads $\widehat{\mu}_{q'}$, and $\widehat{\mu}'_{q'}$ are independent of them.

In this iteration, sk_q is made *apparently* useless for decrypting ct.

$\widehat{L}_{q,j}$ is randomly generated and independent of the other variables in $\mathsf{H}^b_{4,q}$ and $\mathsf{H}^b_{4,q+1}$, it still applies. Second, the hybrids in 1-ABE security proof only modify the values in $\mathbf{v}_{q,j}|_{\mathsf{s}_{1\text{-ABE}}}$ and $\mathbf{u}[\mathsf{sim}_1], \mathbf{u}[\mathsf{sim}_\star]$, all of which are in the private slot. Moreover, $\mathbf{v}_{q',\mathrm{pad}}, \mathbf{v}_{q',j}$ in the additional secret keys (for $q' \neq q$) have values at $\mathsf{sim}_1, \mathsf{sim}_\star$ set to zero hence cannot "detect" the changes in $\mathbf{u}[\mathsf{sim}_1], \mathbf{u}[\mathsf{sim}_\star]$. Therefore, all the inner products are kept the same and the function-hiding property of IPFE still applies. We indeed conclude $\mathsf{H}^b_{4,q} \approx \mathsf{H}^b_{4,q+1}$.

We now have completed the second step of the proof. Observe that H^b_3 and $\mathsf{H}^b_{4,1}$ are identical. We need another invocation of the function-hiding property of IPFE to remove h from $\mathbf{u}$ (in ict).

- H^b_5 proceeds identically to H^b_3, except that all the pads $\widehat{\mu}_q$ in $\mathbf{v}_{q,\mathrm{pad}}$'s are replaced by independent random values $\widehat{\mu}'_q$, as illustrated in Table 4.5. Observe that $\mathsf{H}^b_{4,Q+1} \equiv \mathsf{H}^b_5$, i.e., the loop hybrids connect H^b_3 and H^b_5.

  In this hybrid, the inner products of $\mathbf{u}$ and $\mathbf{v}_{q,\mathrm{pad}}$'s are $\{h + \widehat{\mu}'_q\}_q$, hiding h. However, h still appears in $\mathbf{u}$. In the next hybrid, we remove h from $\mathbf{u}$ and directly hardwire the inner products $\{h + \widehat{\mu}'_q\}_q$ in $\mathbf{v}_{q,\mathrm{pad}}$'s so that h becomes information-theoretically hidden.

- H^b_6 proceeds identically to H^b_5, except that the pads h and $\widehat{\mu}'_q$ are removed from $\mathbf{u}[\mathsf{pad}]$ and $\mathbf{v}_{q,\mathrm{pad}}[\mathsf{const}]$ respectively, and that their sums $\{h + \widehat{\mu}'_q\}_q$ are put into $\mathbf{v}_{q,\mathrm{pad}}[\mathsf{const}]$. The change is illustrated in Table 4.5. Since the inner products remain the same as in H^b_5, by the function-hiding property of IPFE, $\mathsf{H}^b_5 \approx \mathsf{H}^b_6$.

Lastly, we argue $\mathsf{H}^0_6 \equiv \mathsf{H}^1_6$. The secret keys are independent of h since each $\widehat{\mu}_q$ perfectly hides h. The only other place where h appears is for padding the message, i.e., $(g_b + [\![h]\!]_\mathrm{T})$ in ct. Therefore, g_b is information-theoretically hidden by h.

By hybrid argument, $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$, i.e., Construction 4.3 is adaptively secure (Definition 2.2). □

Combining Lemma 4.2 and Theorems 4.6 and 4.8, we obtain the following:

Corollary 4.9. *Assuming* MDDH_k *(Assumption 2.1) in pairing groups (Definition 2.5), there is a* compact *and* adaptively secure *KP-ABE for ABP (Definitions 2.2, 2.3, and 4.2).*

4.6 KP-ABE for Logspace Turing Machines (Uniform)

In this section, we construct a KP-ABE scheme for L, uniform deterministic logspace computation. Here, each secret key is associated with a Turing machine M (with an arbitrary number Q of states), and each ciphertext, an input $\mathbf{x}$ (of arbitrary length N) and time/space complexity bounds $T, S \geq 1$. Decryption succeeds if and only if M accepts $\mathbf{x}$ within time T and space S. More formally,

$$\begin{aligned} \mathsf{Params}_{\mathsf{gp}} &= \{\bot\}, & F_{\mathsf{gp}} &= \mathsf{TM}, \\ X_{\mathsf{gp}} &= \left\{ (\mathbf{x}, 1^T, 1^{2^S}) \;\middle|\; \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1 \text{ and } T, S \geq 1 \right\}, \\ P_{\mathsf{gp}}(M, (\mathbf{x}, 1^T, 1^{2^S})) &= M|_{N,T,S}(\mathbf{x}) \text{ for } \mathbf{x} \in \{0,1\}^N, \end{aligned}$$

where $M|_{N,T,S}$ (see Definition 4.3) indicates whether M accepts $\mathbf{x}$ within time T and space S. Here, $\bot$ is a placeholder value, i.e., there is the only predicate to choose from. In our construction, the size of a secret key grows linearly in Q, and that of a ciphertext, $NTS2^S$.

Given our construction of ABE for L, we can derive ABE for DFA as a special case, since DFA can be regarded as a Turing machine with time complexity $T = N$ and space complexity $S = 1$, which always moves the input tape pointer to the right and never uses the working tape. Moreover, our ABE scheme for L extends immediately to *non-deterministic, unambiguous* logspace Turing machines (UL). Such machines have at most one accepting path for any input. In fact, our construction handles all *non-deterministic* logspace Turing machines (NL), except when the number of accepting paths is exactly *a multiple of* p, the order of the pairing groups.

Below, we start with our AKGS for Turing machines, then build a 1-ABE scheme based on the AKGS and a function-hiding secret-key IPFE, and lastly lift it to a full-fledged KP-ABE scheme using slotted IPFE. For simplicity of exposition, we describe our construction based on SXDH (Assumption 4.1). It readily extends to be based on MDDH_k for any $k \geq 1$.

4.6.1 AKGS for Turing Machines with Time/Space Bounds

To obtain our AKGS for Turing machines with time/space bounds, we first represent

the computation of Turing machines as a sequence of matrix multiplications, and then design an AKGS for matrix multiplication. See Section 4.2.3 for an overview of our AKGS.

Transition Matrix. Given a Turing machine $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$, upper bounds of time and space complexity $T, S \geq 1$, and an input $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, we consider the length-T *computation path* of M with input $\mathbf{x}$ and space bound S. Recall that the set of internal configurations is

$$\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q].$$

An internal configuration $z = (i, j, \mathbf{W}, q) \in \mathcal{C}_{M,N,S}$ specifies that the input and working tape pointers are respectively at positions i and j, the working tape has content $\mathbf{W}$, and the current state is q. In particular, the initial configuration is $(1, 1, \mathbf{0}_S, 1)$ — the input/working tape pointers point to the first cell, the working tape is all-0, and the state is the initial state 1. An accepting configuration satisfies $\mathbf{y}_{\text{acc}}[q] = 1$.

We construct a transition matrix $\mathbf{M}_{N,S}(\mathbf{x}) \in \{0,1\}^{\mathcal{C}_{M,N,S} \times \mathcal{C}_{M,N,S}}$ where $\mathbf{M}_{N,S}(\mathbf{x})[z, z']$ is 1 if and only if the internal configuration of M is z' after one step of computation starting from internal configuration z. By how the Turing machine operates in each step depending on the transition function δ, we have

$$\begin{aligned}
&\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] \\
&\quad= \begin{cases} 1, & \text{if } \delta(q,\mathbf{x}[i],\mathbf{W}[j]) = (q',\mathbf{W}'[j],i'-i,j'-j) \\ & \qquad \text{and } \mathbf{W}'[j''] = \mathbf{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise;} \end{cases} \\
&\quad= \mathbf{x}[i] \cdot \begin{cases} 1, & \text{if } \delta(q,1,\mathbf{W}[j]) = (q',\mathbf{W}'[j],i'-i,j'-j) \\ & \qquad \text{and } \mathbf{W}'[j''] = \mathbf{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise;} \end{cases} \\
&\qquad + (1-\mathbf{x}[i]) \cdot \begin{cases} 1, & \text{if } \delta(q,0,\mathbf{W}[j]) = (q',\mathbf{W}'[j],i'-i,j'-j) \\ & \qquad \text{and } \mathbf{W}'[j''] = \mathbf{W}[j''] \text{ for all } j'' \neq j; \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

With the transition matrix, we can now write the computation of Turing machine as a sequence of matrix multiplications. We represent internal configurations using one-hot encoding — the internal configuration z is represented by the basis vector $\boldsymbol{\iota}_z \in \{0,1\}^{C_{M,N,S}}$ whose z-entry is 1 and whose other entries are 0. Note that multiplying $\boldsymbol{\iota}_z^\intercal$ on the right by the transition matrix $\mathbf{M}_{N,S}(\mathbf{x})$ produces exactly the next internal configuration. If there is no valid internal configuration of M after one step of computation starting from z, we have $\boldsymbol{\iota}_z^\intercal \mathbf{M}_{N,S}(\mathbf{x}) = \mathbf{0}^\intercal$. Otherwise, $\boldsymbol{\iota}_z^\intercal \mathbf{M}_{N,S}(\mathbf{x}) = \boldsymbol{\iota}_{z'}^\intercal$ for the unique next internal configuration z'. The function $M|_{N,T,S}(\mathbf{x})$ can be written as

$$M|_{N,T,S}(\mathbf{x}) = \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T \big(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\big),$$

where $\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}$ represents the initial internal configuration. The multiplications advance the computation by T steps and then test whether the final internal configuration is in an accepting state. We elaborate on the last step. The tensor product $\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}$ is a vector in $\{0,1\}^{C_{M,N,S}}$ such that its $(i,j,\mathbf{W},q)$-entry is 1 if and only if $\mathbf{y}_{\text{acc}}[q] = 1$, i.e., q is an accepting state. Therefore, taking the inner product of $\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^T = \boldsymbol{\iota}_{z'}^\intercal$ (z' being the final internal configuration) or $\mathbf{0}^\intercal$ (no valid final internal configuration) with the tensor product indicates whether M accepts $\mathbf{x}$ within time T and space S.

Transition Blocks. The transition matrix has the following two useful properties.

- $\mathbf{M}_{N,S}(\mathbf{x})$ is affine in $\mathbf{x}$ when regarded as an integer matrix.
- $\mathbf{M}_{N,S}(\mathbf{x})$ has the following block structure. There is a constant number of $Q \times Q$ matrices $\{\mathbf{M}_\tau\}_\tau$ defined by the transition function δ, called the *transition blocks*, such that for every $(i,j,\mathbf{W})$ and $(i',j',\mathbf{W}')$ in $[N] \times [S] \times \{0,1\}^S$, the submatrix $\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\text{␣}),(i',j',\mathbf{W}',\text{␣})]$ is either some $\mathbf{M}_\tau$ or $\mathbf{0}$.

We define the transition blocks below.

Definition 4.12 (transition blocks). Let $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$ be a Turing machine (Definition 4.3) and $\mathcal{T} = \{0,1\}^3 \times \{0,\pm 1\}^2$ the set of *transition types*. The *transition blocks* of M consist of 72 matrices $\mathbf{M}_\tau \in \{0,1\}^{Q\times Q}$ for $\tau = (x,w,w',\Delta i,\Delta j) \in \mathcal{T}$, each encoding the possible transitions among the states given the following information: the input

tape symbol x under scan, the working tape symbol w under scan, the symbol w' overwriting w, the direction Δi to which the input tape pointer moves, and the direction Δj to which the working tape pointer moves. Formally,

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = \begin{cases} 1, & \text{if } \delta(q,x,w) = (q',w',\Delta i,\Delta j); \\ 0, & \text{otherwise.} \end{cases}$$

In $\mathbf{M}_{N,S}(\mathbf{x})$, each $Q \times Q$ block is either one of the transition blocks or $\mathbf{0}$:

$$\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\textvisiblespace),(i',j',\mathbf{W}',\textvisiblespace)]$$
$$= \begin{cases} \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}, & \text{if } i'-i, j'-j \in \{0,\pm 1\} \text{ and} \\ & \quad \mathbf{W}[j''] = \mathbf{W}'[j''] \text{ for all } j'' \neq j; \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Observe further that in $\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\textvisiblespace),(\textvisiblespace,\textvisiblespace,\textvisiblespace,\textvisiblespace)]$, each transition block appears at most once.

AKGS for Turing machines. We have represented the Turing machine computation as a sequence of matrix multiplications *over the integers*:

$$M|_{N,T,S}(\mathbf{x}) = \boldsymbol{\iota}^{\intercal}_{(1,1,\mathbf{0}_S,1)} \left(\mathbf{M}_{N,S}(\mathbf{x})\right)^T \left(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\right) \quad \text{for } \mathbf{x} \in \{0,1\}^N.$$

We can formally extend[20] $M|_{N,T,S} : \{0,1\}^N \to \{0,1\}$ to a $\mathbb{Z}_p^N \to \mathbb{Z}_p$ function using the same matrix multiplication formula, preserving its behavior when the input comes from $\{0,1\}^N$. When p is clear from the context, we use $M|_{N,T,S}$ to represent its extension over $\mathbb{Z}_p$.

We now construct an AKGS for Turing machine computations and prove its special piecewise security. It is constructed via a recursive mechanism for garbling matrix multiplications, which is inspired[21] by the garbling scheme for arithmetic NC^1 circuits in [AIK11] and the garbling mechanism for multiplication gates in [BGG+14].

[20] This is done to arithmetize the computation, adhering to the formalism of AKGS. The blocks are computed as $((1-\mathbf{x}[i])\mathbf{M}_{0,\dots} + \mathbf{x}[i]\mathbf{M}_{1,\dots})$, which may be neither $\mathbf{M}_{0,\dots}$ nor $\mathbf{M}_{1,\dots}$ when $\mathbf{x}[i] \notin \{0,1\}$, but we will not consider non-binary $\mathbf{x}$ in ABE.

[21] As a historical fact, the authors first came up with the AKGS for arithmetic formulae, abstracted out piecewise security, and only verified that the PGS for ABP [IW14] is also a piecewise secure AKGS.

Construction 4.4 (AKGS for $M|_{N,T,S}$). Let

$$\mathcal{F} = \left\{ M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p \;\middle|\; M \in \mathsf{TM},\ N, T, S \geq 1,\ p \text{ prime} \right\}$$

be the set of time/space bounded Turing machine computations. The AKGS for $\mathcal{F}$ operates as follows.

- $\mathsf{Garble}\big((M, 1^N, 1^T, 1^{2^S}, p), \alpha, \beta\big)$ takes as input a function $M|_{N,T,S}$ over $\mathbb{Z}_p$ from $\mathcal{F}$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$. Suppose $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, the algorithm samples the randomness $\mathbf{r}$ by

$$\begin{aligned} (\text{for } t \in [0,T]) \qquad \mathbf{r}_t &\xleftarrow{\$} \mathbb{Z}_p^{\mathcal{C}_{M,N,S}} \qquad (\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]), \\ \mathbf{r} &\in \mathbb{Z}_p^{[0,T] \times \mathcal{C}_{M,N,S}}, \qquad \mathbf{r}[(t,i,j,\mathbf{W},q)] = \mathbf{r}_t[(i,j,\mathbf{W},q)]. \end{aligned}$$

  It computes the transition matrix $\mathbf{M}_{N,S}(\mathbf{x})$ as a function of $\mathbf{x}$ and defines the label functions by

$$\begin{aligned} L_{\mathrm{init}}(\mathbf{x}) &= \beta + \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}} \mathbf{r}_0, \\ (\text{for } t \in [T]) \qquad \big(L_{t,z}(\mathbf{x})\big)_{z \in \mathcal{C}_{M,N,S}} &= -\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t, \\ \big(L_{T+1,z}(\mathbf{x})\big)_{z \in \mathcal{C}_{M,N,S}} &= -\mathbf{r}_T + \alpha \cdot \mathbf{1}_{[N] \times [S] \times \{0,1\}^S} \otimes \mathbf{y}_{\mathrm{acc}}. \end{aligned}$$

  The algorithm collects and outputs the coefficients of those label functions.

- $\mathsf{Eval}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, \ell_{\mathrm{init}}, (\ell_{t,z})_{t \in [T+1], z \in \mathcal{C}_{M,N,S}}\big)$ takes as input $M|_{N,T,S}$, an input string $\mathbf{x} \in \mathbb{Z}_p^N$, and the labels. It first computes the transition matrix $\mathbf{M}_{N,S}(\mathbf{x})$ with $\mathbf{x}$ substituted into it and sets $\boldsymbol{\ell}_t = (\ell_{t,z})_{z \in \mathcal{C}_{M,N,S}}$ for $t \in [T+1]$. The algorithm computes and outputs

$$\ell_{\mathrm{init}} + \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^{\mathsf{T}} \sum_{t=1}^{T+1} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \boldsymbol{\ell}_t.$$

Syntax and Correctness. We verify that Construction 4.4 is indeed a linear AKGS (Definitions 4.7 and 4.8).

- L_{init} and $L_{t,z}$'s are affine functions of $\mathbf{x}$. The functions L_{init} and $L_{T+1,z}$'s are constant with respect to $\mathbf{x}$. The rest are $L_{t,z}(x) = \big(-\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\big)[z]$. Since $\mathbf{M}_{N,S}(\mathbf{x})$ is affine in $\mathbf{x}$ and $\mathbf{r}_{t-1}, \mathbf{r}_t$ are constant with respect to $\mathbf{x}$, these label functions are also affine in $\mathbf{x}$.

- Shape determinism holds. The garbling size of $M|_{N,T,S}$ is $\big(1 + N(T+1)S2^S Q\big)$.
- Garble is linear in $(\alpha, \beta, \mathbf{r})$. Note that $\mathbf{M}_{N,S}(\mathbf{x}), \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}, \mathbf{y}_{\text{acc}}$ are constant with respect to $(\alpha, \beta, \mathbf{r})$ and $\alpha, \beta, \{\mathbf{r}_t\}_{t\in[0,T]}$ are linear in $(\alpha, \beta, \mathbf{r})$. By the definition of the label functions, their coefficients are linear in $(\alpha, \beta, \mathbf{r})$.
- Correctness. Substituting $\ell_{t,z} = L_{t,z}(\mathbf{x})$ and the formula for $M|_{N,T,S}$ into the summation, we find that it is a telescoping sum:

$$\begin{aligned}
\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \sum_{t=1}^{T+1} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \boldsymbol{\ell}_t &= \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \sum_{t=1}^{T} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \big(-\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\big) \\
&\quad + \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{T} \big(-\mathbf{r}_T + \alpha \cdot \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\text{acc}}\big) \\
&= \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \sum_{t=1}^{T} \Big(-\big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \mathbf{r}_{t-1} + \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t} \mathbf{r}_t\Big) \\
&\quad - \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{T} \mathbf{r}_T + \alpha M|_{N,T,S}(\mathbf{x}) \\
&= -\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \mathbf{r}_0 + \alpha M|_{N,T,S}(\mathbf{x}).
\end{aligned}$$

  The output of Eval is

$$\begin{aligned}
\ell_{\text{init}} + \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \sum_{t=1}^{T+1} \big(\mathbf{M}_{N,S}(\mathbf{x})\big)^{t-1} \boldsymbol{\ell}_t &= \big(\beta + \boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \mathbf{r}_0\big) + \big(-\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^\intercal \mathbf{r}_0 + \alpha M|_{N,T,S}(\mathbf{x})\big) \\
&= \beta + \alpha M|_{N,T,S}(\mathbf{x}).
\end{aligned}$$

  Therefore, the scheme is correct.
- Eval is linear in ℓ_{init} and $\ell_{t,z}$'s. This can be verified by inspection.

Security. We show that Construction 4.4 is special piecewise secure.

Theorem 4.10 (¶). *Construction 4.4 is* special piecewise secure *(Definition 4.11). More specifically, the label functions are ordered as* $L_{\text{init}}, (L_{1,z})_{z\in\mathcal{C}_{M,N,S}}, (L_{2,z})_{z\in\mathcal{C}_{M,N,S}}, \ldots, (L_{T+1,z})_{z\in\mathcal{C}_{M,N,S}}$*, the randomness is ordered as* $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_T$*, and the randomizer of* $L_{t,z}$ *is* $\mathbf{r}_{t-1}[z]$*. For each* $t \in [T+1]$*, the ordering of the components in* $(L_{t,z})_{z\in\mathcal{C}_{M,N,S}}$ *and* $\mathbf{r}_{t-1}$ *can be arbitrary, as long as the two are consistent.*

Proof (Theorem 4.10). By definition, the coefficient of ℓ_{init} in Eval is $1 \neq 0$. Now we prove the marginal randomness property. Recall that all the other label functions are

grouped by time step $t \in [T+1]$ into

$$\begin{aligned} (\text{for } t \in [T]) \qquad \left(L_{t,z}(\mathbf{x})\right)_{z \in \mathcal{C}_{M,N,S}} &= -\mathbf{r}_{t-1} + \mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t, \\ \left(L_{T+1,z}(\mathbf{x})\right)_{z \in \mathcal{C}_{M,N,S}} &= -\mathbf{r}_T + \alpha \cdot \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathrm{acc}}. \end{aligned}$$

Fix a group $t \in [T+1]$ of the label functions. Each $L_{t,z}$ (for $z \in \mathcal{C}_{M,N,S}$), aside from $\mathbf{r}_{t-1}[z]$ (which is in the constant), only uses components in $\mathbf{r}_t$ (if $t \le T$) as randomness. Moreover, all the label functions in groups $t+1, t+2, \ldots, T+1$ only use components in $\mathbf{r}_t, \mathbf{r}_{t+1}, \ldots, \mathbf{r}_T$ as randomness. Therefore, all those label functions are in the special form, the randomizer of $L_{t,z}$ is $\mathbf{r}_{t-1}[z]$, and the components in each group of $L_{t,z}$'s and $\mathbf{r}_{t-1}$ can be reordered as long as they are consistent. □

4.6.2 1-ABE for L

With the AKGS for Turing machines, we are ready to construct our 1-ABE for L and prove its security. To focus on the core idea, we present our construction assuming SXDH in the pairing groups and briefly discuss how to generalize the construction to be based on MDDH_k for any $k \ge 1$.

New Challenge. As discussed in the technical overview (see Section 4.2.3), building ABE for L from AKGS faces new challenges comparing with ABE for ABP. In particular, the size of the AKGS garbling is roughly $NTS2^S Q$. While the key generation algorithm knows Q and the encryption algorithms knows N, T, S, neither of them alone has full information of the size of the AKGS garbling. Furthermore, the total size of a pair of secret key and ciphertext is way smaller than the garbling size. Therefore, it is impossible to encode the label functions of AKGS in either the secret key or the ciphertext (in contrast, in ABE for ABP, the label functions are completely encoded in the secret key). To overcome this challenge, we will *i)* let the secret key and the ciphertext jointly generate the label functions, and *ii)* use pseudorandomness in the exponent.

An Exercise of Algebra. Recall that the AKGS for L samples $\mathbf{r} \in \mathbb{Z}_p^{[0,T]\times\mathcal{C}_{M,N,S}}$ as its randomness. We will use "structured" vector $\mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}$ for $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$

and $\mathbf{r}_\mathrm{f} \xleftarrow{\$} \mathbb{Z}_p^Q$ as *the* randomness for the AKGS garbling. Before laying out the construction, we first show that $\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t$ (a central part of the label functions) can be expressed as a *bilinear* function of $\mathbf{x}, \mathbf{r}_\mathrm{x}, \mathbf{x} \otimes \mathbf{r}_\mathrm{x}$ (known at encryption time) and $\{\mathbf{M}_\tau \mathbf{r}_\mathrm{f}\}_\tau, \mathbf{r}_\mathrm{f}$ (known at key generation time), and hence can be computed as the inner products of vectors depending on these two groups of variables separately.

By our choice of randomness, $\mathbf{r}_t = \mathbf{r}[(t, \sqcup, \sqcup, \sqcup, \sqcup)]$ is a block vector with each block being a multiple of $\mathbf{r}_\mathrm{f}$. More precisely, $\mathbf{r}_t[(i, j, \mathbf{W}, \sqcup)] = \mathbf{r}_\mathrm{x}[(t, i, j, \mathbf{W})] \cdot \mathbf{r}_\mathrm{f}$. We compute each block of the product $\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t$:

$$
\begin{aligned}
& \big(\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\big)[(i, j, \mathbf{W}, \sqcup)] \\
\left(\begin{smallmatrix}\text{row } r \text{ of } \mathbf{Ab} \text{ is} \\ \text{row } r \text{ of } \mathbf{A} \text{ times } \mathbf{b}\end{smallmatrix}\right) \quad & = \mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (\sqcup, \sqcup, \sqcup, \sqcup)] \cdot \mathbf{r}_t \\
\left(\begin{smallmatrix}\text{block matrix} \\ \text{multiplication}\end{smallmatrix}\right) \quad & = \sum_{\substack{i' \in [N], j' \in [S] \\ \mathbf{W}' \in \{0,1\}^S}} \mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)] \cdot \mathbf{r}_t[(i', j', \mathbf{W}', \sqcup)] \\
& = \sum_{\substack{i' \in [N], j' \in [S] \\ \mathbf{W}' \in \{0,1\}^S}} \mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)] \cdot \mathbf{r}_\mathrm{x}[(t, i', j', \mathbf{W}')] \cdot \mathbf{r}_\mathrm{f}.
\end{aligned}
$$

Recall that in $\mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (\sqcup, \sqcup, \sqcup, \sqcup)]$, each transition block appears at most once, and the other $Q \times Q$ blocks are $\mathbf{0}$. More specifically, $\mathbf{M}_{x,w,w',\Delta i,\Delta j}$ appears at $\mathbf{M}_{N,S}(\mathbf{x})[(i, j, \mathbf{W}, \sqcup), (i', j', \mathbf{W}', \sqcup)]$ if $\mathbf{x}[i] = x$, $\mathbf{W}[j] = w$, $i' - i = \Delta i$, $j' - j = \Delta j$, and $\mathbf{W}'$ is $\mathbf{W}$ with its j^{th} entry changed to w'. Therefore, we have

$$
\begin{aligned}
& \big(\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\big)[(i, j, \mathbf{W}, \sqcup)] \\
= & \sum_{\substack{w' \in \{0,1\},\ \Delta i, \Delta j \in \{0, \pm 1\} \\ i + \Delta j \in [N],\ j + \Delta j \in [S]}} \mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \cdot \mathbf{r}_\mathrm{x}[(t, i + \Delta i, j + \Delta j, \mathbf{W}')] \cdot \mathbf{r}_\mathrm{f} \\
= & \sum_{\substack{x, w, w' \in \{0,1\} \\ \Delta i, \Delta j \in \{0, \pm 1\}}} \mathbf{M}_{x,w,w',\Delta i,\Delta j} \cdot \mathbf{r}_\mathrm{f} \cdot \begin{cases} \mathbf{r}_\mathrm{x}[(t, i + \Delta i, j + \Delta j, \mathbf{W}')], & \text{if } x = \mathbf{x}[i], \quad i + \Delta i \in [N], \\ & \qquad w = \mathbf{W}[j],\ j + \Delta j \in [S]; \\ 0, & \text{otherwise.} \end{cases} \qquad (4.6)
\end{aligned}
$$

Here, $\mathbf{W}' \in \{0,1\}^S$, $\mathbf{W}'[j] = w'$, and $\mathbf{W}'[j''] = \mathbf{W}[j'']$ for all $j'' \neq j$. Note that in the last summation formula, there are exactly 72 summands. Moreover, each summand is $\mathbf{M}_{x,w,w',\Delta i,\Delta j} \cdot \mathbf{r}_\mathrm{f}$ (depending only on $\mathbf{r}_\mathrm{f}$ and the transition blocks) multiplied by either an entry in $\mathbf{r}_\mathrm{x}$ or 0 (depending only on $\mathbf{r}_\mathrm{x}$ and $\mathbf{x}$). To simplify notations, we define *transition coefficients*.

Definition 4.13 (transition coefficients). Let $\mathcal{T} = \{0,1\}^3 \times \{0,\pm 1\}^2$ be the set of *transition types* (Definition 4.12). For all $\tau = (x, w, w', \Delta i, \Delta j) \in \mathcal{T}$, $N, T, S \geq 1$, $\mathbf{x} \in \{0,1\}^N$, $t \in [T]$, $i \in [N]$, $j \in [S]$, $\mathbf{W} \in \{0,1\}^S$, $\mathbf{r}_{\mathrm{x}} \in \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$, define the *transition coefficient* as

$$\mathfrak{c}_{x,w,w',\Delta i,\Delta j}(\mathbf{x}; t, i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}}) = \begin{cases} \mathbf{r}_{\mathrm{x}}[(t, i+\Delta i, j+\Delta j, \mathbf{W}')], & \text{if } x = \mathbf{x}[i], \quad i+\Delta i \in [N], \\ & \quad w = \mathbf{W}[j],\ j+\Delta j \in [S]; \\ 0, & \text{otherwise;} \end{cases}$$

where $\mathbf{W}' \in \{0,1\}^S$, $\mathbf{W}'[j] = w'$, and $\mathbf{W}'[j''] = \mathbf{W}[j'']$ for all $j'' \neq j$.

With the above definition, Equation (4.6) can be restated as

$$\left(\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\right)[(i, j, \mathbf{W}, \sqcup)] = \sum_{\tau \in \mathcal{T}} \mathfrak{c}_\tau(\mathbf{x}; t, i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}}) \cdot \mathbf{M}_\tau \mathbf{r}_{\mathrm{f}}. \tag{4.7}$$

We are now ready to construct 1-ABE for L.

Ingredients of Construction 4.5. Let

- $(\mathsf{Garble}, \mathsf{Eval})$ be Construction 4.4, the AKGS for

$$\mathcal{F} = \left\{ M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p \;\middle|\; M \in \mathsf{TM},\ N, T, S \geq 1,\ p \text{ prime} \right\},$$

- $\mathsf{GroupGen}$ a pairing group sampler (Definition 2.5), and
- $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a secret-key IPFE (Definition 4.4) based on $\mathsf{GroupGen}$.

Construction 4.5 (1-ABE for L). Define (see Definitions 2.3 and 4.6)

$$\begin{aligned} \mathsf{Params}_{\mathsf{gp}} &= \{\perp\}, & F_{\mathsf{gp}} &= \mathsf{TM}, \\ X_{\mathsf{gp}} &= \left\{ (\mathbf{x}, 1^T, 1^{2^S}) \;\middle|\; \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1 \text{ and } T, S \geq 1 \right\}, \\ P_{\mathsf{gp}}\left(M, (\mathbf{x}, 1^T, 1^{2^S})\right) &= M|_{N,T,S}(\mathbf{x}) \text{ for } \mathbf{x} \in \{0,1\}^N. \end{aligned}$$

Our 1-ABE scheme based on $\mathsf{GroupGen}$ operates as follows.

- $\mathsf{Setup}(\mathsf{gp})$ sets up the IPFE scheme $\mathsf{imsk} \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, \mathfrak{s}_{\text{1-ABE}})$ for

$$\mathfrak{s}_{\text{1-ABE}} = \left\{ \begin{array}{r} \mathsf{init},\ \mathsf{rand}, \mathsf{rand}^{\mathsf{comp}}, \mathsf{rand}^{\mathsf{temp}}, \mathsf{rand}^{\mathsf{temp,comp}}, \\ \mathsf{acc}, \mathsf{acc}^{\mathsf{temp}},\ \mathsf{sim}, \mathsf{sim}^{\mathsf{temp}}, \mathsf{sim}^{\mathsf{comp}} \end{array} \right\} \cup \left\{ \mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{comp}}, \mathsf{tb}_\tau^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{temp,comp}} \;\middle|\; \tau \in \mathcal{T} \right\}.$$

  It outputs $\mathsf{msk} = \mathsf{imsk}$.

  The names of the indices are as follows. The index init is used for computing ℓ_{init}. It is also used in the security proof. The indices tb_τ ("transition blocks") are used for computing $\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t$ in $\ell_{t,z}$'s, the index rand ("randomizer") for $(-\mathbf{r}_{t-1})$, and acc ("acceptance vector") for $\alpha \cdot \mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathsf{acc}}$ in $\ell_{T+1,z}$'s. Values at all the other indices are set to zero by the honest algorithms, and those indices are only used in the security proof. Jumping ahead, sim is used for simulating the labels, and the rest of the indices are used for holding temporary values in the hybrids — "temp" is short for "temporary", and "comp" for "compensation". More details in the proof.

- $\mathsf{KeyGen}(\mathsf{msk}, M, \mu)$ takes as input msk, some $M = (Q, \mathbf{y}_{\mathsf{acc}}, \delta) \in \mathsf{TM} = F_{\mathsf{gp}}$, and $\mu \in \mathbb{Z}_p$. The algorithm computes the transition blocks $\mathbf{M}_\tau$ for $\tau \in \mathcal{T}$ from δ, samples $\mathbf{r}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$, and sets the vectors $\mathbf{v}_{\mathsf{init}}$ and $\mathbf{v}_q$ for $q \in [Q]$ as follows:

vector	init	rand	acc	tb_τ	the other indices
$\mathbf{v}_{\mathsf{init}}$	$\mathbf{r}_{\mathrm{f}}[1]$	0	0	0	0
$\mathbf{v}_q$	0	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu \cdot \mathbf{y}_{\mathsf{acc}}[q]$	$(\mathbf{M}_\tau \mathbf{r}_{\mathrm{f}})[q]$	

  It generates an IPFE secret key for each vector defined above by

$$\mathsf{isk}_{\mathsf{init}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_{\mathsf{init}}]\!]_2),$$
$$(\text{for } q \in [Q]) \qquad \mathsf{isk}_q \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_q]\!]_2).$$

  The algorithm outputs $\mathsf{sk} = (\mathsf{isk}_{\mathsf{init}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_Q)$.

  We note that KeyGen creates the garbling (or rather, half of it) with secrets $\alpha = \mu$ and $\beta = 0$ so that μ can be recovered using Eval if and only if $M|_{N,T,S}(\mathbf{x}) \neq 0$.[22]

[22] For comparison, in Chapter 7, decryption is authorized for the computation yielding zero.

- $\mathsf{Enc}(\mathsf{msk}, (\mathbf{x}, 1^T, 1^{2^S}))$ takes as input msk, an input string $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, and time/space complexity bounds $T, S \geq 1$ (encoded as 1^T and 1^{2^S}). It samples $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$ and sets the vectors $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{u}_{t,i,j,\mathbf{W}}$ for $t \in [T+1]$, $i \in [N]$, $j \in [S]$, $\mathbf{W} \in \{0,1\}^S$ as follows:

vector	init	rand	acc	tb_τ	the other indices
$\mathbf{u}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]$	0	0	0	
$(t \leq T)$ $\mathbf{u}_{t,i,j,\mathbf{W}}$	0	$\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0	$\mathfrak{c}_\tau(\mathbf{x}; t, i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}})$	0
$\mathbf{u}_{T+1,i,j,\mathbf{W}}$	0	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	1	0	

  The algorithm then generates IPFE ciphertexts for the vectors defined above by

$$\mathsf{ict}_{\mathrm{init}} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{imsk}, [\![\mathbf{u}_{\mathrm{init}}]\!]_1),$$
$$(\text{for } t \in [T+1], i \in [N], j \in [S], \mathbf{W} \in \{0,1\}^S) \qquad \mathsf{ict}_{t,i,j,\mathbf{W}} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{imsk}, [\![\mathbf{u}_{t,i,j,\mathbf{W}}]\!]_1).$$

  The algorithm outputs $\mathsf{ct} = \big(\mathsf{ict}_{\mathrm{init}}, \{\mathsf{ict}_{t,i,j,\mathbf{W}}\}_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big)$.

- $\mathsf{Dec}\big(M, \mathsf{sk}, (\mathbf{x}, 1^T, 1^{2^S}), \mathsf{ct}\big)$ outputs $\perp$ if $M|_{N,T,S}(\mathbf{x}) = 0$, where $\mathbf{x} \in \{0,1\}^N$. Otherwise, it parses sk, ct as in $\mathsf{KeyGen}, \mathsf{Enc}$ and computes the labels (recall that $\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]$)

$$[\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{init}}, \mathsf{ict}_{\mathrm{init}}),$$
$$(\text{for } t \in [T+1], (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S}) \qquad [\![\ell_{t,i,j,\mathbf{W},q}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_q, \mathsf{ict}_{t,i,j,\mathbf{W}}).$$

  The algorithm performs the AKGS evaluation and outputs

$$[\![\mu']\!]_{\mathrm{T}} \leftarrow \mathsf{Eval}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, [\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}}, ([\![\ell_{t,z}]\!]_{\mathrm{T}})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

Correctness. We verify that the scheme is correct. Note that $\ell_{\mathrm{init}}, (\ell_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}$ is a valid garbling of $M|_{N,T,S}$ with secrets $\alpha = \mu$ and $\beta = 0$. Consider

$$\alpha = \mu, \qquad \beta = 0, \qquad \mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}, \qquad \mathbf{r}_t = \mathbf{r}[(t, \sqcup, \sqcup, \sqcup, \sqcup)] \qquad (t \in [0,T]),$$
$$\big(\mathbf{L}_{\mathrm{init}}, (\mathbf{L}_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big) \leftarrow \mathsf{Garble}\big((M, 1^N, 1^T, 1^{2^S}, p), \alpha, \beta; \mathbf{r}\big),$$

where the randomness used by Garble is explicitly written. Let L_{init} and $L_{t,z}$'s be the label functions (affine in $\mathbf{x}$ with coefficient vectors being $\mathbf{L}_{\mathrm{init}}$ and $\mathbf{L}_{t,z}$'s, respectively).

First, by the correctness of IPFE and the definition of vectors $\mathbf{v}_{\mathsf{init}}$ and $\mathbf{u}_{\mathsf{init}}$, we have

$$\ell_{\mathsf{init}} = \mathbf{r}_{\mathsf{x}}[(0,1,1,\mathbf{0}_S)] \cdot \mathbf{r}_{\mathsf{f}}[1] = \mathbf{r}_0[(1,1,\mathbf{0}_S,1)] = \beta + \boldsymbol{\iota}^{\mathsf{T}}_{(1,1,\mathbf{0}_S,1)}\mathbf{r}_0 = L_{\mathsf{init}}(\mathbf{x}).$$

Next, by the correctness of IPFE and the definition of vectors $\mathbf{v}_q$ and $\mathbf{u}_{t,i,j,\mathbf{W}}$, we have

$$\begin{aligned}
\ell_{t,i,j,\mathbf{W},q} &= -\mathbf{r}_{\mathsf{x}}[(t-1,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathsf{f}}[q] + \sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathsf{x}}) \cdot (\mathbf{M}_\tau \mathbf{r}_{\mathsf{f}})[q] \\
&= -\mathbf{r}_{t-1}[(i,j,\mathbf{W},q)] + \left(\sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathsf{x}}) \cdot \mathbf{M}_\tau \mathbf{r}_{\mathsf{f}}\right)[q] \\
\text{(Equation (4.7))} \quad &= -\mathbf{r}_{t-1}[(i,j,\mathbf{W},q)] + \left(\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\right)[(i,j,\mathbf{W},\textvisiblespace)][q] \\
&= -\mathbf{r}_{t-1}[(i,j,\mathbf{W},q)] + \left(\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t\right)[(i,j,\mathbf{W},q)] \\
&= L_{t,i,j,\mathbf{W},q}(\mathbf{x}) \qquad \text{for all } t \in [T],\ (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S}, \\
\ell_{T+1,i,j,\mathbf{W},q} &= -\mathbf{r}_{\mathsf{x}}[(T,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathsf{f}}[q] + \alpha \cdot \mathbf{y}_{\mathsf{acc}}[q] \\
&= -\mathbf{r}_T[(i,j,\mathbf{W},q)] + \alpha \cdot \left(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S} \otimes \mathbf{y}_{\mathsf{acc}}\right)[(i,j,\mathbf{W},q)] \\
&= L_{T+1,i,j,\mathbf{W},q}(\mathbf{x}) \qquad \text{for all } (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S}.
\end{aligned}$$

Lastly, by the linearity of Eval in the labels, we can evaluate the garbling in the exponent of the target group and obtain $\mu' = \alpha M|_{N,T,S}(\mathbf{x}) + \beta = \mu$ (note that $M|_{N,T,S}(\mathbf{x})$ is either 0 or 1 for $\mathbf{x} \in \{0,1\}^N$) in the exponent. Therefore, correctness holds.

Extension to MDDH. The 1-ABE scheme for L based on MDDH_k can be obtained by modifying the above scheme to use pseudorandomness

$$\mathbf{r} = \mathbf{R}_{\mathsf{x}}^{\mathsf{T}}\mathbf{R}_{\mathsf{f}} \qquad \text{for} \qquad \mathbf{R}_{\mathsf{x}} \xleftarrow{\$} \mathbb{Z}_p^{[k]\times([0,T]\times[N]\times[S]\times\{0,1\}^S)} \text{ and } \mathbf{R}_{\mathsf{f}} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q}.$$

Security. We state and prove the security of our 1-ABE for L.

Theorem 4.11 (¶). *Suppose in Construction 4.5, SXDH (Assumption 4.1) holds over* GroupGen *and the IPFE scheme is* function-hiding *(Definition 4.5), then the resultant 1-ABE scheme is* 1-key 1-ciphertext secure *(Definition 4.6).*

Proof Idea. We first discuss the high-level ideas of the proof.

Recall that in the proof of adaptive security of 1-ABE for ABP (Theorem 4.7), for the case where the secret key query appears before the ciphertext challenge, we gradually

replace the label functions in the secret key by labels simulated as random values, except that the first label is reversely sampled.

That approach, however, fails for Construction 4.5 for two reasons: *i)* the garbling uses pseudorandomness instead of true randomness; *ii)* there is not enough space in the ciphertext, the secret key, or both to embed $\Omega(NTS2^S Q)$ labels simulated as random values. We make two tweaks to the proof of Theorem 4.7 to solve the problems: *i)* we rely on SXDH to gradually switch pseudorandomness to true randomness so that by special piecewise security, we can simulate the labels as random; *ii)* in the final hybrid, the labels are simulated as pseudorandom values, which can be embedded in the ciphertext and the secret key.

At a high level, the proof involves three groups of hybrids manipulating the vectors encrypted by IPFE.

1. The first few hybrids replace the first label function L_{init} jointly encoded in $\mathbf{u}_{\text{init}}, \mathbf{v}_{\text{init}}$ by the reversely sampled first label ℓ_{init} hardwired in either $\mathbf{u}_{\text{init}}$ or $\mathbf{v}_{\text{init}}$, whichever is generated later.

2. The middle hybrids are a loop. Along the hybrids, we gradually replace the label functions $L_{t,i,j,\mathbf{W},q}$ (jointly encoded in $\mathbf{u}_{t,i,j,\mathbf{W}}$'s and $\mathbf{v}_q$'s) generated using pseudorandomness by simulated labels $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$, where $\mathbf{s}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$ is embedded in $\mathbf{u}_{t,i,j,\mathbf{W}}$'s and $\mathbf{s}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ is embedded in $\mathbf{v}_q$'s. How the loop proceeds depends on whether the ciphertext challenge comes first or second.

3. The final hybrids do some clean-up work to completely remove μ^0 from the vectors so that it is information-theoretically hidden.

We elaborate on the loop hybrids. The reason why the strategy depends on whether the ciphertext challenge comes first or second is that we need to invoke DDH to temporarily make some label, say $\ell_{t,i,j,\mathbf{W},q}$, truly random. The reduction algorithm (reducing neighboring indistinguishability to DDH) needs to reversely sample ℓ_{init} using RevSamp, which takes as input $M, \mathbf{x}, T, S$ and all the other labels $\ell_{t',i',j',\mathbf{W}',q'}$, including $\ell_{t,i,j,\mathbf{W},q}$. On one hand, since $M, \mathbf{x}, T, S$ are only fully specified after both the

key and ciphertext queries are made, ℓ_{init} can only be embedded in the key if it comes after the ciphertext, and in the ciphertext otherwise. On the other hand, since $\ell_{t,i,j,\mathbf{W},q}$ is being switched from pseudorandom to random by DDH, the reduction only has access to $\ell_{t,i,j,\mathbf{W},q}$ in the exponent of the group $\mathbb{G}_b$ where DDH is invoked. Therefore, ℓ_{init} must be reversely sampled in the exponent of $\mathbb{G}_b$. Adding these requirements together necessitates that DDH must be invoked in the same group where ℓ_{init} is hardwired.

- Case 1: The ciphertext challenge appears before the secret key query. For this case, $\ell_{\mathsf{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ is hardwired in $\mathbf{v}_{\mathsf{init}}$ (in $\mathsf{isk}_{\mathsf{init}}$, encoded in $\mathbb{G}_2$). The middle hybrids loop over $(t, i, j, \mathbf{W})$ in lexicographical order. In each iteration, we first move everything from $\mathbf{u}_{t,i,j,\mathbf{W}}$ to all the $\mathbf{v}_q$'s in one shot, hardwiring the honest labels $\ell_{t,i,j,\mathbf{W},q}$ into $\mathbf{v}_q$ for all q. Next, we invoke DDH over $\mathbb{G}_2$ to switch $\mathbf{r}_t[(i, j, \mathbf{W}, q)]$ from $\mathbf{r}_{\mathrm{x}}[(t, i, j, \mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q]$ to truly random for all q, which makes the labels $\ell_{t,i,j,\mathbf{W},q}$ truly random for all q. Then, we invoke DDH over $\mathbb{G}_2$ again to make $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$ pseudorandom for all q. Lastly, we move $\mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})]$ from all $\mathbf{v}_q$'s back to $\mathbf{u}_{t,i,j,\mathbf{W}}$.
- Case 2: The secret key query appears before the ciphertext challenge. For this case, $\ell_{\mathsf{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ is hardwired in $\mathbf{u}_{\mathsf{init}}$ (in $\mathsf{ict}_{\mathsf{init}}$, encoded in $\mathbb{G}_1$). The middle hybrids form a two-level loop with outer loop over t in increasing order and inner loop over q in increasing order. In each iteration, we first move all occurrences of $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ into all the $\mathbf{u}_{t',i',j',\mathbf{W}'}$'s in one shot (using the extra indices) and hardwire the honest labels $\ell_{t,i,j,\mathbf{W},q}$ into $\mathbf{u}_{t,i,j,\mathbf{W}}$'s for all $i, j, \mathbf{W}$. Next, we invoke DDH over $\mathbb{G}_1$ to switch $\mathbf{r}_t[(i, j, \mathbf{W}, q)]$ from $\mathbf{r}_{\mathrm{x}}[(t, i, j, \mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q]$ to truly random for all $i, j, \mathbf{W}$, making the labels $\ell_{t,i,j,\mathbf{W},q}$ truly random for all $i, j, \mathbf{W}$. Then, we invoke DDH over $\mathbb{G}_1$ again to make $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$ pseudorandom for all $i, j, \mathbf{W}$. Lastly, we move $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ back from all the $\mathbf{u}_{t',i',j',\mathbf{W}'}$'s.

The values must be carefully embedded in appropriate locations, subject to the compactness requirement and the constraint that labels are still correctly computed, so that all the invocations of DDH are indeed valid.

We start with the first/final hybrids, for which the two cases can be handled together, and then complete the proof by connecting the hybrids in two separate claims for the two cases.

Proof (Theorem 4.11). The statement is $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$. Recall that in $\mathsf{Exp}^b_{\text{1-ABE}}$, the adversary is allowed to obtain a single key sk encoding $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$ and μ^0 and a single ciphertext ct encoding $(\mathbf{x}, 1^T, 1^{2^S})$. The only difference is that the adversary additionally receives either μ^0 or μ^1. By construction, the ciphertext ct contains a set of IPFE ciphertexts $\mathsf{ict}_{\text{init}}$ and $\mathsf{ict}_{t,i,j,\mathbf{w}}$'s encrypting $\mathbf{u}_{\text{init}}$ and $\mathbf{u}_{t,i,j,\mathbf{w}}$'s, and sk contains IPFE secret keys $\mathsf{isk}_{\text{init}}$ and isk_q's encrypting $\mathbf{v}_{\text{init}}$ and $\mathbf{v}_q$'s, respectively.

The first/final hybrids are illustrated in Table 4.7. The indices with superscripts are suppressed as they are only used in the loop hybrids. Note that though $\mathbf{v}$'s (in isk's) are listed before $\mathbf{u}$'s (in ict's), the argument about these hybrids is unaffected by whether the ciphertext challenge comes first or second.

- H^b_0 is $\mathsf{Exp}^b_{\text{1-ABE}}$, where $\mathbf{v}$'s contain μ^0, $\mathbf{r}_{\text{f}}$, and the transition blocks, and $\mathbf{u}$'s contain $\mathbf{x}$ and $\mathbf{r}_{\text{x}}$.

- H^b_1 proceeds identically to H^b_0, except that ℓ_{init} is hardwired in $\mathbf{u}_{\text{init}}$ or $\mathbf{v}_{\text{init}}$, whichever is generated later, and that $\mathbf{s}_{\text{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ is embedded in $\mathbf{v}_q[\mathsf{sim}]$'s. More specifically, the first change is implemented as follows.

  - If the *ciphertext* challenge comes first, $\mathbf{u}_{\text{init}}[\mathsf{init}]$ is set to 1 (at encryption time) and $\mathbf{v}_{\text{init}}[\mathsf{init}]$ is set to $\mathbf{r}_{\text{x}}[(0, 1, 1, \mathbf{0}_S)] \cdot \mathbf{r}_{\text{f}}[1]$ (at key generation time).
  - If the *secret key* query comes first, $\mathbf{v}_{\text{init}}[\mathsf{init}]$ is set to 1 (at key generation time) and $\mathbf{u}_{\text{init}}[\mathsf{init}]$ is set to $\mathbf{r}_{\text{x}}[(0, 1, 1, \mathbf{0}_S)] \cdot \mathbf{r}_{\text{f}}[1]$ (at encryption time).

  The first change prepares ℓ_{init} to be reversely sampled starting in the next hybrid by hardwiring it. The second change prepares the $\ell_{t,i,j,\mathbf{w},q}$'s to be simulated as pseudorandom values in the loop hybrids — it has no effect at the moment since the newly embedded values in $\mathbf{v}$'s multiply with 0 in $\mathbf{u}$'s. The inner products in H^b_0 and H^b_1 remain unchanged, therefore, by the function-hiding property of IPFE, we have $\mathsf{H}^b_0 \approx \mathsf{H}^b_1$.

Table 4.7. Hybrids, 1-ABE for L, first and final.

hybrid	vector	init	rand, acc, tb_τ	sim
$\mathsf{H}_0^b \equiv \mathsf{Exp}^b_{\text{1-ABE}}$	$\mathbf{v}_{\text{init}}$	$\boxed{\mathbf{r}_{\mathrm{f}}[1]}$		
	$\mathbf{v}_q$		normal	$\boxed{0}$
	$\mathbf{u}_{\text{init}}$	$\boxed{\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]}$		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		normal	0
H_1^b	$\mathbf{v}_{\text{init}}$	$\boxed{1}$ or $\boxed{\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\cdot\mathbf{r}_{\mathrm{f}}[1]}$		
	$\mathbf{v}_q$		normal	$\boxed{\mathbf{s}_{\mathrm{f}}[q]}$
	$\mathbf{u}_{\text{init}}$	$\boxed{\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\cdot\mathbf{r}_{\mathrm{f}}[1]}$ or $\boxed{1}$		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		normal	0
H_2^b	$\mathbf{v}_{\text{init}}$	1 or $\boxed{\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)}$		
	$\mathbf{v}_q$		normal	$\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\boxed{\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)}$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		$\boxed{\text{normal}}$	$\boxed{0}$
loop		.....		
H_4^b	$\mathbf{v}_{\text{init}}$	1 or $\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$		
	$\mathbf{v}_q$		$\boxed{\text{normal}}$	$\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		$\boxed{0,0,0}$	$\boxed{\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]}$
H_5^b	$\mathbf{v}_{\text{init}}$	1 or $\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$		
	$\mathbf{v}_q$		$\boxed{0,0,0}$	$\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{\text{init}}$	$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ or 1		
	$\mathbf{u}_{t,i,j,\mathbf{W}}$		0,0,0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$

The "normal" in...		rand	acc	tb_τ
$(t \le T)$	$\mathbf{u}_{t,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0	$\mathsf{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$
	$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	1	0
	$\mathbf{v}_q$:	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\cdot\mathbf{y}_{\text{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$

In H_1^b,	if...	$\boxed{\text{sk}, M \text{ is before ct}, \mathbf{x}}$	$\boxed{\text{ct}, \mathbf{x} \text{ is before sk}, M}$
	$\mathbf{v}_{\text{init}}[\text{init}] =$	1	$\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\cdot\mathbf{r}_{\mathrm{f}}[1]$
	$\mathbf{u}_{\text{init}}[\text{init}] =$	$\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]\cdot\mathbf{r}_{\mathrm{f}}[1]$	1

In $\mathsf{H}_2^b, \mathsf{H}_4^b, \mathsf{H}_5^b$,	if...	$\boxed{\text{sk}, M \text{ is before ct}, \mathbf{x}}$	$\boxed{\text{ct}, \mathbf{x} \text{ is before sk}, M}$
	$\mathbf{v}_{\text{init}}[\text{init}] =$	1	$\mathsf{RevSamp}(\cdots)$
	$\mathbf{u}_{\text{init}}[\text{init}] =$	$\mathsf{RevSamp}(\cdots)$	1

The reverse sampling works by (due to the constraint, $\mu^0 \cdot M|_{N,T,S}(\mathbf{x}) = 0$)
$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, 0, (\ell_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big)$.

In H_2^b, the labels $\ell_{t,i,j,\mathbf{W},q} = L_{t,i,j,\mathbf{W},q}(\mathbf{x})$ are computed honestly using IPFE.
The garbling randomness is $\mathbf{r} = \mathbf{r}_{\mathrm{x}} \otimes \mathbf{r}_{\mathrm{f}}$ with $\mathbf{r}_{\mathrm{x}}$ in ct and $\mathbf{r}_{\mathrm{f}}$ in sk.

In H_4^b, the labels $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\cdot\mathbf{s}_{\mathrm{f}}[q]$ are simulated and computed using IPFE with $\mathbf{s}_{\mathrm{x}}$ in ct and $\mathbf{s}_{\mathrm{f}}$ in sk.

In H_5^b, the labels are simulated and μ^0 does not appear.

- H_2^b proceeds identically to H_1^b, except that ℓ_{init} (was $\mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)] \cdot \mathbf{r}_{\mathrm{f}}[1]$) in $\mathbf{u}_{\text{init}}$ or $\mathbf{v}_{\text{init}}$ is reversely sampled from the other labels. By the special piecewise security of AKGS, $\mathsf{H}_1^b \equiv \mathsf{H}_2^b$.

- H_4^b proceeds identically to H_2^b, except that the inner products between $\mathbf{u}_{t,i,j,\mathbf{W}}$'s and $\mathbf{v}_q$'s change from the honest labels to simulated labels $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$. In Table 4.7, this is reflected by the $\mathbf{u}_{t,i,j,\mathbf{W}}$'s having their values at rand, acc, and tb_τ's cleared and embedding $\mathbf{s}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[T+1]\times[N]\times[S]\times\{0,1\}^S}$ at sim. We will show $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ as two separate claims.

  Claim 4.12 (¶). $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ *conditioned on Case 1.*

  Claim 4.13 (¶). $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ *conditioned on Case 2.*

- H_5^b proceeds identically to H_4^b, except that all the $\mathbf{v}_q$'s have their (leftover) values at rand, acc, and tb_τ's cleared. These values multiply with 0 in $\mathbf{u}$'s in H_4^b, so the inner products remain unchanged, and $\mathsf{H}_5^b \approx \mathsf{H}_4^b$ by the function-hiding property of IPFE.

Observe that $\mathsf{H}_5^0 \equiv \mathsf{H}_5^1$ — since μ^0 never appears in the vectors, both μ^0 and μ^1 are just uniformly random values independent of everything else in the hybrids. By hybrid argument, we conclude $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$. □

Proof (Claim 4.12). For Case 1, we show $\mathsf{H}_2^b \approx \mathsf{H}_4^b$ via a series of hybrids $\mathsf{H}_{3,t,i,j,\mathbf{W},1}, \ldots,$ $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ for $(t,i,j,\mathbf{W}) \in [T+1] \times [N] \times [S] \times \{0,1\}^S$ in lexicographical order. Denote the next tuple of indices in increasing order by $((t,i,j,\mathbf{W})+1)$.

The hybrids are illustrated in Table 4.8. Note that the $\mathbf{u}$'s are listed before the $\mathbf{v}$'s, because in Case 1, the ciphertext is generated before the secret key. In the table, all the indices with superscripts except $\mathsf{sim}^{\text{temp}}$ ("temp" for "temporary") are suppressed as they are not used.[23]

[23] Case 1 is the simpler case, though it is already not simple. Note that this coincides with the selective (or semi-adaptive) case, but the simplicity is more of a consequence of the AKGS structure than that of selectivity.

Table 4.8. Hybrids, 1-ABE for L, Case 1 ($\mathbf{x}$ then M), loop (middle).

hybrid	vector	rand	acc	tb_τ	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$	$\mathbf{u}_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{u}_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\cdot\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	0
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$	$\mathbf{u}_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	0	1
	$\mathbf{u}_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\cdot\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	honest $\ell_{t,i,j,\mathbf{W},q}=-\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\cdot\mathbf{r}_{\mathrm{f}}[q]+\cdots$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$	$\mathbf{u}_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	0	1
	$\mathbf{u}_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\cdot\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	$\ell_{t,i,j,\mathbf{W},q}\xleftarrow{\$}\mathbb{Z}_p$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$	$\mathbf{u}_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	0	1
	$\mathbf{u}_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\cdot\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	simulated $\ell_{t,i,j,\mathbf{W},q}=\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\cdot\mathbf{s}_{\mathrm{f}}[q]$
$\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ $\equiv\mathsf{H}^b_{3,t',i',j',\mathbf{W}',1}$ for $(t',i',j',\mathbf{W}')=(t,i,j,\mathbf{W})+1$	$\mathbf{u}_{<(t,i,j,\mathbf{W})}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0	0	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{>(t,i,j,\mathbf{W})}$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i',j',\mathbf{W}')]$	0 or 1	$\mathfrak{c}_\tau(\mathbf{x};t',i',j',\mathbf{W}';\mathbf{r}_{\mathrm{x}})$ or 0	0	0
	$\mathbf{v}_q$	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0\cdot\mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau\mathbf{r}_{\mathrm{f}})[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	0

For brevity, $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{v}_{\mathrm{init}}$ are suppressed. The reversely sampled ℓ_{init} is hardwired in $\mathbf{v}_{\mathrm{init}}$, and is only needed (and can only be computed so by the reduction) in the exponent of $\mathbb{G}_2$:

$$[\![\ell_{\mathrm{init}}]\!]_2\leftarrow\mathsf{RevSamp}\big((M,1^N,1^T,1^{2^S},p),\mathbf{x},[\![0]\!]_2,([\![\ell_{t,z}]\!]_2)_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The choices between "0 or 1" and "$\mathfrak{c}_\tau(\cdots)$ or 0" are the former if $t'\le T$, and the latter if $t'=T+1$.

In this interation, the labels $\ell_{t',i',j',\mathbf{W}',q}$ with $(t',i',j',\mathbf{W}')$... are...

- $<(t,i,j,\mathbf{W})$: simulated as $\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\cdot\mathbf{s}_{\mathrm{f}}[q]$ and computed using IPFE
- $=(t,i,j,\mathbf{W})$: computed honestly using IPFE
  - → computed honestly and hardwired in sk
  - → simulated as random and hardwired in sk
  - → simulated as $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\cdot\mathbf{s}_{\mathrm{f}}[q]$ and hardwired in sk
  - → simulated as $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\cdot\mathbf{s}_{\mathrm{f}}[q]$ and computed using IPFE
- $>(t,i,j,\mathbf{W})$: computed honestly using IPFE

When a label is computed honestly, the garbling randomness is $\mathbf{r}=\mathbf{r}_{\mathrm{x}}\otimes\mathbf{r}_{\mathrm{f}}$.

When a label is computed using IPFE, $\mathbf{r}_{\mathrm{x}},\mathbf{s}_{\mathrm{x}}$ components are in ct and $\mathbf{r}_{\mathrm{f}},\mathbf{s}_{\mathrm{f}}$ components are in sk.

- $\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$ proceeds identically to H^b_2, except that for all $(t',i',j',\mathbf{W}') < (t,i,j,\mathbf{W})$, the vector $\mathbf{u}_{t',i',j',\mathbf{W}'}$ has its values in rand, acc, and tb_τ's cleared, and that a random value $\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$ is embedded in $\mathbf{u}_{t',i',j',\mathbf{W}'}[\mathsf{sim}]$. This means all the labels with $(t',i',j',\mathbf{W}') < (t,i,j,\mathbf{W})$ are simulated, the first label ℓ_{init} is reversely sampled, and the rest are honestly computed.

- $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$, except that the values in $\mathbf{u}_{t,i,j,\mathbf{W}}$ are zeroed out and its inner products with all $\mathbf{v}_q$'s — the labels $\ell_{t,i,j,\mathbf{W},q}$ for all q's — are hardwired into $\mathbf{v}_q$'s.

  - The values of $\mathbf{u}_{t,i,j,\mathbf{W}}$ at rand, acc, and tb_τ's are set to 0.
  - $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathrm{temp}}]$ is set to 1 to pick up the values $\mathbf{v}_q[\mathsf{sim}^{\mathrm{temp}}]$ for all q.
  - The honest labels $\ell_{t,i,j,\mathbf{W},q} = -\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q] + \cdots$ are embedded in $\mathbf{v}_q[\mathsf{sim}^{\mathrm{temp}}]$ for each $q \in [Q]$, where "$\cdots$" is either $\mu^0 \cdot \mathbf{y}_{\mathrm{acc}}[q]$ (when $t = T+1$) or $\sum_{\tau\in\mathcal{T}} \mathfrak{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}) \cdot (\mathbf{M}_\tau \mathbf{r}_{\mathrm{f}})[q]$ (when $t \le T$; this value is $(\mathbf{M}_{N,S}(\mathbf{x}) \cdot \mathbf{r}_t)[(t,i,j,\mathbf{W},q)]$ for pseudorandom $\mathbf{r}_t = \mathbf{r}_{\mathrm{x}}[(t,\sqcup,\sqcup,\sqcup)] \otimes \mathbf{r}_{\mathrm{f}}$).

  As depicted in Table 4.8, the inner products in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},1}$ and $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$ remain unchanged, so they are indistinguishable by the function-hiding property of IPFE.

- $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$, except that the labels $\ell_{t,i,j,\mathbf{W},q}$ are replaced by truly random values. Observe that in both $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$ and $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$, the random values $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$ and $\mathbf{r}_{\mathrm{f}}[q]$'s only appear in $\mathbb{G}_2$ (the group encoding secret key vectors). Moreover, $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$ is only multiplied to $\mathbf{r}_{\mathrm{f}}[q]$'s and does not appear elsewhere (in particular, not on its own). The indistinguishability between them reduces to DDH over $\mathbb{G}_2$.

  The reduction roughly works as follows. Given an $\mathsf{MDDH}_{1,q}$ challenge

$$[\![\mathbf{r}_{\mathrm{f}}[1],\ldots,\mathbf{r}_{\mathrm{f}}[Q],\, z_1,\ldots,z_Q]\!]_2$$

$$\text{with } \begin{cases} z_q = \mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q], & \text{if a DDH tuple is given;} \\ z_q \xleftarrow{\$} \mathbb{Z}_p, & \text{if a truly random tuple is given;} \end{cases}$$

we set $\ell_{t,i,j,\mathbf{W},q} = -z_q + \cdots$ for all $q \in [Q]$. If a DDH tuple is given, those labels use pseudorandom *randomizers* $\mathbf{r}_{t-1}[(i,j,\mathbf{W},\llcorner\lrcorner)] = \mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}$ as in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},2}$. If a truly random tuple is given, those labels use truly random *randomizers* $\mathbf{r}_{t-1}[(i,j,\mathbf{W},\llcorner\lrcorner)] \xleftarrow{\$} \mathbb{Z}_p^Q$, hence are themselves truly random (as in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$) due to the special piecewise security of AKGS. Note that everything else in the two hybrids can be efficiently computed — in particular, the value $[\![\ell_{\mathrm{init}}]\!]_2 \leftarrow \mathsf{RevSamp}(\cdots)$ can be computed efficiently *in the exponent*.

- $\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},3}$, except that the truly random labels $\ell_{t,i,j,\mathbf{W},q}$ for all $q \in [Q]$ are replaced by pseudorandom values $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$ with[24] $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})] \xleftarrow{\$} \mathbb{Z}_p$. The indistinguishability between them reduces to DDH over $\mathbb{G}_2$.

- $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ proceeds identically to $\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$, except that the pseudorandom labels $\ell_{t,i,j,\mathbf{W},q} = \mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$ hardwired in $\mathbf{v}_q[\mathsf{sim}^{\mathrm{temp}}]$'s are *split* into $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}]$ (embedding the factor $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$) and $\mathbf{v}_q[\mathsf{sim}]$'s (embedding the factor $\mathbf{s}_{\mathrm{f}}[q]$), as shown in Table 4.8. The inner products in $\mathsf{H}^b_{3,t,i,j,\mathbf{W},4}$ and $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5}$ remain unchanged, so the two hybrids are indistinguishable by the function-hiding property of IPFE. Moreover, $\mathsf{H}^b_{3,t,i,j,\mathbf{W},5} \equiv \mathsf{H}^b_{3,t',i',j',\mathbf{W}',1}$ for $(t',i',j',\mathbf{W}') = (t,i,j,\mathbf{W}) + 1$.

By hybrid argument, we have $\mathsf{H}^b_{3,1,1,1,\mathbf{0}_S,1} \approx \mathsf{H}^b_{3,T+1,N,S,\mathbf{1}_S,5}$. Observe that $\mathsf{H}^b_2 \equiv \mathsf{H}^b_{3,1,1,1,\mathbf{0}_S,1}$ and $\mathsf{H}^b_4 \equiv \mathsf{H}^b_{3,T+1,N,S,\mathbf{1}_S,5}$. Therefore, $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ conditioned on Case 1. □

Proof (Claim 4.13). The proof of $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ is more involved in Case 2. In this case, the ciphertext challenge appears after the secret key query, and we must hardwire the reversely sampled ℓ_{init} in the ciphertext.

Suppose at some point, we want to make $\ell_{t,i,j,\mathbf{W},q}$ truly random by invoking DDH. In Case 1, we move $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$ into the secret key vectors, which is easy because by the time we need to invoke DDH on $\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$, it only appears in one place (namely, in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{rand}]$) and goes away after the invocation of DDH. In Case 2, since

[24] Note that this is *the first hybrid* in which $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$ appears — each such value is introduced only when needed.

$\ell_{\text{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ must be computed in the exponent of $\mathbb{G}_1$, the only option is to move $\mathbf{r}_{\text{f}}[q]$ into the ciphertext vectors. However, depending on the transition blocks, $\mathbf{r}_{\text{f}}[q]$ might appear in $(\mathbf{M}_\tau \mathbf{r}_{\text{f}})[q']$ of any $\mathbf{v}_{q'}$. There are many limitations to take into account when sorting out the proof.

- The special piecewise security only allows us to simulate $\ell_{t,i,j,\mathbf{W},q}$'s in increasing order of t.
- To simulate $\ell_{t,i,j,\mathbf{W},q}$, all occurrences of $\mathbf{r}_{\text{f}}[q]$ must be in the ciphertext.
- There is not enough space in the ciphertext to embed all the $\mathbf{r}_{\text{f}}[q]$'s at the same time.
- The value $\mathbf{r}_{\text{f}}[q]$ must *not* go away until *all* $\ell_{t,i,j,\mathbf{W},q}$'s are simulated. Indeed, $\mathbf{r}_{\text{f}}[q]$ still resides in $\mathbf{v}_{q'}$'s in H_4^b, the last hybrid (for this claim).

Combining these factors, a feasible strategy is to switch (in increasing order of t, q) batches of $NS2^S$ label functions (i.e., for fixed t, q and all $i, j, \mathbf{W}$) into simulated labels by moving $\mathbf{r}_{\text{f}}[q]$'s back and forth in each iteration (to keep the labels correctly computed) until all the labels are simulated. The above also applies to $\mathbf{s}_{\text{f}}[q]$ when we consider invoking DDH again to make truly random labels only pseudorandom.

More specifically, in iteration t, q, when moving $\mathbf{r}_{\text{f}}[q]$ into the ciphertext vectors, we erase all its occurrences in the secret key vectors and must *compensate* some $\ell_{t',i,j,\mathbf{W},q'}$'s for their loss of $\mathbf{r}_{\text{f}}[q]$ using the indices with superscript "comp". This is done by separating out the terms with $\mathbf{r}_{\text{f}}[q]$. See the *compensation identity* (Equation (4.8) in the notes of Table 4.10) for details.

To better understand the procedure, we define the *modes* of a label $\ell_{t',i,j,\mathbf{W},q'}$. There are three orthogonal groups of modes. Each label has one mode in each group. We will refer to these modes in the description of hybrids below. Their meaning will also become clearer in the context of hybrids.

- The first group is about the value of the label. A label is *honest* if its value is $L_{t',i,j,\mathbf{W},q'}(\mathbf{x})$ with garbling randomness set to $\mathbf{r} = \mathbf{r}_{\text{x}} \otimes \mathbf{r}_{\text{f}}$; it is *random* if its value is sampled uniformly at random; it is *simulated* if its value is $\mathbf{s}_{\text{x}}[(t', i, j, \mathbf{W})] \cdot \mathbf{s}_{\text{f}}[q']$.

- The second group is about where the terms $\mathbf{r}_{\mathsf{f}}, \mathbf{s}_{\mathsf{f}}$ of a label reside. A label is normal (this is the default) if $\mathbf{r}_{\mathsf{f}}, \mathbf{s}_{\mathsf{f}}$ reside in the secret key; it is *compensated* if $\mathbf{r}_{\mathsf{f}}[q], \mathbf{s}_{\mathsf{f}}[q]$ are in the ciphertext with the other components of $\mathbf{r}_{\mathsf{f}}, \mathbf{s}_{\mathsf{f}}$ in the secret key; it is *hardwired* if the value (in its entirety) is hardwired in the ciphertext (for conceptual simplicity, this mode only applies to the labels with $t' = t, q' = q$). A non-normal label will use indices with superscript "comp".

- The last group is a highly technical one. A label is normal (default) if it is computed without indices with superscript "temp"; otherwise it is *temporary* (a label can be in this mode only when $t' = t$).

The loop hybrids are a *two-level loop* with outer loop over $t = 1, \ldots, T + 1$ (Table 4.9) and inner loop over $q = 1, \ldots, Q$ (Table 4.10). In these hybrids, $\mathbf{u}$'s (in ict's) appear after $\mathbf{v}$'s (in isk's). The outer loop consists of the following hybrids.

- $\mathsf{H}^b_{3,t,1}$ proceeds identically to H^b_2, except that for all $t' < t$ and all $i, j, \mathbf{W}$, the vectors $\mathbf{u}_{t',i,j,\mathbf{W}}$ have their values at rand, acc, and tb_τ's cleared and embed $\mathbf{s}_{\mathsf{x}}[(t', i, j, \mathbf{W})]$ at sim. In this hybrid, labels with $t = t'$ are in the *honest* mode.

- $\mathsf{H}^b_{3,t,2}$ proceeds identically to $\mathsf{H}^b_{3,t,1}$, except that the mode of $\ell_{t,i,j,\mathbf{W},q}$'s (for all $i, j, \mathbf{W}, q$) are changed to *honest and temporary* and a random value $\mathbf{s}_{\mathsf{x}}[(t, i, j, \mathbf{W})]$ is embedded in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathsf{temp}}]$ for all $i, j, \mathbf{W}$. The first change is implemented as follows.

  - For all q, the values of $\mathbf{v}_q$'s at rand, acc, and tb_τ are *copied* to their counterparts with superscript "temp".
  - For all $i, j, \mathbf{W}$, the values of $\mathbf{u}_{t,i,j,\mathbf{W}}$'s at rand, acc, and tb_τ are *moved* to their counterparts with superscript "temp".

  This way, the labels are still correctly computed, i.e., the inner products remain unchanged. The second change prepares $\ell_{t,i,j,\mathbf{W},q}$'s to be simulated and currently has no effect — the newly embedded values multiply with 0 in the $\mathbf{v}_q$'s. By the function-hiding property of IPFE, we have $\mathsf{H}^b_{3,t,1} \approx \mathsf{H}^b_{3,t,2}$.

Table 4.9. Hybrids, 1-ABE for L, Case 2 (M then $\mathbf{x}$), outer loop (indexed by t).

hybrid	vector	$\mathsf{rand}, \mathsf{acc}, \mathsf{tb}_\tau$	$\mathsf{rand}^{\mathsf{temp}}, \mathsf{acc}^{\mathsf{temp}}, \mathsf{tb}_\tau^{\mathsf{temp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$
$\mathsf{H}^b_{3,t,1}$	$\mathbf{v}_q$	normal	0, 0, 0	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	0, 0, 0	0, 0, 0	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	normal	0, 0, 0	0	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	0, 0, 0	0	0
$\mathsf{H}^b_{3,t,2}$ ≡ $\mathsf{H}^b_{3,t,3,1,1}$	$\mathbf{v}_q$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	0, 0, 0	0, 0, 0	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0, 0, 0	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	0, 0, 0	0	0
$\mathsf{H}^b_{3,t,3,1\sim Q,1\sim 5}$	.....				
$\mathsf{H}^b_{3,t,4}$ ≡ $\mathsf{H}^b_{3,t,3,Q,5}$	$\mathbf{v}_q$	normal	0, 0, 0	$\mathbf{s}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	0, 0, 0	0, 0, 0	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0, 0, 0	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	0, 0, 0	0	0
$\mathsf{H}^b_{3,t,5}$ ≡ $\mathsf{H}^b_{3,t+1,1}$	$\mathbf{v}_q$	normal	0, 0, 0	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	0, 0, 0	0, 0, 0	$\mathbf{s}_{\mathrm{x}}[(t',i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	0, 0, 0	0, 0, 0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	0, 0, 0	0	0

For brevity, $\mathbf{u}_{\mathrm{init}}$ and $\mathbf{v}_{\mathrm{init}}$ are suppressed. The reversely sampled ℓ_{init} is hardwired in $\mathbf{u}_{\mathrm{init}}$:

$$\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, 0, (\ell_{t,z})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The "normal" in...		$\mathsf{rand}, \mathsf{rand}^{\mathsf{temp}}$	$\mathsf{acc}, \mathsf{acc}^{\mathsf{temp}}$	$\mathsf{tb}_\tau, \mathsf{tb}_\tau^{\mathsf{temp}}$
	$\mathbf{v}_q$:	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu^0 \cdot \mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau \mathbf{r}_{\mathrm{f}})[q]$
$(t' \le T)$	$\mathbf{u}_{t',i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]$	0	$\mathsf{c}_\tau(\mathbf{x}; t', i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}})$
	$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	1	0

In this iteration, the *modes* of $\ell_{t',i,j,\mathbf{W},q}$ are...

- $t' < t$: simulated
- $t' = t$: honest → honest and temporary → simulated and temporary → simulated
- $t' > t$: honest

Table 4.10. Hybrids, 1-ABE for L, Case 2 (M then $\mathbf{x}$), inner loop (indexed by t, q).

hybrid	vector	$\mathsf{rand},\mathsf{rand}^{\mathsf{comp}},$ $\mathsf{acc},\mathsf{tb}_r,\mathsf{tb}_r^{\mathsf{comp}}$	$\mathsf{rand}^{\mathsf{temp}},\mathsf{rand}^{\mathsf{temp,comp}},$ $\mathsf{acc}^{\mathsf{temp}},\mathsf{tb}_r^{\mathsf{temp}},\mathsf{tb}_r^{\mathsf{temp,comp}}$	sim	$\mathsf{sim}^{\mathsf{temp}}$	$\mathsf{sim}^{\mathsf{comp}}$
$\mathsf{H}^b_{3,t,3,q,1}$	$\mathbf{v}_{q'<q}$	normal	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q]$	0	0
	$\mathbf{v}_{q'>q}$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,2}$	$\mathbf{v}_{q'<q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	1
	$\mathbf{v}_{q'>q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\cdot\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	honest $\ell_{t,i,j,\mathbf{W},q}=$ $-\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]\cdot\mathbf{r}_{\mathrm{f}}[q]+\cdots$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,3}$	$\mathbf{v}_{q'<q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	1
	$\mathbf{v}_{q'>q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\cdot\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	$\ell_{t,i,j,\mathbf{W},q}\xleftarrow{\$}\mathbb{Z}_p$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,4}$	$\mathbf{v}_{q'<q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	1
	$\mathbf{v}_{q'>q}$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	✗ $\mathbf{r}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]\cdot\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	simulated $\ell_{t,i,j,\mathbf{W},q}=$ $\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]\cdot\mathbf{s}_{\mathrm{f}}[q]$
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	✓ $\mathbf{r}_{\mathrm{f}}[q]$	$0,0,0,0,0$	0	0	0
$\mathsf{H}^b_{3,t,3,q,5}$ ≡ $\mathsf{H}^b_{3,t,3,q+1,1}$	$\mathbf{v}_{q'<q}$	normal	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q']$	$\mathbf{s}_{\mathrm{f}}[q']$	0
	$\mathbf{v}_{q}$	normal	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{f}}[q]$	$\mathbf{s}_{\mathrm{f}}[q]$	0
	$\mathbf{v}_{q'>q}$	normal	normal	$\mathbf{s}_{\mathrm{f}}[q']$	0	0
	$\mathbf{u}_{t'<t,i,j,\mathbf{W}}$	$0,0,0,0,0$	$0,0,0,0,0$	$\mathbf{s}_{\mathrm{x}}[(t',i',j',\mathbf{W}')]$	0	0
	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$0,0,0,0,0$	normal	0	$\mathbf{s}_{\mathrm{x}}[(t,i,j,\mathbf{W})]$	0
	$\mathbf{u}_{t'>t,i,j,\mathbf{W}}$	normal	$0,0,0,0,0$	0	0	0

Continued on the next page with notes...

Continuing from the previous page of hybrids...

For brevity, $\mathbf{u}_{\text{init}}$ and $\mathbf{v}_{\text{init}}$ are suppressed. The reversely sampled ℓ_{init} is hardwired in $\mathbf{u}_{\text{init}}$,
and is only needed (and can only be computed so by the reduction) in the exponent of $\mathbb{G}_1$:

$$[\![\ell_{\text{init}}]\!]_1 \leftarrow \mathsf{RevSamp}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, [\![0]\!]_1, ([\![\ell_{t,z}]\!]_1)_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The "normal" in...		$\mathsf{rand}^?$	$\mathsf{acc}^?$	$\mathsf{tb}_\tau^?$	$\mathsf{rand}^{?,\mathsf{comp}}, \mathsf{tb}_\tau^{?,\mathsf{comp}}$
	$\mathbf{v}_{q'}$:	$-\mathbf{r}_{\mathrm{f}}[q']$	$\mu^0 \cdot \mathbf{y}_{\mathrm{acc}}[q']$	$(\mathbf{M}_\tau \mathbf{r}_{\mathrm{f}})[q']$	0
$(t \le t' \le T)$	$\mathbf{u}_{t',i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]$	0	$\mathsf{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	0
	$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	1	0	0

The compensation components ("✗ $\mathbf{r}_{\mathrm{f}}[q]$" and "✓ $\mathbf{r}_{\mathrm{f}}[q]$") in...

		$\mathsf{rand}^?$	$\mathsf{rand}^{?,\mathsf{comp}}$	$\mathsf{acc}^?$	$\mathsf{tb}_\tau^?$	$\mathsf{tb}_\tau^{?,\mathsf{comp}}$
$(q' \ne q)$	$\mathbf{v}_{q'}$:	$-\mathbf{r}_{\mathrm{f}}[q']$	0	$\mu^0 \cdot \mathbf{y}_{\mathrm{acc}}[q']$	$(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q] \cdot \boldsymbol{\iota}_q))[q']$	$(\mathbf{M}_\tau \boldsymbol{\iota}_q)[q']$
	$\mathbf{v}_q$:	0	-1	$\mu^0 \cdot \mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q] \cdot \boldsymbol{\iota}_q))[q]$	$(\mathbf{M}_\tau \boldsymbol{\iota}_q)[q]$
when $t \le T$...						
	$\mathbf{u}_{t,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0	0	$\mathsf{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	$\mathsf{c}_\tau(\mathbf{x};t,i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}) \cdot \mathbf{r}_{\mathrm{f}}[q]$
$(t < t' \le T)$	$\mathbf{u}_{t',i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]$	$\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q]$	0	$\mathsf{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})$	$\mathsf{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}) \cdot \mathbf{r}_{\mathrm{f}}[q]$
	$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q]$	1	0	0
when $t = T+1$...						
	$\mathbf{u}_{T+1,i,j,\mathbf{W}}$:	$\mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	0	1	0	0

In the tables above, "?" is either nothing or "temp", i.e.,
if the values are set in both non-temporary slots and temporary slots, they are the same.

Note that $(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q] \cdot \boldsymbol{\iota}_q)$ is simply $\mathbf{r}_{\mathrm{f}}$ with its q^{th} entry changed to 0, so $\mathbf{r}_{\mathrm{f}}[q]$ does not appear.

The compensation is governed by the following identity:

$$\begin{aligned}
\ell_{t',i,j,\mathbf{W},q'} &= -\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})] \cdot \mathbf{r}_{\mathrm{f}}[q'] + \sum_{\tau\in\mathcal{T}} \mathsf{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}) \cdot (\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q] \cdot \boldsymbol{\iota}_q + \mathbf{r}_{\mathrm{f}}[q] \cdot \boldsymbol{\iota}_q))[q'] \\
&= \underbrace{(-1)}_{\mathbf{v}[\mathsf{rand}^{?,\mathsf{comp}}]} \cdot \underbrace{\mathbf{r}_{\mathrm{x}}[(t'-1,i,j,\mathbf{W})]}_{\mathbf{u}[\mathsf{rand}^{?,\mathsf{comp}}]} \cdot \mathbf{r}_{\mathrm{f}}[q'] \\
&\quad + \sum_{\tau\in\mathcal{T}} \overbrace{\mathsf{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}})}^{\mathbf{u}[\mathsf{tb}_\tau^?]} \cdot \overbrace{(\mathbf{M}_\tau(\mathbf{r}_{\mathrm{f}} - \mathbf{r}_{\mathrm{f}}[q] \cdot \boldsymbol{\iota}_q))[q']}^{\mathbf{v}[\mathsf{tb}_\tau^?]} \\
&\quad + \sum_{\tau\in\mathcal{T}} \underbrace{\mathsf{c}_\tau(\mathbf{x};t',i,j,\mathbf{W};\mathbf{r}_{\mathrm{x}}) \cdot \mathbf{r}_{\mathrm{f}}[q]}_{\mathbf{u}[\mathsf{tb}_\tau^{?,\mathsf{comp}}]} \cdot \underbrace{(\mathbf{M}_\tau \boldsymbol{\iota}_q)[q']}_{\mathbf{v}[\mathsf{tb}_\tau^{?,\mathsf{comp}}]}.
\end{aligned} \tag{4.8}$$

In this iteration, the *modes* of $\ell_{t',i,j,\mathbf{W},q'}$ are...

	$q' < q$	$q' = q$	$q' > q$
$t' < t$	S	S → SC → S	S
$t' = t$	ST	HT → HW → RW → SW → ST	HT → HCT → HT
$t' > t$	H → HC → H	H → HC → H	H → HC → H

The shorthands are **H**onest, **R**andom, **S**imulated, **C**ompensated, hard**W**ired, **T**emporary.

For three-mode chains, the middle one spans $\mathsf{H}^b_{3,t,3,q,2\sim4}$.

The net effect is that $\ell_{t,i,j,\mathbf{W},q}$'s change from *honest and temporary* to *simulated and temporary*.

Table 4.10 (continued, notes). Hybrids, 1-ABE for L, Case 2 (M then $\mathbf{x}$),
inner loop (indexed by t, q).

- $\mathsf{H}^b_{3,t,4}$ proceeds identically to $\mathsf{H}^b_{3,t,2}$, except that the mode of $\ell_{t,i,j,\mathbf{W},q}$'s (for all $i, j, \mathbf{W}, q$) is switched from *honest and temporary* to *simulated and temporary*. In Table 4.9, this is reflected by $\mathbf{v}_q$'s (for all q) having their values at $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$, and $\mathsf{tb}^{\mathsf{temp}}_\tau$'s cleared and embedding $\mathbf{s}_{\mathrm{f}}[q]$ at $\mathsf{sim}^{\mathsf{temp}}$. We will show $\mathsf{H}^b_{3,t,2} \approx \mathsf{H}^b_{3,t,4}$ via a subseries of hybrids (the inner loop hybrids).

- $\mathsf{H}^b_{3,t,5}$ proceeds identically to $\mathsf{H}^b_{3,t,4}$, except that the mode of $\ell_{t,i,j,\mathbf{W},q}$'s is switched from *simulated and temporary* to *simulated* and the following clean-up work is done in preparation for the next iteration.

  - $\mathbf{v}_q$'s (for all q) have their values at $\mathsf{sim}^{\mathsf{temp}}$ cleared.
  - $\mathbf{u}_{t,i,j,\mathbf{W}}$'s (for all $i, j, \mathbf{W}$) have $\mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})]$'s moved from $\mathsf{sim}^{\mathsf{temp}}$ into sim.
  - $\mathbf{u}_{t,i,j,\mathbf{W}}$'s (for all $i, j, \mathbf{W}$) have their values at $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$, and $\mathsf{tb}^{\mathsf{temp}}_\tau$'s cleared.

  It is readily verified that the inner products remain unchanged, so $\mathsf{H}^b_{3,t,4} \approx \mathsf{H}^b_{3,t,5}$ by the function-hiding property of IPFE. Furthermore, we have $\mathsf{H}^b_{3,t,5} \equiv \mathsf{H}^b_{3,t+1,1}$.

Note that $\mathsf{H}^b_2 \equiv \mathsf{H}^b_{3,1,1}$ and $\mathsf{H}^b_4 \equiv \mathsf{H}^b_{3,T+1,5}$. Once we show $\mathsf{H}^b_{3,t,2} \approx \mathsf{H}^b_{3,t,4}$, we can conclude $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ by hybrid argument.

The inner loop hybrids connect $\mathsf{H}^b_{3,t,2}$ and $\mathsf{H}^b_{3,t,4}$ by switching $\ell_{t,i,j,\mathbf{W},q}$ from *honest and temporary* to *simulated and temporary* one by one for $q = 1, \ldots, Q$.

- $\mathsf{H}^b_{3,t,3,q,1}$ proceeds identically to $\mathsf{H}^b_{3,t,2}$, except that for $q' < q$, all the $\mathbf{v}_{q'}$ have their values at $\mathsf{rand}^{\mathsf{temp}}$, $\mathsf{acc}^{\mathsf{temp}}$, and $\mathsf{tb}^{\mathsf{temp}}_\tau$'s cleared and the value $\mathbf{s}_{\mathrm{f}}[q']$ is embedded at $\mathsf{sim}^{\mathsf{temp}}$, i.e., the labels $\ell_{t,i,j,\mathbf{W},q'}$ for all $i, j, \mathbf{W}$ and all $q' < q$ have been switched from *honest and temporary* to *simulated and temporary*.

- $\mathsf{H}^b_{3,t,3,q,2}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,1}$, except that all occurrences of $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ are moved from $\mathbf{v}_{q'}$'s into $\mathbf{u}_{t',i,j,\mathbf{W}}$'s according to the *compensation identity* (Equation (4.8)). The implementation is illustrated in Table 4.10. The labels with $q' = q$ or $t' > t$ or $t' = t, q' > q$ gain *compensated* mode on top of their existing modes, and the labels $\ell_{t,i,j,\mathbf{W},q}$ for all $i, j, \mathbf{W}$ become *honest and hardwired*

(hardwired in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathrm{comp}}]$). The inner products (labels) remain unchanged, so $\mathsf{H}^b_{3,t,3,q,1} \approx \mathsf{H}^b_{3,t,3,q,2}$ by the function-hiding property of IPFE.

- $\mathsf{H}^b_{3,t,3,q,3}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,2}$, except that the labels $\ell_{t,i,j,\mathbf{W},q}$ (for all $i, j, \mathbf{W}$) hardwired in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathrm{comp}}]$ become *random and hardwired*. The indistinguishability between $\mathsf{H}^b_{3,t,3,q,2}$ and $\mathsf{H}^b_{3,t,3,q,3}$ reduces to DDH over $\mathbb{G}_1$.

- $\mathsf{H}^b_{3,t,3,q,4}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,2}$, except that the labels $\ell_{t,i,j,\mathbf{W},q}$ (for all $i, j, \mathbf{W}$) hardwired in $\mathbf{u}_{t,i,j,\mathbf{W}}[\mathsf{sim}^{\mathrm{comp}}]$ are changed to be pseudorandom $\mathbf{s}_{\mathrm{x}}[(t, i, j, \mathbf{W})] \cdot \mathbf{s}_{\mathrm{f}}[q]$, which makes them *simulated and hardwired*. The indistinguishability between $\mathsf{H}^b_{3,t,3,q,2}$ and $\mathsf{H}^b_{3,t,3,q,3}$ again reduces to DDH over $\mathbb{G}_1$.

- $\mathsf{H}^b_{3,t,3,q,5}$ proceeds identically to $\mathsf{H}^b_{3,t,3,q,4}$, except that all occurrences of $\mathbf{r}_{\mathrm{f}}[q]$ and $\mathbf{s}_{\mathrm{f}}[q]$ are moved back to $\mathbf{v}_{q'}$'s (including putting $\mathbf{s}_{\mathrm{f}}[q]$ back to $\mathbf{v}_q[\mathsf{sim}]$) — compensation is undone — and some clean-up work is done to prepare for the next iteration. More specifically, the clean-up work involves clearing the values of $\mathbf{v}_q, \mathbf{u}_{t,i,j,\mathbf{W}}$ at $\mathsf{sim}^{\mathrm{comp}}$ for all $i, j, \mathbf{W}$ and putting $\mathbf{s}_{\mathrm{f}}[q]$ into $\mathbf{v}_q[\mathsf{sim}^{\mathrm{temp}}]$. As a result, all the labels lose their *compensated* mode, and the labels $\ell_{t,i,j,\mathbf{W},q}$ for all $i, j, \mathbf{W}$ become *simulated and temporary*. Note that $\mathsf{H}^b_{3,t,3,q,5} \equiv \mathsf{H}^b_{3,t,3,q+1,1}$.

Lastly, observe that $\mathsf{H}^b_{3,t,3,1,1} \equiv \mathsf{H}^b_{3,t,2}$ and $\mathsf{H}^b_{3,t,3,Q,5} \equiv \mathsf{H}^b_{3,t,4}$. By a hybrid argument over the inner loop hybrids, we have $\mathsf{H}^b_{3,t,2} \approx \mathsf{H}^b_{3,t,4}$.

Now, by a hybrid argument over the outer loop, $\mathsf{H}^b_2 \approx \mathsf{H}^b_4$ in Case 2. □

Extension to MDDH. For the version based on MDDH_k, the labels are simulated as

$$\ell_{t,i,j,\mathbf{W},q} = (\mathbf{S}_{\mathrm{x}}^{\top}\mathbf{S}_{\mathrm{f}})[(t, i, j, \mathbf{W}), q] \quad \text{for} \quad \mathbf{S}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[k]\times([T+1]\times[N]\times[S]\times\{0,1\}^S)} \text{ and } \mathbf{S}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q}.$$

The proof for Case 1 goes by invoking MDDH_k instead of DDH. We discuss the change in the proof for Case 2. At some point, we need to switch the randomizers of the labels with some specific t, q to truly random. Invoking MDDH_k apparently requires completely removing the q^{th} column of $\mathbf{R}_{\mathrm{f}}$, yet this column is used to compute the randomizers for the labels with $q' = q$ that are still honest. We got away with this when assuming DDH because both distributions give us access to $\mathbf{r}_{\mathrm{f}}[q]$ (a convenient

shortcut), i.e.,

$$\begin{aligned} & [\![\mathbf{r}_{\mathrm{x}}[(t-1,\_,\_,\_)], \mathbf{r}_{\mathrm{f}}[q], \mathbf{r}_{\mathrm{x}}[(t-1,\_,\_,\_)] \cdot \mathbf{r}_{\mathrm{f}}[q]]\!]_1 \\ \text{(DDH)} \quad \approx\ & [\![\mathbf{r}_{\mathrm{x}}[(t-1,\_,\_,\_)], \mathbf{r}_{\mathrm{f}}[q], \quad \mathbf{r}[(t-1,\_,\_,\_,q)] \quad]\!]_1. \end{aligned}$$

A naïve attempt to generalize the proof to work with MDDH_k is to say

$$\begin{aligned} & [\![\mathbf{R}_{\mathrm{x}}[\_,(t-1,\_,\_,\_)], \mathbf{R}_{\mathrm{f}}[\_,q], (\mathbf{R}_{\mathrm{x}}[\_,(t-1,\_,\_,\_)])^{\mathsf{T}} \mathbf{R}_{\mathrm{f}}[\_,q]]\!]_1 \\ \overset{?}{\approx}\ & [\![\mathbf{R}_{\mathrm{x}}[\_,(t-1,\_,\_,\_)], \mathbf{R}_{\mathrm{f}}[\_,q], \quad \mathbf{r}[(t-1,\_,\_,\_),q] \quad]\!]_1, \end{aligned}$$

which does *not* follow from MDDH_k (and is false for symmetric pairing groups). The correct proof uses

$$\begin{aligned} & [\![\mathbf{R}_{\mathrm{x}}, \overbrace{\{(\mathbf{R}_{\mathrm{x}}[\_,(t'-1,\_,\_,\_)])^{\mathsf{T}} \mathbf{R}_{\mathrm{f}}[\_,q]\}_{t'>t}}^{\substack{\text{in } \mathbf{u}_{t'',\_,\_,\_} \text{ at } \mathsf{rand}^{\mathsf{comp}}, \mathsf{tb}_\tau^{\mathsf{comp}}\text{'s} \\ \text{and in } \mathbf{u}_{t,\_,\_,\_} \text{ at } \mathsf{tb}_\tau^{\mathsf{temp,comp}}\text{'s}}}, \overbrace{(\mathbf{R}_{\mathrm{x}}[\_,(t-1,\_,\_,\_)])^{\mathsf{T}} \mathbf{R}_{\mathrm{f}}[\_,q]}^{\text{in } \mathbf{u}_{t,\_,\_,\_} \text{ at } \mathsf{sim}^{\mathsf{comp}}}]\!]_1 \\ (\mathsf{MDDH}_k) \approx\ & [\![\mathbf{R}_{\mathrm{x}}, \{ \quad \mathbf{r}[(t'-1,\_,\_,\_),q] \quad \}_{t'>t}, \quad \mathbf{r}[(t-1,\_,\_,\_),q] \quad]\!]_1 \\ (\mathsf{MDDH}_k) \approx\ & [\![\mathbf{R}_{\mathrm{x}}, \{(\mathbf{R}_{\mathrm{x}}[\_,(t'-1,\_,\_,\_)])^{\mathsf{T}} \mathbf{R}_{\mathrm{f}}[\_,q]\}_{t'>t}, \quad \mathbf{r}[(t-1,\_,\_,\_),q] \quad]\!]_1. \end{aligned}$$

In other words, we first switch all the *randomizers* of the labels with $q' = q$ to truly random, then switch back those with $t' > t, q' = q$ to pseudorandom (the randomizers for the labels with $t' < t$ are already gone in that iteration, because those labels are already simulated). A similar strategy is used for $\mathbf{S}_{\mathrm{x}}, \mathbf{S}_{\mathrm{f}}$ when we switch the truly random *labels* to pseudorandom simulated ones.

4.6.3 KP-ABE for L

We use the same technique in Section 4.5.3 to convert 1-ABE for L into a full-fledged ABE for L. The only difference is that we will need *two* copies of the indices in the private slot.

In the security proof of KP-ABE for ABP, the private slot can be regarded as a compartment isolating the interaction between all the secret keys and *the challenge ciphertext* from the other ciphertexts (which can be generated by the adversary using the master public key in the public slot). All the secret keys can share the same private slot because the security of 1-ABE for ABP is unaffected by the extra keys. However,

the security of 1-ABE for L does not automatically extend to the multi-key setting, and we need an extra copy of the indices to further isolate the interaction between the challenge ciphertext and *the secret key being modified* from the other secret keys.

Ingredients of Construction 4.6. Let

- $(\mathsf{Garble}, \mathsf{Eval})$ be Construction 4.4, the AKGS for

$$\mathcal{F} = \left\{ M|_{N,T,S} : \mathbb{Z}_p^N \to \mathbb{Z}_p \;\middle|\; M \in \mathsf{TM},\ N, T, S \geq 1,\ p \text{ prime} \right\},$$

- $\mathsf{GroupGen}$ a pairing group sampler (Definition 2.5), and
- $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec}, \mathsf{IPFE.SlotEnc})$ a slotted IPFE (Definition 4.4) based on $\mathsf{GroupGen}$.

Construction 4.6 (KP-ABE for L). Define (see Definition 2.3)

$$\mathsf{Params}_{\mathsf{gp}} = \{\bot\}, \qquad F_{\mathsf{gp}} = \mathsf{TM}, \qquad Y_{\mathsf{gp}} = \mathbb{G}_T,$$
$$X_{\mathsf{gp}} = \left\{ (\mathbf{x}, 1^T, 1^{2^S}) \;\middle|\; \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1 \text{ and } T, S \geq 1 \right\},$$
$$P_{\mathsf{gp}}\left(M, (\mathbf{x}, 1^T, 1^{2^S})\right) = M|_{N,T,S}(\mathbf{x}) \text{ for } \mathbf{x} \in \{0,1\}^N.$$

Our KP-ABE scheme based on $\mathsf{GroupGen}$ operates as follows.

- $\mathsf{Setup}(\mathsf{gp})$ takes the group description gp as input. It generates IPFE master public/secret key pair $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, \mathfrak{s}_{\mathrm{pub}}, \mathfrak{s}_{\mathrm{priv}})$ for

$$\mathfrak{s}_{\mathrm{pub}} = \left\{ \mathsf{pad}^{\mathrm{ct}}, \mathsf{pad}^{\mathrm{sk}}, \mathsf{init}^{\mathrm{pub}}, \mathsf{rand}^{\mathrm{pub}}, \mathsf{acc}^{\mathrm{pub}} \right\} \cup \left\{ \mathsf{tb}_\tau^{\mathrm{pub}} \;\middle|\; \tau \in \mathcal{T} \right\},$$
$$\mathfrak{s}_{\mathrm{copy}} = \left\{ \mathsf{init}^{\mathrm{copy}}, \mathsf{rand}^{\mathrm{copy}}, \mathsf{acc}^{\mathrm{copy}} \right\} \cup \left\{ \mathsf{tb}_\tau^{\mathrm{copy}} \;\middle|\; \tau \in \mathcal{T} \right\},$$
$$\mathfrak{s}_{\mathrm{priv}} = \mathfrak{s}_{\mathrm{copy}} \cup \mathfrak{s}_{\text{1-ABE}} \cup \left\{ \mathsf{pad}^{\mathrm{copy}}, \mathsf{pad}^{\mathrm{temp}} \right\},$$

  where $\mathfrak{s}_{\text{1-ABE}}$ is the index set defined in Construction 4.5. The algorithm outputs $\mathsf{mpk} = \mathsf{impk}$ and $\mathsf{msk} = \mathsf{imsk}$.

  The naming is as follows. The indices $\mathsf{pad}^{\mathrm{ct}}, \mathsf{pad}^{\mathrm{sk}}$ are used to compute $(h + \mu)$ in the exponent, where h (resp. μ) is the random pad in ct (resp. sk) embedded at $\mathsf{pad}^{\mathrm{ct}}$ (resp. $\mathsf{pad}^{\mathrm{sk}}$). The values in $\mathfrak{s}_{\mathrm{priv}}$ are set to 0 by the honest algorithms and reserved for the security proof. The names of the other indices follow the convention in 1-ABE.

- $\mathsf{KeyGen}(\mathsf{msk}, M)$ takes as input msk and $M = (Q, \mathbf{y}_{\mathrm{acc}}, \delta) \in \mathsf{TM} = F_{\mathrm{gp}}$. It computes the transition blocks $\mathbf{M}_\tau$ for $\tau \in \mathcal{T}$ from δ, samples $\mu \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{r}_{\mathrm{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$, and sets the vectors $\mathbf{v}_{\mathrm{pad}}$, $\mathbf{v}_{\mathrm{init}}$, and $\mathbf{v}_q \in \mathbb{Z}_p^{\mathfrak{s}}$ for $q \in [Q]$ as follows:

vector	$\mathsf{pad}^{\mathsf{ct}}$	$\mathsf{pad}^{\mathsf{sk}}$	$\mathsf{init}^{\mathsf{pub}}$	$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_\tau^{\mathsf{pub}}$	in $\mathfrak{s}_{\mathrm{priv}}$
$\mathbf{v}_{\mathrm{pad}}$	1	μ	0	0	0	0	
$\mathbf{v}_{\mathrm{init}}$	0	0	$\mathbf{r}_{\mathrm{f}}[1]$	0	0	0	0
$\mathbf{v}_q$	0	0	0	$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu \cdot \mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_\tau \mathbf{r}_{\mathrm{f}})[q]$	

  The algorithm then generates IPFE secret keys for them by

$$\begin{aligned} \mathsf{isk}_{\mathrm{pad}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_{\mathrm{pad}}]\!]_2), \\ \mathsf{isk}_{\mathrm{init}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_{\mathrm{init}}]\!]_2), \\ (\text{for } q \in [Q]) \qquad \mathsf{isk}_q &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_q]\!]_2). \end{aligned}$$

  The algorithm outputs $\mathsf{sk} = (\mathsf{isk}_{\mathrm{pad}}, \mathsf{isk}_{\mathrm{init}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_Q)$.

  Same as in our 1-ABE for L (Construction 4.5), the algorithm KeyGen creates the garbling with secrets $\alpha = \mu$ and $\beta = 0$.

- $\mathsf{Enc}\big(\mathsf{mpk}, (\mathbf{x}, 1^T, 1^{2^S}), g\big)$ takes as input mpk, input $\mathbf{x} \in \{0,1\}^N$ for some $N \geq 1$, time/space complexity bounds $T, S \geq 1$ (encoded as 1^T and 1^{2^S}), and message $g \in \mathbb{G}_{\mathrm{T}}$. It samples $h \xleftarrow{\$} \mathbb{Z}_p$, $s \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{r}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S}$ and sets the vectors $\mathbf{u}_{\mathrm{pad}}$, $\mathbf{u}_{\mathrm{init}}$, and $\mathbf{u}_{t,i,j,\mathbf{W}} \in \mathbb{Z}_p^{\mathfrak{s}_{\mathrm{pub}}}$ for $t \in [T+1]$, $i \in [N]$, $j \in [S]$, $\mathbf{W} \in \{0,1\}^S$ as follows:

vector	$\mathsf{pad}^{\mathsf{ct}}$	$\mathsf{pad}^{\mathsf{sk}}$	$\mathsf{init}^{\mathsf{pub}}$	the other indices
$\mathbf{u}_{\mathrm{pad}}$	h	s	0	0
$\mathbf{u}_{\mathrm{init}}$	0	0	$s \cdot \mathbf{r}_{\mathrm{x}}[(0,1,1,\mathbf{0}_S)]$	

vector		$\mathsf{rand}^{\mathsf{pub}}$	$\mathsf{acc}^{\mathsf{pub}}$	$\mathsf{tb}_\tau^{\mathsf{pub}}$	the other indices
$(t \leq T)$	$\mathbf{u}_{t,i,j,\mathbf{W}}$	$s \cdot \mathbf{r}_{\mathrm{x}}[(t-1,i,j,\mathbf{W})]$	0	$s \cdot \mathfrak{c}_\tau(\mathbf{x}; t, i, j, \mathbf{W}; \mathbf{r}_{\mathrm{x}})$	0
	$\mathbf{u}_{T+1,i,j,\mathbf{W}}$	$s \cdot \mathbf{r}_{\mathrm{x}}[(T,i,j,\mathbf{W})]$	s	0	

The algorithm generates IPFE ciphertexts by

$$\mathsf{ict}_{\mathrm{pad}} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{impk}, [\![\mathbf{u}_{\mathrm{pad}}]\!]_1), \qquad \mathsf{ict}_{\mathrm{init}} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{impk}, [\![\mathbf{u}_{\mathrm{init}}]\!]_1),$$
$$(\text{for } t \in [T+1], i \in [N], j \in [S], \mathbf{W} \in \{0,1\}^S)\ \mathsf{ict}_{t,i,j,\mathbf{W}} \xleftarrow{\$} \mathsf{IPFE.SlotEnc}(\mathsf{impk}, [\![\mathbf{u}_{t,i,j,\mathbf{W}}]\!]_1).$$

It outputs $\mathsf{ct} = \big(g + [\![h]\!]_{\mathrm{T}}, \mathsf{ict}_{\mathrm{pad}}, \mathsf{ict}_{\mathrm{init}}, \{\mathsf{ict}_{t,i,j,\mathbf{W}}\}_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big)$.

- $\mathsf{Dec}\big(\mathsf{mpk}, M, \mathsf{sk}, (\mathbf{x}, 1^T, 1^{2^S}), \mathsf{ct}\big)$ outputs $\perp$ if $M|_{N,T,S}(\mathbf{x}) = 0$ for $\mathbf{x} \in \{0,1\}^N$. Otherwise, it parses sk as in KeyGen and ct as

$$\big([\![z]\!]_{\mathrm{T}}, \mathsf{ict}_{\mathrm{pad}}, \mathsf{ict}_{\mathrm{init}}, \{\mathsf{ict}_{t,i,j,\mathbf{W}}\}_{t\in[T+1],i\in[N],j\in[S],\mathbf{W}\in\{0,1\}^S}\big),$$

and runs (recall that $\mathcal{C}_{M,N,S} = [N] \times [S] \times \{0,1\}^S \times [Q]$)

$$[\![z']\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{pad}}, \mathsf{ict}_{\mathrm{pad}}), \qquad [\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_{\mathrm{init}}, \mathsf{ict}_{\mathrm{init}}),$$
$$(\text{for } t \in [T+1], (i,j,\mathbf{W},q) \in \mathcal{C}_{M,N,S}) \quad [\![\ell_{t,i,j,\mathbf{W},q}]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_q, \mathsf{ict}_{t,i,j,\mathbf{W}}),$$
$$[\![\mu']\!]_{\mathrm{T}} \leftarrow \mathsf{Eval}\big((M, 1^N, 1^T, 1^{2^S}, p), \mathbf{x}, [\![\ell_{\mathrm{init}}]\!]_{\mathrm{T}}, ([\![\ell_{t,z}]\!]_{\mathrm{T}})_{t\in[T+1],z\in\mathcal{C}_{M,N,S}}\big).$$

The algorithm computes and outputs $([\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}})$.

Correctness. Combining the arguments for the correctness of 1-ABE for L (Construction 4.5) and KP-ABE for ABP (Construction 4.3), we see

$$\mu' = s \cdot \mu \cdot M|_{N,T,S}(\mathbf{x}) = s\mu \qquad \text{and} \qquad z' = h + s\mu$$
$$\implies \qquad [\![z]\!]_{\mathrm{T}} + [\![\mu']\!]_{\mathrm{T}} - [\![z']\!]_{\mathrm{T}} = g + [\![h]\!]_{\mathrm{T}} + [\![s\mu]\!]_{\mathrm{T}} - [\![h + s\mu]\!]_{\mathrm{T}} = g,$$

i.e., Dec correctly recovers the message.

Security. We state and prove the security of our ABE construction.

Theorem 4.14 (¶). *Suppose in Construction 4.6, SXDH (Assumption 4.1) holds over* GroupGen *and the slotted IPFE scheme is* function-hiding *(Definition 4.5), then the resultant ABE scheme is* adaptively secure *(Definition 2.2).*

Proof (Theorem 4.14). The statement is $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$. Recall that in $\mathsf{Exp}^b_{\mathrm{PHFE}}$, the adversary can receive many secret keys $\{\mathsf{sk}_\varphi\}_{\varphi\in[\Phi]}$ for different machines $\{M_\varphi\}_\varphi$ and a ciphertext ct for input $\mathbf{x}$, time/space bounds T, S, and message g_b, where

$\{M_\varphi\}_\varphi, \mathbf{x}, T, S, g_0, g_1$ are chosen adaptively by the adversary subject to $M_\varphi|_{N,T,S}(\mathbf{x}) = 0$ for all $\varphi \in [\Phi]$, where $\mathbf{x} \in \{0,1\}^N$. (If the constraint is not satisfied, the outcome of the experiment is set to 0.)

At a high level, the proof involves three steps, similar to those in the security proof of KP-ABE for ABP (Theorem 4.8).

- First, the garblings (in both ct and sk_φ) as well as the secret key pads μ_φ are removed from $\mathfrak{s}_{\text{pub}}$. Instead, ct embeds in $\mathfrak{s}_{\text{copy}}$ (half of) the garbling without s, and each sk_φ embeds in $\mathfrak{s}_{\text{copy}}$ (half) a garbling generated with pad $\widehat{\mu}_\varphi^0$ and randomness $\widehat{\mathbf{r}}_{\varphi,\mathsf{f}}$ *independent* of those in $\mathfrak{s}_{\text{pub}}$.

- Next, we go through a loop of Φ iterations. In the φ^{th} iteration, we replace $\widehat{\mu}_\varphi^0$ in $\mathbf{v}_{\varphi,\text{pad}}$ by an independent random value $\widehat{\mu}_\varphi^1$ (inconsistent with $\widehat{\mu}_\varphi^0$ used to generate $\mathbf{v}_{\varphi,\text{init}}$ and $\mathbf{v}_{\varphi,q}$'s). However, this is not as simple as the case of KP-ABE for ABP, because the security of 1-ABE for L (Theorem 4.11) does not automatically generalize to the case of multiple secret key queries, which can be regarded as a consequence of reusing the randomness in the ciphertext *vectors* (i.e., the garbling is already not truly random in 1-ABE). We employ the trick of *temporary* mode to resolve the problem. Roughly speaking, $\mathfrak{s}_{\text{1-ABE}}$ is used for the interaction between sk_φ (the secret key being modified) and ct, while $\mathfrak{s}_{\text{copy}}$ is used for the interaction between all the other secret keys and ct. The ciphertext uses two sets of *independent* randomness — $\mathbf{r}_{\mathsf{x}}$ in $\mathfrak{s}_{\text{copy}}$ and $\widehat{\mathbf{r}}_{\mathsf{x}}$ in $\mathfrak{s}_{\text{1-ABE}}$ — so that the proof of Theorem 4.11 can be invoked in $\mathfrak{s}_{\text{1-ABE}}$.

- When all the secret keys have their pads $\widehat{\mu}_\varphi^0$ replaced by $\widehat{\mu}_\varphi^1$, we move the random pad h from the ciphertext into all the secret keys, by which point h will be perfectly hidden by the pads $\widehat{\mu}_\varphi^1$ and perfectly hide the message.

We start with the first step, which is the first step in the proof of Theorem 4.8 with DDH instead of MDDH.

- H_0^b is $\mathsf{Exp}_{\text{PHFE}}^b$, where the ict's (in ct) are generated using $\mathsf{IPFE.SlotEnc}$. This is depicted by "$\perp$" in the $\mathbf{u}$'s in Table 4.11.

Table 4.11. Hybrids, KP-ABE for L, first and final.

hybrid	vector	$\mathsf{pad}^{\mathsf{ct}}$, $\mathsf{pad}^{\mathsf{sk}}$	$\mathsf{init}^{\mathsf{pub}}$,$\mathsf{rand}^{\mathsf{pub}}$, $\mathsf{acc}^{\mathsf{pub}}$,$\mathsf{tb}_{\mathsf{r}}^{\mathsf{pub}}$	$\mathsf{pad}^{\mathsf{copy}}$	in $\mathsf{s}_{\mathsf{copy}}$
$\mathsf{H}_0^b \equiv \mathsf{Exp}_{\mathrm{PHFE}}^b$	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		0	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		0
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ s$		$\perp$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		$\propto (s, s\mathbf{r}_{\mathrm{x}})$		$\perp$
H_1^b	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\boxed{0}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\boxed{0}$
	$\boxed{\mathbf{u}_{\mathrm{pad}}}$	$h,\ \boxed{s}$		$\boxed{0}$	
	$\boxed{\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}}$		$\boxed{\propto (s, s\mathbf{r}_{\mathrm{x}})}$		$\boxed{0}$
H_2^b	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\boxed{s\mu_\varphi}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\boxed{\propto (s\mu_\varphi, s\mathbf{r}_{\varphi,\mathrm{f}})}$
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ \boxed{0}$		$\boxed{1}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		$\boxed{0}$		$\boxed{\propto (1, \mathbf{r}_{\mathrm{x}})}$
$\mathsf{H}_3^b \equiv \mathsf{H}_{4,1}^b$	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\boxed{\widehat{\mu}_\varphi^0}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\boxed{\propto (\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})}$
	$\mathbf{u}_{\mathrm{pad}}$	$h,\ 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$
$\mathsf{H}_{4,1\sim\Phi+1}^b$			.....		
$\mathsf{H}_5^b \equiv \mathsf{H}_{4,\Phi+1}^b$	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\boxed{\widehat{\mu}_\varphi^1 \xleftarrow{\$} \mathbb{Z}_p}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\propto (\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$	$\boxed{h}, 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$
H_6^b	$\mathbf{v}_{\varphi,\mathrm{pad}}$	$1,\ \mu_\varphi$		$\boxed{h + \widehat{\mu}_\varphi^1}$	
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\propto (\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}})$		$\propto (\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{u}_{\mathrm{pad}}$	$\boxed{0}, 0$		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$		0		$\propto (1, \mathbf{r}_{\mathrm{x}})$

For brevity, the vectors for computing the labels are not spelled out.

The shorthand "$\propto \mathbf{z}$" means that the components there are linear in $\mathbf{z}$ and efficiently computable given $\mathbf{z}$ in the exponent, and that there is only one natural way of computing them (see Construction 4.6).

- H_1^b proceeds identically to H_0^b, except that the ict's are generated using $\mathsf{IPFE.Enc}$ with $\mathbf{u}|_{\mathbb{s}_{\mathrm{priv}}}$ set to $\mathbf{0}$. By the slot-mode correctness of the slotted IPFE, we have $\mathsf{H}_0^b \equiv \mathsf{H}_1^b$.

- H_2^b proceeds identically to H_1^b, except that how the (secret key) pads and the labels are computed (as inner products) is changed — instead of multiplying with s in the ciphertext, s is multiplied in the secret key (which prepares the secret key randomness to be rerandomized).

  - In the ciphertext vectors, values at $\mathsf{pad}^{\mathrm{sk}}$, $\mathsf{init}^{\mathrm{pub}}$, $\mathsf{rand}^{\mathrm{pub}}$, $\mathsf{acc}^{\mathrm{pub}}$, and $\mathsf{tb}_\tau^{\mathrm{pub}}$'s (in the public slot) are *moved* to $\mathsf{pad}^{\mathrm{copy}}$ and $\mathbb{s}_{\mathrm{copy}}$ (in the private slot), and the multiplier s is removed.
  - In the secret key vectors, values at $\mathsf{pad}^{\mathrm{sk}}$, $\mathsf{init}^{\mathrm{pub}}$, $\mathsf{rand}^{\mathrm{pub}}$, $\mathsf{acc}^{\mathrm{pub}}$, and $\mathsf{tb}_\tau^{\mathrm{pub}}$'s (in the public slot) are *copied* to $\mathsf{pad}^{\mathrm{copy}}$ and $\mathbb{s}_{\mathrm{copy}}$ (in the private slot), and s is multiplied to the copy in the private slot.

  The hybrid is illustrated in Table 4.11. Since the inner products remain the same and the secret key vectors have their values in the public slot unchanged, we have $\mathsf{H}_1^b \approx \mathsf{H}_2^b$ by the function-hiding property of IPFE. An alternative view of this hybrid is that in each secret key sk_φ, two (half) garblings are embedded, one generated for μ_φ using randomness $\mathbf{r}_{\varphi,\mathrm{f}}$ in the public slot, the other generated for $s\mu_\varphi$ using randomness $s\mathbf{r}_{\varphi,\mathrm{f}}$ in the private slot.

- H_3^b proceeds identically to H_2^b, except that the garbling in the private slots of the secret key vectors are generated for independent pad $\widehat{\mu}_\varphi^0$ with independent randomness $\widehat{\mathbf{r}}_{\varphi,\mathrm{f}}$. Note that the only difference between H_2^b and H_3^b is that in the former, the randomness and the pads of the garblings in the private slot are *random multiples* of those in the public slot, and that in the latter, those in the private slot are independently sampled. DDH over $\mathbb{G}_2$ says

$$\left\{[\![\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}},\ s\mu_\varphi, s\mathbf{r}_{\varphi,\mathrm{f}}]\!]_2\right\}_{\varphi\in[\Phi]} \approx \left\{[\![\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}},\ \widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}}]\!]_2\right\}_{\varphi\in[\Phi]}.$$

  Given $[\![\mu_\varphi, \mathbf{r}_{\varphi,\mathrm{f}}]\!]_2$ and either $[\![s\mu_\varphi, s\mathbf{r}_{\varphi,\mathrm{f}}]\!]_2$ or $[\![\widehat{\mu}_\varphi^0, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}}]\!]_2$, the secret key vectors can be efficiently computed *in the exponent*. Plus, everything else can be efficiently computed. Therefore, $\mathsf{H}_2^b \approx \mathsf{H}_3^b$ by DDH over $\mathbb{G}_2$.

We have completed the first step of the proof.

Now we proceed to the second step, in which we switch $\widehat{\mu}_\varphi^0$ to independent random values $\widehat{\mu}_\varphi^1$ (inconsistent with $\mathbf{v}_{\varphi,\mathsf{init}}$ and $\mathbf{v}_{\varphi,q}$'s) one by one:

- $\mathsf{H}_{4,\varphi}^b$ proceeds identically to H_3^b, except that for $\varphi' < \varphi$ (the first $(\varphi-1)$ secret keys), $\mathbf{v}_{\varphi',\mathsf{pad}}$ (in $\mathsf{isk}_{\varphi',\mathsf{pad}}$) stores an independent random value $\widehat{\mu}_\varphi^1 \xleftarrow{\$} \mathbb{Z}_p$ at $\mathsf{pad}^{\mathsf{copy}}$.

Observe that $\mathsf{H}_3^b \equiv \mathsf{H}_{4,1}^b$. We claim the following:

Claim 4.15 (¶). $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$ *for all* $\varphi \in [\Phi]$, $b \in \{0,1\}$.

For now, we focus on the last step.

- H_5^b proceeds identically to H_3^b, except that all the pads $\widehat{\mu}_\varphi^0$ in $\mathbf{v}_{\varphi,\mathsf{pad}}[\mathsf{pad}^{\mathsf{copy}}]$ are replaced by independent random values $\widehat{\mu}_\varphi^1 \xleftarrow{\$} \mathbb{Z}_p$. The hybrid is shown in Table 4.11. Observe that $\mathsf{H}_5^b \equiv \mathsf{H}_{4,\Phi+1}^b$.

  Note that in this hybrid, the inner products of $\mathbf{u}_{\mathsf{pad}}$ and $\mathbf{v}_{\varphi,\mathsf{pad}}$'s are $\{h + \widehat{\mu}_\varphi^1\}_\varphi$, which hide h. However, h still appears in $\mathbf{u}_{\mathsf{pad}}$. In the next hybrid, we remove h from $\mathbf{u}_{\mathsf{pad}}$ and directly hardwire the inner products $\{h + \widehat{\mu}_\varphi^1\}_\varphi$ in $\mathbf{v}_{\varphi,\mathsf{pad}}$'s so that h becomes information-theoretically hidden.

- H_6^b proceeds identically to H_5^b, except that the random pad h is removed from $\mathbf{u}_{\mathsf{pad}}[\mathsf{pad}^{\mathsf{ct}}]$ and each $\mathbf{v}_{\varphi,\mathsf{pad}}$ stores $(h + \widehat{\mu}_\varphi^1)$ at $\mathsf{pad}^{\mathsf{copy}}$. Since the inner products and the public slot of the secret key vectors remain unchanged, we have $\mathsf{H}_5^b \approx \mathsf{H}_6^b$ by the function-hiding property of IPFE.

We have $\mathsf{H}_6^0 \equiv \mathsf{H}_6^1$ because h is perfectly hidden by each $\widehat{\mu}_\varphi^1$ and perfectly hides the message. Therefore, we conclude $\mathsf{Exp}_{\mathsf{PHFE}}^0 \approx \mathsf{Exp}_{\mathsf{PHFE}}^1$. □

Proof (Claim 4.15). Fixing some $\varphi \in [\Phi]$ and $b \in \{0,1\}$, we want to show $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$, for which we must modify sk_φ. There are three key ideas in the proof, which are logically chained in an anadiplosis fashion.

- To modify sk_φ, we temporarily use $\mathsf{pad}^{\mathsf{temp}}$ and $\mathfrak{s}_{\text{1-ABE}}$ exclusively for the interaction between ct and sk_φ, and use $\mathsf{pad}^{\mathsf{copy}}$ and $\mathfrak{s}_{\mathsf{copy}}$ for the interaction

between ct and the other secret keys, similar to the *temporary* mode in the proof of Claim 4.13. With the interaction between ct and sk_φ isolated, we could hope to syntactically invoke 1-ABE security for them.

- To invoke 1-ABE security in $\mathbb{s}_{\text{1-ABE}}$, we must make sure that in $\mathbb{s}_{\text{1-ABE}}$, the ciphertext vectors use randomness independent of those in $\mathbb{s}_{\text{copy}}$. This is done by invoking DDH twice in a row to make ct use $\mathbf{r}_{\mathrm{x}}$ in $\mathsf{pad}^{\text{copy}}, \mathbb{s}_{\text{copy}}$ and independent $\widehat{\mathbf{r}}_{\mathrm{x}}$ in $\mathsf{pad}^{\text{temp}}, \mathbb{s}_{\text{1-ABE}}$, similar to how the labels are simulated in the proof of Theorem 4.11 by invoking DDH twice in a row.

- To invoke DDH to rerandomize $\mathbf{r}_{\mathrm{x}}$, there needs to be a random value multiplied to it. However, in $\mathsf{H}^b_{4,\varphi}$, nothing is multiplied to μ's, $\mathbf{r}_{\mathrm{f}}$'s, $\widehat{\mu}$'s, $\widehat{\mathbf{r}}_{\mathrm{f}}$'s, nor $\mathbf{r}_{\mathrm{x}}$. This can be resolved by a change of variable introducing a negligible statistical error:

$$(\widehat{\mu}^0_\varphi, \widehat{\mu}^1_\varphi, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}}) \approx_{\mathrm{s}} (\widehat{s}\,\widehat{\mu}^0_\varphi, \widehat{s}\,\widehat{\mu}^1_\varphi, \widehat{s}\,\widehat{\mathbf{r}}_{\varphi,\mathrm{f}}) \quad \text{for} \quad \widehat{s} \xleftarrow{\$} \mathbb{Z}_p.$$

  We then move $\widehat{s}$ to $\mathbf{r}_{\mathrm{x}}$ using the function-hiding property of IPFE, after which DDH can be invoked.

We now implement these ideas in the following hybrids $\mathsf{G}^\psi_0, \ldots, \mathsf{G}^\psi_6$.

- G^ψ_0 proceeds identically to $\mathsf{H}^b_{4,\varphi+\psi}$, where the first $(\varphi - 1)$ secret keys have their $\widehat{\mu}^0_{\varphi'}$ replaced by $\widehat{\mu}^1_{\varphi'}$ in $\mathbf{v}_{\varphi',\mathsf{pad}}[\mathsf{pad}^{\text{copy}}]$, the key sk_φ stores either $\widehat{\mu}^0_\varphi$ (consistent with $\mathbf{v}_{\varphi,\text{init}}$ and $\mathbf{v}_{\varphi,q}$'s) or $\widehat{\mu}^1_\varphi$ (inconsistent). The hybrid is shown in Table 4.12.

- G^ψ_1 proceeds identically to G^ψ_0, except that a random multiplier $\widehat{s} \xleftarrow{\$} \mathbb{Z}_p$ is multiplied to the values at $\mathsf{pad}^{\text{copy}}, \mathbb{s}_{\text{copy}}$ for sk_φ. Conditioned on $\widehat{s} \neq 0$ (which happens with overwhelming probability), G^ψ_1 and G^ψ_0 are identical. Therefore, $\mathsf{G}^\psi_0 \approx_{\mathrm{s}} \mathsf{G}^\psi_1$ taking into account the case of $\widehat{s} = 0$.

- G^ψ_2 proceeds identically to G^ψ_1, except that the interaction between ct and sk_φ is isolated from those between ct and the other secret keys, and that s is multiplied in ct instead of sk_φ.

  - In the vectors in *the φ^{th} secret key* sk_φ, the values at $\mathsf{pad}^{\text{copy}}, \mathbb{s}_{\text{copy}}$ are *moved* to $\mathsf{pad}^{\text{temp}}, \mathbb{s}_{\text{1-ABE}}$, and the multiplier $\widehat{s}$ is removed.

Table 4.12. Hybrids, KP-ABE for L, loop (middle).

hybrid	vector	in $\mathsf{s}_{\mathrm{pub}}$	$\mathsf{pad}^{\mathrm{copy}}$	in $\mathsf{s}_{\mathrm{copy}}$	$\mathsf{pad}^{\mathrm{temp}}$	in $\mathsf{s}_{\text{1-ABE}}$
G_0^{ψ} ≡ $\mathsf{H}_{4,\varphi+\psi}^{b}$	$\varphi' < \varphi$ { $\mathbf{v}_{\varphi',\mathrm{pad}}$; $\mathbf{v}_{\varphi',\mathrm{init}}, \mathbf{v}_{\varphi',q}$ }	μ, $\mathbf{r}_{\mathrm{f}}$'s, ($\widehat{\mu}, \widehat{\mathbf{r}}_{\mathrm{f}}$'s independent) and h	$\widehat{\mu}^1_{\varphi'} \xleftarrow{\$} \mathbb{Z}_p$	$\propto (\widehat{\mu}^0_{\varphi'}, \widehat{\mathbf{r}}_{\varphi',\mathrm{f}})$	0	0
	$\mathbf{v}_{\varphi,\mathrm{pad}}$		$\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{\mu}^0_{\varphi}, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$	0	0
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$		$\widehat{\mu}^0_{\varphi'}$	$\propto (\widehat{\mu}^0_{\varphi'}, \widehat{\mathbf{r}}_{\varphi',\mathrm{f}})$	0	0
	$\varphi' > \varphi$ { $\mathbf{v}_{\varphi',\mathrm{pad}}$; $\mathbf{v}_{\varphi',\mathrm{init}}, \mathbf{v}_{\varphi',q}$ }					
	$\mathbf{u}_{\mathrm{pad}}$		1	$\propto (1, \mathbf{r}_{\mathrm{x}})$	0	0
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$					
G_1^{ψ}	$\mathbf{v}_{\varphi,\mathrm{pad}}$	μ, $\mathbf{r}_{\mathrm{f}}$'s, h	$\widehat{s}\,\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{s}\,\widehat{\mu}^0_{\varphi}, \widehat{s}\,\widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$	0	0
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		0	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		0
G_2^{ψ}	$\mathbf{v}_{\varphi,\mathrm{pad}}$	μ, $\mathbf{r}_{\mathrm{f}}$'s, h	0	0	$\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{\mu}^0_{\varphi}, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		$\widehat{s}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (\widehat{s}, \widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}})$
G_3^{ψ}	$\mathbf{v}_{\varphi,\mathrm{pad}}$	μ, $\mathbf{r}_{\mathrm{f}}$'s, h	0	0	$\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{\mu}^0_{\varphi}, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		$\widehat{s}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (\widehat{s}, \widehat{\mathbf{s}})$
G_4^{ψ}	$\mathbf{v}_{\varphi,\mathrm{pad}}$	μ, $\mathbf{r}_{\mathrm{f}}$'s, h	0	0	$\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{\mu}^0_{\varphi}, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		$\widehat{s}$	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (\widehat{s}, \widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}})$
G_5^{ψ}	$\mathbf{v}_{\varphi,\mathrm{pad}}$	μ, $\mathbf{r}_{\mathrm{f}}$'s, h	0	0	$\widehat{s}\,\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{s}\,\widehat{\mu}^0_{\varphi}, \widehat{s}\,\widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (1, \widehat{\mathbf{r}}_{\mathrm{x}})$
G_6^{ψ}	$\mathbf{v}_{\varphi,\mathrm{pad}}$	μ, $\mathbf{r}_{\mathrm{f}}$'s, h	0	0	$\widehat{\mu}^{\psi}_{\varphi}$	$\propto (\widehat{\mu}^0_{\varphi}, \widehat{\mathbf{r}}_{\varphi,\mathrm{f}})$
	$\mathbf{v}_{\varphi,\mathrm{init}}, \mathbf{v}_{\varphi,q}$					
	$\mathbf{u}_{\mathrm{pad}}$		1		1	
	$\mathbf{u}_{\mathrm{init}}, \mathbf{u}_{t,i,j,\mathbf{W}}$			$\propto (1, \mathbf{r}_{\mathrm{x}})$		$\propto (1, \widehat{\mathbf{r}}_{\mathrm{x}})$

For brevity, the vectors for computing the labels are not spelled out, and the secret key vectors other than those for sk_{φ} are suppressed except in G_0^{ψ}.

See Table 4.11 for the meaning of "$\propto \mathbf{z}$".

  - In the vectors in ct, the values at $\mathsf{pad}^{\mathsf{copy}}$, $\mathsf{s}_{\mathsf{copy}}$ are *copied* to $\mathsf{pad}^{\mathsf{temp}}$, $\mathsf{s}_{\text{1-ABE}}$, and $\widehat{s}$ is multiplied to the new copy.

  As illustrated in Table 4.12, the inner products and the public slots of the secret key vectors remain unchanged, so $\mathsf{G}_1^{\psi} \approx \mathsf{G}_2^{\psi}$ by the function-hiding property of IPFE.

- G_3^{ψ} proceeds identically to G_2^{ψ}, except that in the ciphertext vectors, the $\widehat{s}\,\mathbf{r}_{\mathrm{x}}$ in $\mathsf{s}_{\text{1-ABE}}$ is replaced by an independent and uniformly random $\widehat{\mathbf{s}}$. The indistinguishability between G_2^{ψ} and G_3^{ψ} follows from DDH over $\mathbb{G}_1$:

$$[\![\mathbf{r}_{\mathrm{x}}, \widehat{s}, \widehat{s}\,\mathbf{r}_{\mathrm{x}}]\!]_1 \approx [\![\mathbf{r}_{\mathrm{x}}, \widehat{s}, \widehat{\mathbf{s}}]\!]_1 \qquad \text{for} \qquad \mathbf{r}_{\mathrm{x}}, \widehat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S},\ \widehat{s} \xleftarrow{\$} \mathbb{Z}_p.$$

- G_4^{ψ} proceeds identically to G_3^{ψ}, except that in the ciphertext vectors, the $\widehat{\mathbf{s}}$ in $\mathsf{s}_{\text{1-ABE}}$ is replaced by $\widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}}$. The indistinguishability between G_3^{ψ} and G_4^{ψ} again follows from DDH over $\mathbb{G}_1$:

$$[\![\widehat{s}, \widehat{\mathbf{s}}]\!]_1 \approx [\![\widehat{s}, \widehat{s}\,\widehat{\mathbf{r}}_{\mathrm{x}}]\!]_1 \qquad \text{for} \qquad \widehat{\mathbf{s}}, \widehat{\mathbf{r}}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0,T]\times[N]\times[S]\times\{0,1\}^S},\ \widehat{s} \xleftarrow{\$} \mathbb{Z}_p.$$

- G_5^{ψ} proceeds identically to G_4^{ψ}, except that the multiplier $\widehat{s}$ is moved from $\mathbf{u}$'s back to $\mathbf{v}_{\varphi}$'s as depicted in Table 4.12. By the function-hiding property of IPFE, we have $\mathsf{G}_4^{\psi} \approx \mathsf{G}_5^{\psi}$.

- G_6^{ψ} proceeds identically to G_5^{ψ}, except that the multiplier $\widehat{s}$ is removed. We have $\mathsf{G}_5^{\psi} \approx_{\mathrm{s}} \mathsf{G}_6^{\psi}$.

Lastly, observe that in G_6^{ψ}, only the ciphertext vectors and the vectors for the φ^{th} secret key have non-zero values in $\mathsf{s}_{\text{1-ABE}}$ and those values are generated exactly as specified by our 1-ABE for L (Construction 4.5). We use syntactically the same security proof of 1-ABE for L (Theorem 4.11) to argue $\mathsf{G}_6^0 \approx \mathsf{G}_6^1$.

- Each step in the proof of Theorem 4.11 uses the function-hiding property, DDH, or the special piecewise security, all of which are still valid when translated into the case of G_6^0 and G_6^1 in the only natural way. More specifically, *i)* the function-hiding property is unaffected by the public slot, $\mathsf{s}_{\mathsf{copy}}$, $\mathsf{pad}^{\mathsf{copy}}$, and the additional

vectors (in the other secret keys), as the values being manipulated during the proof are all in $\mathbb{s}_{\text{1-ABE}}$ in the private slot and the additional vectors (having values 0 in $\mathbb{s}_{\text{1-ABE}}$) cannot "detect" the changes in $\mathbb{s}_{\text{1-ABE}}$; and *ii)* the applications of DDH and special piecewise security still go through, because the garbling in $\mathbb{s}_{\text{1-ABE}}$ is generated using randomness independent of everything else in G_6^{ψ}.

- In the translated version of the proof of Theorem 4.11, $\mathbf{v}_{\varphi,\text{pad}}[\mathsf{pad}^{\text{temp}}]$ is always set to $\widehat{\mu}_{\varphi}^{\psi}$. In the final hybrids, both $\widehat{\mu}_{\varphi}^{0}$ and $\widehat{\mu}_{\varphi}^{1}$ are uniformly random and independent of everything else, so they can replace each other.

Therefore, $\mathsf{G}_6^0 \approx \mathsf{G}_6^1$, and by hybrid argument, we conclude $\mathsf{H}_{4,\varphi}^b \approx \mathsf{H}_{4,\varphi+1}^b$. □

Extension to MDDH. The natural way to generalize Construction 4.6 to MDDH_k is to let the public slot store k independent garblings and the private slot provide space for $(k+1)$ garblings — among the $(k+1)$ copies in the private slot, k of them are the counterpart of $\mathbb{s}_{\text{copy}}$ and the last one of them is the counterpart of $\mathbb{s}_{\text{1-ABE}}$. Of course, the underlying 1-ABE scheme must also be based on MDDH_k. This natural generalization uses vectors of dimension $\Theta(k^2)$, because each copy of the garbling already needs $\Theta(k)$ indices.

A simple optimization that shortens the vectors to only $\Theta(k)$ is to realize that the rerandomization of the (half) garblings in the security proof of ABE can be done without creating k copies of 1-ABE, essentially because each 1-ABE already uses k copies of random values to jointly generate the label functions. For this optimization to work, the copy in $\mathbb{s}_{\text{pub}}$ needs to embed k independent pads in the secret keys and k independent random multipliers in the ciphertext, and decryption gives the inner product of the two. For the other two "compartments", $\mathbb{s}_{\text{copy}}$ and $\mathbb{s}_{\text{1-ABE}}$, each only needs to provide space for one pad.

The ABE scheme still uses three copies, and we use $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ to represent an ABE key/ciphertext generated with randomness $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ in $\mathbb{s}_{\text{pub}}, \mathbb{s}_{\text{copy}}, \mathbb{s}_{\text{1-ABE}}$ ($\mathbf{0}$ means no copy is there; pads are omitted). Roughly speaking, the first step (isolating the interaction between sk's and ct from the other ciphertexts) is done as follows.

	Real	Hybrid 1	Hybrid 2	Hybrid 3
sk_φ:	$(\mathbf{R}_{\mathrm{f},\varphi}, \mathbf{0}, \mathbf{0})$	$(\mathbf{R}_{\mathrm{f},\varphi}, \mathbf{0}, \mathbf{0})$	$(\mathbf{R}_{\mathrm{f},\varphi}, \mathbf{A}^\intercal\mathbf{R}_{\mathrm{f},\varphi}, \mathbf{0})$	$(\mathbf{R}_{\mathrm{f},\varphi}, \widehat{\mathbf{R}}_{\mathrm{f},\varphi}, \mathbf{0})$
ct:	$(\mathbf{R}_{\mathrm{x}}, \mathbf{0}, \mathbf{0})$	$(\mathbf{A}\mathbf{R}_{\mathrm{x}}, \mathbf{0}, \mathbf{0})$	$(\mathbf{0}, \mathbf{R}_{\mathrm{x}}, \mathbf{0})$	$(\mathbf{0}, \mathbf{R}_{\mathrm{x}}, \mathbf{0})$

Here, $\mathbf{R}_{\mathrm{f},\varphi}, \widehat{\mathbf{R}}_{\mathrm{f},\varphi} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q_\varphi}$, $\mathbf{R}_{\mathrm{x}} \xleftarrow{\$} \mathbb{Z}_p^{[k]\times([0,T]\times[N]\times[S]\times\{0,1\}^S)}$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}$. The first transition is a change of variable introducing only a negligible statistical error. The second one is due to the function-hiding property of IPFE. The third reduces to $\mathsf{MDDH}^k_{k,\Phi+Q_1+\cdots+Q_\Phi}$.

The second step is to change the pads in $\mathsf{s}_{\mathrm{copy}}$ to be inconsistent one by one. Similar to the proof of Claim 4.15, we achieve this via a loop of Φ iterations, one for each secret key. In the φ^{th} iteration, we *i)* introduce a random multiplier (now a $k \times k$ matrix) in $\mathsf{s}_{\mathrm{copy}}$ of sk_φ, *ii)* isolate the interaction between ct and sk_φ into $\mathsf{s}_{\text{1-ABE}}$ with the multiplier moved into ct, *iii)* switch the randomness in $\mathsf{s}_{\text{1-ABE}}$ of ct to be sampled independently from that in $\mathsf{s}_{\mathrm{copy}}$ in ct, and *iv)* invoke 1-ABE security.

The last step is to remove the ciphertext pad from ct and hide it using the inconsistent pads in sk's, which is straight-forward.

4.6.4 Extension to NL

Applying our construction of KP-ABE for L to non-deterministic Turing machines yields a KP-ABE for NL. In this section, we discuss what modifications need to be made to handle NL.

Non-Deterministic Turing Machines. The definitions of non-deterministic Turing machines (NTM) and time/space bounded computation are the same as Definition 4.3, except with these changes.

- The transition criterion δ can be any relation between (i.e., any subset of the Cartesian product of) $[Q] \times \{0,1\}^2$ and $[Q] \times \{0,1\} \times \{0,\pm 1\}^2$, where $\big((q,x,w),(q',w',\Delta i,\Delta j)\big) \in \delta$ means that if the current state is q, the input tape symbol under scan is x, and the working tape symbol under scan is w, then it is *valid* to transit into state q', overwrite w with w', and move the input and working tape pointer by offsets Δi and Δj, respectively.

- The definition of *hanging in accepting states* is that for all $q \in [Q]$ such that $\mathbf{y}_{\text{acc}}[q] = 1$ and all $x, w \in \{0,1\}$,

$$\delta \cap \big(\{(q,x,w)\} \times ([Q] \times \{0,1\} \times \{0,\pm 1\}^2)\big) = \big\{\big((q,x,w),(q,w,0,0)\big)\big\}.$$

- In the definition of acceptance,

$$\begin{aligned} & \delta(q_t, \mathbf{x}[i_t], \mathbf{W}_t[j_t]) = (q_{t+1}, \mathbf{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t) \\ \text{is changed to} \quad & \big((q_t, \mathbf{x}[i_t], \mathbf{W}_t[j_t]), (q_{t+1}, \mathbf{W}_{t+1}[j_t], i_{t+1} - i_t, j_{t+1} - j_t)\big) \in \delta. \end{aligned}$$

Note that in the case of non-deterministic machines, the machine might move off the tapes with certain choices of transitions, but that does not necessarily lead to rejection — it is just that particular path that is silently dropped. As long as one valid path does not exceed the space bound and land in an accepting state after T steps, the input is accepted.

Arithmetizing NTM Computation. The same matrix multiplication formula (thus the same AKGS) works for non-deterministic computation. We just need to use the non-deterministic version of the transition matrix, which share the same block structure as the deterministic ones.

Let $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$ be an NTM. The *transition blocks* of M are

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j} \in \mathbb{Z}_p^{Q \times Q} \text{ for } x, w, w' \in \{0,1\}, \Delta i, \Delta j \in \{0,\pm 1\}:$$

$$\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = \begin{cases} 1, & \text{if } \big((q,x,w),(q',w',\Delta i,\Delta j)\big) \in \delta; \\ 0, & \text{otherwise.} \end{cases}$$

The transition matrix of M for input of length N and space bound S is

$$\mathbf{M}_{N,S}(\mathbf{x}) \in \{0,1\}^{\mathcal{C}_{M,N,S} \times \mathcal{C}_{M,N,S}},$$

$$\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\text{␣}),(i',j',\mathbf{W}',\text{␣})]$$

$$= \begin{cases} \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}, & \text{if } i'-i, j'-j \in \{0,\pm 1\} \text{ and} \\ & \quad \mathbf{W}[j''] = \mathbf{W}'[j''] \text{ for all } j'' \neq j; \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

It is readily verified that in $\mathbf{M}_{N,S}(\mathbf{x})[(i,j,\mathbf{W},\sqcup),(\sqcup,\sqcup,\sqcup,\sqcup)]$, each transition block appears at most once. The matrix multiplication formula yields the number of accepting paths, as summarized below.

Lemma 4.16. *Let $M=(Q,\mathbf{y}_{\text{acc}},\delta)$ be an NTM, $\mathbf{x}\in\{0,1\}^N$ for some $N\geq 1$, and $T,S\geq 1$. The number of valid computation paths reaching internal configuration $(i,j,\mathbf{W},q)$ when running M starting from the initial configuration $(1,1,\mathbf{0}_S,1)$ with input $\mathbf{x}\in\{0,1\}^N$ and space bound S for T steps is*

$$\left(\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^{\intercal}\left(\mathbf{M}_{N,S}(\mathbf{x})\right)^T\right)^{\intercal}[(i,j,\mathbf{W},q)].$$

Moreover, computing $\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^{\intercal}\left(\mathbf{M}_{N,S}(\mathbf{x})\right)^T\left(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S}\otimes\mathbf{y}_{\text{acc}}\right)$ over the integers *yields the number of accepting computation paths when running M starting from the initial configuration $(1,1,\mathbf{0}_S,1)$ with input $\mathbf{x}\in\{0,1\}^N$ within time T and space S, which is non-zero if and only if M accepts $\mathbf{x}$ within time T and space S.*

We arithmetize time/space bounded computation of M by defining

$$M|_{N,T,S}(\mathbf{x})=\boldsymbol{\iota}_{(1,1,\mathbf{0}_S,1)}^{\intercal}\left(\mathbf{M}_{N,S}(\mathbf{x})\right)^T\left(\mathbf{1}_{[N]\times[S]\times\{0,1\}^S}\otimes\mathbf{y}_{\text{acc}}\right)\quad\text{over }\mathbb{Z}_p,\tag{4.9}$$

which evaluates to 0 if (but not necessarily only if) $\mathbf{x}\in\{0,1\}^N$ is not accepted by M within time T and space S.

Discussion on Correctness. The arithmetization is precise for the complexity class $\mathsf{mod}\mathbb{Z}_p\mathsf{L}$ [BDHM92] (restricted logspace counting class modulo p). Each language $L\in\mathsf{mod}\mathbb{Z}_p\mathsf{L}$ is associated with a non-deterministic logspace Turing machine M such that

$$\begin{aligned}\mathbf{x}\in L\quad&\Longrightarrow\quad\#[\text{accepting paths of }M(\mathbf{x})]\not\equiv 0\pmod p,\\ \mathbf{x}\notin L\quad&\Longrightarrow\quad\#[\text{accepting paths of }M(\mathbf{x})]=0\quad\text{(as an integer)}.\end{aligned}$$

As a special case, $\mathsf{UL}\subseteq\mathsf{mod}\mathbb{Z}_p\mathsf{L}$ for all p, where UL means *unambiguous* logspace — these are languages recognized by a logspace NTM such that for any input, there is at most one accepting path.

However, it is not known whether $\mathsf{NL}\subseteq\mathsf{mod}\mathbb{Z}_p\mathsf{L}$ for any p. Indeed, if the number of accepting paths is a non-zero multiple of p, the computation will be mistakenly

rejected under our arithmetization. We circumvent this correctness issue using pairing groups of *entropic* order.

Definition 4.14 (entropic order). A pairing group sampler (Definition 2.5) GroupGen has *entropic order* if the min-entropy H_λ of p is $\omega(\log \lambda)$:

$$H_\lambda = -\log_2 \max_{p'} \Pr\left[(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda) \ : \ p = p'\right].$$

For 1-ABE and ABE, we relax the correctness requirement so that decryption only needs to succeed with overwhelming probability for all sequence of polynomial-size inputs.

Definition 4.15 (relaxed correctness). An ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ per Definition 2.3 is *correct in the relaxed sense* if the following two conditions hold.

- Either the scheme is not group-based or F, X do not depend on gp (P, Y could depend on gp).
- For all sequence $\{(\mathsf{param}_\lambda, f_\lambda, x_\lambda)\}_{\lambda \in \mathbb{N}}$ such that $\mathsf{param}_\lambda \in \mathsf{Params}_\lambda$, $f_\lambda \in F_{\lambda,\mathsf{param}_\lambda}$, $x_\lambda \in X_{\lambda,\mathsf{param}_\lambda}$, and the lengths of $\mathsf{param}_\lambda, f_\lambda, x_\lambda$ are $\mathrm{poly}(\lambda)$-bounded, it holds that

$$\Pr\left[\begin{array}{c} \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, \mathsf{param}_\lambda) \\ \mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, f_\lambda) \end{array} : \begin{array}{l} P_{\lambda,\mathsf{gp},\mathsf{param}_\lambda}(f_\lambda, x_\lambda) = 1 \text{ but} \\ \text{there exists } y \in Y_{\lambda,\mathsf{gp},\mathsf{param}_\lambda} \text{ such that} \\ \Pr\left[\mathsf{Dec}\left(\begin{array}{l} 1^\lambda, \mathsf{mpk}, f_\lambda, \mathsf{sk}, x_\lambda, \\ \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x_\lambda, y) \end{array}\right) = y\right] \neq 1 \end{array}\right]$$

  is negligible in λ.

The version for 1-ABE is defined analogously.

AKGS, 1-ABE, and ABE for NL. We simply plug in the matrix multiplication formula for NTM into our AKGS, 1-ABE, and ABE constructions for L.

Construction 4.7 (AKGS, 1-ABE, and ABE for NL). The AKGS, 1-ABE, and ABE for NL are obtained by plugging Equation (4.9) into Constructions 4.4 (AKGS), 4.5 (1-ABE), and 4.6 (ABE). Formally (see Definitions 2.3 and 4.6),

$$\mathsf{Params}_{\mathsf{gp}} = \{\bot\}, \qquad F_{\mathsf{gp}} = \{\, M \mid M \text{ is an NTM} \,\}, \qquad Y_{\mathsf{gp}} = \mathbb{G}_T,$$
$$X_{\mathsf{gp}} = \left\{ (\mathbf{x}, 1^T, 1^{2^S}) \,\middle|\, \mathbf{x} \in \{0,1\}^N \text{ for some } N \geq 1 \text{ and } T, S \geq 1 \right\},$$

$$P_{\mathsf{gp}}\big(M,(\mathbf{x},1^T,1^{2^S})\big)=\begin{cases}1, & \text{if } M \text{ accepts } \mathbf{x} \text{ within time } T \text{ and space } S;\\ 0, & \text{otherwise.}\end{cases}$$

In 1-ABE and ABE, Dec outputs $\perp$ if $M|_{N,T,S}(\mathbf{x}) \equiv 0 \pmod{p}$; otherwise, it recovers the message or the pad by dividing the Eval result by $M|_{N,T,S}(\mathbf{x})$.

Theorem 4.17 (¶). *The AKGS in Construction 4.7 is* correct, linear, *and* special piecewise secure *(Definitions 4.7, 4.8, and 4.11) with the same ordering of label functions and randomizers as described in Theorem 4.10.*

Suppose in Construction 4.7, GroupGen *has* entropic order *(Definition 4.14), SXDH (Assumption 4.1) holds over* GroupGen*, and the IPFE schemes are* function-hiding *(Definition 4.5), then the resultant 1-ABE and ABE schemes are* correct in the relaxed sense *(Definition 4.15) and* adaptively secure *(Definitions 4.6 and 2.2).*

If the non-deterministic machines are restricted to unambiguous *ones (Definition 4.3),* GroupGen *need not have* entropic order *and the schemes enjoy* perfect correctness *(Definition 2.1).*

Proof (Theorem 4.17). The AKGS correctness, linearity, and special piecewise security (and ordering) are straight-forward. The security of 1-ABE and ABE also generalizes from the proofs of Theorems 4.11 and 4.14 — by Lemma 4.16, $M|_{N,T,S}$ for non-deterministic Turing machine M always evaluates to 0 if the input is not accepted within the time/space bounds. We focus on the relaxed correctness of 1-ABE and ABE, which amounts to showing that $M|_{N,T,S}(\mathbf{x}) \equiv 0 \pmod{p}$ happens with only negligible probability for accepting computations.

Let the sequence in Definition 4.15 be

$$\mathsf{param}_\lambda = \perp, \qquad M_\lambda = (Q_\lambda, \mathbf{y}_{\mathrm{acc},\lambda}, \delta_\lambda), \qquad (\mathbf{x}_\lambda, 1^{T_\lambda}, 1^{2^{S_\lambda}}) \text{ with } \mathbf{x} \in \{0,1\}^{N_\lambda}.$$

Since the terms are polynomially long, the number of possible transitions (i.e., $|\delta_\lambda|$) and the time constraint T_λ are bounded by $\mathrm{poly}(\lambda)$, which implies that

$$\begin{aligned} A_\lambda &\overset{\text{def}}{=} \#[\text{accepting paths of } M_\lambda(\mathbf{x}_\lambda) \text{ in } T_\lambda \text{ steps}] \\ &\le \#[\text{valid computation paths of } M_\lambda(\mathbf{x}_\lambda) \text{ in } T_\lambda \text{ steps}] \\ &\le |\delta_\lambda|^{T_\lambda} \le 2^{C+\lambda^C} \end{aligned}$$

for some constant $C > 0$. In particular, A_λ has no more than $(C + \lambda^C)$ prime factors. Let the min-entropy of p from GroupGen be $H_\lambda = \omega(\log \lambda)$. By the premise, M_λ accepts $\mathbf{x}_\lambda$ within time T_λ and space S_λ, so $A_\lambda > 0$ and

$$\begin{aligned}&\Pr\left[(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda) \ : \ p \mid A_\lambda\right] \\ &\quad \le \sum_{\substack{p' \text{ prime} \\ p' \mid A_\lambda}} \Pr\left[(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda) \ : \ p = p'\right] \le (C + \lambda^C) \cdot 2^{-H_\lambda}\end{aligned}$$

is negligible in λ. □

4.6.5 ABE for DFA/NFA

A deterministic finite automaton (DFA) can be converted (in polynomial time) to a deterministic Turing machine with space complexity 1 and time complexity N, where N is the length of the input. Similarly, an NFA can be converted to an NTM with space complexity 1 and time complexity N. Therefore, ABE schemes for DFA and NFA are just special cases of our ABE for L and NL.

As an optimization, DFA and NFA do not need to keep track of the input tape pointer in their internal configurations — it is always the current time step. This corresponds to a simpler formula (hence a simpler AKGS) for arithmetizing DFA and NFA. We formally define NFA (with DFA as a special kind of NFA), present the AKGS for it, and discuss how to construct 1-ABE and ABE for NFA. Again, we only consider the binary alphabet. The schemes readily extend to handle any polynomial-size alphabet fixed at set-up time.

Definition 4.16 (NFA). A *non-deterministic finite automaton* is a tuple $(Q, \mathbf{y}_{\mathrm{acc}}, \delta)$, where $Q \ge 1$ is the number of states (we use $[Q]$ as the set of states and 1 the initial state), $\mathbf{y}_{\mathrm{acc}} \in \{0,1\}^Q$ indicates whether each state is accepting, and δ is a relation (state transition relation) between $[Q] \times \{0,1\}$ and $[Q]$. For $\mathbf{x} \in \{0,1\}^N$ for some $N \ge 1$, the NFA *accepts* $\mathbf{x}$ if there exists $q_0, \dots, q_N \in [Q]$ (called an *accepting path*) such that

$$q_0 = 1, \qquad \big((q_{i-1}, \mathbf{x}[i]), q_i\big) \in \delta \quad (\text{for } i \in [N]), \qquad \mathbf{y}_{\mathrm{acc}}[q_N] = 1.$$

A path without the last condition is a *computation path*. An NFA is *deterministic*, if δ is

a function relation. An NFA is *unambiguous,* if for any input $\mathbf{x}$, there is at most one accepting path.

Clearly, a DFA is always unambiguous.

Transition Matrix and Blocks. We use $\boldsymbol{\iota}_q \in \{0,1\}^Q$ to represent the current state of an NFA. For an NFA $M = (Q, \mathbf{y}_{\text{acc}}, \delta)$, its *transition matrix* is

$$\mathbf{M}(x)[q, q'] = \begin{cases} 1, & \text{if } \big((q,x), q'\big) \in \delta; \\ 0, & \text{otherwise.} \end{cases}$$

For all $q \in [Q]$ and $x \in \{0,1\}$, consider $\mathbf{c}^\intercal = \boldsymbol{\iota}_q^\intercal \mathbf{M}(x)$, then $\mathbf{c} \in \{0,1\}^Q$ and $\mathbf{c}[q'] = 1$ if and only if q' is a valid state after the NFA reads x in state q. Inductively, $\boldsymbol{\iota}_q^\intercal \mathbf{M}(x_1) \cdots \mathbf{M}(x_n)$ is a vector that counts the number of computation paths reaching each state starting from state q after reading $x_1, \ldots, x_n$. Let the *transition blocks* of M be $\mathbf{M}_x = \mathbf{M}(x)$ for $x \in \{0,1\}$, then $\mathbf{M}(x) = (1-x)\mathbf{M}_0 + x\mathbf{M}_1$. We arithmetize the computation of NFA by defining

$$M|_N(\mathbf{x}) = \boldsymbol{\iota}_1^\intercal \prod_{i=1}^{N} \big((1 - \mathbf{x}[i])\mathbf{M}_0 + \mathbf{x}[i]\mathbf{M}_1\big) \cdot \mathbf{y}_{\text{acc}} \qquad \text{over } \mathbb{Z}_p \text{ for } \mathbf{x} \in \mathbb{Z}_p^N.$$

In case the NFA is unambiguous, $M|_N$ is binary over $\{0,1\}^N$ and indicates whether M accepts the input.

AKGS for NFA. Garbling $M|_N$ using the recursive mechanism for garbling matrix multiplication yields a special piecewise secure AKGS for NFA.

Construction 4.8 (AKGS for NFA). Let

$$\mathcal{F} = \left\{ M|_N : \mathbb{Z}_p^N \to \mathbb{Z}_p \;\middle|\; M \text{ is an NFA and } p \text{ is prime} \right\}$$

be the set of NFA computations of fixed lengths. The AKGS for $\mathcal{F}$ operates as follows.

- $\mathsf{Garble}\big((M, 1^N, p), \alpha, \beta\big)$ takes as input $M|_N$ and two secrets α, β. It computes the transition blocks $\mathbf{M}_0, \mathbf{M}_1$ of M, samples $\mathbf{r}_0, \ldots, \mathbf{r}_N \xleftarrow{\$} \mathbb{Z}_p^Q$, and defines the label

functions by

$$L_{\mathsf{init}}(\mathbf{x}) = \beta + \boldsymbol{\iota}_1^\intercal \mathbf{r}_0,$$
$$(\text{for } i \in [N]) \qquad \big(L_{i,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_{i-1} + \big((1-\mathbf{x}[i])\mathbf{M}_0 + \mathbf{x}[i]\mathbf{M}_1\big)\mathbf{r}_i,$$
$$\big(L_{N+1,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_N + \alpha \cdot \mathbf{y}_{\mathsf{acc}}.$$

The algorithm collects and outputs the coefficients of those label functions.

- $\mathsf{Eval}\big((M, 1^N, p), \mathbf{x}, \ell_{\mathsf{init}}, (\ell_{i,q})_{i\in[N+1],q\in[Q]}\big)$ takes as input $M|_N$, string $\mathbf{x} \in \{0,1\}^N$, and the labels. It computes the transition blocks $\mathbf{M}_0, \mathbf{M}_1$ of M, sets $\boldsymbol{\ell}_i = (\ell_{i,q})_{q\in[Q]}$ for $i \in [N+1]$, and computes and outputs

$$\ell_{\mathsf{init}} + \boldsymbol{\iota}_1^\intercal \sum_{i=1}^{N+1} \prod_{j=1}^{i-1} \big((1-\mathbf{x}[j])\mathbf{M}_0 + \mathbf{x}[j]\mathbf{M}_1\big) \cdot \boldsymbol{\ell}_i.$$

Remarks. Construction 4.8 coincides with the secret sharing scheme for DFA due to Waters [Wat12]. However, extending it to NFA via path-tracking makes the secret sharing scheme insecure due to *backtracking* attacks, as observed in [Wat12]. In contrast, generalizing it to NFA via path-counting using the matrix multiplication formula retains the special piecewise security. The concurrent work of [GW20a] observed the same generalization.

Syntax, Correctness, and Security. All of them are straight-forward to verify, similarly to Construction 4.4.

Theorem 4.18. *Construction 4.8 is* special piecewise secure *(Definition 4.11) with* L_{init} *being the first label function, the other label functions sorted in increasing order of* i*, and the randomness sorted in the same order as the label functions.*

ABE for NFA. ABE for NFA can be obtained following the same blueprint of Constructions 4.5 and 4.6 and using pairing groups of entropic order. With the pseudo-randomness set to $\mathbf{r}_i = \mathbf{r}_{\mathsf{x}}[i] \cdot \mathbf{r}_{\mathsf{f}}$ for $\mathbf{r}_{\mathsf{x}} \xleftarrow{\$} \mathbb{Z}_p^{[0,N]}$ sampled at encryption time and $\mathbf{r}_{\mathsf{f}} \xleftarrow{\$} \mathbb{Z}_p^Q$ sampled at key generation time, the labels for $\alpha = \mu$ and $\beta = 0$ can be computed as

$$\ell_{\mathsf{init}} = \mathbf{r}_0[1] = \mathbf{r}_{\mathsf{x}}[0] \cdot \mathbf{r}_{\mathsf{f}}[1] = \langle \mathbf{u}_{\mathsf{init}}, \mathbf{v}_{\mathsf{init}} \rangle,$$

$$
\begin{aligned}
(\text{for } i \in [N],\, q \in [Q]) \qquad \ell_{i,q} &= -\mathbf{r}_{i-1}[q] + \big(((1-\mathbf{x}[i])\mathbf{M}_0 + \mathbf{x}[i]\mathbf{M}_1)\mathbf{r}_i\big)[q] \\
&= -\mathbf{r}_{\mathrm{x}}[i-1] \cdot \mathbf{r}_{\mathrm{f}}[q] + \mathbf{r}_{\mathrm{x}}[i] \cdot (1-\mathbf{x}[i])(\mathbf{M}_0\mathbf{r}_{\mathrm{f}})[q] \\
&\qquad + \mathbf{r}_{\mathrm{x}}[i] \cdot \mathbf{x}[i](\mathbf{M}_1\mathbf{r}_{\mathrm{f}})[q] \\
&= \langle \mathbf{u}_i, \mathbf{v}_q \rangle, \\
(\text{for } q \in [Q]) \qquad \ell_{N+1,q} &= -\mathbf{r}_N[q] + \mu \cdot \mathbf{y}_{\mathrm{acc}}[q] = -\mathbf{r}_{\mathrm{x}}[N] \cdot \mathbf{r}_{\mathrm{f}}[q] + \mu \cdot \mathbf{y}_{\mathrm{acc}}[q] \\
&= \langle \mathbf{u}_{N+1}, \mathbf{v}_q \rangle,
\end{aligned}
$$

where the vectors are as follows:

vector	init	rand	acc	tb_0	tb_1	the other indices
$\mathbf{u}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{x}}[0]$	0				
$\mathbf{v}_{\mathrm{init}}$	$\mathbf{r}_{\mathrm{f}}[1]$					
$(i \in [N])$ $\mathbf{u}_i$		$\mathbf{r}_{\mathrm{x}}[i-1]$	0	$\mathbf{r}_{\mathrm{x}}[i] \cdot (1-\mathbf{x}[i])$	$\mathbf{r}_{\mathrm{x}}[i] \cdot \mathbf{x}[i]$	0
$\mathbf{u}_{N+1}$	0	$\mathbf{r}_{\mathrm{x}}[N]$	1	0	0	
$(q \in [Q])$ $\mathbf{v}_q$		$-\mathbf{r}_{\mathrm{f}}[q]$	$\mu \cdot \mathbf{y}_{\mathrm{acc}}[q]$	$(\mathbf{M}_0\mathbf{r}_{\mathrm{f}})[q]$	$(\mathbf{M}_1\mathbf{r}_{\mathrm{f}})[q]$	

As is the case for the other 1-ABE schemes, 1-ABE for NFA will need extra indices for the security proof, the exact number of which can be figured out easily but tediously. Likewise, the ABE scheme will need more copies of the indices for computing the labels (see Construction 4.6).

Corollary 4.19. *Assuming SXDH (Assumption 4.1) over pairing groups of* entropic order *(Definition 4.14), there is a* compact *and* adaptively secure *(Definition 2.2) ABE for NFA (satisfying* relaxed correctness *per Definition 4.15). If the NFA are restricted to* unambiguous *ones (Definition 4.16), the pairing groups need not have* entropic order *and the scheme enjoys* perfect correctness *(Definition 2.1). Moreover, SXDH can be relaxed to* MDDH_k *(Assumption 2.1) for any integer* $k \geq 1$.

4.7 Further Discussions

Here are some further comments about the research around [LL20a,LL20b].

Two Propositions. Lemma 4.2 (function-hiding IPFE from pairing and k-Lin) follows from applying double encryption [LV16,Lin17] to the ALS IPFE scheme [ALS16,Wee17].

Theorem 4.5 (piecewise security becomes special piecewise security after a change of variable for the randomness) is not necessary to derive our ABE schemes, but I personally like it because it reveals the structural essence of piecewise security. Quoting Kaplansky (in *Paul Halmos: Celebrating 50 Years of Mathematics*):

> *We think basis-free, we write basis-free, but when the chips are down*
> *we close the office door and compute with matrices like fury.*

Theorem 4.5 indicates that what special piecewise security is to piecewise security is what matrices are to linear maps.

For the proofs of Lemma 4.2 and Theorem 4.5, see the appendix of [LL20b]. There, we also show how to make our KP-ABE for ABP support delegation.

Limit of AKGS. Despite the claim that combining AKGS and IPFE is agnostic to the underlying computation model, one might wonder for what kinds of computation there exists AKGS. After all, logspace computation and finite automata are represented by repeated matrix multiplication, which is just layered ABP. Understanding the scope of AKGS as defined in this chapter allows us to know the limit of this method for adaptive security. Unfortunately, it turns out that the AKGS size of a function is somewhat tightly related to its ABP size.

Theorem ([Luo20]). *If f has a* piecewise secure *AKGS (Definitions 4.7 and 4.10) with* garbling size *m, then there exists a function g such that g never vanishes and both f/g and $1/g$ can be computed by ABP (Definition 4.2) of* size *no more than $(m+1)$.*

Note that for non-vanishing g, the zero-test predicates of (i.e., the decryption policies defined by) f and f/g are the same. The theorem is similar to the relation [KW93] between the monotone span program size of a function and its linear secret sharing (LSS) size. The class of functions with polynomial-size LSS and AKGS is contained in NC, as both LSS and AKGS are linear-algebraic. Therefore, to prove adaptive security for a larger class of policies,[25] a new method is needed.

[25] From k-Lin and pairing, the most expressive ABE for non-uniform computation is for arithmetic *span*

As shown in Chapters 6 and 7, relaxing the notion of LSS could make it work with arbitrary polynomial-size circuits, a much larger complexity class. Is it possible to relax the notion of AKGS for larger class of functions while keeping it useful for proving adaptive security?

programs, a class slightly larger than ABP. But such ABE is either only selectively secure or read-once. Closing this gap would be satisfying, although it is niche as a research question.

Chapter 5

Succinct and Adaptively Secure ABE for ABP from k-Lin

(Pairing-Based IPFE, Noiseless Linear Garbling)

This chapter is an adaptation of [LL20d,LL20c].[26]

In this chapter, we present an amelioration of the framework in Chapter 4 and construct *succinct* and *adaptively secure* ABE for arithmetic branching programs (ABP), still from k-Lin in asymmetric pairing groups.

5.1 Introduction

Over the past decade, as the research of ABE blossomed, ABE as a primitive has found numerous cryptographic and security applications. To be deployed in practice, it is paramount for a scheme to be efficient, in particular, having fast decryption algorithms and small ciphertexts. It turns out that the size of ABE ciphertexts can be *independent* of the length of the attribute $\mathbf{x}$, and dependent only on the message length and the security parameter — we say such ciphertexts are *succinct* or have *constant size* (in attribute length). Proposed first in [EMN+09] as a goal, succinct ciphertexts are possible because ABE does not require hiding the attribute $\mathbf{x}$, and the decryption algorithm can take $\mathbf{x}$ as input in the clear. Consequently, ciphertexts only need to contain enough information of $\mathbf{x}$ to enforce the *integrity* of computation on $\mathbf{x}$, which does not necessitate encoding the entire $\mathbf{x}$.

Succinct ABE are highly desirable. Real-world applications call for long attributes to achieve sophisticated access control, so succinct ciphertexts are much more

[26] Huijia Lin, Ji Luo: *Succinct and Adaptively Secure ABE for ABP from k-Lin,*

 DOI 10.1007/978-3-030-64840-4_15.

preferable. From a theoretical point of view, succinct ciphertexts have (asymptotically) *optimal* size, as the dependency on the message length and the security parameter is inevitable. From a technical point of view, succinct ABE provides an interesting mechanism for enforcing the integrity of computation without encoding the input. So far, several succinct ABE schemes have been proposed [ALdP11,YAHK14,Tak14,Att16, AHY15,ZGT+16,AT20], but almost all schemes either rely on non-standard assumptions or provide only weak security, as summarized in Tables 5.1 and 5.2.

Table 5.1. Comparison among select KP-ABE schemes with succinct ciphertexts.

reference	**policy**	**assumption**	**adaptive**	|mpk|	|sk|	|ct|	Dec
[ALdP11]	MSP	q-type		$2n+1$	$m(n+1)$	3	3
[YAHK14]	MSP	q-type		$n+2$	$m(n+1)$	2	2
[Tak14]	MSP	2-Lin ✓		$18(2n+1)$	$6m(n+1)$	17	17
[Att16]	MSP	q-type	✓	$6n+42$	$3m(n+3)+9$	18	18
[ZGT+16]	MSP	k-Lin ✓		$2k^2(n+1)$	$2km(n+1)$	$4k$	$4k$
[AT20]	NC^1	MDDH$_k$ ✓	✓	$k(k+1)(n+3)$	$(k+1)m(n+2)$	$2k+2$	$2k+2$
Section 5.6	ABP	MDDH$_k$ ✓	✓	$k(k+2)(n+2)$	$(k+1)m(n+2)+m$	$2k+3$	$2k+3$

MSP, monotone span programs; NC^1, Boolean formulae; ABP, arithmetic branching programs.

n = attribute length, m = policy size, p = group order.

|mpk|, |sk|, |ct| count non-generator elements from the source groups.

Dec counts pairing operations during decryption.

Schemes based on k-Lin can be based on MDDH$_k$ at the cost of a few more elements in mpk.

ABE for arithmetic span programs can be obtained by reduction to MSP [AHY15].

Table 5.2. Comparison among select CP-ABE schemes with succinct secret keys.

reference	**policy**	**assumption**	**adaptive**	|mpk|	|sk|	|ct|	Dec
[Att16]	MSP	q-type	✓	$6n+54$	24	$3m(n+3)+15$	24
[AHY15]	ASP	q-type	✓	$O(n \log p)$	$O(1)$	$O(mn \log p)$	$O(1)$
[AT20]	NC^1	MDDH$_k$ ✓	✓	$O(k^2 n)$	$O(k)$	$O(kmn)$	$O(k)$
Section 5.7	ABP	MDDH$_k$ ✓	✓	$k(k+1)(n+4)+k$	$3k+4$	$(k+1)m(n+2)+m+k+1$	$3k+4$

See Table 5.1 for notations and notes.

Our Results. We first construct a *succinct* key-policy ABE (KP-ABE) simultaneously satisfying the following properties.

(1) *Expressiveness* — It supports policies expressed as arithmetic branching programs.

(2) *Security* — It satisfies adaptive security, as opposed to selective or semi-adaptive security.

(3) *Assumption* — Its security is based on the standard assumptions as opposed to, e.g., q-type assumptions. Specifically, our scheme relies on the matrix decisional Diffie–Hellman (MDDH) assumption over pairing groups.

(4) *Efficiency* — It has succinct ciphertexts. Concretely, each ciphertext consists of 5 group elements when assuming SXDH, and $(2k + 3)$ elements for MDDH_k (implied by k-Lin). Decryption involves the same number of pairing operations. Moreover, our scheme can work with the more efficient asymmetric prime-order pairing groups.

Next, we construct a ciphertext-policy ABE (CP-ABE) with the same properties. Here, the secret keys are tied to attributes and ciphertexts to policies, and succinctness refers to having constant-size secret keys. Our scheme has keys consisting of 7 group elements based on SXDH and $(3k + 4)$ based on MDDH_k.

Besides succinctness (4), achieving the strong notion of adaptive security (2) based on standard assumptions (3) is also highly desirable from both a practical and a theoretical point of view. Prior to the research underlying this chapter, only the recent construction of (KP- and CP-) ABE schemes in [AT20] simultaneously achieves (2)–(4), and their scheme handles policies expressed as Boolean formulae. Our construction expands the class of policies to *arithmetic* branching programs, which is a more expressive model of computation. Our succinct ABE is also the first scheme natively supporting *arithmetic* computation over large fields,[27] whereas all prior succinct ABE schemes (even ones relying on q-type assumptions and/or achieving only selective security) only work natively with Boolean computation. Lastly, we note that even

[27] One can always convert an arithmetic computation into a Boolean one, which we consider non-native.

when relaxing the efficiency requirement from having succinct ciphertext to *compact* ciphertext, whose size grows linearly with the length of the attribute, only a few schemes (see [KW19,GW20a] and Chapter 4) simultaneously achieve (2)–(4), and the most expressive class of policies supported is also ABP, constructed in Chapter 4.

Our Techniques. In Chapter 4, we presented a general framework for constructing *compact* adaptively secure ABE from MDDH. In this chapter, we improve that framework to achieve *succinctness*. The framework in Chapter 4 yields linear-size ciphertexts because it crucially relies on *function-hiding* IPFE [DDM16,LV16]. To recall, IPFE allows issuing secret keys and ciphertexts tied to vectors $\mathbf{v}, \mathbf{u}$ respectively, and decryption reveals their inner product $\langle \mathbf{u}, \mathbf{v} \rangle$. The function-hiding property guarantees that nothing about $\mathbf{u}, \mathbf{v}$ beyond the inner product is revealed, which entails that ciphertexts and secret keys must have size linear in the length of the vectors.

Towards succinctness, our key idea is relaxing function-hiding to a *new and weaker* guarantee, called *gradual simulation security*, where only the vectors encrypted in the ciphertexts are hidden. Such IPFE can have succinct (constant-size) secret keys and can be public-key. We use new ideas to modify the framework in Chapter 4 to work with the weaker gradual simulation security and obtain succinct ciphertexts. Furthermore, we extend the framework to construct ciphertext-policy ABE, which is not handled in Chapter 4. In summary, our techniques give a general and modular approach for constructing succinct and adaptively secure (KP- and CP-) ABE from MDDH.

Related Works. We discuss related works by features.

SUCCINCT ABE. We compare our scheme with previous KP-ABE schemes with constant-size ciphertexts in Table 5.1 and CP-ABE schemes with constant-size secret keys in Table 5.2.

COMPACT ABE. Previous schemes achieving compactness (linear-size keys and ciphertexts, also known as "unbounded multi-use of attributes") and adaptive security based on standard assumptions are [KW19,AT20] for Boolean formulae, [GW20a] for

Boolean branching programs, and Chapter 4 for arithmetic branching programs. Among them, only the work of [AT20] achieves succinctness.

ABE WITH SUCCINCT f-PART. From pairing, we know several ABE schemes with succinct $\mathbf{x}$-part (ciphertexts in KP-ABE and keys in CP-ABE) and compact f-part (linear in the size of f), including ones in this chapter. One can also investigate succinctness in f-part (keys in KP-ABE and ciphertexts in CP-ABE). So far, the only schemes with succinct f-part are KP-ABE for polynomial-size circuits based on LWE [BGG+14] and CP-ABE schemes for NC^1 based on LWE and pairing [AY20], in which the size of f-part depends on the depth but not the size of the circuit. Yet these schemes have compact but non-succinct $\mathbf{x}$-part. This direction is explored in later chapters.

UNBOUNDED ABE. Our succinct ABE schemes have master public keys of size linear in the attribute length. In general, one can further improve the size of master keys to be a constant, which requires the scheme to be able to handle attributes of any polynomial length. Such schemes are called unbounded ABE. So far, there are unbounded and compact ABE schemes (e.g., [KW19] for NC^1). It remains an interesting open problem to construct unbounded succinct schemes.

In summary, to the best of our knowledge, our schemes achieve one of the currently best trade-offs in terms of master public key, secret key, ciphertext sizes.

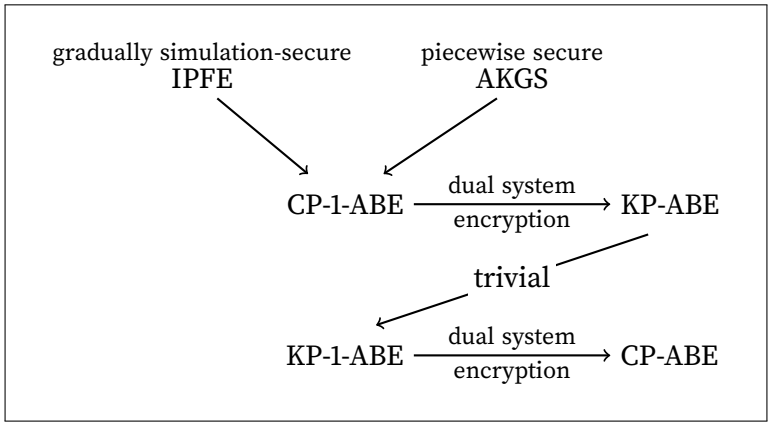


Figure 5.1. Roadmap of ABE constructions in this chapter.

5.2 Technical Overview

We present an overview of our construction of succinct ABE schemes, following the roadmap shown in Figure 5.1.

5.2.1 1-ABE Using Function-Hiding IPFE (Review of Chapter 4)

The core of many ABE schemes is a 1-key 1-ciphertext secure secret-key ABE, or 1-ABE for short. The construction in this chapter improves the 1-ABE scheme for ABP in Chapter 4, which achieves adaptive security but not succinctness.

Suppose we want decryption to recover the message $\mu \in \mathbb{Z}_p$ if (and only if) $f(\mathbf{x}) \neq 0$ for policy function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ and attribute $\mathbf{x} \in \mathbb{Z}_p^n$. This is equivalent to computing $\mu f(\mathbf{x})$ upon decryption. The basic idea in Chapter 4 is that when a key (for f, μ)[28] and a ciphertext (for $\mathbf{x}$) are put together, one can compute a randomized encoding of $\mu f(\mathbf{x})$, denoted by $\widehat{\mu f(\mathbf{x})}$, which reveals $\mu f(\mathbf{x})$ and hence μ if $f(\mathbf{x}) \neq 0$. Since in ABE, we do not try to hide f or $\mathbf{x}$, the randomized encoding only needs to hide μ beyond the output $\mu f(\mathbf{x})$, referred to as the partially hiding property, first introduced by [IW14]. Thanks to the weak security requirement, partially hiding randomized encoding can have extremely simple structure. In Chapter 4, we defined a refined version of it, arithmetic key garbling scheme (AKGS), with the following properties.

- Linear Encoding. The encoding is in the form of

$$\widehat{\mu f(\mathbf{x})} = \big(L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})\big),$$

  where L_j's are *affine* functions of $\mathbf{x}$ and the coefficients of L_j's are *linear* in the message μ and the garbling randomness. The L_j's are called the *label functions* and $\ell_j = L_j(\mathbf{x})$, the *labels*.

- Linear Evaluation. There is a procedure Eval that can compute $\mu f(\mathbf{x})$ from $f, \mathbf{x}$ and the labels, i.e.,

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \mu f(\mathbf{x}).$$

  Importantly, Eval is linear in the labels.[29]

[28] The reason why we put the message μ in the key will become clear later in the overview.

[29] In contrast, linear evaluation is impossible for fully hiding randomized encoding that hides $\mathbf{x}, f$.

The basic security notion of AKGS is simulation security. There needs to be an efficient simulator Sim that can perfectly simulate the labels given $f, \mathbf{x}, \mu f(\mathbf{x})$, i.e.,

$$\big(L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})\big) \equiv (\ell_1, \ldots, \ell_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \mu f(\mathbf{x})).$$

Since the label functions are affine in $\mathbf{x}$ thus linear in $(1, \mathbf{x}^\intercal)$, the labels $\ell_j = L_j(\mathbf{x})$ can be securely computed using a *function-hiding* IPFE. In IPFE, keys $\mathsf{isk}(\mathbf{v})$ and ciphertexts $\mathsf{ict}(\mathbf{u})$ are generated for vectors $\mathbf{v}, \mathbf{u}$, and decryption yields their inner product $\langle \mathbf{u}, \mathbf{v} \rangle$ but nothing else. More precisely, function-hiding says that two sets of keys and ciphertexts encoding different vectors are indistinguishable as long as they yield identical inner products:

$$\big(\{\mathsf{isk}_j(\mathbf{v}_j)\}, \{\mathsf{ict}_i(\mathbf{u}_i)\}\big) \approx \big(\{\mathsf{isk}_j(\mathbf{v}'_j)\}, \{\mathsf{ict}_i(\mathbf{u}'_i)\}\big) \qquad \text{if } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = \langle \mathbf{u}'_i, \mathbf{v}'_j \rangle \text{ for all } i, j.$$

In other words, all the vectors, no matter whether encoded in keys or ciphertexts, are protected. Moreover, function-hiding should hold even when those vectors are chosen adaptively by the adversary, depending on previously observed keys and ciphertexts.

In the 1-ABE scheme in Chapter 4, an ABE key consists of many IPFE keys encoding the coefficients (denoted $\mathbf{L}_j$) of the label functions, and an ABE ciphertext is an IPFE ciphertext encrypting $(1, \mathbf{x}^\intercal)$, as shown below (real algorithms). When they are put together, IPFE decryption recovers exactly the labels $\ell_j = L_j(\mathbf{x}) = \langle \mathbf{L}_j, (1, \mathbf{x}^\intercal) \rangle$, from which we can recover $\mu f(\mathbf{x})$ using the evaluation procedure. A technicality is that known IPFE are built from pairing groups, and decryption only reveals $\mu f(\mathbf{x})$ in the exponent of the target group. Nevertheless, we can obtain $\mu f(\mathbf{x})$ in the exponent, thanks to the linearity of AKGS evaluation.

Intuitively, the scheme in Chapter 4 is secure since IPFE only reveals the labels, and AKGS security guarantees that given the labels, only $\mu f(\mathbf{x})$ is revealed. This idea is simple to formalize in the selective setting, where $\mathbf{x}$ is chosen before querying the key for f. By the function-hiding property, it is indistinguishable to hardwire the labels in the IPFE keys as follows.

$$\begin{array}{cc} \textsc{Real Algorithms} & \textsc{Hybrid} \\ \left\{ \begin{array}{ll} \mathsf{ct}_{\mathbf{x}}: & \mathsf{ict}\,(\ 1, \mathbf{x}^\intercal\ ) \\ \mathsf{sk}_f(\mu): & \{\mathsf{isk}_j(\ \mathbf{L}_j\ )\}_{j\in[m]} \end{array} \right\} & \approx \left\{ \begin{array}{ll} \mathsf{ct}_{\mathbf{x}}: & \mathsf{ict}\,(\ 1\ ,\ \mathbf{x}\ ) \\ \mathsf{sk}_f(\mu): & \{\mathsf{isk}_j(\ L_j(\mathbf{x})\ ,\ \mathbf{0}\ )\}_{j\in[m]} \end{array} \right\} \end{array}$$

After the labels $L_j(\mathbf{x})$ are hardwired and all the label functions are removed, the security of AKGS guarantees that the labels only reveal $\mu f(\mathbf{x})$, and μ is hidden if $f(\mathbf{x}) = 0$. Observe that for selective security, we only need hiding in the IPFE keys and not the IPFE ciphertext.

The above proof fails for adaptive security, in particular in the case where the secret key is queried before the ciphertext (we will focus on this harder case below). At key generation time, $\mathbf{x}$ is unknown and consequently the labels $L_j(\mathbf{x})$ are unknown. We also do not want to hardwire all the labels in the ciphertext as that would make the ciphertext as large as the policy. This problem was solved in Chapter 4 by relying on a stronger security notion of AKGS called *piecewise security*.

- The marginal distribution of $\ell_2, \ldots, \ell_m$ is uniformly random, and ℓ_1 can be *reversely computed* from these other labels $\ell_2, \ldots, \ell_m$ and $f, \mathbf{x}$, by finding the unique ℓ_1 satisfying the constraint of evaluation correctness.[30]

- Every label, except the first, is *conditionally random* even given the coefficients of all subsequent label functions, i.e.,

$$\left(L_j(\mathbf{x}), \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\right) \equiv \left(z, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\right) \qquad \text{for } z \xleftarrow{\$} \mathbb{Z}_p, \quad \text{for all } j > 1.$$

The first property implies a specific simulator — simply sample $\ell_2, \ldots, \ell_m$ at random, then solve for ℓ_1 from the correctness constraint. This strategy is particularly suitable for the adaptive setting, as only the simulation of ℓ_1 depends on the input $\mathbf{x}$. Thus, a conceivable simulation strategy for 1-ABE is to hardwire $\ell_2, \ldots, \ell_m$ in the secret key and ℓ_1 in the ciphertext. This would not hurt the compactness of the ciphertext.

Proving the indistinguishability of the real and the simulated worlds takes two steps. In the first step, the first label $\ell_1 = L_1(\mathbf{x})$ is hardwired into the IPFE ciphertext ict, and then changed to be reversely computed from the other labels and $f, \mathbf{x}$, which is possible since by the time we generate ict, we know both f and $\mathbf{x}$. In the second

[30] The original definition only requires ℓ_1 to be reversely *sampleable*. In Chapter 4, it is shown that the two are equivalent for piecewise security, and we stick to the simpler definition in this overview. In the full definition, ℓ_1 also depends on the computation result. For the purpose of this overview, the result is always $\mu f(\mathbf{x}) = 0$ as the adversary is restricted to non-decrypting queries.

step, each isk_j (for $j > 1$) is, one by one, switched from encoding the label function to encoding a random label. To do so, the j^{th} label $\ell_j = L_j(\mathbf{x})$ is first hardwired into ict, after which it is switched to random relying on piecewise security, and lastly moved back to isk_j. Observe that the proof uses two extra slots in the vectors (one for ℓ_1, the other for each ℓ_j temporarily) and relies on hiding in both the IPFE keys and the IPFE ciphertext.

5.2.2 Lightweight Alternative to Function-Hiding

In a function-hiding IPFE, keys and ciphertexts must be of size at least linear in the vector dimension. This means the resulting ABE scheme can never be succinct. Our first observation is that function-hiding IPFE is an overkill. Since in ABE, $\mathbf{x}$ is not required to be hidden, it is quite wasteful to protect it inside an IPFE ciphertext. Indeed, selective security of the scheme in Chapter 4 does not rely on hiding in the ciphertext.

Our idea to achieve succinctness is to use a non-function-hiding IPFE scheme instead, e.g., a public-key IPFE. Usually the vector in the key is included verbatim as part of the key,[31] and the "essence" of the secret key (excluding the vector itself) could be significantly shorter than the vector. Indeed, many known public-key IPFE schemes [ABDP15,ALS16] have succinct keys.

Since the coefficients of the label functions (which contains information about μ and the garbling randomness) *must* be hidden for the 1-ABE to be secure, and $\mathbf{x}$ is public, we should encrypt the coefficients of the label functions in IPFE *ciphertexts* and use an IPFE *key* for $(1, \mathbf{x}^\intercal)$ to compute the garbling. Since the message μ is together with f and the generation of IPFE ciphertexts is public-key, the 1-ABE scheme is more like a *public-key ciphertext-policy* ABE than a secret-key ABE, except that we only consider security given a single key for some attribute $\mathbf{x}$. Therefore, we redefine 1-ABE as 1-key secure public-key CP-ABE,[32] and the idea is to construct it from a public-key

[31] This alludes to an alternative formulation of public-key PHFE different from Definition 2.1, where Dec is not explicitly given access to a verbatim copy of f. See also discussion in Section 9.1.2.

[32] This definition has the advantage of automatically being multi-ciphertext secure (if secure at all)

IPFE and AKGS as follows.

$$\left.\begin{array}{rl} \mathsf{sk}_{\mathbf{x}}\colon & \mathsf{isk}\,(\ 1, \mathbf{x}^{\mathsf{T}}\ ) \\ \mathsf{ct}_f(\mu)\colon & \{\mathsf{ict}_j(\ \ \mathbf{L}_j\ \ )\}_{j\in[m]} \end{array}\right\} \xrightarrow[\text{decryption}]{\text{IPFE}} \{\langle \mathbf{L}_j, (1, \mathbf{x}^{\mathsf{T}})\rangle = L_j(\mathbf{x}) = \ell_j\}_{j\in[m]} \xrightarrow[\text{evaluation}]{\text{AKGS}} \mu f(\mathbf{x}).$$

Our CP-1-ABE is $\mathbf{x}$-selectively secure if the underlying IPFE is indistinguishability-secure, similar to the selective security proven in Chapter 4.

However, it is not immediate that we can prove adaptive security of this new scheme. The adaptive security proof in Chapter 4 requires hardwiring ℓ_1 and one of ℓ_j's with $\mathbf{x}$, which is now encoded in the secret key without hiding property. Taking a step back, hardwiring a label is really about *removing its label function* and *only using the label*, which is the inner product yielded by IPFE decryption. Our idea is to use simulation security to achieve this goal. A simulator for a public-key IPFE can simulate the master public key, the secret keys, and one (or a few) ciphertext, using only the inner products, and the simulator can do so adaptively. Let's take simulating one ciphertext as an example.

$$\begin{array}{ccc} \textsc{Real} & & \textsc{Simulation} \\ \left\{\begin{array}{c} \mathsf{impk} \\ \{\mathsf{isk}_j(\ \ \mathbf{v}_j\ \ )\}_{j\le J^*} \\ \mathsf{ict}\ (\ \ \mathbf{u}\ \ ) \\ \{\mathsf{isk}_j(\ \ \mathbf{v}_j\ \ )\}_{j>J^*} \end{array}\right\} & \approx & \left\{\begin{array}{c} \widetilde{\mathsf{impk}} \\ \{\widetilde{\mathsf{isk}}_j(\ \ \mathbf{v}_j\ \ |\ \ \varnothing\ \ )\}_{j\le J^*} \\ \widetilde{\mathsf{ict}}\ (\ \ \varnothing\ \ |\ \ \{\langle \mathbf{u}, \mathbf{v}_j\rangle\}_{j\le J^*}\ \ ) \\ \{\widetilde{\mathsf{isk}}_j(\ \ \mathbf{v}_j\ \ |\ \ \langle \mathbf{u}, \mathbf{v}_j\rangle\ \ )\}_{j>J^*} \end{array}\right\} \end{array} \tag{5.$\star$}$$

Here, J^* is the number of keys issued before ciphertext generation. On the left are the honestly generated master public key, secret keys, and ciphertext. On the right is their simulation. The vertical bar separates what the real algorithms use and what the simulator (additionally) uses. Since public-key IPFE completely reveals the key vectors,[33] they are always provided to the simulator. As for the other values...

- Before ciphertext simulation, there is no additional information supplied.
- When the ciphertext is simulated, the vector $\mathbf{u}$ is *not* provided, but its inner

over the secret-key definition. It is also more convenient to use in reductions for full ABE.

[33] Anyone can encrypt the standard basis vectors using impk then decrypt to obtain each component of the vector in a secret key.

products with already simulated keys are provided to the simulator.[34]

- After ciphertext simulation, when simulating a key for $\mathbf{v}_j$, the inner product $\langle \mathbf{u}, \mathbf{v}_j \rangle$ is provided with $\mathbf{v}_j$.

Observe that the values after the vertical bar are exactly those computable using the functionality of IPFE *at that time*, so during simulation, anything about the plaintext vector *not yet* computable by the functionality of IPFE, simply does not exist (information-theoretically) at all. In the setting of our CP-1-ABE, we will simulate an IPFE ciphertext to remove its corresponding label function and only retain the label. Looking from the perspective of hardwiring, when we issue $\mathsf{sk}_{\mathbf{x}} = \mathsf{isk}(1, \mathbf{x}^\intercal)$ after we have created the ciphertext $\mathsf{ct}_f(\mu)$ (in which ict_j has been simulated), the inner product ℓ_j is supplied to the simulator only when we simulate isk, after the simulation of ict_j. This means the label ℓ_j is hardwired into isk.

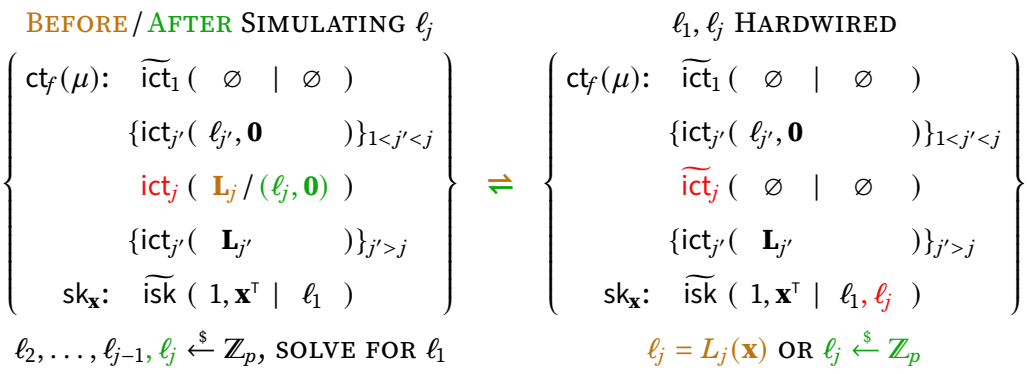
Let's exemplify the proof of adaptive security in the more difficult case, where $\mathsf{sk}_{\mathbf{x}}$ is queried after $\mathsf{ct}_f(\mu)$. We first simulate ict_1 to hardwire the first label into isk.

$$\begin{array}{cc} \textsc{Real Algorithms} & \ell_1 \textsc{ Hardwired} \\ \left\{\begin{array}{rl} \mathsf{ct}_f(\mu)\colon & \mathsf{ict}_1(\ \mathbf{L}_1\ ) \\ & \{\mathsf{ict}_j(\ \mathbf{L}_j\ )\}_{j>1} \\ \mathsf{sk}_{\mathbf{x}}\colon & \mathsf{isk}\,(\,1, \mathbf{x}^\intercal\,) \end{array}\right\} & \approx \left\{\begin{array}{rl} \mathsf{ct}_f(\mu)\colon & \widetilde{\mathsf{ict}}_1(\ \varnothing\ \mid\ \varnothing\ ) \\ & \{\mathsf{ict}_j(\ \mathbf{L}_j\ )\}_{j>1} \\ \mathsf{sk}_{\mathbf{x}}\colon & \widetilde{\mathsf{isk}}\,(\,1, \mathbf{x}^\intercal \mid\ \ell_1 = L_1(\mathbf{x})\ ) \end{array}\right\} \end{array}$$

(We omit the master public key for brevity.) Note that ict_j's for $j > 1$ do not use *ciphertext* simulation, but are created using impk or $\widetilde{\mathsf{impk}}$. Once ℓ_1 is hardwired, we can instead solve for it from the correctness equation.

The second step is to switch $\mathsf{ict}_j(\mathbf{L}_j)$ to $\mathsf{ict}_j(\ell_j, \mathbf{0})$ for $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ one by one, i.e., to simulate ℓ_j as random. To do so, we first simulate ict_j (hardwiring $\ell_j = L_j(\mathbf{x})$ into isk), then switch ℓ_j to random (using piecewise security), and lastly revert ict_j back to encryption (not simulated), but encrypting $(\ell_j, \mathbf{0})$ instead.

[34] Though the number J^* of inner products with already simulated keys is unbounded, since the vectors $\{\mathbf{v}_j\}_{j \le J^*}$ in the keys are public, these inner products are determined by those with any maximal subset of linearly independent $\mathbf{v}_j$'s, the cardinality of which will not exceed the dimension. As such, the simulated ciphertext can still be compact.



During the proof, there are at most two simulated ciphertexts at any time, so it appears that we can simply use a simulation-secure IPFE capable of simulating at most two ciphertexts. This is *not* the case. The tricky part is that the usual formulation of simulation security, i.e., Equation (5.⋆), only requires the *real world* to be indistinguishable from *simulation*. However, in the step of simulating ℓ_j as random, we need to switch ict_j to simulation when $\widetilde{\mathsf{ict}}_1$ is already simulated (and symmetrically, reverting $\widetilde{\mathsf{ict}}_j$ back to encryption while keeping $\widetilde{\mathsf{ict}}_1$ simulated). It is unclear whether this transition is indistinguishable just by simulation security, because the definition says nothing about the indistinguishability of simulating *one more* ciphertext when there is already one simulated ciphertext, i.e.,

$$(\widetilde{\mathsf{impk}}, \widetilde{\mathsf{ict}}_1, \mathsf{ict}_2, \{\widetilde{\mathsf{isk}}_j\}_j) \overset{?}{\approx} (\widetilde{\mathsf{impk}}, \widetilde{\mathsf{ict}}_1, \widetilde{\mathsf{ict}}_2, \{\widetilde{\mathsf{isk}}_j\}_j).$$

Note that when we want to simulate ℓ_j, the computation of ℓ_1 is dependent on $\mathbf{x}$ in some complex[35] manner. We cannot hope to get around the issue by first reverting $\widetilde{\mathsf{ict}}_1$ back to normal encryption then simultaneously simulating $\mathsf{ict}_1, \mathsf{ict}_j$, as we do not know what to encrypt in ict_1.

Gradually Simulation-Secure IPFE. To solve the problem above, we define a stronger notion of simulation security, called *gradual simulation security*. It bridges the gap by capturing the idea that it is indistinguishable to simulate more ciphertexts even when some ciphertexts (and all the keys) are already simulated, as long as the total number of simulated ciphertexts does not exceed a preselected threshold. We show that the IPFE scheme in [ALS16] can be adapted for gradual simulation security. The length

[35] In fact, it is as complex as the computation of $f(\mathbf{x})$.

of secret keys grows linearly in the maximum number of simulated ciphertexts, but not in the vector dimension. Plugging it into our CP-1-ABE construction, we obtain a CP-1-ABE with succinct keys.

Lesser Solution. We remark that another way to get around the issue of simulation security is to notice that there are at most two ciphertexts simulated at any time and one of them is ict_1. Therefore, we can simply prepare two instances of IPFE (with independently generated master public and secret keys), one dedicated to ict_1 and the other to ict_j's (for $j > 1$). During the proof, the instance for ict_1 is always simulated, and the other instance is switched between simulation and normal. This idea appears in a concurrent work [AGW20]. The downside of this method is that using two instances doubles 1-ABE key size. In contrast, the solution using gradually simulation-secure IPFE only needs one more element of $\mathbb{Z}_p$ in CP-1-ABE key.

Comparison with Previous Techniques. Previous works constructing succinct ABE only natively support Boolean computations, whereas our method natively supports *arithmetic* computations. In [ALdP11,YAHK14,Tak14,Att16,AHY15], succinct ABE schemes are obtained from a special succinct ABE for set-membership policies (keys are tied to a set S, ciphertexts are tied to an element x, and decryption succeeds if $x \in S$). Based on ABE for set-membership policies, one can construct ABE for monotone span programs, or equivalently, policies admitting linear secret sharing schemes. Those ingredients (the special ABE, MSP, LSS) are inherently only native to Boolean computations. Among them, the work of [AHY15] constructs succinct ABE for arithmetic span programs by reduction to MSP at the cost of a $\Theta(\log p)$ blow-up in key sizes.

The succinct ABE schemes in [ZGT+16,AT20] are implicitly based on IPFE with succinct keys. The IPFE is used to compute LSS and it is used in a non-black-box way. In contrast, our 1-ABE can be constructed from any IPFE in a modular and black-box fashion, and we use it for arithmetic branching programs.

5.2.3 Dual System Encryption for Full ABE

To lift our CP-1-ABE to full KP-ABE, we need to flip the position of attributes and

policies. Our idea is to use CP-1-ABE as a key encapsulation mechanism. More specifically, a KP-ABE key for policy f is a CP-1-ABE ciphertext $\mathsf{cpct}_f(\mu)$, where μ is the message in CP-1-ABE and encapsulated key in KP-ABE. A KP-ABE ciphertext for attribute $\mathbf{x}$ and message m consists of a CP-1-ABE key $\mathsf{cpsk}_{\mathbf{x}}$ and the masked message $(\mu + m)$. If decryption is authorized, we can recover μ by running CP-1-ABE decryption, then use it to unmask the message. Observe that the security of KP-ABE aligns with the security of CP-1-ABE, namely, in the KP-ABE security game...

- We only need to handle one ciphertext, for which we rely on 1-key security of CP-1-ABE.
- We need to handle multiple keys, which corresponds to multi-ciphertext security of CP-1-ABE. Since our CP-1-ABE is public-key, it indeed satisfies multi-ciphertext security given only one key.

However, we need to resolve the issue that encryption of KP-ABE is now secret-key, since we need to know both the master secret key of CP-1-ABE and μ (part of the master secret key of KP-ABE) to generate KP-ABE ciphertext.

We observe that our CP-1-ABE is linear, i.e., the spaces of $\mathsf{cpmsk}, \mathsf{cpsk}, \mathsf{cpct}, \mu$ are vector spaces over $\mathbb{Z}_p$, and[36]

$$k_1\mathsf{cpsk}_{\mathbf{x}}(\mathsf{cpmsk}_1) + k_2\mathsf{cpsk}_{\mathbf{x}}(\mathsf{cpmsk}_2) = \mathsf{cpsk}_{\mathbf{x}}(k_1\mathsf{cpmsk}_1 + k_2\mathsf{cpmsk}_2),$$
$$k_1\mathsf{cpct}_f(\mathsf{cpmsk}_1, \mu_1) + k_2\mathsf{cpct}_f(\mathsf{cpmsk}_2, \mu_2) = \mathsf{cpct}_f(k_1\mathsf{cpmsk}_1 + k_2\mathsf{cpmsk}_2, k_1\mu_1 + k_2\mu_2).$$

Here, $\mathsf{cpsk}_{\mathbf{x}}(\mathsf{cpmsk})$ and $\mathsf{cpct}_f(\mathsf{cpmsk}, \mu)$ represent that they are generated in the CP-1-ABE instance whose master secret key is cpmsk. We instantiate our CP-1-ABE with an IPFE such that the keys are linear in the master secret key and the ciphertexts are linear in the concatenation of the master secret key and the encrypted vector. CP-1-ABE master secret key and keys are IPFE master secret key and keys, so cpsk's are linear in cpmsk. CP-1-ABE ciphertexts are IPFE ciphertexts for the label functions of AKGS, and AKGS is linear with respect to the message μ, so cpct's are linear in (cpmsk, μ).

[36]The randomness in key generation/encryption should also take part in the linear homomorphism, but we omit it in this overview for brevity.

Let $\mathbb{G}$ be a group of prime order p. Concretely, our cpmsk and cpsk's consist of elements of $\mathbb{Z}_p$. Now if we encode cpmsk in $\mathbb{G}$, by linearity we can compute cpsk in $\mathbb{G}$, and we denote this fact by

$$[\![\mathsf{cpsk}_{\mathbf{x}}(\mathsf{cpmsk})]\!] = \mathsf{cpsk}_{\mathbf{x}}([\![\mathsf{cpmsk}]\!]).$$

Assume for the moment that this can also be done for cpct's and decryption still works.[37] Given the linearity, we can employ dual system encryption [Wat09] to make the scheme public-key. In prime-order groups, the classic dual system encryption can be regarded as hash proof systems based on MDDH [CS02,EHK+13].[38]

Take MDDH_1 (DDH assumption) for example. KP-ABE prepares two instances of CP-1-ABE and two messages, and publishes the projection of them along a randomly sampled vector (b_1, b_2) in the exponent:

$$\begin{aligned}\mathsf{kpmpk} &= [\![b_1, b_2,\ b_1\mathsf{cpmsk}_1 + b_2\mathsf{cpmsk}_2,\ b_1\mu_1 + b_2\mu_2]\!] \qquad \text{for } b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p,\\ \mathsf{kpmsk} &= (\mathsf{cpmpk}_1, \mathsf{cpmpk}_2,\ \mathsf{cpmsk}_1, \mathsf{cpmsk}_2,\ \mu_1, \mu_2).\end{aligned}$$

Encryption is now public-key. A KP-ABE ciphertext simply uses a random CP-1-ABE master secret key in the projected space (a.k.a. the *normal* space in dual system encryption) and use the projected μ to mask the message. A KP-ABE key consists of two CP-1-ABE ciphertexts, one in each instance encrypting the corresponding encapsulated key.

$$\begin{aligned}\mathsf{kpct}_{\mathbf{x}}(m) &= \big(s[\![b_1, b_2]\!],\ \mathsf{cpsk}_{\mathbf{x}}(s[\![b_1\mathsf{cpmsk}_1 + b_2\mathsf{cpmsk}_2]\!]),\ m + s[\![b_1\mu_1 + b_2\mu_2]\!]\big)\\ &\qquad\qquad \text{for } s \xleftarrow{\$} \mathbb{Z}_p,\\ \mathsf{kpsk}_f &= \big(\mathsf{cpct}_f(\mathsf{cpmsk}_1, \mu_1), \mathsf{cpct}_f(\mathsf{cpmsk}_2, \mu_2)\big).\end{aligned}$$

To decrypt, we first use linearity to combine the two CP-1-ABE ciphertexts into

$$\begin{aligned}&\mathsf{cpct}_f([\![sb_1\mathsf{cpmsk}_1 + sb_2\mathsf{cpmsk}_2,\ sb_1\mu_1 + sb_2\mu_2]\!])\\ &\qquad = [\![sb_1]\!]\mathsf{cpct}_f(\mathsf{cpmsk}_1, \mu_1) + [\![sb_2]\!]\mathsf{cpct}_f(\mathsf{cpmsk}_2, \mu_2).\end{aligned}$$

[37] In our case, cpct's are already group-encoded, and this is where pairing comes in.

[38] A few examples are [Wee17,ALS16,KW19,GW20a]. Wee [Wee14] also notices that certain usage of dual system encryption in composite-order groups is reminiscent of hash proof systems. There are other ways to use dual system encryption that do not resemble hash proof systems.

The master secret key of the combined cpct matches that of the cpsk in the KP-ABE ciphertext, and CP-1-ABE decryption will recover $[\![sb_1\mu_1 + sb_2\mu_2]\!]$, using which we can unmask to obtain the message m.

To argue security, we first replace $[\![sb_1, sb_2]\!]$ used in the challenge ciphertext by $[\![a_1, a_2]\!]$ for random $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_p$ (using DDH), which is not colinear with (b_1, b_2) with overwhelming probability. Ciphertexts in this form are said to be *semi-functional* in dual system encryption.

By the linearity, we can look at the ABE scheme from a new basis, namely (b_1, b_2) and (a_1, a_2). We denote the CP-1-ABE components and μ's in this basis with primes, e.g., $\mathsf{cpmsk}'_1 = b_1\mathsf{cpmsk}_1 + b_2\mathsf{cpmsk}_2$ and $\mu'_2 = a_1\mu_1 + a_2\mu_2$. The KP-ABE master public key reveals cpmsk'_1 but is independent of cpmsk'_2. A KP-ABE secret key for policy f is essentially $\mathsf{cpct}_f(\mathsf{cpmsk}'_1, \mu'_1)$ and $\mathsf{cpct}_f(\mathsf{cpmsk}'_2, \mu'_2)$. The challenge ciphertext has $\mathsf{cpsk}_{\mathbf{x}}(\mathsf{cpmsk}'_2)$, and the message is masked by μ'_2. By CP-1-ABE security, μ'_2 (in cpct's) should be hidden, which means the message in the challenge ciphertext is hidden by μ'_2.

The proof completes by replacing μ'_2 in all the KP-ABE keys by random. ABE keys in this form are said to be *semi-functional* in dual system encryption.

Lastly, to base the scheme on MDDH_k, we use $(k+1)$ instances of CP-1-ABE, publish a k-dimensional projection (*normal* space), and reserve the unpublished dimension for the security proof (*semi-functional* space).

CP-ABE from KP-1-ABE. By symmetry, we can apply the transformation to obtain CP-ABE from KP-1-ABE. Moreover, our KP-ABE trivially serves as a KP-1-ABE. Therefore, the DDH-based scheme is (ignoring group encoding)

$$\begin{aligned}
\mathsf{cpmpk} &= (d_1, d_2,\ d_1\mathsf{kpmsk}_1 + d_2\mathsf{kpmsk}_2,\ d_1\nu_1 + d_2\nu_2) && \text{for } d_1, d_2 \xleftarrow{\$} \mathbb{Z}_p,\\
\mathsf{cpmsk} &= (\mathsf{kpmpk}_1, \mathsf{kpmpk}_2,\ \mathsf{kpmsk}_1, \mathsf{kpmsk}_2,\ \nu_1, \nu_2),\\
\mathsf{cpsk} &= \big(\mathsf{kpct}_{\mathbf{x}}(\mathsf{kpmsk}_1, \nu_1), \mathsf{kpct}_{\mathbf{x}}(\mathsf{kpmsk}_2, \nu_2)\big),\\
\mathsf{cpct} &= \big(td_1, td_2,\ \mathsf{kpsk}_f(t(d_1\mathsf{kpmsk}_1 + d_2\mathsf{kpmsk}_2)),\ m + t(d_1\nu_1 + d_2\nu_2)\big) && \text{for } t \xleftarrow{\$} \mathbb{Z}_p.
\end{aligned}$$

Again, KP-1-ABE is used to encapsulate keys ν_1, ν_2, whose projection masks the message in CP-ABE. Dual system encryption or hash proof system is used to obtain public-key

encryption by publishing a random projection of KP-1-ABE master secret keys (in this case, along (d_1, d_2)).

One final observation is that only μ_1, μ_2 in KP-(1-)ABE need to be duplicated and projected, yielding only a small overhead in CP-ABE compared to KP-ABE. We leave the details to the main body and further discussions to Section 5.8.

We note that once we obtain KP-ABE from CP-1-ABE, going to CP-ABE using the same method is natural and simple.

5.3 Preliminaries

Table 5.3 explains select single-letter symbols used in this chapter.

Table 5.3. Select single-letter symbols in this chapter.

symbol	meaning
b	challenge bit
$f, \mathbf{x}, n$	function, input, input length (ABP)
α, β, γ	linear coefficient, constant, evaluation result
$L, \mathbf{L}, \ell, m, j$	label function, coefficients, label, count, index
n	vector dimension (IPFE)
T	maximum number of simulated ciphertexts (IPFE)
$\mathbf{w}$	master secret key of OTP-IPFE
$\mathbf{W}$	master secret key of various schemes
$\mathbf{A}, \mathbf{s}$	MDDH_k matrix, randomness
$\mathbf{c}$	vector outside row space of $\mathbf{A}$
$\mathbf{a}^\perp$	vector annihilated by (rows of) $\mathbf{A}$ but not $\mathbf{c}^\intercal$
$\mathbf{B}, \mathbf{r}, \mathbf{c}, \mathbf{b}^\perp$	similar to $\mathbf{A}, \mathbf{s}, \mathbf{c}, \mathbf{a}^\perp$
$\mathbf{D}, \mathbf{t}, \mathbf{c}, \mathbf{d}^\perp$	—〃—
$\boldsymbol{\mu}, \mathbf{U}, \boldsymbol{\nu}$	materials for key encapsulation
Q, q, δ	query count, query index, one-time pad

This chapter reuses preliminaries and concepts from Chapter 4. One exception is that we no longer use general index sets and instead revert back to integer-based indexing. The relevant parts are restated, followed by additional preliminaries.

Coefficient Vector. An affine function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ is conveniently associated with its coefficient vector $\mathbf{f} \in \mathbb{Z}_p^{n+1}$ (the same letter in boldface) such that $f(\mathbf{x}) = \mathbf{f}^\intercal \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$.

Definitions 4.2 and 4.1 (ABP and zero-test predicates). An *arithmetic branching program* over $\mathbb{Z}_p^n$ is a weighted directed acyclic graph (V, E, w, s, t) with two distinguished vertices, where V, E are the sets of vertices and edges, $w : E \to (\mathbb{Z}_p^n \to \mathbb{Z}_p)$ specifies an affine weight function for each edge, and $s, t \in V$ are the distinguished vertices. The in-degree of s and the out-degree of t are 0. Such a program *computes* a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ defined by

$$f(\mathbf{x}) = \sum_{\substack{s\text{-}t \text{ path} \\ e_1 \cdots e_i}} \prod_{j=1}^{i} w(e_j)(\mathbf{x}).$$

Further define its *zero-test predicates* as

$$f_{\neq 0}(\mathbf{x}) = \begin{cases} 0, & \text{if } f(\mathbf{x}) = 0; \\ 1, & \text{if } f(\mathbf{x}) \neq 0; \end{cases} \qquad f_{=0}(\mathbf{x}) = \neg f_{\neq 0}(\mathbf{x}) \qquad \text{for } \mathbf{x} \in \mathbb{Z}_p^n.$$

The *size* of the ABP is $|V|$, the number of vertices. The class of all ABP is

$$\mathsf{ABP} = \left\{ f \,\middle|\, f \text{ is an ABP over } \mathbb{Z}_p^n \text{ for some prime } p \text{ and input length } n \right\}.$$

When there is no ambiguity, we use f to represent both the ABP and the function it computes, and ABP both the class of all ABP and the class of all functions computed by some ABP.

Arithmetic Key Garbling. We rely on an arithmetic key garbling scheme for ABP.

Definitions 4.7 and 4.8 ((linear) AKGS). A *(linear) arithmetic key garbling scheme* for a function class $\mathcal{F} = \{f\}$, where each $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ for some p, n specified by f, consists of two efficient algorithms.

- $\mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ takes as input $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$. It uses *uniform* randomness $\mathbf{r} \in \mathbb{Z}_p^{m'}$. The algorithm outputs the coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m \in \mathbb{Z}_p^{n+1}$ of m affine functions $L_1, \ldots, L_m : \mathbb{Z}_p^n \to \mathbb{Z}_p$ (the *label functions*). The vectors $\mathbf{L}_j$ are *linear* in $(\alpha, \beta, \mathbf{r})$. The dimension m' of randomness and the number m of label functions (the *garbling size* of f) are solely determined by f.

- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ takes as input $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}$, $\mathbf{x} \in \mathbb{Z}_p^n$, and $\ell_1, \ldots, \ell_m \in \mathbb{Z}_p$ (the *labels*). The algorithm is deterministic and outputs a value in $\mathbb{Z}_p$ that is *linear* in the labels.

The scheme must be *correct*, i.e., for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^n$,

$$\Pr\left[\begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_j \leftarrow L_j(\mathbf{x}) \text{ for all } j \in [m] & : \quad \mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \alpha f(\mathbf{x}) + \beta \end{array}\right] = 1.$$

Definition 4.10 (piecewise security). An AKGS (Definition 4.7) is *piecewise secure* if the following conditions hold.

- The first label is *reversely sampleable* from the other *labels* together with f and $\mathbf{x}$. The sampling must be perfect even given all the other *label functions*. Formally, there exists an efficient algorithm $\mathsf{RevSamp}$ such that for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^n$, the following distributions are identical:

$$\left\{\begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_1 \leftarrow L_1(\mathbf{x}) & : \quad (\ell_1, \mathbf{L}_2, \ldots, \mathbf{L}_m) \end{array}\right\},$$

$$\left\{\begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \qquad\qquad \ell_j \leftarrow L_j(\mathbf{x}) \text{ for all } j \in [m],\ j > 1 & : \quad (\ell_1, \mathbf{L}_2, \ldots, \mathbf{L}_m) \\ \qquad\qquad \ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m) & \end{array}\right\}.$$

- For the other *labels*, each is *marginally random*[39] even given all the *label functions* after it. Formally, for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}$, $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{x} \in \mathbb{Z}_p^n$, and all $j \in [m]$ such

[39] This is an unfortunate misnomer since [LL20a,LL20b]. The correct wording is "conditionally random". We keep the original version for consistency.

that $j > 1$, the following distributions are identical:

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \leftarrow L_j(\mathbf{x}) & : \quad (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \end{array} \right\},$$

$$\left\{ \begin{array}{ll} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \xleftarrow{\$} \mathbb{Z}_p & : \quad (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \end{array} \right\}.$$

Construction 4.1 and Theorem 4.6 (piecewise secure AKGS for ABP). *There exists a* piecewise secure *AKGS (Definitions 4.7 and 4.10) for ABP (Definition 4.2), for which the* garbling size *of an ABP is the same as its* size.

Batch Notation. Throughout this chapter, we will use a vectorized version of the garbling algorithm. Let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^k$, then $\mathsf{Garble}(f, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is executed component-wise with independent randomness and the outputs are concatenated:

$$\begin{aligned} &(\text{for } t \in [k]) \qquad (\mathbf{L}_1^t, \ldots, \mathbf{L}_m^t) \xleftarrow{\$} \mathsf{Garble}(f, \boldsymbol{\alpha}[t], \boldsymbol{\beta}[t]), \\ &(\text{for } j \in [m]) \qquad \mathbf{L}_j = \begin{pmatrix} \mathbf{L}_j^1 \\ \vdots \\ \mathbf{L}_j^k \end{pmatrix} = \sum_{t=1}^{k} \boldsymbol{\iota}_t \otimes \mathbf{L}_j^t, \\ &\qquad\qquad \text{output } (\mathbf{L}_1, \ldots, \mathbf{L}_m). \end{aligned}$$

In the vectorized version, the randomness is a matrix and each row of the matrix is used for one invocation of the non-vectorized garbling. This notation is compatible with tensor products, as summarized in the following lemma.

Lemma 5.1 (mixing and stitching). *Suppose* $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$.

Let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^k$, $\mathbf{R} \in \mathbb{Z}_p^{k \times m'}$, $\mathbf{c} \in \mathbb{Z}_p^k$, *and define*

$$(\mathbf{L}_1, \ldots, \mathbf{L}_m) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}, \boldsymbol{\beta}; \mathbf{R}), \qquad (\mathbf{L}'_1, \ldots, \mathbf{L}'_m) \leftarrow \mathsf{Garble}(f, \mathbf{c}^\intercal \boldsymbol{\alpha}, \mathbf{c}^\intercal \boldsymbol{\beta}; \mathbf{c}^\intercal \mathbf{R}),$$

then $\mathbf{L}_j^\intercal (\mathbf{c} \otimes \mathbf{I}_{n+1}) = (\mathbf{L}'_j)^\intercal$ *for all* $j \in [m]$.

Now let $\alpha, \beta \in \mathbb{Z}_p$, $\mathbf{r} \in \mathbb{Z}_p^{m'}$, $\mathbf{d} \in \mathbb{Z}_p^k$, *and define*

$$(\mathbf{L}'_1, \ldots, \mathbf{L}'_j) \leftarrow \mathsf{Garble}(f, \alpha, \beta; \mathbf{r}), \qquad (\mathbf{L}_1, \ldots, \mathbf{L}_j) \leftarrow \mathsf{Garble}(f, \alpha\mathbf{d}, \beta\mathbf{d}; \mathbf{d}\mathbf{r}^\intercal),$$

then $\mathbf{d} \otimes \mathbf{L}'_j = \mathbf{L}_j$ *for all* $j \in [m]$.

5.4 IPFE with Gradual Simulation Security

In an inner-product functional encryption scheme, secret keys and ciphertexts are associated with vectors. Decryption reveals the inner product and nothing more about the plaintext vector. We will use a public-key IPFE as our building block for 1-ABE, a precursor to full ABE.

In this chapter, we consider IPFE schemes based on MDDH-hard groups (not necessarily with pairing), where the ciphertext encodes the encrypted vector in the exponent of the group and decryption computes the inner product in the exponent.

Definition 5.1 (IPFE). Let GroupGen be a group sampler (Definition 2.4). An *inner-product functional encryption scheme* based on GroupGen is a PHFE (Definition 2.1) with

$$\begin{aligned} \mathsf{Params}_{\lambda,\mathsf{gp}} &= \{\,(1^n, 1^T) \mid n, T \in \mathbb{N}\,\}, & \mathbf{v} &\in F_{\lambda,\mathsf{gp},1^n,1^T} = \mathbb{Z}_p^n, \\ X_{\lambda,\mathsf{gp},1^n,1^T} &= \{\bot\}, & [\![\mathbf{u}]\!] &\in Y_{\lambda,\mathsf{gp},1^n,1^T} = \mathbb{G}^n, \\ Z_{\lambda,\mathsf{gp},1^n,1^T} &= \mathbb{G}, & \varphi_{\lambda,\mathsf{gp},1^n,1^T}(\mathbf{v}, \bot, [\![\mathbf{u}]\!]) &= [\![\mathbf{u}^\intercal \mathbf{v}]\!]. \end{aligned}$$

The quantity T is only relevant for security (Definition 5.2). The scheme is *succinct* if the length of isk[40] is independent of n and only depends on λ, gp, T.

The level of security increases with the value of T, as explained in the following definition.

Gradual Simulation Security. When building the 1-ABE scheme, we rely on the notion of gradual simulation security, which is stronger than the usual simulation security (see [ALMT20]). Roughly speaking, on top of the requirement that simulation should be indistinguishable from the real scheme, the notion stipulates that even when some ciphertexts are already simulated, whether another ciphertext is honest or simulated should be indistinguishable. The parameter T specifies the maximum number of ciphertexts that can be simulated.

[40] Although this kind of IPFE is formally defined as a PHFE, by convention, we still write $\mathsf{impk}, \mathsf{imsk}, \mathsf{isk}, \mathsf{ict}$ for its components. Compare with Definition 4.4, which directly defines another kind of IPFE and specifies its component names.

To navigate around the many indices involved in the definition, it is the easiest to keep in mind that i (hence I, I^*, I_t) always counts the ciphertexts, and that j (hence J, J^*, J_t) always counts the keys.

Definition 5.2 (gradual simulation security). A *simulator* for an IPFE scheme (Definition 5.1) consists of three efficient algorithms.[41]

- $\mathsf{SimSetup}(1^\lambda, \mathsf{gp}, 1^n, 1^T)$ takes the same input as Setup. It outputs a simulated master public key impk and an internal state σ.[42]
- $\mathsf{SimKeyGen}(\sigma, \mathbf{v}, z_1, \ldots, z_I)$ takes as input the internal state σ, a vector $\mathbf{v}$, and a list $z_1, \ldots, z_I$ of inner products in $\mathbb{Z}_p$ (which are the intended inner products between this simulated key and all previously simulated ciphertexts). It outputs a simulated secret key isk and a new state σ'.
- $\mathsf{SimEnc}(\sigma, z_1, \ldots, z_J)$ takes as input the internal state σ and a list $z_1, \ldots, z_J$ of inner products in $\mathbb{Z}_p$ (which are the intended inner products between this simulated ciphertext and all previously simulated keys). It outputs a simulated ciphertext ict and a new state σ'.

The simulator *gradually T-simulates* the scheme if it satisfies both *key simulation security* and *T-ciphertext simulation security* defined below.

The scheme is *gradually T-simulation-secure* if it can be gradually T-simulated by some simulator. It is *gradually simulation-secure* if there exists a simulator that can gradually T-simulate the scheme for all $T = \mathrm{poly}(\lambda)$.

KEY SIMULATION SECURITY. Roughly speaking, this captures the idea that it is indistinguishable to interact with the real authority (who generates and distributes impk and isk's) versus the simulator issuing simulated impk and isk's (without simulating any ciphertext). We require $\mathsf{Exp}_{\mathrm{real}} \approx \mathsf{Exp}_{\mathrm{sim}}$, which proceed as follows when run with an adversary $\mathcal{A}$.

[41] To be precise, the efficiency requirement is that whenever $\mathsf{Exp}_{\mathrm{real}}$ finishes in polynomial time, so must $\mathsf{Exp}_{\mathrm{sim}}$ and $\mathsf{Exp}^b_{T\text{-GS}}$. (See below for the definitions of these experiments.)

[42] It is understood that the state is maintained by one instance of simulator. Except in definitions, its creation, persistence, and update are suppressed when there is no danger of ambiguity.

- **Setup.** Run $(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda)$. Launch $\mathcal{A}(1^\lambda, \mathsf{gp})$ and receive $(1^n, 1^T)$ from it. Run

$$(\text{in } \mathsf{Exp}_{\text{real}}) \qquad (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, 1^n, 1^T);$$
$$(\text{in } \mathsf{Exp}_{\text{sim}}) \qquad (\mathsf{impk}, \sigma) \xleftarrow{\$} \mathsf{SimSetup}(1^\lambda, \mathsf{gp}, 1^n, 1^T).$$

  Send impk to $\mathcal{A}$.

- **Challenge.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ submits a vector $\mathbf{v}_j$. Upon this challenge, run

$$(\text{in } \mathsf{Exp}_{\text{real}}) \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, \mathbf{v}_j);$$
$$(\text{in } \mathsf{Exp}_{\text{sim}}) \qquad (\mathsf{isk}_j, \sigma') \xleftarrow{\$} \mathsf{SimKeyGen}(\sigma, \mathbf{v}_j).$$

  Update $\sigma \leftarrow \sigma'$ and send isk_j to $\mathcal{A}$.

- **Guess.** The adversary outputs a bit b', the outcome of the experiment.

Note that there is no ciphertext challenge in the experiments. The adversary can generate ciphertexts on its own using impk.

T-Ciphertext Simulation Security. Roughly speaking, this captures the idea that when interacting with the simulator, it is indistinguishable whether any subset of ciphertexts are normally generated or simulated, as long as the number of simulated ciphertexts does not exceed T. In the experiments below, we denote by $z_{i,j} \in \mathbb{Z}_p$ the decryption outcome (inner product, as a number instead of a group element) between the j^{th} simulated secret key (ordered temporally among all queried secret keys) and the i^{th} simulated ciphertext (ordered temporally among all queried ciphertexts, excluding the challenge ciphertext). We also let I_t, J_t be the number of simulated ciphertexts (excluding the challenge ciphertext) and secret keys at any time t. The requirement is $\mathsf{Exp}^0_{T\text{-GS}} \approx \mathsf{Exp}^1_{T\text{-GS}}$, where $\mathsf{Exp}^b_{T\text{-GS}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Run $(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(1^\lambda)$. Launch $\mathcal{A}(1^\lambda, \mathsf{gp})$ and receive $(1^n, 1^T)$ from it. Run

$$(\mathsf{impk}, \sigma) \xleftarrow{\$} \mathsf{SimSetup}(1^\lambda, \mathsf{gp}, 1^n, 1^T)$$

  and send impk to $\mathcal{A}$.

- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ has two options.
  - *Key Simulation Query.* $\mathcal{A}$ can submit a vector $\mathbf{v}_j$ with a list $z_{\leq I_t,j}$ of inner products for a secret key isk_j. The list $z_{\leq I_t,j}$ consists of $z_{1,j},\ldots,z_{I_t,j}$, all the decryption outcomes between isk_j and the simulated ciphertexts queried up to this point. Upon this query, run
  $$(\mathsf{isk}_j,\sigma') \xleftarrow{\$} \mathsf{SimKeyGen}(\sigma,\mathbf{v}_j,z_{1,j},\ldots,z_{I_t,j}),\qquad \sigma\leftarrow\sigma',$$
  and send isk_j to $\mathcal{A}$.
  - *Ciphertext Simulation Query.* $\mathcal{A}$ can submit a list $z_{i,\leq J_t}$ of inner products for a simulated ciphertext ict_i. The list $z_{i,\leq J_t}$ consists of $z_{i,1},\ldots,z_{i,J_t}$, all the decryption outcomes between ict_i and the simulated secret keys queried up to this point. Upon this query, run
  $$(\mathsf{ict}_i,\sigma') \xleftarrow{\$} \mathsf{SimEnc}(\sigma,z_{i,1},\ldots,z_{i,J_t}),\qquad \sigma\leftarrow\sigma',$$
  and send ict_i to $\mathcal{A}$.
- **Challenge.** The adversary submits a vector $\mathbf{u}^*$. Upon the challenge, let the total number of secret key and ciphertext queries in Query I be J^*, I^*, respectively. Run
  $$(b=0)\qquad \mathsf{ict}^* \xleftarrow{\$} \mathsf{Enc}(1^\lambda,\mathsf{impk},[\![\mathbf{u}^*]\!]);$$
  $$(b=1)\qquad (\mathsf{ict}^*,\sigma') \xleftarrow{\$} \mathsf{SimEnc}(\sigma,(\mathbf{u}^*)^\top\mathbf{v}_1,\ldots,(\mathbf{u}^*)^\top\mathbf{v}_{J^*}),\qquad \sigma\leftarrow\sigma'.$$
  Send ict^* to $\mathcal{A}$.
- **Query II.** Same as Query I, except that in $\mathsf{Exp}^1_{T\text{-}\mathrm{GS}}$, for each secret key query $\mathbf{v}_j$, we put $(\mathbf{u}^*)^\top\mathbf{v}_j$ immediately after $z_{I^*,j}$ in the argument list of $\mathsf{SimKeyGen}$ so that the simulator gets the correct list of inner products, i.e.,
  $$(b=0)\quad (\mathsf{isk}_j,\sigma') \xleftarrow{\$} \mathsf{SimKeyGen}(\sigma,\mathbf{v}_j,z_{1,j},\ldots,z_{I^*,j},\quad z_{I^*+1,j},\ldots,z_{I_t,j});$$
  $$(b=1)\quad (\mathsf{isk}_j,\sigma') \xleftarrow{\$} \mathsf{SimKeyGen}(\sigma,\mathbf{v}_j,z_{1,j},\ldots,z_{I^*,j},(\mathbf{u}^*)^\top\mathbf{v}_j,z_{I^*+1,j},\ldots,z_{I_t,j});$$
  $$\sigma\leftarrow\sigma'\qquad\text{(in either case)}.$$

- **Guess.** The adversary outputs a bit b'. The outcome of the experiment is b' if
  - the total number of ciphertext simulation queries in Query I and II is less than T and
  - the equation $\{\mathbf{u}_i^\intercal \mathbf{v}_j = z_{i,j} \ \ \forall i, j\}$ (about $\mathbf{u}_i$'s) has a solution.

  Otherwise, the outcome is set to 0.

Remarks. In $\mathsf{Exp}^1_{T\text{-GS}}$, the challenge ciphertext ict^* is generated in the same way as the other simulated ciphertexts, and in Query II, the inner products between isk_j and ict^* are appropriately positioned. From the simulator's perspective, there is no indication which ciphertext is the challenge ciphertext. This definition ensures that the simulator *cannot behave differently depending on whether a particular ciphertext is the challenge or not*, and simplifies the application of gradual simulation security in our construction of ABE.

Note that the simulator receives the inner products $z_{i,j}$ in the clear and the adversary submits challenge $\mathbf{u}^*$ in $\mathsf{Exp}^b_{T\text{-GS}}$ in the clear, though the input to encryption and the output of decryption are group-encoded. This is necessary as otherwise, the simulator must solve discrete logarithm in $\mathbb{G}$.

We note that when $T = 1$, gradual simulation security boils down to the standard notion of simulation security. On the other hand, simulation security does not imply gradual simulation security. So Definition 5.2 is a strict generalization of simulation security.

Schemes. We can generically upgrade any selectively IND-CPA secure IPFE scheme for gradual simulation security.

Theorem 5.2. *Let* GroupGen *be a group sampler (Definition 2.4). Suppose there exists an IPFE scheme (Definition 5.1) based on* GroupGen *that is* selectively IND-CPA secure *(Definition 2.1), then there exists a* gradually simulation-secure *one (Definition 5.2) based on* GroupGen*. If the underlying scheme has* succinct *keys, then so does the resultant scheme.*

See Section 5.8 for further discussion of Theorem 5.2.

In the rest of this section, we present a direct construction adapted from [ALS16], which is used in our ABE for the best efficiency.

5.4.1 Direct Construction

The IPFE scheme in [ALS16] has been proven simulation-secure [ALMT20]. We show that it can be adapted for gradual simulation security. The scheme has succinct keys, whose length grows linearly in T and polynomially in λ, and is independent of n, which eventually translates into the succinctness of our ABE scheme.

Ingredients of Construction 5.1. Let GroupGen be a (prime-order) group sampler (Definition 2.4) over which MDDH_k (Assumption 2.1) holds.

Construction 5.1 (modified from [ALS16]). Our IPFE scheme based on GroupGen works as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^n, 1^T)$ takes as input the group description gp the dimension n, and the maximum number T of simulated ciphertexts. It samples $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+T)}$ and $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{(k+T)\times n}$. The algorithm outputs $\mathsf{impk} = [\![\mathbf{A}, \mathbf{AW}]\!]$ and $\mathsf{imsk} = \mathbf{W}$.
- $\mathsf{KeyGen}(\mathsf{imsk}, \mathbf{v})$ outputs $\mathsf{isk} = \mathbf{Wv}$.
- $\mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}]\!])$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and outputs $\mathsf{ict} = (\mathbf{s}^\top[\![\mathbf{A}]\!], \mathbf{s}^\top[\![\mathbf{AW}]\!] + [\![\mathbf{u}^\top]\!])$.
- $\mathsf{Dec}(\mathsf{impk}, \mathbf{v}, \mathsf{isk}, \mathsf{ict})$ parses ict as $([\![\mathbf{c}^\top]\!], [\![\mathbf{t}^\top]\!])$ and outputs $(-[\![\mathbf{c}^\top]\!] \cdot \mathsf{isk} + [\![\mathbf{t}^\top]\!] \cdot \mathbf{v})$.

The correctness is readily verified by

$$-[\![\mathbf{c}^\top]\!] \cdot \mathsf{isk} + [\![\mathbf{t}^\top]\!] \cdot \mathbf{v} = [\![-(\mathbf{s}^\top\mathbf{A})(\mathbf{Wv}) + (\mathbf{s}^\top\mathbf{AW} + \mathbf{u}^\top)\mathbf{v}]\!] = [\![\mathbf{u}^\top\mathbf{v}]\!].$$

The scheme is succinct as isk consists of $(k + T)$ elements of $\mathbb{Z}_p$, independent of n.

Theorem 5.3 (¶). *Suppose in Construction 5.1,* MDDH_k *(Assumption 2.1) holds over* GroupGen*, then the resultant scheme is* gradually simulation-secure *(Definition 5.2).*

To prove this theorem, we first study the simple one-time pad IPFE scheme (OTP-IPFE). The simulator for Construction 5.1 can be constructed from the simulator of OTP-IPFE modularly.

5.4.2 One-Time Pad IPFE Scheme

OTP-IPFE is a secret-key IPFE scheme, whose syntax and security are defined below.

Definition 5.3 (secret-key IPFE). A *secret-key IPFE scheme* consists of four efficient algorithms.

- $\mathsf{Setup}(p, 1^n)$ takes as input the modulus p and the dimension n. It outputs a master secret key imsk.
- $\mathsf{KeyGen}(\mathsf{imsk}, \mathbf{v})$ generates the secret key isk for $\mathbf{v} \in \mathbb{Z}_p^n$.
- $\mathsf{Enc}(\mathsf{imsk}, \mathbf{u})$ encrypts $\mathbf{u} \in \mathbb{Z}_p^n$ into a ciphertext ict.
- $\mathsf{Dec}(\mathbf{v}, \mathsf{isk}, \mathsf{ict})$ is supposed to compute the inner product.

The scheme must be *correct*, i.e., for all prime p, natural number n, and $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^n$,

$$\Pr\left[\begin{array}{l|c|l} \mathsf{imsk} \xleftarrow{\$} \mathsf{Setup}(p, 1^n) & & \\ \mathsf{isk} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, \mathbf{v}) & : & \mathsf{Dec}(\mathbf{v}, \mathsf{isk}, \mathsf{ict}) = \mathbf{u}^\mathsf{T}\mathbf{v} \\ \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(\mathsf{imsk}, \mathbf{u}) & & \end{array}\right] = 1.$$

Definition 5.4 (simulation security). A *simulator* for a secret-key IPFE scheme (Definition 5.3) consists of three efficient algorithms.[43]

- $\mathsf{SimSetup}(p, 1^n)$ takes the same input as Setup. It outputs an internal state σ.
- $\mathsf{SimKeyGen}(\sigma, \mathbf{v}, z)$ takes as input the state, the vector, and optionally an inner product $z \in \mathbb{Z}_p$. It outputs a simulated secret key isk and a new state σ'.
- $\mathsf{SimEnc}(\sigma, z_1, \ldots, z_J)$ takes as input the state and a list of inner products. It outputs a simulated ciphertext ict and a new state σ'.

The simulator *perfectly simulates* the scheme if $\mathsf{Exp}_{\mathsf{real}} \equiv \mathsf{Exp}_{\mathsf{sim}}$, where the experiments with an (unbounded) adversary $\mathcal{A}$ proceed as follows.

[43] To be precise, the efficiency requirement is that whenever $\mathsf{Exp}_{\mathsf{real}}$ finishes in polynomial time, so must $\mathsf{Exp}_{\mathsf{sim}}$. (See below for the definitions of these experiments.)

- **Setup.** Launch $\mathcal{A}()$ and receive from it $(p, 1^n)$. Run

$$\begin{aligned}
&(\text{in } \mathsf{Exp}_{\text{real}}) && \mathsf{imsk} \xleftarrow{\$} \mathsf{Setup}(p, 1^n);\\
&(\text{in } \mathsf{Exp}_{\text{sim}}) && \sigma \xleftarrow{\$} \mathsf{SimSetup}(p, 1^n).
\end{aligned}$$

- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ submits a vector $\mathbf{v}_j$. Upon this query, run

$$\begin{aligned}
&(\text{in } \mathsf{Exp}_{\text{real}}) && \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, \mathbf{v}_j);\\
&(\text{in } \mathsf{Exp}_{\text{sim}}) && (\mathsf{isk}_j, \sigma') \xleftarrow{\$} \mathsf{SimKeyGen}(\sigma, \mathbf{v}_j), \qquad \sigma \leftarrow \sigma';
\end{aligned}$$

  and send isk_j to $\mathcal{A}$.

- **Challenge.** The adversary submits a vector $\mathbf{u}^*$. Upon the challenge, let the total number of secret key queries in Query I be J^*, run

$$\begin{aligned}
&(\text{in } \mathsf{Exp}_{\text{real}}) && \mathsf{ict}^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{imsk}, \mathbf{u}^*);\\
&(\text{in } \mathsf{Exp}_{\text{sim}}) && (\mathsf{ict}^*, \sigma') \xleftarrow{\$} \mathsf{SimEnc}(\sigma, (\mathbf{u}^*)^\top \mathbf{v}_1, \ldots, (\mathbf{u}^*)^\top \mathbf{v}_{J^*}), \qquad \sigma \leftarrow \sigma';
\end{aligned}$$

  and send ict^* to $\mathcal{A}$.

- **Query II.** Same as Query I, except that in $\mathsf{Exp}_{\text{sim}}$, we supply the inner product to the simulator, i.e.,

$$(\mathsf{isk}_j, \sigma') \xleftarrow{\$} \mathsf{SimKeyGen}(\sigma, \mathbf{v}_j, (\mathbf{u}^*)^\top \mathbf{v}_j), \qquad \sigma \leftarrow \sigma'.$$

- **Guess.** The adversary outputs a bit b', which is the outcome of the experiment.

We now describe the one-time pad IPFE scheme.

Construction 5.2 (OTP-IPFE). The one-time pad IPFE works as follows.

- $\mathsf{Setup}(p, 1^n)$ takes as input the prime p and the dimension n. It samples $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^n$ and outputs $\mathsf{imsk} = \mathbf{w}$.
- $\mathsf{KeyGen}(\mathsf{imsk}, \mathbf{v})$ outputs $\mathsf{isk} = \mathbf{w}^\top \mathbf{v}$.
- $\mathsf{Enc}(\mathsf{imsk}, \mathbf{u})$ outputs $\mathsf{ict} = (\mathbf{w} + \mathbf{u})^\top$.

- $\mathsf{Dec}(\mathbf{v}, \mathsf{isk}, \mathsf{ict})$ outputs $(-\mathsf{isk} + \mathsf{ict} \cdot \mathbf{v})$.

The correctness follows from $-\mathbf{w}^\intercal\mathbf{v} + (\mathbf{w}+\mathbf{u})^\intercal\mathbf{v} = \mathbf{u}^\intercal\mathbf{v}$.

Wee [Wee17] implicitly constructed a simulator for OTP-IPFE where the adversary cannot query secret keys before the ciphertext challenge. Abdalla, Catalano, Fiore, Gay, and Ursu [ACF+18] explicitly gave that construction. Agrawal, Libert, Maitra, and Titiu [ALMT20] implicitly constructed the adaptive simulator.

Construction 5.3 (simulator for OTP-IPFE). The simulator works as follows.

- $\mathsf{SimSetup}(p, 1^n)$ samples $\widetilde{\mathbf{w}} \xleftarrow{\$} \mathbb{Z}_p^n$ and outputs $\sigma = (\widetilde{\mathbf{w}}, \perp)$.
- $\mathsf{SimKeyGen}(\sigma, \mathbf{v}, z)$ takes as input the state σ, a vector $\mathbf{v}$, and optionally an inner product z. It parses σ as $(\widetilde{\mathbf{w}}, \mathsf{ict}^*, \mathbf{v}_1, \ldots, \mathbf{v}_J)$ and outputs

  $$\begin{aligned} &(\text{Query I, i.e., if } \widetilde{\mathbf{w}} \neq \perp) && \mathsf{isk} = \widetilde{\mathbf{w}}^\intercal\mathbf{v}, && \sigma' = (\widetilde{\mathbf{w}}, \perp, \mathbf{v}_1, \ldots, \mathbf{v}_J, \mathbf{v}); \\ &(\text{Query II, i.e., if } \widetilde{\mathbf{w}} = \perp) && \mathsf{isk} = \mathsf{ict}^* \cdot \mathbf{v} - z, && \sigma' = (\perp, \mathsf{ict}^*). \end{aligned}$$

- $\mathsf{SimEnc}(\sigma, z_1, \ldots, z_J)$ takes as input the state σ and a list $z_1, \ldots, z_J$ of inner products. It parses σ as $(\widetilde{\mathbf{w}}, \perp, \mathbf{v}_1, \ldots, \mathbf{v}_{J'})$. If $J \neq J'$, the algorithm outputs $\perp$ and terminates. Otherwise, it outputs the simulated ciphertext $\mathsf{ict}^* \in \mathbb{Z}_p^{1\times n}$ to be a uniformly random solution of

  $$-\widetilde{\mathbf{w}}^\intercal\mathbf{v}_j + \mathsf{ict}^* \cdot \mathbf{v}_j = z_j \qquad \forall j \in [J],$$

  and the updated state $\sigma' = (\perp, \mathsf{ict}^*)$.

The following lemma shows that the simulator is perfect:

Lemma 5.4 (¶). *The simulator in Construction 5.3* perfectly simulates *(Definition 5.4) the one-time pad scheme in Construction 5.2.*

Proof (Lemma 5.4). In the security experiment, the adversary can query arbitrarily many keys and one challenge ciphertext, both with vectors of its choice. The keys and ciphertext are either honestly generated or simulated, and we want to show that the adversary has no advantage in distinguishing the two cases. We consider the view of the adversary at each phase of the security experiment.

Before the ciphertext challenge, the simulator is perfect, since the distribution of $\widetilde{\mathbf{w}}$ ($\mathbf{w}$ in the real scheme) and how the secret keys are generated are the same as in the real scheme.

Once the adversary chooses the challenge vector $\mathbf{u}^*$, the distribution of $\mathbf{w}$ is uniformly random conditioned on $\mathbf{w}^\intercal \mathbf{v}_j = \widetilde{\mathbf{w}}^\intercal \mathbf{v}_j$ for all j. Moreover, $\mathbf{u}^*$ is a particular solution of $\mathbf{u}^\intercal \mathbf{v}_j = z_j$ for all j. This implies $\mathsf{ict}^* = (\mathbf{w} + \mathbf{u}^*)^\intercal$ is a uniformly random solution of $\mathsf{ict}^* \cdot \mathbf{v}_j = \widetilde{\mathbf{w}}^\intercal \mathbf{v}_j + z_j$ for all j, and the simulator does the same thing.

After the ciphertext challenge, the secret key of $\mathbf{v}$ is

$$\mathbf{w}^\intercal \mathbf{v} = \left(\mathsf{ict}^* - (\mathbf{u}^*)^\intercal\right)\mathbf{v} = \mathsf{ict}^* \cdot \mathbf{v} - \underbrace{(\mathbf{u}^*)^\intercal \mathbf{v}}_{=z},$$

and the simulator computes the same value. □

5.4.3 Security of Direct Construction

We first discuss the intuition of our simulator. Recall that in the scheme, the public key consists of a random matrix $\mathbf{A}$ and the projection $\mathbf{AW}$ of the master secret key $\mathbf{W}$. A ciphertext can be regarded as an OTP-IPFE ciphertext using $\mathbf{s}^\intercal\mathbf{AW}$ as *its* master secret key, which is a random combination of $\mathbf{AW}$. It also includes the combination coefficients $\mathbf{s}^\intercal\mathbf{A}$ (with respect to $\mathbf{W}$). The MDDH assumption tells us that $\mathbf{s}^\intercal\mathbf{AW}$ is a pseudorandom combination of $\mathbf{W}$, so it is indistinguishable for the ciphertext to use $\mathbf{c}^\intercal\mathbf{W}$ as *its* OTP-IPFE master secret key for a random $\mathbf{c}$, which lies outside the row space of $\mathbf{A}$ with overwhelming probability. A ciphertext in this form is using an OTP imsk independent of impk, or equivalently, a fresh instance of OTP-IPFE dedicated to this ciphertext.

A recurring technique in dual system encryption is to perform a change of variable for $\mathbf{W}$ to explicitly separate out *the* instance for the challenge ciphertext. We program $\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{a}^\perp \mathbf{w}^\intercal$, where $\mathbf{w}$ is an OTP-IPFE master secret key and $\mathbf{a}^\perp$ is any vector such that $\mathbf{A}\mathbf{a}^\perp = \mathbf{0}$ and $\mathbf{c}^\intercal\mathbf{a}^\perp = 1$. (This step is known as using the *parameter hiding* property.) Under this change of variable, keys and ciphertext become

$$\mathsf{isk} = \mathbf{W}\mathbf{v} = \widetilde{\mathbf{W}}\mathbf{v} + \mathbf{a}^\perp \boxed{\mathbf{w}^\intercal\mathbf{v}}, \qquad \mathsf{ict} = [\![\mathbf{c}^\intercal, \mathbf{c}^\intercal\mathbf{W} + \mathbf{u}^\intercal]\!] = [\![\mathbf{c}^\intercal, \mathbf{c}^\intercal\widetilde{\mathbf{W}} + \boxed{(\mathbf{w}+\mathbf{u})^\intercal}]\!].$$

Note that the terms in the boxes are precisely OTP-IPFE keys and ciphertext and those outside the boxes do not depend on $\mathbf{u}$. Therefore, our simulator prepares an OTP-IPFE simulator instance, and samples $\mathbf{A}, \mathbf{c}, \widetilde{\mathbf{W}}$ and computes $\mathbf{a}^\perp$ on its own. To simulate keys and ciphertext, it invokes the OTP-IPFE simulator and stitches the result. The construction below extends this idea to handle multiple simulated ciphertexts.

Ingredients of Construction 5.4. Let

- GroupGen be the group sampler (Definition 2.4) used in Construction 5.1, over which MDDH_k (Assumption 2.1) holds, and
- $(\mathsf{OTP.SimSetup}, \mathsf{OTP.SimKeyGen}, \mathsf{OTP.SimEnc})$ the simulator (Construction 5.3) for OTP-IPFE (Construction 5.2).

Construction 5.4 (simulator for Construction 5.1). The simulator works as follows.

- $\mathsf{SimSetup}(\mathsf{gp}, 1^n, 1^T)$ takes as input the group description gp, the dimension n, and the maximum number T of simulated ciphertexts. It samples

$$\mathbf{T} \xleftarrow{\$} \mathbb{Z}_p^{k\times k}, \qquad \widetilde{\mathbf{W}} \xleftarrow{\$} \mathbb{Z}_p^{(k+T)\times n},$$
$$\mathbf{P} = (\mathbf{a}_1^\perp \ \cdots \ \mathbf{a}_T^\perp \ \mathbf{p}_1 \ \cdots \ \mathbf{p}_k) \xleftarrow{\$} \mathbb{Z}_p^{(k+T)\times(k+T)},$$

  conditioned on $\mathbf{P}$ being invertible. The algorithm prepares T instances of the OTP-IPFE simulator by running $\sigma_t \xleftarrow{\$} \mathsf{OTP.SimSetup}(p, 1^n)$ for $t \in [T]$. It sets

$$(\mathbf{c}_1 \ \cdots \ \mathbf{c}_{T-1} \ \mathbf{c}_T \ \mathbf{A}^\intercal) = (\mathbf{P}^\intercal)^{-1} \begin{pmatrix} \mathbf{I}_{T-1} & & \\ & 1 & \\ & & \mathbf{T} \end{pmatrix}$$

  and outputs $\mathsf{impk} = [\![\mathbf{A}, \mathbf{A}\widetilde{\mathbf{W}}]\!]$ and $\sigma = (0, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1, \ldots, \mathbf{a}_T^\perp, [\![\mathbf{c}_T^\intercal]\!], \sigma_T)$.

  Conceptually, we program $\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{a}_1^\perp \mathbf{w}_1^\intercal + \cdots + \mathbf{a}_T^\perp \mathbf{w}_T^\intercal$, where $\mathbf{w}_t$ is the OTP-IPFE master secret key for the t^{th} simulated ciphertext.[44] By definition, $\mathbf{A}\mathbf{a}_i^\perp = \mathbf{0}$, $\mathbf{c}_i^\intercal \mathbf{a}_{i'}^\perp = 0$ for $i \neq i'$, and $\mathbf{c}_i^\intercal \mathbf{a}_i^\perp = 1$. The first component of σ keeps track of the number of simulated ciphertext at any point.

[44] The programming of imsk's is only conceptual because they are in fact simulated.

- $\mathsf{SimKeyGen}(\sigma, \mathbf{v}, z_1, \ldots, z_I)$ takes as input the state σ, a vector $\mathbf{v}$, and a list of inner products $z_1, \ldots, z_I$. It parses σ as $(I', \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1, \ldots, \mathbf{a}_T^\perp, [\![\mathbf{c}_T^\intercal]\!], \sigma_T)$. If $I \neq I'$, the algorithm outputs $\perp$ and terminates. Otherwise, it runs

$$\begin{aligned} &(\text{for } t \le I) \qquad (\mathsf{OTP.isk}_t, \sigma_t') \xleftarrow{\$} \mathsf{OTP.SimKeyGen}(\sigma_t, \mathbf{v}, z_t); \\ &(\text{for } t > I) \qquad (\mathsf{OTP.isk}_t, \sigma_t') \xleftarrow{\$} \mathsf{OTP.SimKeyGen}(\sigma_t, \mathbf{v}). \end{aligned}$$

  The algorithm outputs

$$\begin{aligned} \mathsf{isk} &= \widetilde{\mathbf{W}}\mathbf{v} + \mathbf{a}_1^\perp \cdot \mathsf{OTP.isk}_1 + \cdots + \mathbf{a}_T^\perp \cdot \mathsf{OTP.isk}_T, \\ \sigma' &= (I, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1', \ldots, \mathbf{a}_T^\perp, [\![\mathbf{c}_T^\intercal]\!], \sigma_T'). \end{aligned}$$

- $\mathsf{SimEnc}(\sigma, z_1, \ldots, z_J)$ takes as input the state σ and a list of inner products $z_1, \ldots, z_J$. The algorithm parses σ as $(I, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1, \ldots, \mathbf{a}_T^\perp, [\![\mathbf{c}_T^\intercal]\!], \sigma_T)$. If $I = T$, it outputs $\perp$ and terminates. Otherwise, the algorithm runs

$$(\mathsf{OTP.ict}, \sigma_{I+1}') \xleftarrow{\$} \mathsf{OTP.SimEnc}(\sigma_{I+1}, z_1, \ldots, z_J), \qquad \sigma_t' \leftarrow \sigma_t \text{ for } t \neq I+1,$$

  and outputs

$$\begin{aligned} \mathsf{ict} &= ([\![\mathbf{c}_{I+1}^\intercal]\!], [\![\mathbf{c}_{I+1}^\intercal]\!]\widetilde{\mathbf{W}} + [\![\mathsf{OTP.ict}]\!]), \\ \sigma' &= (I+1, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1', \ldots, \mathbf{a}_T^\perp, [\![\mathbf{c}_T^\intercal]\!], \sigma_T'). \end{aligned}$$

Proving Theorem 5.3 amounts to proving the following two claims.

Theorem 5.5 (¶). *Construction 5.4 (simulator) satisfies key simulation security (Definition 5.2) for Construction 5.1 (scheme).*

Theorem 5.6 (¶). *Suppose in Constructions 5.1 (scheme) and 5.4 (simulator),* MDDH_k *(Assumption 2.1) holds over* $\mathsf{GroupGen}$*, then the simulator satisfies* T*-ciphertext simulation security (Definition 5.2) for the scheme for all* $T = \mathrm{poly}(\lambda)$.

Proof (Theorem 5.5). Recall that in $\mathsf{Exp}_{\mathrm{real}}$ and $\mathsf{Exp}_{\mathrm{sim}}$, the adversary only gets the master public key and the secret keys. We show that their distributions are identical via the following hybrids.

- H_1 proceeds identically as $\mathsf{Exp}_{\mathsf{real}}$, except that $\mathbf{A}$ is sampled as in $\mathsf{Exp}_{\mathsf{sim}}$.
- H_2 proceeds identically as H_1, except that we set

$$\mathbf{W} = \widetilde{\mathbf{W}} + \mathbf{a}_1^\perp \mathbf{w}_1^\intercal + \cdots + \mathbf{a}_T^\perp \mathbf{w}_T^\intercal$$

  for uniformly random $\widetilde{\mathbf{W}}, \mathbf{w}_1, \ldots, \mathbf{w}_T$.

In both $\mathsf{Exp}_{\mathsf{real}}$ and $\mathsf{Exp}_{\mathsf{sim}}$, the matrix $\mathbf{A}$ is uniformly random, so $\mathsf{Exp}_{\mathsf{real}} \equiv \mathsf{H}_1$. We also have $\mathsf{H}_1 \equiv \mathsf{H}_2$ as the change of variable does not alter the distribution of $\mathbf{W}$ and $\mathbf{AW} = \mathbf{A}\widetilde{\mathbf{W}}$. In H_2, the secret key for $\mathbf{v}$ is

$$\mathbf{Wv} = \widetilde{\mathbf{W}}\mathbf{v} + \mathbf{a}_1^\perp \mathbf{w}_1^\intercal \mathbf{v} + \cdots + \mathbf{a}_T^\perp \mathbf{w}_T^\intercal \mathbf{v}.$$

In $\mathsf{Exp}_{\mathsf{sim}}$, the terms $\mathbf{w}_i^\intercal \mathbf{v}$ are replaced by their simulation. Since the OTP-IPFE simulator is perfect, we conclude $\mathsf{H}_2 \equiv \mathsf{Exp}_{\mathsf{sim}}$. □

Proof (Theorem 5.6). Recall that in $\mathsf{Exp}^b_{T\text{-}\mathsf{GS}}$, the adversary receives the simulated impk, can query for arbitrarily many simulated secret keys and at most $(T-1)$ simulated ciphertexts, and submits one challenge vector to obtain either a challenge ciphertext encrypted using the simulated impk or a simulated challenge ciphertext. We want to show that the two cases are indistinguishable.

To simulate the challenge ciphertext, we need to first switch it from using $\mathbf{s}^\intercal \mathbf{A}$ to $\mathbf{c}^\intercal$. At first sight, it seems that the reduction cannot know $\mathbf{a}^\perp$'s (since it receives $[\![\mathbf{A}]\!]$ as an MDDH_k challenge) hence cannot simulate *non-challenge* ciphertexts. This is why the simulator samples $\mathbf{A}$ in a fashion similar to the reduction of $\mathsf{MDDH}_{k,k+T}$ to MDDH_k, which originates from a technique in [CGKW18a].

We go through the following hybrids.

- H_1 proceeds identically to $\mathsf{Exp}^0_{T\text{-}\mathsf{GS}}$, except that the T^{th} instance of OTP-IPFE simulator is replaced by an instance of OTP-IPFE itself and $\mathbf{c}_T$ is removed. In particular, the state of the simulator is

$$\sigma = (I, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1, \ldots, \mathbf{a}_{T-1}^\perp, [\![\mathbf{c}_{T-1}^\intercal]\!], \sigma_{T-1}, \mathbf{a}_T^\perp, \boxed{\mathbf{w}_T})$$

  and a simulated secret key for $\mathbf{v}$ is

$$\mathsf{isk} = \widetilde{\mathbf{W}}\mathbf{v} + \mathbf{a}_1^\perp \cdot \mathsf{OTP.isk}_1 + \cdots + \mathbf{a}_{T-1}^\perp \cdot \mathsf{OTP.isk}_{T-1} + \mathbf{a}_T^\perp \boxed{\mathbf{w}_T^\intercal \mathbf{v}}.$$

Since the adversary can query at most $(T-1)$ simulated ciphertexts (excluding the challenge ciphertext), this instance is never used to simulate a ciphertext in $\mathsf{Exp}^0_{T\text{-GS}}$, and it does not matter if we remove $\mathbf{c}_T$. The transition from $\mathsf{Exp}^0_{T\text{-GS}}$ to H_1 is essentially undoing simulation for the last OTP-IPFE instance. Since the OTP-IPFE simulator is perfect, $\mathsf{Exp}^0_{T\text{-GS}} \equiv \mathsf{H}_1$.

- H_2 proceeds identically to H_1, except that $\mathbf{a}_T^\perp$ and the last OTP-IPFE instance are removed. The state of the simulator and a simulated secret key for $\mathbf{v}$ are

$$\sigma = (I, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1, \ldots, \mathbf{a}_{T-1}^\perp, [\![\mathbf{c}_{T-1}^\intercal]\!], \sigma_{T-1}, \cdot),$$
$$\mathsf{isk} = \widetilde{\mathbf{W}}\mathbf{v} + \mathbf{a}_1^\perp \cdot \mathsf{OTP.isk}_1 + \cdots + \mathbf{a}_{T-1}^\perp \cdot \mathsf{OTP.isk}_{T-1} + 0.$$

  This can be interpreted as a change of variable $\widetilde{\mathbf{W}} \to \widetilde{\mathbf{W}} - \mathbf{a}_T^\perp \mathbf{w}_T^\intercal$, which does not affect the distribution of $\widetilde{\mathbf{W}}$, so $\mathsf{H}_1 \equiv \mathsf{H}_2$.

- H_3 proceeds identically to H_2, except that the challenge ciphertext becomes

$$\mathsf{ict}^* = ([\![\boxed{\mathbf{c}^\intercal}]\!], [\![\boxed{\mathbf{c}^\intercal}\, \widetilde{\mathbf{W}} + (\mathbf{u}^*)^\intercal]\!]) \quad \text{for} \quad \mathbf{c} \xleftarrow{\$} \mathbb{Z}_p^{k+T}.$$

  Claim 5.7 (¶). $\mathsf{H}_2 \approx \mathsf{H}_3$ *reduces to* MDDH_k *(Assumption 2.1) over* GroupGen.

- H_4 proceeds identically to H_3, except that $\mathbf{c}$ in the challenge ciphertext becomes $\mathbf{c}_T$ sampled by SimSetup. Note that we are not using $\mathbf{a}_T^\perp$ and $\mathbf{c}_T$ is statistically close to $\mathbf{c}$. Therefore, $\mathsf{H}_3 \approx_s \mathsf{H}_4$.

- H_5 proceeds identically to H_4, except that we perform a change of variable $\widetilde{\mathbf{W}} \to \widetilde{\mathbf{W}} + \mathbf{a}_T^\perp \mathbf{w}_T^\intercal$. Clearly, $\mathsf{H}_4 \equiv \mathsf{H}_5$.

- H_6 proceeds identically to H_5, except that we resurrect the last instance of OTP-IPFE simulator and use it to simulate the challenge ciphertext. In H_5, the challenge ciphertext and a simulated secret key for $\mathbf{v}$ are

$$\mathsf{ict}^* = ([\![\mathbf{c}_T^\intercal]\!], [\![\mathbf{c}_T^\intercal \widetilde{\mathbf{W}} + \boxed{(\mathbf{w}_T + \mathbf{u}^*)^\intercal}\,]\!]),$$
$$\mathsf{isk} = \widetilde{\mathbf{W}} + \mathbf{a}_1^\perp \cdot \mathsf{OTP.isk}_1 + \cdots + \mathbf{a}_{T-1}^\perp \cdot \mathsf{OTP.isk}_{T-1} + \mathbf{a}_T^\perp \boxed{\mathbf{w}_T^\intercal \mathbf{v}}.$$

  The terms in boxes are OTP-IPFE ciphertexts and keys, and in H_6 they are replaced by simulation. Since the OTP-IPFE simulator is perfect, $\mathsf{H}_5 \equiv \mathsf{H}_6$.

We argue $\mathsf{H}_6 \equiv \mathsf{Exp}^1_{T\text{-GS}}$. Note that the only difference between the two is which instance is used to simulate which ciphertext. This amounts to permuting the $(\mathbf{a}_t^\perp, [\![\mathbf{c}_t^\intercal]\!], \sigma_t)$'s. It is easy to verify that the distribution of these tuples is invariant under permutation. Once we prove $\mathsf{H}_2 \approx \mathsf{H}_3$ (Claim 5.7), we conclude $\mathsf{Exp}^0_{T\text{-GS}} \approx \mathsf{Exp}^1_{T\text{-GS}}$ by hybrid argument. □

Proof (Claim 5.7). Let $\mathcal{A}$ be an efficient adversary distinguishing H_2 and H_3. We construct $\mathcal{B}$ against MDDH_k over $\mathsf{GroupGen}$. Upon receiving MDDH_k challenge $[\![\mathbf{B}, \mathbf{d}^\intercal]\!]$ where $\mathbf{d}^\intercal = \mathbf{s}^\intercal\mathbf{B}$ or $\mathbf{d}^\intercal$ is random, $\mathcal{B}$ launches $\mathcal{A}()$, receives $(1^n, 1^T)$ from it, and samples and defines

$$\widetilde{\mathbf{W}} \xleftarrow{\$} \mathbb{Z}_p^{(k+T)\times n}, \qquad \mathbf{P} = (\mathbf{a}_1^\perp \ \cdots \ \mathbf{a}_{T-1}^\perp \ \mathbf{p}_0 \ \mathbf{p}_1 \ \cdots \ \mathbf{p}_k) \xleftarrow{\$} \mathbb{Z}_p^{(k+T)\times(k+T)},$$
$$(\mathbf{c}_1 \ \cdots \ \mathbf{c}_{T-1} \ \mathbf{c} \ \mathbf{A}^\intercal) = (\mathbf{P}^\intercal)^{-1} \begin{pmatrix} \mathbf{I}_{T-1} & & \\ & \mathbf{d} & \mathbf{B}^\intercal \end{pmatrix}.$$

Here, $(\mathbf{d} \ \mathbf{B}^\intercal)$ occupies the position of $\begin{pmatrix} 1 & \\ & \mathbf{T} \end{pmatrix}$. The algorithm $\mathcal{B}$ prepares $(T-1)$ instances of OTP-IPFE simulator by running $\sigma_t \xleftarrow{\$} \mathsf{OTP.SimSetup}(p, 1^n)$ for $t \in [T-1]$. It sets

$$\mathsf{impk} = ([\![\mathbf{A}]\!], [\![\mathbf{A}]\!]\widetilde{\mathbf{W}}), \qquad \sigma = (0, \mathbf{a}_1^\perp, [\![\mathbf{c}_1^\intercal]\!], \sigma_1, \ \ldots, \ \mathbf{a}_{T-1}^\perp, [\![\mathbf{c}_{T-1}^\intercal]\!], \sigma_{T-1}).$$

The algorithm $\mathcal{B}$ answers queries from $\mathcal{A}$ as defined in $\mathsf{H}_2, \mathsf{H}_3$. It responds to the ciphertext challenge by setting

$$\mathsf{ict}^* = ([\![\mathbf{c}^\intercal]\!], [\![\mathbf{c}^\intercal]\!]\widetilde{\mathbf{W}} + [\![(\mathbf{u}^*)^\intercal]\!]).$$

Lastly, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Clearly, $\mathcal{B}$ is efficient, produces the correct distribution for $(\mathbf{a}_t^\perp, [\![\mathbf{c}_t^\intercal]\!], \sigma_t)$'s and $\mathbf{A}, \widetilde{\mathbf{W}}$, and answers queries (non-challenges) as required. If $\mathbf{d}^\intercal = \mathbf{s}^\intercal\mathbf{B}$, we have $\mathbf{c}^\intercal = \mathbf{s}^\intercal\mathbf{A}$ and $\mathcal{B}$ has simulated H_2 for $\mathcal{A}$. If $\mathbf{d}^\intercal$ is random, so is $\mathbf{c}^\intercal$ and $\mathcal{B}$ has simulated H_3 for $\mathcal{A}$. Therefore, their advantages are the same.

If MDDH_k holds over $\mathsf{GroupGen}$, we have $\mathsf{H}_2 \approx \mathsf{H}_3$. □

5.5 Ciphertext-Policy 1-ABE for ABP

In this section, we construct the core component of our adaptively secure ABE, called *1-ABE*, from any gradually 2-simulation-secure IPFE.

Definition 5.5 (1-ABE). Let GroupGen be a group sampler (Definition 2.4). A *(public-key) 1-ABE scheme* based on GroupGen is an ABE (Definition 2.3) except that Enc takes $\mu \in \mathbb{Z}_p$ as the message and Dec recovers $[\![\mu]\!]$ if decryption is authorized. Such a scheme is *1-key secure* (or simply *secure*) if $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$, where $\mathsf{Exp}^b_{\text{1-ABE}}$ is $\mathsf{Exp}^b_{\text{PHFE}}$ (Definition 2.2) with the following modifications.

- In **Query I/II**, the adversary is allowed to query at most one secret key in total.
- In **Challenge**, the adversary chooses x without the messages. Instead, the message is 0 in $\mathsf{Exp}^0_{\text{1-ABE}}$ and uniformly random in $\mathsf{Exp}^1_{\text{1-ABE}}$. The adversary does not receive the potential random message.

The relaxation of decryption correctness and the change of messages in the security definition are because 1-ABE will be used to encapsulate keys for full ABE. In full ABE, the group-encoded decryption result of 1-ABE is used to mask the message, and we argue security by replacing the encapsulated key by random.

ABE constructions in many previous works such as [KW19] and Chapter 4 go through an intermediate step of building a *secret-key* 1-ABE that is 1-key *1-ciphertext* secure. In the secret-key setting, keys and ciphertexts are symmetric. Consequently, there is no distinction between ciphertext-policy and key-policy 1-ABE. In contrast, 1-ABE in this chapter is *public-key* and *1-key* secure. This asymmetry separates CP-1-ABE and KP-1-ABE. We remark that the 1-ABE in [KW19] and Chapter 4 can be easily modified to fit Definition 5.5 as CP-1-ABE. We will see that Definition 5.5 is easier to use in reductions for full ABE.

Ingredients of Construction 5.5. Let

- $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS (Definitions 4.7 and 4.8) for $\mathcal{F}$,
- GroupGen a group sampler (Definition 2.4), and

- (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec) an IPFE scheme (Definition 5.1) based on GroupGen.

Construction 5.5 (CP-1-ABE). Define (see Definitions 2.3 and 5.5)

$$\mathsf{Params}_{\mathsf{gp}} = \{\, 1^n \mid n \in \mathbb{N} \,\}, \qquad F_{\mathsf{gp},1^n} = \mathbb{Z}_p^n,$$
$$X_{\mathsf{gp},1^n} = \{\, f_{\neq 0}, f_{=0} \mid f \in \mathcal{F},\ f : \mathbb{Z}_p^n \to \mathbb{Z}_p \,\}, \qquad P_{\mathsf{gp},1^n}(\mathbf{x}, f') = f'(\mathbf{x}),$$

where p is the group order in gp. Our 1-ABE scheme based on GroupGen works as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^n)$ takes as input the group description gp and the attribute length n. It runs

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^{n+1}, 1^2),$$

  and outputs $(\mathsf{mpk}, \mathsf{msk}) = (\mathsf{impk}, \mathsf{imsk})$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ takes as input msk and $\mathbf{x} \in \mathbb{Z}_p^n$. It runs

$$\mathsf{isk} \xleftarrow{\$} \mathsf{IPFE.KeyGen}\big(\mathsf{imsk}, (1, \mathbf{x}^\intercal)^\intercal\big)$$

  and outputs $\mathsf{sk} = \mathsf{isk}$.

- $\mathsf{Enc}(\mathsf{mpk}, f', \mu)$ takes as input mpk, some f' (either $f_{\neq 0}$ or $f_{=0}$), and $\mu \in \mathbb{Z}_p$. It garbles f underlying f' with μ and encrypts the label functions into IPFE ciphertexts by running

$$\begin{aligned}
&(\text{if } f' = f_{\neq 0}) \qquad \alpha \leftarrow \mu, \qquad \beta \leftarrow 0;\\
&(\text{if } f' = f_{=0}) \qquad \alpha \xleftarrow{\$} \mathbb{Z}_p, \qquad \beta \leftarrow \mu;\\
&(\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta),\\
&(\text{for } j \in [m]) \qquad \mathsf{ict}_j \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_j]\!]).
\end{aligned}$$

  The algorithm outputs $\mathsf{ct} = (\mathsf{ict}_1, \ldots, \mathsf{ict}_m)$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}, \mathsf{sk}, f', \mathsf{ct})$ takes as input mpk, a secret key sk for $\mathbf{x}$, and a ciphertext ct for f'. If $f'(\mathbf{x}) = 0$, the algorithm outputs $\perp$ and terminates. Otherwise, it parses

sk, ct as in KeyGen, Enc, computes

$$(\text{for } j \in [m]) \qquad [\![\ell_j]\!] \leftarrow \mathsf{IPFE.Dec}\big(\mathsf{impk}, (1, \mathbf{x}^\intercal)^\intercal, \mathsf{isk}, \mathsf{ict}_j\big),$$

$$[\![\mu']\!] \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1, \ldots, \ell_m]\!]), & \text{if } f' = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1, \ldots, \ell_m]\!]), & \text{if } f' = f_{=0}; \end{cases}$$

and outputs $[\![\mu']\!]$.

Correctness and Efficiency. We verify that the scheme is correct. By the correctness of IPFE, we have

$$\ell_j = \mathbf{L}_j^\intercal \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = L_j(\mathbf{x}),$$

where L_j's are the label functions defined by Garble. Since Eval is linear in the labels, it can be performed in the exponent. By the correctness of AKGS,

$$(\text{if } f' = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0) \qquad \mu' = \tfrac{1}{f(\mathbf{x})}(\alpha f(\mathbf{x}) + \beta) = \tfrac{1}{f(\mathbf{x})}(\mu f(\mathbf{x}) + 0) = \mu;$$

$$(\text{if } f' = f_{=0} \text{ and } f(\mathbf{x}) = 0) \qquad \mu' = \alpha f(\mathbf{x}) + \beta = \alpha \cdot 0 + \mu = \mu.$$

Therefore, Construction 5.5 is correct.

Since the CP-1-ABE secret key is simply an IPFE secret key, if the underlying IPFE has succinct keys, so does our CP-1-ABE. When instantiated using Construction 5.1, each secret key consists of just three elements of $\mathbb{Z}_p$.

Security. We state and prove the security of Construction 5.5.

Theorem 5.8 (¶). *Suppose in Construction 5.5, the IPFE is* gradually 2-simulation-secure *(Definition 5.2) and the AKGS is* piecewise secure *(Definition 4.10), then the resultant CP-1-ABE scheme is* (1-key) secure *(Definition 5.5).*

Proof (Theorem 5.8). Recall that in $\mathsf{Exp}^b_{\text{1-ABE}}$, the adversary receives mpk and can query one secret key for attribute $\mathbf{x} \in \mathbb{Z}_p^n$. In addition, it receives a challenge ciphertext tied to some policy f' of its choice, encrypting either 0 (in $\mathsf{Exp}^0_{\text{1-ABE}}$) or $\delta \xleftarrow{\$} \mathbb{Z}_p$ (in $\mathsf{Exp}^1_{\text{1-ABE}}$). The adversary is free to ask for either the secret key or the ciphertext first. Its goal is to distinguish the two cases, and we want to show $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$.

Suppose the garbling size of the underlying ABP of f' is m, we go through a series of hybrids $\mathsf{H}^b_{1\sim m,1\sim 3}$. In these hybrids, we gradually modify the ciphertext queried by the adversary, which starts out as encrypting label functions $\mathbf{L}_j$ generated by AKGS Garble and ends up by encrypting simulated labels $\ell_j \xleftarrow{\$} \mathbb{Z}_p$. The proof interleaves between using the gradual simulation security of IPFE and the piecewise security of AKGS, in a fashion similar to Chapter 4.

The hybrids $\mathsf{H}^b_{1,1\sim 3}$ change ict_1 from encrypting the label function $\mathbf{L}_1$ to being simulated using the reversely sampled first label ℓ_1.

- $\mathsf{H}^b_{1,1}$ proceeds identically to $\mathsf{Exp}^b_{\text{1-ABE}}$, except that we use the simulator of IPFE to generate mpk and sk for $\mathbf{x}$, instead of using the real IPFE algorithms. By the gradual 2-simulation security of IPFE ($\mathsf{Exp}_{\text{real}} \approx \mathsf{Exp}_{\text{sim}}$), we have $\mathsf{Exp}^b_{\text{1-ABE}} \approx \mathsf{H}^b_{1,1}$.
- $\mathsf{H}^b_{1,2}$ proceeds identically to $\mathsf{H}^b_{1,1}$, except that ict_1 (encoding the label function L_1 in the challenge ciphertext) is simulated. By the gradual 2-simulation security of IPFE ($\mathsf{Exp}^0_{\text{2-GS}} \approx \mathsf{Exp}^1_{\text{2-GS}}$), we have $\mathsf{H}^b_{1,1} \approx \mathsf{H}^b_{1,2}$. Note that in $\mathsf{H}^b_{1,2}$, the first label function L_1 is not directly used, and only the first label $\ell_1 = L_1(\mathbf{x})$ is used.
- $\mathsf{H}^b_{1,3}$ proceeds identically to $\mathsf{H}^b_{1,2}$, except that the ℓ_1 used to simulate ict_1 (or isk, whichever comes later; same below for every ciphertext simulation) is reversely sampled instead of being computed from L_1. By the reverse sampleability of AKGS, $\mathsf{H}^b_{1,2} \equiv \mathsf{H}^b_{1,3}$.

The remaining hybrids $\mathsf{H}^b_{j,1\sim 3}$ (for $j = 2, \ldots, m$) change ict_j from encrypting the label function $\mathbf{L}_j$ to encrypting the simulated label $\ell_j \xleftarrow{\$} \mathbb{Z}_p$, via a temporary simulated form.

- $\mathsf{H}^b_{j,1}$ proceeds identically to $\mathsf{H}^b_{1,3}$, except that $\mathsf{ict}_{j'}$ for $1 < j' < j$ encrypts $(\ell_{j'}, \mathbf{0})$ for $\ell_{j'} \xleftarrow{\$} \mathbb{Z}_p$ (here, $\mathbf{0}$ is of the same length as $\mathbf{x}$), and ict_j is simulated using $\ell_j = L_j(\mathbf{x})$. By the gradual 2-simulation security of IPFE ($\mathsf{Exp}^0_{\text{2-GS}} \approx \mathsf{Exp}^1_{\text{2-GS}}$), we have $\mathsf{H}^b_{1,3} \approx \mathsf{H}^b_{2,1}$.

  Similarly, $\mathsf{H}^b_{j-1,3} \approx \mathsf{H}^b_{j,1}$, which should be clear once we specify $\mathsf{H}^b_{j,3}$.
- $\mathsf{H}^b_{j,2}$ proceeds identically to $\mathsf{H}^b_{j,1}$, except that ℓ_j is sampled uniformly at random instead of being computed as $L_j(\mathbf{x})$. Note that both $\mathsf{H}^b_{j,1}$ and $\mathsf{H}^b_{j,2}$ can be generated

using just ℓ_j and $\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$ from the garbling, so the marginal randomness property applies and $\mathsf{H}^b_{j,1} \equiv \mathsf{H}^b_{j,2}$.

- $\mathsf{H}^b_{j,3}$ proceeds identically to $\mathsf{H}^b_{j,2}$, except that ict_j is reverted to a normal ciphertext, now encrypting $(\ell_j, \mathbf{0})$, where $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{0}$ is of the same length as $\mathbf{x}$. By the gradual 2-simulation security of IPFE ($\mathsf{Exp}^1_{\text{2-GS}} \approx \mathsf{Exp}^0_{\text{2-GS}}$), we have $\mathsf{H}^b_{j,2} \approx \mathsf{H}^b_{j,3}$.

  Note that in $\mathsf{H}^b_{j+1,1}$, the next hybrid, ict_{j+1} becomes simulated using $\ell_{j+1} = L_{j+1}(\mathbf{x})$. We have $\mathsf{H}^b_{j,3} \approx \mathsf{H}^b_{j+1,1}$ again by the gradual 2-simulation security of IPFE.

Lastly, we argue $\mathsf{H}^0_{m,3} \equiv \mathsf{H}^1_{m,3}$. In both hybrids, ict_j's for $1 < j \le m$ encrypt $(\ell_j, \mathbf{0})$ for $\ell_j \xleftarrow{\$} \mathbb{Z}_p$, and ict_1 is simulated using ℓ_1, which is reversely sampled. We have two cases under the constraint $f'(\mathbf{x}) = 0$:

$$\text{if } f' = f_{\neq 0}, \quad \ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \delta f(\mathbf{x}) + 0, \ell_2, \ldots, \ell_m) \equiv \mathsf{RevSamp}(f, \mathbf{x}, 0, \ell_2, \ldots, \ell_m);$$

$$\text{if } f' = f_{=0}, \quad \ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \delta, \ell_2, \ldots, \ell_m) \equiv \mathsf{RevSamp}(f, \mathbf{x}, \gamma, \ell_2, \ldots, \ell_m) \text{ for } \gamma \xleftarrow{\$} \mathbb{Z}_p.$$

Here, δ is either 0 (in $\mathsf{H}^0_{m,3}$) or random (in $\mathsf{H}^1_{m,3}$), and when $f' = f_{=0}$ and $f(\mathbf{x}) \neq 0$, the random α is not used elsewhere so $\alpha f(\mathbf{x})$ is a one-time pad that hides δ. In both cases, ℓ_1 as well as any other value used to generate the response to the adversary contains no information about δ. Therefore, $\mathsf{H}^0_{m,3} \equiv \mathsf{H}^1_{m,3}$.[45] By hybrid argument, we have $\mathsf{Exp}^0_{\text{1-ABE}} \approx \mathsf{Exp}^1_{\text{1-ABE}}$. □

5.6 Key-Policy ABE for ABP

In this section, we apply the classic dual system encryption to obtain full KP-ABE from CP-1-ABE instantiated with the IPFE in Section 5.4.1.

Ingredients of Construction 5.6. Let

- $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS (Definitions 4.7 and 4.8) for $\mathcal{F}$,

[45] In fact, $\mathsf{H}^0_{m,2} \equiv \mathsf{H}^1_{m,2}$, but that requires more careful argument.

- GroupGen a pairing group sampler (Definition 2.5) over which MDDH_k (Assumption 2.1) holds.

Construction 5.6 (KP-ABE). Define (see Definition 2.3)

$$\mathsf{Params}_{\mathsf{gp}} = \{\, 1^n \mid n \in \mathbb{N} \,\}, \qquad F_{\mathsf{gp},1^n} = \{\, f_{\neq 0}, f_{=0} \mid f \in \mathcal{F},\ f : \mathbb{Z}_p^n \to \mathbb{Z}_p \,\},$$
$$X_{\mathsf{gp},1^n} = \mathbb{Z}_p^n, \qquad Y_{\mathsf{gp},1^n} = \mathbb{G}_{\mathrm{T}}, \qquad P_{\mathsf{gp},1^n}(f', \mathbf{x}) = f'(\mathbf{x}),$$

where p is the group order in gp. Our KP-ABE scheme based on GroupGen operates as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^n)$ takes as input the group description gp and the attribute length n. It samples and sets

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+2)}, \quad \mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}, \qquad \mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times(k+1)(n+1)}, \quad \boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^{k+1},$$
$$\mathsf{xpk} = ([\![\mathbf{B}^\intercal]\!]_1, [\![\mathbf{W}(\mathbf{B}^\intercal \otimes \mathbf{I}_{n+1})]\!]_1), \qquad \mathsf{fpk} = ([\![\mathbf{A}]\!]_2, [\![\mathbf{AW}]\!]_2),$$
$$\mathsf{mpk} = ([\![\boldsymbol{\mu}^\intercal \mathbf{B}^\intercal]\!]_{\mathrm{T}}, \mathsf{xpk}), \qquad \mathsf{msk} = (\mathsf{fpk}, \boldsymbol{\mu}).$$

  The algorithm outputs $(\mathsf{mpk}, \mathsf{msk})$.

  Here, xpk (resp. fpk) means "public key for $\mathbf{x}$" (resp. f), and security holds even if both are included in mpk. For KP-ABE (resp. CP-ABE in Section 5.7.2), Enc does not need fpk (resp. xpk), hence its omission. The connection of our KP-ABE with CP-1-ABE and dual system encryption (as discussed in Section 5.2.3) is as follows. The matrix $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_{k+1})$ consists of $(k+1)$ master secret keys of CP-1-ABE concatenated by columns, each of shape $(k+2) \times (n+1)$. Its projection along a vector $\mathbf{b} = (b_1, \ldots, b_{k+1})^\intercal$ is

$$\begin{aligned} b_1\mathbf{W}_1 + \cdots + b_{k+1}\mathbf{W}_{k+1} &= \mathbf{W}_1 \cdot b_1\mathbf{I}_{n+1} + \cdots + \mathbf{W}_{k+1} \cdot b_{k+1}\mathbf{I}_{n+1} \\ &= (\mathbf{W}_1, \ldots, \mathbf{W}_{k+1}) \begin{pmatrix} b_1\mathbf{I}_{n+1} \\ \vdots \\ b_{k+1}\mathbf{I}_{n+1} \end{pmatrix} = \mathbf{W}(\mathbf{b} \otimes \mathbf{I}_{n+1}). \end{aligned}$$

  The matrix $\mathbf{B} = (\mathbf{b}_1^\intercal, \ldots, \mathbf{b}_k^\intercal)^\intercal$ consists of all the projection vectors, and the matrix $\mathbf{W}(\mathbf{B}^\intercal \otimes \mathbf{I}_{n+1})$ is the projections of $\mathbf{W}$ along $\mathbf{B}$ concatenated by columns.

- $\mathsf{KeyGen}(\mathsf{msk}, f')$ garbles f' with $\boldsymbol{\mu}$ by

$$\begin{aligned}
&(\text{if } f' = f_{\neq 0}) \qquad \boldsymbol{\alpha} \leftarrow \boldsymbol{\mu}, \qquad \boldsymbol{\beta} \leftarrow \mathbf{0};\\
&(\text{if } f' = f_{=0}) \qquad \boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{k+1}, \qquad \boldsymbol{\beta} \leftarrow \boldsymbol{\mu};\\
&\qquad (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \boldsymbol{\alpha}, \boldsymbol{\beta}).
\end{aligned}$$

  It samples $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [m]$ and sets

$$\mathsf{sk}_{j,1} = \mathbf{s}_j^\top [\![\mathbf{A}]\!]_2, \qquad \mathsf{sk}_{j,2} = \mathbf{s}_j^\top [\![\mathbf{AW}]\!]_2 + [\![\mathbf{L}_j^\top]\!]_2.$$

  The algorithm outputs $\mathsf{sk} = (\mathsf{sk}_{1,1}, \mathsf{sk}_{1,2}, \ldots, \mathsf{sk}_{m,1}, \mathsf{sk}_{m,2})$.

  Generating a key in KP-ABE means encrypting $\boldsymbol{\mu}$ in each CP-1-ABE instance, which boils down to generating IPFE ciphertexts, as shown above.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, g)$ samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets

$$\mathsf{ct}_1 = [\![\mathbf{B}^\top]\!]_1 \mathbf{r}, \qquad \mathsf{ct}_2 = [\![\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1})]\!]_1 \left(\mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}\right), \qquad \mathsf{ct}_3 = g + [\![\boldsymbol{\mu}^\top \mathbf{B}^\top]\!]_\mathrm{T} \mathbf{r}.$$

  The algorithms outputs $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$.

  We remark that $\mathbf{r}$ (resp. $\mathbf{B}^\top \mathbf{r}$) is the coefficients of random linear combination with respect to the projections (resp. the CP-1-ABE instances). Here, ct_2 is a CP-1-ABE key with respect to the projected master secret key $\mathbf{W}(\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1})$, which is an IPFE secret key for $\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$, i.e.,

$$\mathsf{ct}_2 = \left[\!\!\left[\mathbf{W}(\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1}) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right]\!\!\right]_1.$$

  The ciphertext consists of $(2k+3)$ elements of $\mathbb{G}_1$ and one element of $\mathbb{G}_\mathrm{T}$, hence is succinct.

- $\mathsf{Dec}(\mathsf{mpk}, f', \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ first checks if $f'(\mathbf{x}) = 1$. If not, it outputs $\perp$ and terminates. Otherwise, the algorithm parses sk, ct as defined in $\mathsf{KeyGen}, \mathsf{Enc}$, computes

$$(\text{for } j \in [m]) \qquad [\![\ell_j]\!]_\mathrm{T} \leftarrow -\mathsf{sk}_{j,1}\mathsf{ct}_2 + \mathsf{sk}_{j,2}\left(\mathsf{ct}_1 \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}\right);$$

$$[\![\mu']\!]_\mathrm{T} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1, \ldots, \ell_m]\!]_\mathrm{T}), & \text{if } f' = f_{\neq 0};\\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1, \ldots, \ell_m]\!]_\mathrm{T}), & \text{if } f' = f_{=0}; \end{cases}$$

and outputs $(\mathsf{ct}_3 - [\![\mu']\!]_T)$.

Correctness. By definition (see also Construction 5.1),

$$\ell_j = -\mathbf{s}_j^\top \mathbf{A}\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) \left(\mathbf{r} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) + (\mathbf{s}_j^\top \mathbf{A}\mathbf{W} + \mathbf{L}_j^\top) \left(\mathbf{B}^\top \mathbf{r} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right)$$
$$= \mathbf{L}_j^\top \left(\mathbf{B}^\top \mathbf{r} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) = \mathbf{L}_j^\top \left(\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1}\right) \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}.$$

By Lemma 5.1, if we define

$$(\mathbf{L}'_1, \ldots, \mathbf{L}'_m) \leftarrow \mathsf{Garble}(f, \mathbf{r}^\top \mathbf{B}\boldsymbol{\alpha}, \mathbf{r}^\top \mathbf{B}\boldsymbol{\beta}; \mathbf{r}^\top \mathbf{B}\mathbf{R}),$$

where $\mathbf{R}$ is the randomness used to generate $\mathbf{L}_j$'s, then

$$\ell_j = \mathbf{L}_j^\top \left(\mathbf{B}^\top \mathbf{r} \otimes \mathbf{I}_{n+1}\right) \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix} = (\mathbf{L}'_j)^\top \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix} = L'_j(\mathbf{x}).$$

By the correctness of AKGS, we have

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \mathbf{r}^\top \mathbf{B}\boldsymbol{\alpha} f(\mathbf{x}) + \mathbf{r}^\top \mathbf{B}\boldsymbol{\beta}.$$

In the two cases where decryption should succeed,

$$(\text{if } f' = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0) \qquad \mu' = \tfrac{1}{f(\mathbf{x})}(\mathbf{r}^\top \mathbf{B}\boldsymbol{\mu} f(\mathbf{x}) + \mathbf{r}^\top \mathbf{B}\mathbf{0}) = \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu};$$
$$(\text{if } f' = f_{=0} \text{ and } f(\mathbf{x}) = 0) \qquad \mu' = \mathbf{r}^\top \mathbf{B}\boldsymbol{\alpha} f(\mathbf{x}) + \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu} = \mathbf{r}^\top \mathbf{B}\boldsymbol{\mu}.$$

Therefore, in both cases,

$$\mathsf{ct}_3 - [\![\mu']\!]_T = g + [\![\boldsymbol{\mu}^\top \mathbf{B}^\top \mathbf{r}]\!]_T - [\![\mathbf{r}^\top \mathbf{B}\boldsymbol{\mu}]\!]_T = g.$$

Minimizing Pairing Operations. The number of pairing operations in Dec appears to depend on the garbling size of the policy and the attribute length. It can be reduced to $(2k + 3)$ as follows.[46] Since Eval is linear in the labels, one can find $\gamma_1, \ldots, \gamma_m \in \mathbb{Z}_p$ (dependent on $f, \mathbf{x}$) such that

$$\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \sum_{j=1}^{m} \gamma_j \ell_j.$$

[46] Syntactically, we use the pairing groups in black box, and there are only $(2k + 3)$ elements of $\mathbb{G}_1$ in a ciphertext and no element of $\mathbb{G}_1$ in a secret key, so the operations can always be regrouped to use at most $(2k + 3)$ pairing operations. The text below provides the concrete regrouping method.

The computation of $[\![\mu']\!]_T$ can be rewritten as

$$[\![\mu']\!]_T = \sum_{j=1}^{m} \gamma_j [\![\ell_j]\!]_T = \sum_{j=1}^{m} \gamma_j \left(-\mathsf{sk}_{j,1}\mathsf{ct}_2 + \mathsf{sk}_{j,2}\left(\mathsf{ct}_1 \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right)\right)$$
$$= -\left(\sum_{j=1}^{m} \gamma_j \mathsf{sk}_{j,1}\right)\mathsf{ct}_2 + \left(\sum_{j=1}^{m} \gamma_j \mathsf{sk}_{j,2}\right)\left(\mathbf{I}_{k+1} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right)\mathsf{ct}_1.$$

Note that ct_1 and ct_2 consist of $(k+1)$ and $(k+2)$ group elements respectively and only these elements (of $\mathbb{G}_1$) take part in pairing. Therefore, the formula above only uses $(2k+3)$ pairing operations.

There are further optimizations possible, such as appropriately choosing which of the two source groups to use for the secret key to reduce the cost of exponentiation in decryption.

Security. We state and prove the security of our KP-ABE.

Theorem 5.9 (¶). *Suppose in Construction 5.6, the AKGS is* piecewise secure *(Definition 4.10) and* MDDH_k *(Assumption 2.1) holds over* GroupGen*, then the resultant ABE scheme is* adaptively secure *(Definition 2.2).*

Proof (Theorem 5.9). Recall that in the security experiments, the adversary receives the master public key, many secret keys, and a challenge ciphertext. Suppose the queried keys are for $f'_1, \ldots, f'_Q$ and the challenge ciphertext encrypts one of g_0, g_1 with attribute $\mathbf{x}$. The constraint is $f'_q(\mathbf{x}) = 0$ for all $q \in [Q]$, and we want to show that the adversary cannot distinguish which of g_0, g_1 is encrypted.

The proof follows dual system encryption methodology (see Section 5.2.3). More formally, we go through hybrids $\mathsf{H}_1^b, \mathsf{H}_2^b$.

- H_1^b proceeds identically to $\mathsf{Exp}^b_{\mathrm{PHFE}}$ (Definition 2.2), except that the challenge ciphertext becomes

$$\mathsf{ct}_1 = [\![\ \boxed{\mathbf{c}}\ ]\!]_1, \qquad \mathsf{ct}_2 = \left[\!\!\left[\mathbf{W}\left(\boxed{\mathbf{c}} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right)\right]\!\!\right]_1, \qquad \mathsf{ct}_3 = g_b + [\![\boldsymbol{\mu}^\intercal \boxed{\mathbf{c}}\ ]\!]_T,$$

  where $\mathbf{c}$ is uniformly random outside the row space of $\mathbf{B}$.

- H_2^b proceeds identically to H_1^b, except that each queried key[47] is garbled for $(\boldsymbol{\mu} + \delta_q \mathbf{b}^\perp)$, i.e.,

$$\begin{aligned}
&(\text{if } f'_q = f_{q,\neq 0}) \qquad \boldsymbol{\alpha}_q \leftarrow \boldsymbol{\mu} + \boxed{\delta_q}\,\mathbf{b}^\perp, \qquad \boldsymbol{\beta}_q \leftarrow \mathbf{0};\\
&(\text{if } f'_q = f_{q,=0}) \qquad \boldsymbol{\alpha}_q \xleftarrow{\$} \mathbb{Z}_p^{k+1}, \qquad\quad \boldsymbol{\beta}_q \leftarrow \boldsymbol{\mu} + \boxed{\delta_q}\,\mathbf{b}^\perp;\\
&\qquad (\mathbf{L}_{q,1}, \ldots, \mathbf{L}_{q,m_q}) \xleftarrow{\$} \mathsf{Garble}(f_q, \boldsymbol{\alpha}_q, \boldsymbol{\beta}_q).
\end{aligned}$$

  Here, $\mathbf{b}^\perp$ is a fixed vector such that $\mathbf{B}\mathbf{b}^\perp = \mathbf{0}$ and $\mathbf{c}^\intercal \mathbf{b}^\perp = 1$, and $\delta_q \xleftarrow{\$} \mathbb{Z}_p$ is fresh per key.

We have the following claims.

Claim 5.10 (¶). $\mathsf{Exp}^b_{\mathrm{PHFE}} \approx \mathsf{H}_1^b$ *reduces to* MDDH_k *(Assumption 2.1) over* $\mathbb{G}_1$.

Claim 5.11 (¶). $\mathsf{H}_1^b \approx \mathsf{H}_2^b$ *reduces to the security of CP-1-ABE (Construction 5.5 instantiated with gradually* 2*-simulation-secure Construction 5.1 using* $\mathbb{G}_2$*, which relies on the piecewise security of AKGS and* MDDH_k *over* $\mathbb{G}_2$*).*

Claim 5.12 (¶). $\mathsf{H}_2^0 \equiv \mathsf{H}_2^1$.

Once we prove them, we have $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$ by hybrid argument. □

Proof (Claim 5.10). Let $\mathcal{A}$ be an efficient adversary distinguishing $\mathsf{Exp}^b_{\mathrm{PHFE}}$ and H_1^b, we construct $\mathcal{B}$ against MDDH_k over $\mathbb{G}_1$. Upon receiving MDDH_k challenge $[\![\mathbf{B}, \mathbf{c}^\intercal]\!]_1$ where either $\mathbf{c}^\intercal = \mathbf{r}^\intercal \mathbf{B}$ or $\mathbf{c}^\intercal$ is random, $\mathcal{B}$ launches $\mathcal{A}()$, receives the attribute length 1^n from it, samples $\mathbf{A}, \mathbf{W}, \boldsymbol{\mu}$, and sets

$$\begin{aligned}
\mathsf{xpk} &= \left([\![\mathbf{B}^\intercal]\!]_1, \mathbf{W}([\![\mathbf{B}^\intercal]\!]_1 \otimes \mathbf{I}_{n+1})\right), & \mathsf{fpk} &= ([\![\mathbf{A}]\!]_2, [\![\mathbf{A}\mathbf{W}]\!]_2),\\
\mathsf{mpk} &= ([\![\boldsymbol{\mu}^\intercal]\!]_2 [\![\mathbf{B}^\intercal]\!]_1, \mathsf{xpk}), & \mathsf{msk} &= (\mathsf{fpk}, \boldsymbol{\mu}).
\end{aligned}$$

It sends mpk to $\mathcal{A}$ and responds to the key queries normally. To respond to the ciphertext challenge for one of g_0, g_1 with attribute $\mathbf{x}$, our $\mathcal{B}$ sets

$$\mathsf{ct}_1 = [\![\mathbf{c}]\!]_1, \qquad \mathsf{ct}_2 = \mathbf{W}\left([\![\mathbf{c}]\!]_1 \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}\right), \qquad \mathsf{ct}_3 = g_b + [\![\boldsymbol{\mu}^\intercal]\!]_2 [\![\mathbf{c}]\!]_1.$$

[47] In a typical dual system proof, the keys are switched one by one. In our case, we can change them all at once since our 1-ABE is public-key.

Lastly, it outputs whatever $\mathcal{A}$ outputs.

Clearly $\mathcal{B}$ is efficient. Note that in $\mathsf{Exp}^b_{\mathrm{PHFE}}$, the exponents in ct_2 are

$$\mathbf{W}(\mathbf{B}^\intercal \otimes \mathbf{I}_{n+1})\left(\mathbf{r} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) = \mathbf{W}\left(\mathbf{B}^\intercal\mathbf{r} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right).$$

If $\mathbf{c}^\intercal = \mathbf{r}^\intercal\mathbf{B}$, our $\mathcal{B}$ has simulated $\mathsf{Exp}^b_{\mathrm{PHFE}}$ for $\mathcal{A}$. If $\mathbf{c}^\intercal$ is random, our $\mathcal{B}$ has (statistically closely) simulated H^b_1 for $\mathcal{A}$ — a uniformly random $\mathbf{c}^\intercal$ lies outside the row space of $\mathbf{B}$ with overwhelming probability. Therefore, their advantages are negligibly close.

If MDDH_k holds over $\mathbb{G}_1$, we have $\mathsf{Exp}^b_{\mathrm{PHFE}} \approx \mathsf{H}^b_1$. □

Proof (Claim 5.11). Let $\mathbf{b}^\perp$ be such that $\mathbf{B}\mathbf{b}^\perp = \mathbf{0}$ and $\mathbf{c}^\intercal\mathbf{b}^\perp = 1$ for the $\mathbf{B}$ used in the master public key and the $\mathbf{c}^\intercal$ used in the challenge ciphertext. We set[48]

$$\mathbf{W} = \widetilde{\mathbf{W}} + (\mathbf{b}^\perp)^\intercal \otimes \widehat{\mathbf{W}}, \qquad \boldsymbol{\mu}_q = \boldsymbol{\mu} + \delta_q\mathbf{b}^\perp,$$

where $\widetilde{\mathbf{W}} \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times(k+1)(n+1)}$, $\widehat{\mathbf{W}} \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times(n+1)}$, $\boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, and either $\delta_q = 0$ (in H^b_1) or $\delta_q \xleftarrow{\$} \mathbb{Z}_p$ (in H^b_2). Recall that to create a secret key for f'_q, we need to garble f'_q with $\boldsymbol{\mu}_q$. Let m_q, m'_q be the randomness dimension and the garbling size of f'_q. We set

$$\begin{aligned}
&(\text{if } f'_q = f_{q,\neq 0}) \quad && \widetilde{\boldsymbol{\alpha}}_q \leftarrow \boldsymbol{\mu}, && \widehat{\alpha}_q \leftarrow \delta_q, && \widetilde{\boldsymbol{\beta}}_q \leftarrow \mathbf{0}, && \widehat{\beta}_q \leftarrow 0;\\
&(\text{if } f'_q = f_{q,=0}) \quad && \widetilde{\boldsymbol{\alpha}}_q \xleftarrow{\$} \mathbb{Z}_p^{k+1}, && \widehat{\alpha}_q \xleftarrow{\$} \mathbb{Z}_p, && \widetilde{\boldsymbol{\beta}}_q \leftarrow \boldsymbol{\mu}, && \widehat{\beta}_q \leftarrow \delta_q;\\
&&& \boldsymbol{\alpha}_q = \widetilde{\boldsymbol{\alpha}}_q + \widehat{\alpha}_q\mathbf{b}^\perp, && \boldsymbol{\beta}_q = \widetilde{\boldsymbol{\beta}}_q + \widehat{\beta}_q\mathbf{b}^\perp,\\
&&& \widetilde{\mathbf{R}}_q \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times m'_q}, && \mathbf{r}_q \xleftarrow{\$} \mathbb{Z}_p^{m'_q}, && \mathbf{R}_q = \widetilde{\mathbf{R}}_q + \mathbf{b}^\perp\mathbf{r}_q^\intercal,\\
&&& (\widetilde{\mathbf{L}}_{q,1},\dots,\widetilde{\mathbf{L}}_{q,m_q}) \leftarrow \mathsf{Garble}(f, \widetilde{\boldsymbol{\alpha}}_q, \widetilde{\boldsymbol{\beta}}_q; \widetilde{\mathbf{R}}_q),\\
&&& (\widehat{\mathbf{L}}_{q,1},\dots,\widehat{\mathbf{L}}_{q,m_q}) \leftarrow \mathsf{Garble}(f, \widehat{\alpha}_q, \widehat{\beta}_q; \mathbf{r}_q),\\
&&& (\mathbf{L}_{q,1},\dots,\mathbf{L}_{q,m_q}) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}_q, \boldsymbol{\beta}_q; \mathbf{R}_q).
\end{aligned}$$

By Lemma 5.1 and the linearity of Garble, we have $\mathbf{L}_{q,j} = \widetilde{\mathbf{L}}_{q,j} + \mathbf{b}^\perp \otimes \widehat{\mathbf{L}}_{q,j}$.

With the above preparation, we can describe the reduction. Let $\mathcal{A}$ be an efficient adversary distinguishing H^b_1 and H^b_2, we construct $\mathcal{B}$ against the security of CP-1-ABE

[48] Components with hat will come from CP-1-ABE, and components with tilde will be sampled by the reduction algorithm.

(Construction 5.5 instantiated with gradually 2-simulation-secure Construction 5.1 using $\mathbb{G}_2$). The adversary $\mathcal{B}$ launches $\mathcal{A}()$, receives the attribute length 1^n from it, and forwards 1^n to the security experiment of CP-1-ABE. It receives $\widehat{\mathsf{mpk}} = ([\![\mathbf{A}]\!]_2, [\![\mathbf{A}\widehat{\mathbf{W}}]\!]_2)$. The adversary $\mathcal{B}$ samples $\widetilde{\mathbf{W}}, \mathbf{B}, \mathbf{c}, \boldsymbol{\mu}$ and computes $\mathbf{b}^\perp$ as specified above. Note that

$$\begin{aligned}\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) &= (\widetilde{\mathbf{W}} + (\mathbf{b}^\perp)^\top \otimes \widehat{\mathbf{W}})(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) \\ &= \widetilde{\mathbf{W}}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) + (\mathbf{b}^\perp)^\top \mathbf{B}^\top \otimes \widehat{\mathbf{W}} = \widetilde{\mathbf{W}}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}),\end{aligned}$$

so $\mathcal{B}$ sets

$$\mathsf{xpk} = ([\![\mathbf{B}^\top]\!]_1, [\![\widetilde{\mathbf{W}}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1})]\!]_1), \qquad \mathsf{mpk} = ([\![\boldsymbol{\mu}^\top \mathbf{B}^\top]\!]_T, \mathsf{xpk}).$$

It sends mpk to $\mathcal{A}$ and starts to respond to its queries and challenges. To respond to a secret key query for f'_q, our $\mathcal{B}$ computes $\widetilde{\mathbf{L}}_{q,1}, \ldots, \widetilde{\mathbf{L}}_{q,m_q}$ as specified above, and queries the security experiment of CP-1-ABE for a ciphertext for f'_q, which consists of

$$\mathsf{ict}_{q,j} = ([\![\mathbf{s}_{q,j}^\top \mathbf{A}]\!]_2, [\![\mathbf{s}_{q,j}^\top \mathbf{A}\widehat{\mathbf{W}} + \widehat{\mathbf{L}}_{q,j}^\top]\!]_2) \qquad \text{for } j \in [m_q].$$

It stitches the response as $\mathsf{sk}_{q,j,1} = [\![\mathbf{s}_{q,j}^\top \mathbf{A}]\!]_2$ and

$$\begin{aligned}\mathsf{sk}_{q,j,2} = [\![\mathbf{s}_{q,j}^\top \mathbf{A}\mathbf{W} + \mathbf{L}_{q,j}^\top]\!]_2 &= [\![\mathbf{s}_{q,j}^\top \mathbf{A}(\widetilde{\mathbf{W}} + (\mathbf{b}^\perp)^\top \otimes \widehat{\mathbf{W}}) + \widetilde{\mathbf{L}}_{q,j}^\top + (\mathbf{b}^\perp)^\top \otimes \widehat{\mathbf{L}}_{q,j}^\top]\!]_2 \\ &= [\![\mathbf{s}_{q,j}^\top \mathbf{A}]\!]_2 \widetilde{\mathbf{W}} + [\![\widetilde{\mathbf{L}}_{q,j}^\top]\!]_2 + (\mathbf{b}^\perp)^\top \otimes [\![\mathbf{s}_{q,j}^\top \mathbf{A}\widehat{\mathbf{W}} + \widehat{\mathbf{L}}_{q,j}^\top]\!]_2.\end{aligned}$$

To respond to the ciphertext challenge for one of g_0, g_1 with attribute $\mathbf{x}$, our $\mathcal{B}$ queries the security experiment of CP-1-ABE to obtain a secret key for $\mathbf{x}$, which is $\widehat{\mathbf{W}}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}$. It sets $\mathsf{ct}_1 = [\![\mathbf{c}]\!]_1$, $\mathsf{ct}_3 = g_b + [\![\boldsymbol{\mu}^\top \mathbf{c}]\!]_T$, and stitches ct_2 by

$$\begin{aligned}\mathbf{W}\left(\mathbf{c} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) &= (\widetilde{\mathbf{W}} + (\mathbf{b}^\perp)^\top \otimes \widehat{\mathbf{W}})\left(\mathbf{c} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) \\ &= \widetilde{\mathbf{W}}\left(\mathbf{c} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) + (\mathbf{b}^\perp)^\top \mathbf{c} \otimes \widehat{\mathbf{W}}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix} \\ \text{(since } \mathbf{c}^\top \mathbf{b}^\perp = 1) \quad &= \widetilde{\mathbf{W}}\left(\mathbf{c} \otimes \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}\right) + \widehat{\mathbf{W}}\begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}.\end{aligned}$$

Lastly, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Clearly $\mathcal{B}$ is efficient and only queries one secret key in the security experiments of CP-1-ABE. If $\mathcal{B}$ is in (the multi-ciphertext version of) $\mathsf{Exp}^0_{\text{1-ABE}}$ ($\delta_q = 0$ inside $\widehat{\mathbf{L}}_{q,j}$'s), then $\mathcal{A}$ is in H^b_1. Otherwise, $\mathcal{B}$ is in $\mathsf{Exp}^1_{\text{1-ABE}}$ ($\delta_q \xleftarrow{\$} \mathbb{Z}_p$ inside $\widehat{\mathbf{L}}_{q,j}$'s), then $\mathcal{A}$ is in H^b_2. Therefore, their advantages are the same. Since the CP-1-ABE is secure (Theorem 5.8), we have $\mathsf{H}^b_1 \approx \mathsf{H}^b_2$. □

Proof (Claim 5.12). Inspecting the appearances of $\boldsymbol{\mu}$ in H^0_2 and H^1_2, the master public key leaks $\boldsymbol{\mu}^\top \mathbf{B}^\top$, the secret keys leak $\{\boldsymbol{\mu} + \delta_q \mathbf{b}^\perp\}_q$, and the challenge ciphertext leaks $\boldsymbol{\mu}^\top \mathbf{c}$. Consider the change of variable $\boldsymbol{\mu} = \widetilde{\boldsymbol{\mu}} + \delta \mathbf{b}^\perp$ for $\widetilde{\boldsymbol{\mu}} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and $\delta \xleftarrow{\$} \mathbb{Z}_p$, then the master public key leaks $\widetilde{\boldsymbol{\mu}}^\top \mathbf{B}^\top$, the secret keys, $\{\widetilde{\boldsymbol{\mu}} + (\delta_q + \delta)\mathbf{b}^\perp\}_q$, and in the challenge ciphertext, δ appears as

$$g_b + [\![\boldsymbol{\mu}^\top \mathbf{c}]\!]_{\mathrm{T}} = (g_b + [\![\delta]\!]_{\mathrm{T}}) + [\![\widetilde{\boldsymbol{\mu}}^\top \mathbf{c}]\!]_{\mathrm{T}}.$$

Since δ is perfectly hidden elsewhere (by δ_q's), it perfectly hides g_b (hence b).

We conclude $\mathsf{H}^0_2 \equiv \mathsf{H}^1_2$. □

5.7 Ciphertext-Policy ABE for ABP

In this section, we show how to construct CP-ABE using exactly the same method for building KP-ABE. More specifically, in the first step, we simply regard our KP-ABE as a KP-1-ABE. In the second step, we convert KP-1-ABE to a CP-ABE using dual system encryption, similar to the transformation from CP-1-ABE to KP-ABE.

5.7.1 KP-1-ABE

We modify Construction 5.6 to obtain our KP-1-ABE,[49] based on which we construct our CP-ABE. In the modified $\mathsf{Setup}(\mathsf{gp}, 1^n)$,

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+2)}, \quad \mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}, \quad \mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times(k+1)(n+1)}, \quad \boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^{k+1},$$

$$\mathsf{xpk} = ([\![\mathbf{B}^\top]\!]_1, [\![\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1})]\!]_1), \qquad \mathsf{fpk} = ([\![\mathbf{A}]\!]_2, [\![\mathbf{AW}]\!]_2),$$

$$\mathsf{mpk} = ([\![\boldsymbol{\mu}^\top \mathbf{B}^\top]\!]_{\boxed{1}}, \mathsf{xpk}, \boxed{\mathsf{fpk}}), \qquad \mathsf{msk} = (\mathsf{fpk}, \boldsymbol{\mu}).$$

[49] The scheme in fact enjoys multi-key security, but we only need its 1-key security.

The algorithm $\mathsf{KeyGen}(\mathsf{msk}, f')$ remains unchanged. In the modified $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, g)$, the message g is now in $\mathbb{Z}_p$, and the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$ is generated with $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$ as

$$\mathsf{ct}_1 = [\![\mathbf{B}^\intercal]\!]_1 \mathbf{r}, \qquad \mathsf{ct}_2 = [\![\mathbf{W}(\mathbf{B}^\intercal \otimes \mathbf{I}_{n+1})]\!]_1 \left(\mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}\right), \qquad \mathsf{ct}_3 = \boxed{[\![g]\!]_1} + [\![\boldsymbol{\mu}^\intercal \mathbf{B}^\intercal]\!]_{\boxed{1}} \mathbf{r}.$$

In the modified $\mathsf{Dec}(\mathsf{mpk}, f', \mathsf{sk}, \mathbf{x}, \mathsf{ct})$, the message is recovered as $[\![g]\!]_T$.

We change the encoding of $\boldsymbol{\mu}^\intercal \mathbf{B}^\intercal$ and the last component of the ciphertext from $\mathbb{G}_T$ to $\mathbb{G}_1$ so that we do not "use up" the pairing at encryption time. We also include fpk in mpk, which will be useful for reductions in CP-ABE. It is straight-forward to adapt the security proof of KP-ABE (Theorem 5.9) for these changes.

5.7.2 CP-ABE

We again use dual system encryption to lift KP-1-ABE to CP-ABE. Recall that the transformation sets up the system by preparing $(k+1)$ instances of 1-ABE and $(k+1)$ random messages (encrypted under 1-ABE and used to generate encapsulated keys in ABE), and publishing a random projection of them. For example, during the conversion from CP-1-ABE to KP-ABE, $\mathbf{W}$ is enlarged from $(k+2) \times (n+1)$ to $(k+2) \times (k+1)(n+1)$ and a fresh $\boldsymbol{\mu}$ is sampled. In contrast, $\mathbf{A}$ is not enlarged and is shared among all the $(k+1)$ instances. More precisely, it is the master secret key being duplicated.

It turns out that *the* master secret key of KP-1-ABE is essentially the messages $\boldsymbol{\mu}$, not including fpk. This is confirmed by the inclusion of fpk in mpk. Therefore, in CP-ABE, preparing $(k+1)$ instances of KP-1-ABE just means $\boldsymbol{\mu} \in \mathbb{Z}_p^{k+1}$ is enlarged to $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(k+1)}$. Below, we call *the* set of messages $\boldsymbol{\nu}$ in CP-ABE.

Ingredients of Construction 5.7. Let

- $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS (Definitions 4.7 and 4.8) for $\mathcal{F}$,
- $\mathsf{GroupGen}$ a pairing group sampler (Definition 2.5) over which MDDH_k (Assumption 2.1) holds.

Construction 5.7 (CP-ABE). Define (see Definition 2.3)

$$\mathsf{Params}_{\mathsf{gp}} = \{\, 1^n \mid n \in \mathbb{N} \,\}, \qquad F_{\mathsf{gp},1^n} = \mathbb{Z}_p^n,$$
$$X_{\mathsf{gp},1^n} = \{\, f_{\neq 0}, f_{=0} \mid f \in \mathcal{F},\ f : \mathbb{Z}_p^n \to \mathbb{Z}_p \,\}, \qquad Y_{\mathsf{gp},1^n} = \mathbb{G}_{\mathrm{T}}, \qquad P_{\mathsf{gp},1^n}(\mathbf{x}, f') = f'(\mathbf{x}),$$

where p is the group order in gp. Our CP-ABE scheme based on GroupGen operates as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^n)$ takes as input the group description gp and the attribute length n. It samples and sets

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+2)}, \qquad \mathbf{B} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)}, \qquad \mathbf{D} \xleftarrow{\$} \mathbb{Z}_p^{k\times(k+1)},$$
$$\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times(k+1)(n+1)}, \qquad \mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(k+1)}, \qquad \boldsymbol{\nu} \xleftarrow{\$} \mathbb{Z}_p^{k+1},$$
$$\mathsf{xpk} = (\llbracket \mathbf{B}^\top \rrbracket_1, \llbracket \mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) \rrbracket_1), \qquad \mathsf{fpk} = (\llbracket \mathbf{A} \rrbracket_2, \llbracket \mathbf{AW} \rrbracket_2),$$
$$\mathsf{mpk} = (\llbracket \mathbf{D} \rrbracket_2, \llbracket \mathbf{DU} \rrbracket_2, \llbracket \mathbf{D}\boldsymbol{\nu} \rrbracket_{\mathrm{T}}, \mathsf{fpk}), \qquad \mathsf{msk} = (\llbracket \mathbf{UB}^\top \rrbracket_1, \mathsf{xpk}, \boldsymbol{\nu}).$$

  The algorithm outputs $(\mathsf{mpk}, \mathsf{msk})$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets

$$\mathsf{sk}_1 = \llbracket \mathbf{B}^\top \rrbracket_1 \mathbf{r}, \quad \mathsf{sk}_2 = \llbracket \mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1}) \rrbracket_1 \left(\mathbf{r} \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \right), \quad \mathsf{sk}_3 = \llbracket \boldsymbol{\nu} \rrbracket_1 + \llbracket \mathbf{UB}^\top \rrbracket_1 \mathbf{r}.$$

  It outputs $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3)$.

- $\mathsf{Enc}(\mathsf{mpk}, f', g)$ samples $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets

$$\llbracket \mathbf{u}^\top \rrbracket_2 = \mathbf{t}^\top \llbracket \mathbf{DU} \rrbracket_2, \qquad \mathsf{ct}_{0,1} = \mathbf{t}^\top \llbracket \mathbf{D} \rrbracket_2, \qquad \mathsf{ct}_{0,2} = g + \mathbf{t}^\top \llbracket \mathbf{D}\boldsymbol{\nu} \rrbracket_{\mathrm{T}}.$$

  It garbles f' with $\llbracket \mathbf{u} \rrbracket_2$ by

$$(\text{if } f' = f_{\neq 0}) \qquad \boldsymbol{\alpha} \leftarrow \mathbf{u}, \qquad \boldsymbol{\beta} \leftarrow \mathbf{0};$$
$$(\text{if } f' = f_{=0}) \qquad \boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_p^{k+1}, \qquad \boldsymbol{\beta} \leftarrow \mathbf{u};$$
$$\llbracket \mathbf{L}_1, \ldots, \mathbf{L}_m \rrbracket_2 \xleftarrow{\$} \mathsf{Garble}(f, \llbracket \boldsymbol{\alpha} \rrbracket_2, \llbracket \boldsymbol{\beta} \rrbracket_2).$$

  The algorithm then samples $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [m]$ and sets

$$\mathsf{ct}_{j,1} = \mathbf{s}_j^\top \llbracket \mathbf{A} \rrbracket_2, \qquad \mathsf{ct}_{j,2} = \mathbf{s}_j^\top \llbracket \mathbf{AW} \rrbracket_2 + \llbracket \mathbf{L}_j^\top \rrbracket_2.$$

  It outputs $\mathsf{ct} = (\mathsf{ct}_{0,1}, \mathsf{ct}_{0,2}, \mathsf{ct}_{1,1}, \mathsf{ct}_{1,2}, \ldots, \mathsf{ct}_{m,1}, \mathsf{ct}_{m,2})$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}, \mathsf{sk}, f', \mathsf{ct})$ first checks if $f'(\mathbf{x}) = 1$. If not, it outputs $\perp$ and terminates. Otherwise, the algorithm parses sk, ct as defined in $\mathsf{KeyGen}, \mathsf{Enc}$, computes

$$(\text{for } j \in [m]) \qquad [\![\ell_j]\!]_T \leftarrow -\mathsf{ct}_{j,1}\mathsf{sk}_2 + \mathsf{ct}_{j,2}\left(\mathsf{sk}_1 \otimes \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}\right);$$

$$[\![\mu']\!]_T \leftarrow \begin{cases} \frac{1}{f'(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1, \ldots, \ell_m]\!]_T), & \text{if } f' = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1, \ldots, \ell_m]\!]_T), & \text{if } f' = f_{=0}; \end{cases}$$

  and outputs $(\mathsf{ct}_{0,2} + [\![\mu']\!]_T - \mathsf{ct}_{0,1}\mathsf{sk}_3)$.

Correctness and Efficiency. Note that $\mathsf{sk}_1, \mathsf{sk}_2$ of this scheme correspond to $\mathsf{ct}_1, \mathsf{ct}_2$ in Construction 5.6 (KP-ABE) and $\mathsf{ct}_{j,1}, \mathsf{ct}_{j,2}$ correspond to $\mathsf{sk}_{j,1}, \mathsf{sk}_{j,2}$ for $j \in [m]$, except that $\mathsf{ct}_{j,1}, \mathsf{ct}_{j,2}$ are generated with the master secret $\boldsymbol{\mu}$ in KP-ABE replaced by $\mathbf{U}^\top\mathbf{D}^\top\mathbf{t}$. By the same argument for KP-ABE correctness, we have $\mu' = \mathbf{r}^\top\mathbf{B}\mathbf{U}^\top\mathbf{D}^\top\mathbf{t}$ if decryption is authorized. Therefore,

$$\mathsf{ct}_{0,2} + [\![\mu']\!]_T - \mathsf{ct}_{0,1}\mathsf{sk}_3 = g + [\![\mathbf{t}^\top\mathbf{D}\boldsymbol{\nu}]\!]_T + [\![\mathbf{r}^\top\mathbf{B}\mathbf{U}^\top\mathbf{D}^\top\mathbf{t}]\!]_T - [\![\mathbf{t}^\top\mathbf{D}]\!]_2[\![\boldsymbol{\nu} + \mathbf{U}\mathbf{B}^\top\mathbf{r}]\!]_1 = g,$$

i.e., the scheme is correct. Similarly to our KP-ABE, the number of pairing operations in the decryption algorithm can be reduced to $(3k + 4)$.

Security. We state and prove the security of our CP-ABE.

Theorem 5.13 (¶). *Suppose in Construction 5.7, the AKGS is* piecewise secure *(Definition 4.10) and* MDDH_k *(Assumption 2.1) holds over* $\mathsf{GroupGen}$*, then the resultant ABE scheme is* adaptively secure *(Definition 2.2).*

Similarly to the proof for KP-ABE (Theorem 5.9), we first switch $\mathbf{t}^\top\mathbf{D}$ in the challenge ciphertext to random $\mathbf{c}^\top$. This corresponds to a fresh $\mathbf{u}^\top = \mathbf{c}^\top\mathbf{U}$ used as the $\boldsymbol{\mu}^\top$ in KP-1-ABE and a fresh encapsulated key $\mathbf{c}^\top\boldsymbol{\nu}$ encrypted under this KP-1-ABE instance. By KP-1-ABE security, $\mathbf{c}^\top\boldsymbol{\nu}$ is hidden, which in turn hides the message in CP-ABE.

Proof (Theorem 5.13). We go through hybrids $\mathsf{H}_1^b, \mathsf{H}_2^b$.

- H_1^b proceeds identically to $\mathsf{Exp}^b_{\mathrm{PHFE}}$, except that when generating the challenge ciphertext, we set

$$[\![\mathbf{u}^\top]\!]_2 = [\![\boxed{\mathbf{c}^\top}\mathbf{U}]\!]_2, \qquad \mathsf{ct}_{0,1} = [\![\boxed{\mathbf{c}^\top}]\!]_2, \qquad \mathsf{ct}_{0,2} = g_b + [\![\boxed{\mathbf{c}^\top}\boldsymbol{\nu}]\!]_T,$$

where $\mathbf{c}$ is uniformly random outside the row space of $\mathbf{D}$ and the rest of the procedure to generate the ciphertext proceeds normally.

- H_2^b proceeds identically to H_1^b, except that in each queried key,

$$\mathsf{sk}_3 = [\![\boldsymbol{v} + \boxed{\delta_q}\, \mathbf{d}^\perp + \mathbf{U}\mathbf{B}^\intercal \mathbf{r}_q]\!]_1.$$

  Here, $\mathbf{d}^\perp$ is a fixed vector such that $\mathbf{D}\mathbf{d}^\perp = \mathbf{0}$ and $\mathbf{c}^\intercal \mathbf{d}^\perp = 1$, and $\delta_q \xleftarrow{\$} \mathbb{Z}_p$ is fresh per key.

We have the following claims.

Claim 5.14 (¶). $\mathsf{Exp}^b_{\mathrm{PHFE}} \approx \mathsf{H}_1^b$ *reduces to* MDDH_k *(Assumption 2.1) over* $\mathbb{G}_2$.

Claim 5.15 (¶). $\mathsf{H}_1^b \approx \mathsf{H}_2^b$ *reduces to the security of KP-1-ABE (Section 5.7.1), which relies on the piecewise security of AKGS and* MDDH_k *over* $\mathsf{GroupGen}$ *(Theorem 5.9).*

Claim 5.16 (¶). $\mathsf{H}_2^0 \equiv \mathsf{H}_2^1$.

Once we prove them, we have $\mathsf{Exp}^0_{\mathrm{PHFE}} \approx \mathsf{Exp}^1_{\mathrm{PHFE}}$ by hybrid argument. □

Proof (Claim 5.14). This is analogous to the proof of Claim 5.10. Let $\mathcal{A}$ be an efficient adversary distinguishing $\mathsf{Exp}^b_{\mathrm{PHFE}}$ and H_1^b, we construct $\mathcal{B}$ against MDDH_k over $\mathbb{G}_2$. Upon receiving MDDH_k challenge $[\![\mathbf{D}, \mathbf{c}^\intercal]\!]_2$ where either $\mathbf{c}^\intercal = \mathbf{t}^\intercal \mathbf{D}$ or $\mathbf{c}^\intercal$ is random, our $\mathcal{B}$ launches $\mathcal{A}()$, receives the attribute length 1^n from it, and sets up the system using $[\![\mathbf{D}]\!]_2$, sampling and computing everything else as required. It sends mpk to $\mathcal{A}$ and responds to the key queries normally. To respond to the ciphertext challenge for one of g_0, g_1 with policy f', our $\mathcal{B}$ sets

$$[\![\mathbf{u}^\intercal]\!]_2 = [\![\mathbf{c}^\intercal]\!]_2 \mathbf{U}, \qquad \mathsf{ct}_{0,1} = [\![\mathbf{c}^\intercal]\!]_2, \qquad \mathsf{ct}_{0,2} = g_b + [\![\mathbf{c}^\intercal]\!]_2 [\![\boldsymbol{v}]\!]_1,$$

and continues the encryption procedure from there. Lastly, it outputs whatever $\mathcal{A}$ outputs.

Clearly $\mathcal{B}$ is efficient. If $\mathbf{c}^\intercal = \mathbf{t}^\intercal \mathbf{D}$, our $\mathcal{B}$ has simulated $\mathsf{Exp}^b_{\mathrm{PHFE}}$ for $\mathcal{A}$. Otherwise, $\mathbf{c}^\intercal$ is random, and $\mathcal{B}$ has (statistically closely) simulated H_1^b for $\mathcal{A}$. Therefore, their advantages are negligibly close.

If MDDH_k holds over $\mathbb{G}_2$, we have $\mathsf{Exp}^b_{\mathrm{PHFE}} \approx \mathsf{H}_1^b$. □

Proof (Claim 5.15). This is analogous to the proof of Claim 5.11. Let $\mathbf{d}^\perp$ be such that $\mathbf{D}\mathbf{d}^\perp = \mathbf{0}$ and $\mathbf{c}^\top\mathbf{d}^\perp = 1$ for $\mathbf{D}$ in the master public key and $\mathbf{c}^\top$ in the challenge ciphertext. We set

$$\mathbf{U} = \widetilde{\mathbf{U}} + \mathbf{d}^\perp\boldsymbol{\mu}^\top, \qquad \widetilde{\mathbf{U}} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times(k+1)}, \qquad \boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^{k+1}, \qquad \boldsymbol{\nu}_q = \boldsymbol{\nu} + \delta_q\mathbf{d}^\perp,$$

where $\boldsymbol{\mu}$ comes from KP-1-ABE, and $\delta_q = 0$ or $\delta_q \xleftarrow{\$} \mathbb{Z}_p$ for each key queried.

Let $\mathcal{A}$ be an efficient adversary distinguishing H_1^b and H_2^b, we construct $\mathcal{B}$ against the security of KP-1-ABE (Section 5.7.1 and Theorem 5.9). The adversary $\mathcal{B}$ launches $\mathcal{A}()$, receives the attribute length 1^n from it, and forwards 1^n to the security experiment of KP-1-ABE. It receives $\widehat{\mathsf{mpk}} = (\llbracket\boldsymbol{\mu}^\top\mathbf{B}^\top\rrbracket_1, \mathsf{xpk}, \mathsf{fpk})$, where

$$\mathsf{xpk} = (\llbracket\mathbf{B}\rrbracket_1, \llbracket\mathbf{W}(\mathbf{B}^\top \otimes \mathbf{I}_{n+1})\rrbracket_1), \qquad \mathsf{fpk} = (\llbracket\mathbf{A}\rrbracket_2, \llbracket\mathbf{A}\mathbf{W}\rrbracket_2).$$

The adversary $\mathcal{B}$ samples $\mathbf{D}, \mathbf{c}, \widetilde{\mathbf{U}}, \boldsymbol{\nu}$ and computes $\mathbf{d}^\perp$. Note that

$$\mathbf{D}\mathbf{U} = \mathbf{D}\widetilde{\mathbf{U}} + \mathbf{D}\mathbf{d}^\perp\boldsymbol{\mu}^\top = \mathbf{D}\widetilde{\mathbf{U}},$$

so $\mathcal{B}$ sets $\mathsf{mpk} = (\llbracket\mathbf{D}\rrbracket_2, \llbracket\mathbf{D}\widetilde{\mathbf{U}}\rrbracket_2, \llbracket\mathbf{D}\boldsymbol{\nu}\rrbracket_T, \mathsf{fpk})$, sends it to $\mathcal{A}$, and starts to respond to its queries and challenges. To respond to a key query for $\mathbf{x}_q$, our $\mathcal{B}$ queries the security experiment of KP-1-ABE for a ciphertext

$$\widehat{\mathsf{ct}}_{q,1} = \llbracket\mathbf{B}^\top\mathbf{r}_q\rrbracket_1, \qquad \widehat{\mathsf{ct}}_{q,2} = \left\llbracket\mathbf{W}\left(\mathbf{B}^\top\mathbf{r}_q \otimes \begin{pmatrix}1\\ \mathbf{x}_q\end{pmatrix}\right)\right\rrbracket_1, \qquad \widehat{\mathsf{ct}}_{q,3} = \llbracket\boldsymbol{\mu}^\top\mathbf{B}^\top\mathbf{r}_q + \delta_q\rrbracket_1.$$

It sets $\mathsf{sk}_{q,1} = \widehat{\mathsf{ct}}_{q,1}$, $\mathsf{sk}_{q,2} = \widehat{\mathsf{ct}}_{q,2}$, and stitches $\mathsf{sk}_{q,3}$ by

$$\begin{aligned}\mathsf{sk}_{q,3} = \llbracket\boldsymbol{\nu}_q + \mathbf{U}\mathbf{B}^\top\mathbf{r}_q\rrbracket_1 &= \llbracket\boldsymbol{\nu} + \delta_q\mathbf{d}^\perp + \widetilde{\mathbf{U}}\mathbf{B}^\top\mathbf{r}_q + \mathbf{d}^\perp\boldsymbol{\mu}^\top\mathbf{B}^\top\mathbf{r}_q\rrbracket_1\\ &= \llbracket\boldsymbol{\nu}\rrbracket_1 + \widetilde{\mathbf{U}}\llbracket\mathbf{B}^\top\mathbf{r}_q\rrbracket_1 + \mathbf{d}^\perp\llbracket\boldsymbol{\mu}^\top\mathbf{B}^\top\mathbf{r}_q + \delta_q\rrbracket_1.\end{aligned}$$

To respond to the ciphertext challenge for one of g_0, g_1 with policy f', our $\mathcal{B}$ queries the security experiment of KP-1-ABE for a secret key for f', receiving

$$\widehat{\mathsf{sk}}_{j,1} = \llbracket\mathbf{s}_j^\top\mathbf{A}\rrbracket_2, \qquad \widehat{\mathsf{sk}}_{j,2} = \llbracket\mathbf{s}_j^\top\mathbf{A}\mathbf{W} + \widehat{\mathbf{L}}_j^\top\rrbracket_2,$$

where f' is garbled with $\boldsymbol{\mu}$ to get $\widehat{\mathbf{L}}_1, \ldots, \widehat{\mathbf{L}}_m$. Note that

$$\mathbf{u}^\top = \mathbf{c}^\top\mathbf{U} = \mathbf{c}^\top\widetilde{\mathbf{U}} + \mathbf{c}^\top\mathbf{d}^\perp\boldsymbol{\mu}^\top = \mathbf{c}^\top\widetilde{\mathbf{U}} + \boldsymbol{\mu}^\top,$$

so $\mathcal{B}$ can garble f' with $\mathbf{c}^\intercal\widetilde{\mathbf{U}}$ to obtain $\widetilde{\mathbf{L}}_1, \ldots, \widetilde{\mathbf{L}}_m$ and stitch the response as

$$\mathsf{ct}_{0,1} = [\![\mathbf{c}^\intercal]\!]_2,\ \mathsf{ct}_{0,2} = g_b + [\![\mathbf{c}^\intercal\boldsymbol{\nu}]\!]_T, \qquad (j \in [m])\quad \mathsf{ct}_{j,1} = \widehat{\mathsf{sk}}_{j,1},\ \mathsf{ct}_{j,2} = \widehat{\mathsf{sk}}_{j,2} + [\![\widetilde{\mathbf{L}}_j^\intercal]\!]_2.$$

The linearity of Garble guarantees that the result is well-distributed. Lastly, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Clearly $\mathcal{B}$ is efficient and it makes at most one secret key query in the security experiments of KP-1-ABE. If $\mathcal{B}$ is in $\mathsf{Exp}^0_{\text{1-ABE}}$, it has simulated H^b_1 for $\mathcal{A}$. Otherwise, $\mathcal{B}$ is in $\mathsf{Exp}^1_{\text{1-ABE}}$ and has simulated H^b_2 for $\mathcal{A}$. Therefore, the two have the same advantage.

By KP-1-ABE security, $\mathsf{H}^b_1 \approx \mathsf{H}^b_2$. □

Proof (Claim 5.16). This is analogous to the proof of Claim 5.12. Observe that $\boldsymbol{\nu}$ appears in mpk as $\mathbf{D}\boldsymbol{\nu}$, in sk's as $\{\boldsymbol{\nu} + \delta_q\mathbf{d}^\perp\}_q$, and in the challenge ciphertext as $\mathbf{c}^\intercal\boldsymbol{\nu}$. Consider the change of variable $\boldsymbol{\nu} = \widetilde{\boldsymbol{\nu}} + \delta\mathbf{d}^\perp$, then in mpk we only use $\mathbf{D}\widetilde{\boldsymbol{\nu}}$, in sk's we only use $\{\widetilde{\boldsymbol{\nu}} + (\delta_q + \delta)\mathbf{d}^\perp\}_q$, and in the challenge ciphertext the message appears as

$$g_b + [\![\mathbf{c}^\intercal\boldsymbol{\nu}]\!]_T = (g_b + [\![\delta]\!]_T) + [\![\mathbf{c}^\intercal\widetilde{\boldsymbol{\nu}}]\!]_T,$$

which means g_b (thus b) is perfectly hidden. We conclude $\mathsf{H}^0_2 \equiv \mathsf{H}^1_2$. □

5.8 Further Discussions

Here are some further comments about the research around [LL20c,LL20d].

Generic Compiler. Theorem 5.2 shows that selective IND-CPA and gradual simulation security are not existentially separated. Its proof is in [LL20d]. The generically lifted scheme from [ABDP15] can also be used to construct CP-1-ABE, KP-ABE, and CP-ABE. Other than lesser efficiency, that route also leads to non-ergonomic reductions when performing dual system encryption.

Opening Up Abstraction. One might ask whether this chapter truly is within the general paradigm, as we use dual system encryption for lifting 1-ABE to ABE. Instead of answering this question, I cast the research of this chapter as an ablation study of the framework.

What purpose does IPFE serve in each phase of constructing an ABE scheme? In Chapter 4, IPFE was regarded as an all-in-one tool for obtaining ABE from linear garbling. The current chapter provides a refined understanding. In the case of 1-ABE for ABP,[50] IPFE provides *i)* the secure evaluation of garbling, allowing the choices of input and function to be separate while enabling the computation when components are put together. When converting from 1-ABE to ABE, two things are in effect — one must *ii)* rerandomize the garbling properly and *iii)* compute the rerandomized labels securely. In Chapter 4, the three items are achieved by function-hiding, DDH, function-hiding. In the current chapter, they are achieved by simulation security, DDH, "nothing". From this we deduce that function-hiding (hiding *both* key and plaintext vectors) is not essential to the paradigm, as *i)* needs one-side security (hiding just the plaintext vectors) and *iii)* does not require any security from IPFE.[51]

I came to know (the name of) dual system encryption method at the beginning of the research of Chapter 4, but could not construct schemes using it at that time — it was way too complex for me at that time. Some time after completing the research of Chapter 4, a lightbulb suddenly clicked in my head and I started to understand dual system encryption (the version using MDDH in prime-order pairing groups) through the lens of IPFE — I regarded it as a method to rerandomize without the power of IPFE. Moreover, the version using MDDH boils down to projective hash functions (hash proof systems).

A common wisdom in computer science is to understand one level below one's day-to-day abstraction. Chapters 4 and 5 show an example of such. The abstraction of IPFE has taught us one way to construct ABE. Opening it up has enabled us to analyze its purposes. Replacing parts of it by other well-established methods has made the ABE scheme much more efficient. Although we eventually arrive at a scheme reachable purely using dual system encryption, it is the abstraction of IPFE that paved its way. Furthermore, we still use IPFE in the step of constructing 1-ABE to simplify

[50] The case of uniform computation is more complex — even for 1-ABE, the computed garbling is only computationally secure.

[51] In the current chapter, it relies on the structure of the keys of the specific 1-ABE schemes used.

its presentation.

To make my case for *learning through abstraction, then opening it up for improvement*, I present the next two points.

Improved Exposition of Dual System Encryption for CP-ABE. In many dual-system CP-ABE, the proof considers more than two forms (normal and semi-functional) of keys. For example, the CP-ABE in [KW20] uses normal, pseudo-normal, pseudo-semi-functional, semi-functional forms. Their reason[52] of having to consider four forms is that their CP-ABE security proof is by reduction to their core 1-ABE (corresponding to our CP-1-ABE), and the transition with pseudo forms essentially redoes a KP-ABE proof.

The proof of our CP-ABE (Construction 5.7) does *not explicitly* use four forms of keys, because we are reducing to KP-(1-)ABE security. Let P/N/SF mean pseudo, normal, semi-functional. If we open up the proof of the reduced-to KP-ABE security inside the CP-ABE proof, then in Construction 5.7 and its proofs:

- an N secret key uses $\mathsf{sk}_1 = [\![\mathbf{B}^\intercal \mathbf{r}]\!]_1$ and puts $\boldsymbol{\nu}$ in sk_3;
- a PN secret key uses truly random sk_1 and puts $\boldsymbol{\nu}$ in sk_3;
- a PSF secret key uses truly random sk_1 and puts $(\boldsymbol{\nu} + \delta \mathbf{d}^\perp)$ in sk_3;
- an SF secret key uses $\mathsf{sk}_1 = [\![\mathbf{B}^\intercal \mathbf{r}]\!]_1$ and puts $(\boldsymbol{\nu} + \delta \mathbf{d}^\perp)$ in sk_3.

The transitions N $\to$ PN and PSF $\to$ SF are (forward and backward applications of) Claim 5.10, and that from PN to PSF is Claims 5.11 and 5.12 — all of those claims are part of our KP-ABE security proof. We can fully abstract them away, so our CP-ABE proof goes directly from N to SF in Claim 5.15. I regard this as an improved exposition for dual system encryption.

[52] There are other reasons, such as using bilinear entropy expansion for unbounded ABE. The point here is that pseudo forms can be encapsulated during the proof of CP-ABE. In contrast, the entropy-expanded form necessarily appears in an ABE security proof.

Efficiency Comparison with Certain Dual-System CP-ABE. There is a curious efficiency difference between our CP-ABE (Construction 5.7) and the CP-ABE in [KW20].[53] Specifically, we are interested in how size-efficient a CP-ABE scheme is compared to its corresponding KP-ABE scheme.

When relying on MDDH_k, the shapes of the public matrices are often $k \times (k+r)$ for various $r \geq 0$. Roughly speaking, r is the amount of programming space available during the security proof. Typical values of r are $0, 1, k, 2k$. A larger r means larger components. Therefore, we are interested in the change of r from KP-ABE to CP-ABE.

Our CP-1-ABE / KP-ABE / CP-ABE (Constructions 5.5, 5.6, and 5.7) have redundancy 2 / $2, 1$ / $2, 1, 1$ in $\mathbf{A}$ / $\mathbf{A}, \mathbf{B}$ / $\mathbf{A}, \mathbf{B}, \mathbf{D}$. Here, 2 in $\mathbf{A}$ is for simulating two labels at one time (embedding two instances of OTP-IPFE in CP-1-ABE), 1 in $\mathbf{B}$ for embedding one instance of CP-1-ABE in KP-ABE, and 1 in $\mathbf{D}$ for embedding one instance of KP-(1-)ABE ($\boldsymbol{\mu}$) in CP-ABE ($\mathbf{U}$). Compared to KP-ABE, our CP-ABE has one additional public matrix with redundancy 1. In particular, the size dependency on k of a key in our CP-ABE is *the same as* that of a ciphertext in our KP-ABE.

In [KW19,KW20], their 1-ABE has redundancy 0, because its security is proven by guessing reductions, and no embedding of smaller instances is needed. Their KP-ABE has redundancy 1 in their $\mathbf{A}$ (corresponding to our $\mathbf{B}$), for embedding one instance of 1-ABE. Their CP-ABE, however, has redundancy $1, k$ in their $\mathbf{B}, \mathbf{A}$. The $\mathbf{B}$ in their CP-ABE corresponds to our $\mathbf{B}$. The $\mathbf{A}$ in their CP-ABE plays the roles of our $\mathbf{A}, \mathbf{D}$. It appears to me that the direct reduction to 1-ABE, whose secret is k-dimensional (see $\mathbf{w}$ in [KW19; Section 5.4]), necessitated the large redundancy k in their construction. Regardless of the reason, the size dependency on k of a key in their CP-ABE is *twice* that of a ciphertext in their KP-ABE.

A quick survey into other dual-system ABE in prime-order pairings reveals that in many works, the redundancy increases by $(k-1)$ from KP-ABE to CP-ABE. This happens for ABE for branching programs in [GW20b; Appendices H and J] (from 1 to k) and ABE for monotone span programs in [CGKW18b; Sections 7 and 8] (from

[53] Results-wise, the closest would be Section 6.9 of [AT20], but the constants inside the latter is not easily identifiable.

$(k+1)$ to $3k$). The notable exception is [CGW15], where KP-/CP-ABE schemes have the same level of redundancy. This last work is also by direct reduction to the core 1-ABE (predicate encodings), but its core 1-ABE secret is 1-dimensional, which might be the reason why the redundancy did not get bumped up.

Since I have not talked to the authors on this question, it is unclear to me why the over-increased redundancy. In any case, it should be possible to shave a factor of 2 from the secret key size of the CP-ABE scheme of [KW20] by shrinking its $\mathbf{A}$ to $k \times (k+1)$ and using a separate public matrix for projecting its $\mathbf{U}_0$.[54] Note that this is about reducing blow-up in CP-ABE compared to KP-ABE, which is orthogonal to improving ABE to succinct.

[54]This would correspond to forming their core 1-ABE using a public-key encryption, which becomes exactly a CP-1-ABE. The modified version of their CP-ABE is the same as going from their CP-1-ABE to KP-ABE to CP-ABE as done in this chapter.

Chapter 6

Succinct KP-ABE for Circuits and Doubly Succinct CP-ABE for Formulae

(Pairing-Based IPFE, Noisy Linear Garbling)

This chapter is an overhaul of [LLL22b,LLL22a].[55]

In this chapter, we show how to compose pairing-based IPFE and lattice-based garbling to obtain ABE with new efficiency characteristics. We obtain KP-ABE for circuits with *constant-size* keys, independent of the circuit depth, and CP-ABE for formulae with *double succinctness*. In addition, we prove adaptive security under suitable assumptions, which was unknown for lattice-based ABE except for limited classes of policies.

6.1 Introduction

In Chapter 5, we have constructed ABE whose $\mathbf{x}$-part is succinct. There are also known schemes with succinct f-part. It is natural to ask for succinctness on both parts. How close can we get to the *ideal efficiency* of having both constant-size keys and ciphertexts? Despite tremendous effort, the state-of-the-art is still far from ideal. Current ABE schemes with either succinct keys or succinct ciphertexts can be broadly classified as follows (see Figures 6.1 and 6.2).

- The work of [BGG+14], based on LWE, built KP-ABE for polynomial-size circuits with succinct keys $|\mathsf{sk}_f| = \mathrm{poly}(d)$ and ciphertexts of size $|\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(d)$, where d is the maximum depth of the circuit.[56]

[55] Hanjun Li, Huijia Lin, Ji Luo: *ABE for Circuits with Constant-Size Secret Keys and Adaptive Security*, DOI 10.1007/978-3-031-22318-1_24.

[56] Polynomial factors in the security parameter is ignored in this overview.

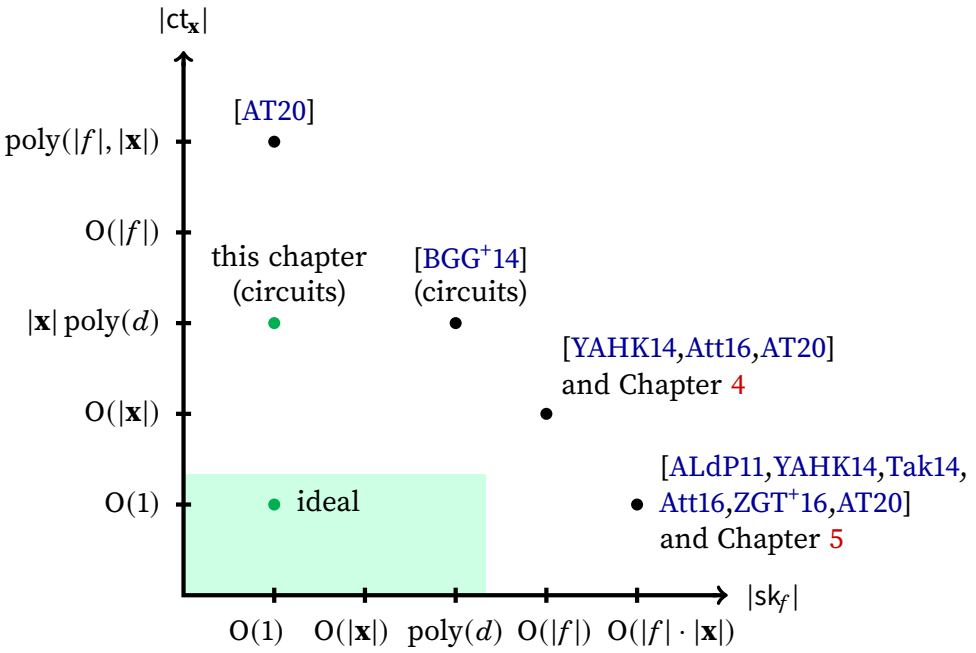


Figure 6.1. Efficiency comparison among KP-ABE schemes.
The shaded region highlights succinctness for $|\mathsf{ct}_{\mathbf{x}}|$ and $|\mathsf{sk}_f|$.
This chapter and [BGG+14] are KP-ABE schemes for circuits.
The rest of the schemes are for low-depth computation.

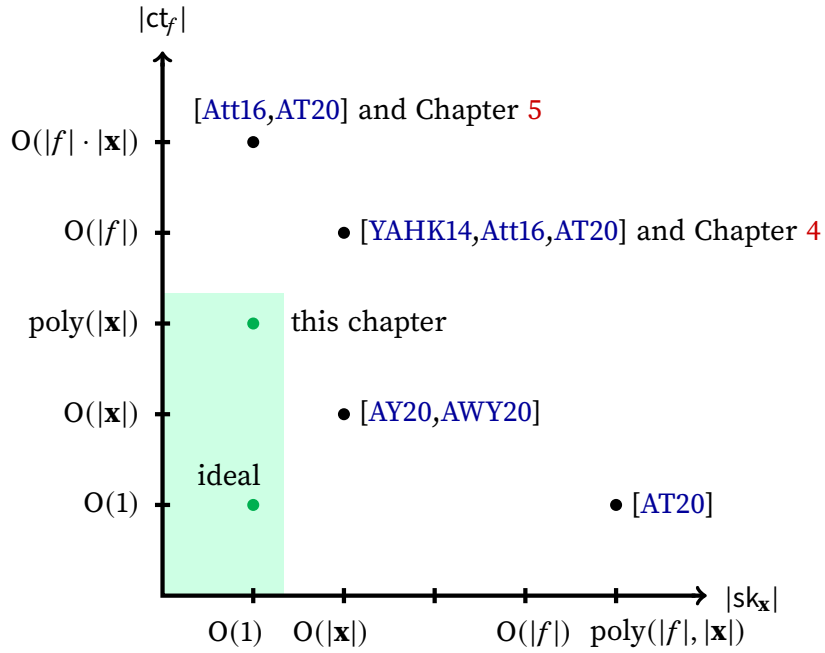


Figure 6.2. Efficiency comparison among CP-ABE schemes.
The shaded region highlights succinctness for $|\mathsf{ct}_f|$ and $|\mathsf{sk}_{\mathbf{x}}|$.
All the schemes shown here are for low-depth computation.

- Several works ([ALdP11,YAHK14,Tak14,Att16,ZGT+16,AT20] and Chapter 5), based on pairing, constructed KP-ABE and CP-ABE for low-depth computations with *either* constant-size secret keys *or* constant-size ciphertexts, i.e., *either* $|\mathsf{sk}| = O(1)$ *or* $|\mathsf{ct}| = O(1)$, at the cost of the other component being much larger, of size $\Omega(|f| \cdot |\mathbf{x}|)$.

- The recent works of [AY20,AWY20] constructed CP-ABE for Boolean formulae with succinct ciphertexts $|\mathsf{ct}_f| = \Theta(|\mathbf{x}|)$ and compact keys $|\mathsf{sk}_{\mathbf{x}}| = \Theta(|\mathbf{x}|)$. These schemes are based on LWE and strong assumptions on pairing groups — either the generic (pairing) group model [AY20] or knowledge assumptions [AWY20].

In this chapter, we set out to improve the state-of-the-art towards the direction of ideal efficiency. We observe that though there are ABE schemes for low-depth computations with constant-size keys, we do not have such ABE for general circuits. We ask:

Can we construct ABE for circuits with constant-size keys?

Furthermore, all of the aforementioned schemes have either succinct keys or succinct ciphertexts, but never *both at the same time*. If we were to eventually achieve ideal efficiency, we would have to first overcome the intermediate barrier of simultaneously having succinct keys and ciphertexts — we refer to this as *double succinctness*. We thus ask:

Can we construct ABE for expressive policies with
both *succinct keys* and *succinct ciphertexts?*

We note that the questions above are unanswered even when assuming the strong primitive of indistinguishability obfuscation ($i\mathcal{O}$). Several works [GGH+13a,GGSW13, KNTY19] constructed ABE for circuits (or even functional encryption for circuits) using indistinguishability obfuscation or related primitives. However, they all have large secret keys of size $\mathrm{poly}(|f|)$. The only work [GKP+13a] that manages to obtain ABE for RAM with constant-size keys relies on a strong primitive called extractable witness encryption, which however lacks provably secure instantiation.

Our Results. We address both questions. For the former, we construct the first KP-ABE scheme for circuits with constant-size keys while keeping the ciphertext size the same as in [BGG+14]. Concretely, each secret key consists of only 3 group elements. For the latter, we present the first CP-ABE scheme for Boolean formulae achieving double succinctness — it has constant-size keys and succinct ciphertexts. Concretely, each secret key consists of only 2 group elements. Both constructions rely on LWE and the generic (pairing) group model, similar to [AY20].

Construction 6.2 (KP-ABE). *Assuming LWE, in the generic (pairing) group model, there is a KP-ABE for circuits that achieves selective security and has key size* $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$ *(concretely, containing* 3 *group elements) and ciphertext size* $|\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(\lambda, d)$*, where* d *is the maximum depth of the policy circuits.*

Construction 6.5 (CP-ABE). *Assuming LWE, in the generic (pairing) group model, there is a CP-ABE for Boolean formulae that achieves very selective security and has constant-size keys* $|\mathsf{sk}_{\mathbf{x}}| = \mathrm{poly}(\lambda)$ *(concretely, containing* 2 *group elements) and ciphertexts of size* $|\mathsf{ct}_f| = |\mathbf{x}|^2\,\mathrm{poly}(\lambda)$ *independent of the formula size* $|f|$*.*

Additional Contribution — Adaptive Security. The standard security property of ABE is collusion resistance, which stipulates that no information of the message μ encrypted in a ciphertext should be revealed even when multiple secret keys are issued, as long as none of the keys alone is authorized to decrypt the ciphertext. Adaptive security requires collusion resistance to hold even when attributes and policies tied to the challenge ciphertext and the secret keys are chosen adaptively by the adversary. The weaker selective security restricts the adversary to commit to the attribute (in KP-ABE) or the policy (in CP-ABE) associated with the challenge ciphertext before seeing any parameters of the system, and very selective security further requires all attributes and policies in both the challenge ciphertext and the secret keys to be chosen statically.

Adaptive security guards against more powerful adversaries than selective security. It is known that the latter implies the former at the cost of subexponential hardness assumptions via complexity leveraging. However, complexity leveraging is undesirable not only because it requires subexponential hardness, but also because it requires

scaling the security parameter to be polynomial in the length of the information to be guessed, $\lambda = \Omega(\text{poly}(|x|))$ in KP-ABE or $\lambda = \Omega(\text{poly}(|f|))$ in CP-ABE. Consequently, complexity leveraging is not a viable solution when aiming for constant-size keys, as key size $\text{poly}(\lambda)$ would already depend on $|x|$ or $|f|$.

Instead, we show that in our constructions of KP- and CP-ABE, if assume adaptive LWE instead of plain LWE, then they achieve *adaptive security* and our reduction only incurs a polynomial amount of security loss. The adaptive LWE assumption [QWW18] postulates that LWE samples of the form $\{\mathbf{s}^\intercal(\mathbf{A}_i - \mathbf{x}[i]\mathbf{G}) + \mathbf{e}_i^\intercal\}_i$ are pseudorandom, even if the adversary adaptively chooses $\mathbf{x}$ depending on the random matrices $\{\mathbf{A}_i\}_i$.

Theorem (adaptive security). *Assuming the* polynomial hardness *of adaptive LWE (instead of LWE), in the generic (pairing) group model, Constructions 6.2 and 6.5 (KP-ABE and CP-ABE) are adaptively secure.*

In the literature, the ABE schemes for circuits based on lattices [GVW12,BGG+14] achieve only selective security (without complexity leveraging). Adapting it to have adaptive security has remained a technical barrier, except for very limited classes of policies such as 3-CNF [Tsa19]. Alternatively, there are schemes based on indistinguishability obfuscation or functional encryption for all circuits that are adaptively secure [Wat15,KNTY19], but requiring stronger assumptions. Our technique can be viewed as making the lattice-based schemes adaptively secure when combined with pairing. Note that this is non-trivial. For instance, the recent CP-ABE schemes in [AY20,AWY20] that combine [BGG+14] with pairing groups inherit the selective security of the former (even when assuming adaptive LWE).

For follow-up discussion on adaptive LWE, see Section 6.8.

6.2 Technical Overview

High-Level Ideas. Let's focus on our KP-ABE scheme for circuits first. The first known construction of KP-ABE for circuits from LWE [GVW13] has keys of size $|f|\,\text{poly}(\lambda, d)$. The scheme of [BGG+14] reduces the key size to $\text{poly}(\lambda, d)$. Both schemes achieve only selective security because they rely on the *lattice trapdoor simulation techniques*.

Consider the BGG$^+$ scheme. Its ciphertext encodes the attribute $\mathbf{x}$ and the message μ as follows:

$$\mathbf{s}^\intercal \mathbf{A} + (\mathbf{e}')^\intercal, \qquad \mathbf{s}^\intercal \big(\overbrace{(\mathbf{A}_1, \ldots, \mathbf{A}_\ell)}^{\mathbf{B}} - \mathbf{x} \otimes \mathbf{G}\big) + \mathbf{e}^\intercal, \qquad \mathbf{s}^\intercal \mathbf{z} + e + \mu \lfloor q/2 \rceil.$$

One can homomorphically evaluate any circuit f on the attribute encoding to obtain $(\mathbf{s}^\intercal(\mathbf{B}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_f^\intercal)$. To decrypt, the secret key sk_f simply is a *short* vector $\mathbf{r}_f$ satisfying $(\mathbf{A}, \mathbf{B}_f)\mathbf{r}_f = \mathbf{z}$, which can be sampled using a trapdoor of $\mathbf{A}$. This approach however has two drawbacks.

- Difficulty Towards Constant-Size Keys. The short vector $\mathbf{r}_f$ contained in the secret key sk_f has size $\mathrm{poly}(\lambda, d)$. This is because it has dimension $m = n\lceil \log_2 q \rceil$ for $\log_2 q = \mathrm{poly}(\lambda, d)$ and entries of magnitude exponential in d.
- Difficulty Towards Adaptive Security. The security proof relies on the ability to simulate trapdoors for matrices $(\mathbf{A}, \mathbf{B}_f)$ corresponding to the secret keys *unauthorized* to decrypt the challenge ciphertext with attribute $\mathbf{x}$, i.e., $f(\mathbf{x}) = 1$. However, to do so, the current technique plants $\mathbf{x}$ in the public matrices $\mathbf{A}_i$'s (in mpk), leading to selective security. Note that even with the stronger adaptive LWE assumption, it is unclear how to simulate these trapdoors in another way. (However, see Section 6.8 for a follow-up discussion on adaptive LWE.)

Towards constant-size keys and adaptive security, our construction circumvents the use of lattice trapdoors altogether. At a high level, we turn our attention to a much weaker lattice primitive called attribute-based laconic function evaluation (AB-LFE) [QWW18], and lift it to a KP-ABE scheme for circuits using pairing. AB-LFE is an interactive protocol. A receiver sends a digest of a function, which is exactly the matrix $\mathbf{B}_f$ in BGG$^+$. The sender then encodes the attribute $\mathbf{x}$ and message μ as follows:

$$\mathbf{s}^\intercal(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}^\intercal, \qquad \mathbf{s}^\intercal \mathbf{B}_f \mathbf{r} + e + \mu \lfloor q/2 \rceil.$$

Here, $\mathbf{r} = \mathbf{G}^{-1}(\mathbf{a})$ is the bit decomposition of a random vector $\mathbf{a}$. Security guarantees that the encoding reveals μ only when $f(\mathbf{x}) = 0$. At a first glance, the LFE encoding appears to be the same as BGG$^+$, but the novelty is in the details. Since the LFE encoding can depend on $\mathbf{B}_f$ (and hence f), it can be generated without using

lattice trapdoors — the short vector $\mathbf{r}$ sampled first, and $\mathbf{B}_f\mathbf{r}$ computed next. When $f(\mathbf{x}) = 1$, the hiding of μ follows directly from the pseudorandomness of LWE samples $(\mathbf{s}^\intercal(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}^\intercal)$ and $(\mathbf{s}^\intercal\mathbf{a} + e)$. When $\mathbf{x}$ is adaptively chosen, security follows naturally from adaptive LWE.

However, AB-LFE is able to avoid lattice trapdoor only because it is significantly weaker than ABE, or even 1-key ABE: *i)* its message encoding depends on $\mathbf{B}_f$ (unknown at ABE encryption time), and *ii)* it is only secure for a single function. Our next challenge is lifting AB-LFE back to full ABE, for which we use pairing.

More specifically, we first modify the AB-LFE scheme of [QWW18] to obtain a *weakly nearly linear secret sharing scheme* for circuits. It contains two parts:

$$\begin{aligned} \mathbf{L}^{\mathbf{x}} &= \mathbf{s}^\intercal(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}^\intercal && (\text{mod } q), \\ L_f &= \mathbf{s}^\intercal \lfloor \mathbf{B}_f\mathbf{r} \cdot p/q \rceil + \mu \lfloor p/2 \rceil && (\text{mod } p). \end{aligned}$$

Note that we round $\mathbf{B}_f\mathbf{r}$ from modulus q of $\text{poly}(\lambda, d)$ bits to p of $\text{poly}(\lambda)$ bits so that the component L_f dependent on f and μ has *constant size*, which is crucial towards constant-size ABE keys. To solve the problem that L_f requires knowledge of $\mathbf{B}_f$ unknown at encryption time, we use a pairing-based inner-product functional encryption (IPFE) to compute L_f in the exponent, by viewing it as an inner product

$$L_f = \langle (\mathbf{s}^\intercal, \mu), (\lfloor \mathbf{B}_f\mathbf{r} \cdot p/q \rceil, \lfloor p/2 \rceil) \rangle,$$

where the two vectors are known respectively at ABE encryption and key generation time. To overcome that AB-LFE only guarantees security for a single L_f. We follow the idea of [AY20,AWY20] to compute $\delta_f \cdot L_f$ in the exponent instead, where δ_f is an independent and random scalar chosen at key generation time. In the GGM, the presence of δ_f's prevents adversaries from meaningfully "combining" information from multiple L_f's for different f's.

Comparison with [AY20,AWY20]. Our way of combining lattice-based LSS with pairing-based IPFE differs from that of [AY20,AWY20], in order to address unique technical difficulties. To start with, they use an LSS scheme based on the BGG$^+$ ABE and inherits the selective security. Second, our KP-ABE scheme reveals part of the shares, $\mathbf{L}^{\mathbf{x}}$, *in the*

clear (in ciphertext), and only computes L_f in the exponent, whereas [AY20,AWY20] compute the entire sharing in the exponent. This is because the decryptor needs to perform the non-linear rounding operation on the result of homomorphic evaluation on $\mathbf{L}^{\mathbf{x}}$, in order to obtain $\lfloor \mathbf{s}^\top \mathbf{B}_f \mathbf{r} \cdot p/q \rceil$ for decryption. Keeping $\mathbf{L}^{\mathbf{x}}$ in the clear allows rounding, but renders security harder to prove.

Furthermore, the security proof of AB-LFE relies on noise flooding — their technique can only show that $(L_f + \widetilde{e})$ is pseudorandom for a super-polynomially large $\widetilde{e}$. But noise flooding is incompatible with computing L_f in the exponent, since we must keep noises polynomially small in order for decryption to be efficient (which brute-forces discrete logarithm). Without noise flooding, we cannot prove that the unauthorized shares are pseudorandom as in [QWW18]. Nevertheless, we show that they are "entropic", captured by a new notion called *non-annihilability*, and that the "entropic" L_f computed in the exponent still hides the message μ. In this dissertation, we present a simplified proof of non-annihilability using union bound (also better because the reduction no longer requires extra non-uniform advice). The original proof in [LLL22a] combines techniques from AB-LFE and leakage simulation [JP14, CCL18], reflecting the idea of L_f being entropic. The work of [AY20,AWY20] does not encounter issues with super-polynomial noises.

We add a note on our doubly succinct CP-ABE for Boolean formulae. It is closer to the CP-ABE scheme of [AY20,AWY20]. However, to obtain constant-size keys, we rely on an IPFE scheme with strong simulation security — it enables simultaneously simulating a polynomial number k of ciphertexts, while keeping the secret key *constant-size* (independent of k). Such strong simulation is impossible in the standard model following an incompressibility argument (each secret key must be prepared to have arbitrary k inner products programmed). We show that this is possible in GGM, and in particular, the IPFE scheme of [ABDP15] satisfies it. IPFE with such strong simulation may find other applications.

Next, we explain our ideas in more details.

Combining LSS with IPFE. An IPFE scheme enables generating keys $\mathsf{isk}(\mathbf{v}_j)$ and ciphertexts $\mathsf{ict}(\mathbf{u}_i)$ associated with vectors $\mathbf{v}_j, \mathbf{u}_i \in \mathbb{Z}_p^N$ such that decryption reveals

only their inner products $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ and hides all other information about $\mathbf{u}_i$ encrypted in the ciphertexts (whereas $\mathbf{v}_j$ associated with the keys are public). It can be based on a variety of assumptions such as MDDH, LWE, or DCR [ABDP15,ALS16].

A *weakly nearly* linear secret sharing scheme enables generating shares $L_f, \mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b$ associated with a policy f and some secret μ, such that for any input $\mathbf{x} \in \{0,1\}^\ell$, its corresponding subset of shares $\mathbf{L}^\mathbf{x} = (\mathbf{L}_0, \{\mathbf{L}_i^{\mathbf{x}[i]}\}_i)$, together with L_f, can be used to *approximately* reconstruct the secret μ if and only if $f(\mathbf{x}) = 0$:

$$(L_f, \mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}) \leftarrow \mathsf{Share}(f, \mu; \mathbf{s}),$$
$$f(\mathbf{x}) = 0 \quad \Longrightarrow \quad \mu \sim \mathsf{Recon}(f, \mathbf{x}, L_f, \mathbf{L}^\mathbf{x}).$$

Weak near linearity means that Recon is linear in the share L_f and that its output is close to the secret μ.

How can we combine these two primitives to construct a KP-ABE? We require $\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b$ to be independent of f and μ, and L_f to be linear in μ and the randomness $\mathbf{s}$ of Share. The first requirement allows us to simply put $\mathbf{L}^\mathbf{x}$ in the ciphertext. The second requirement allows us to encode $\mu, \mathbf{s}$ into ict's and the coefficients (of L_f as a function of $\mu, \mathbf{s}$) into isk's, so that their inner product is exactly L_f. The idea is as follows.

$$\left.\begin{array}{lll} \mathsf{kpsk}_f: & [\![\delta]\!]_2, & \mathsf{isk}(\delta \cdot (\text{coefficients of } L_f)) \\ \mathsf{kpct}_\mathbf{x}: & \mathbf{L}^\mathbf{x}, & \mathsf{ict}(\mu, \mathbf{s}) \end{array}\right\} \quad [\![\delta L_f]\!]_\mathrm{T} \text{ and } \mathbf{L}^\mathbf{x}. \tag{6.1}$$

If $f(\mathbf{x}) = 0$, the linear reconstruction can be carried out in the exponents to approximately obtain $[\![\delta\mu]\!]_\mathrm{T}$. Decryption enumerates all possible errors to recover μ exactly. We stress again that different from [AY20,AWY20], we keep $\mathbf{L}^\mathbf{x}$ in the clear (in the ciphertext), instead of computing the entire secret sharing $\mathbf{L}^\mathbf{x}, L_f$ in the exponent, which is important for achieving constant-size keys, but makes proving security more difficult.

We construct a secret sharing scheme that features L_f of *constant size*, which translates to KP-ABE with constant-size secret keys.

Combining secret sharing and IPFE to construct CP-ABE is similar. We can encode $\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b$ in ict's, and a "selection" vector according to $\mathbf{x}$ in isk's, so that their inner

products are exactly $\mathbf{L}^{\mathbf{x}}$.

$$\left.\begin{array}{ll} \mathsf{cpsk}_{\mathbf{x}}: & [\![\delta]\!]_2, \quad \mathsf{isk}(\delta \cdot (\text{selection vector for } \mathbf{x})) \\ \mathsf{cpct}_f: & \{\mathsf{ict}(\text{component of } L_f, \mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b)\}_{f/0/i,b} \end{array}\right\} \quad [\![\delta L_f]\!]_{\mathrm{T}} \text{ and } [\![\delta \mathbf{L}^{\mathbf{x}}]\!]_{\mathrm{T}}. \tag{6.2}$$

We use an IPFE scheme with secret keys of constant size, independent of the vector dimension or the number of ciphertexts, and a secret sharing scheme whose $L_f, \mathbf{L}^{\mathbf{x}}$ grows only with the input length $|\mathbf{x}|$. This translates to CP-ABE with double succinctness.

6.2.1 KP-ABE with Constant-Size Keys

Lattice-Based Weakly Nearly Linear Secret Sharing. The BGG$^+$ ABE scheme introduces an important homomorphic evaluation procedure. Given a public LWE matrix $\mathbf{B} = (\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell)$ and the following encoding of an input $\mathbf{x}$, one can homomorphically evaluate any circuit f on the input encoding to obtain an output encoding:

$$\begin{aligned} \mathbf{L}^{\mathbf{x}} &= \mathbf{s}^\intercal(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal, \\ \mathsf{EvalCX}(\mathbf{B}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}}) = \mathbf{c}_f^\intercal &= \mathbf{s}^\intercal(\mathbf{B}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_f^\intercal, \text{ where } \mathsf{EvalC}(\mathbf{B}, f) = \mathbf{B}_f. \end{aligned} \tag{6.3}$$

As discussed before, the BGG$^+$ ABE scheme uses lattice trapdoor simulation technique, which we try to avoid in order to get constant-size key and adaptive security.

We hence turn to using the weaker primitive of AB-LFE scheme introduced by [QWW18]. It is a two-party protocol between a sender and a receiver who share the LWE public matrix $\mathbf{B}$ as the common reference string. The receiver first computes a digest $\mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f)$ for a function f and sends it to the sender. Upon receiving the digest, the sender masks a message μ by an LWE sample $L_f = \mathbf{s}^\intercal \mathbf{z}_f + e + \mu\lfloor q/2 \rceil$, where $\mathbf{r} = \mathbf{G}^{-1}(\mathbf{a})$ and $\mathbf{z}_f = \mathbf{B}_f \mathbf{r}$ are the analogues of $\mathbf{r}_f$ and $\mathbf{z}$ in BGG$^+$. It also encodes an attribute $\mathbf{x}$ into LWE samples $\mathbf{L}^{\mathbf{x}}$ as described below.

$$\begin{aligned} \mathsf{crs} &= \mathbf{B}, \\ \mathsf{digest}_f &= \mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f), \end{aligned} \qquad \mathsf{ct}_{f,\mathbf{x}}(\mu) = \begin{cases} \mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n, \\ L_f = \mu\lfloor q/2 \rceil + \mathbf{s}^\intercal \overbrace{\mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a})}^{\mathbf{z}_f} + e, \\ \mathbf{L}^{\mathbf{x}} = \mathbf{s}^\intercal(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal. \end{cases}$$

To decrypt when $f(\mathbf{x}) = 0$, first run $\mathsf{EvalCX}(\mathbf{B}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$ to obtain $\mathbf{c}_f^\intercal = \mathbf{s}^\intercal(\mathbf{B}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_f^\intercal$, then compute $L_f - \mathbf{c}_f^\intercal \mathbf{r} = \mu\lfloor q/2 \rceil + (e - \mathbf{e}_f^\intercal \mathbf{r})$ and round it to recover μ.

Observe that the above scheme can be viewed as a weakly nearly linear secret sharing scheme, where the shares $\mathbf{L}^{\mathbf{x}}$ are chosen by $\mathbf{x}$ and the share L_f is dependent only on f and μ. At the moment, the bit-length of L_f is $\Theta(\log q)$. Since the noise growth during the homomorphic evaluation is exponential in the depth of the computation, q must be a $\mathrm{poly}(\lambda, d)$-bit modulus to accommodate for the noise growth. We next turn to reducing the size of L_f to a constant independent of d.

Rounding to Make L_f Constant-Size. Since the encrypted message is only a single bit, we can afford to lose a lot of precision in the above decryption process. In particular, the scheme is still correct if we round down the digest $\mathbf{B}_f$ to a much smaller, depth-independent, modulus p, and change L_f to use the rounded digest (while keeping $\mathbf{L}^{\mathbf{x}}$ unchanged):

$$L_f = \mu\lfloor p/2 \rceil + \mathbf{s}^\intercal \lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil + e \qquad \text{over } \mathbb{Z}_p.$$

During decryption, one now computes, over $\mathbb{Z}_p$,

$$\begin{aligned} L_f - \lfloor \mathbf{c}_f^\intercal \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil &= L_f - \Big\lfloor \big(\mathbf{s}^\intercal \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) - \overbrace{f(\mathbf{x})}^{=0} \mathbf{s}^\intercal \mathbf{a} + \mathbf{e}_f^\intercal \mathbf{G}^{-1}(\mathbf{a})\big) \cdot p/q \Big\rceil \\ &= L_f - \mathbf{s}^\intercal \lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil - \underbrace{\lfloor \mathbf{e}_f^\intercal \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil}_{e_f} + e_s \\ &= \mu\lfloor p/2 \rceil + (e - e_f + e_s), \end{aligned} \tag{6.4}$$

where the rounding error e_s is of magnitude $|e_s| = O(\|\mathbf{s}\|_1)$. As long as the error terms are much smaller than $p/4$, when $f(\mathbf{x}) = 0$, one can still recover μ. We can now recast the above rounded AB-LFE scheme into a secret sharing scheme (below) with L_f of bit-length $\Theta(\log p)$, independent of depth d. (Only the larger modulus q will, and thus $\mathbf{e}_f$ itself can, grow with d.)

$$\begin{aligned} \mathsf{pp} &= (\mathbf{a}, \mathbf{B}) && (\mathrm{mod}\ q), \\ L_f &= \mu\lfloor p/2 \rceil + \mathbf{s}^\intercal \lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil + e && (\mathrm{mod}\ p), \\ L^{\mathbf{x}} &= \mathbf{s}^\intercal \big(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}\big) + \mathbf{e}^\intercal && (\mathrm{mod}\ q). \end{aligned}$$

As shown in Equation (6.1), to obtain KP-ABE, we will use a pairing-based IPFE to compute L_f in the exponent. In particular, the IPFE secret key isk encodes

Setup():	$\mathsf{pp} = (\mathbf{a}, \mathbf{B}) = (\mathbf{a}, \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell)$.
ShareX(pp):	Compute LWE samples $\mathbf{L}_0 = \mathbf{s}^\intercal \mathbf{A}_0 + \mathbf{e}_0^\intercal, \quad \mathbf{L}_i^b = \mathbf{s}^\intercal(\mathbf{A}_i - b\mathbf{G}) + \mathbf{e}_i^\intercal,$ and output $(\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b, \mathbf{s})$.
ShareF($\mathsf{pp}, f, \mu, \mathbf{s}$):	Compute $\mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f)$, and output $L_f = \mathbf{s}^\intercal \lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil + \mu \lfloor p/2 \rceil$.
For all $\mathbf{x} \in \{0,1\}^\ell$,	let $\mathbf{L}^{\mathbf{x}} = (\mathbf{L}_0, \{\mathbf{L}_i^{\mathbf{x}[i]}\}_i)$.
Recon($\mathsf{pp}, f, \mathbf{x}, L_f, \mathbf{L}^{\mathbf{x}}$):	If $f(\mathbf{x}) = 1$, output $\perp$; otherwise, compute $\mathbf{c}_f^\intercal = \mathsf{EvalCX}(\mathbf{B}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$, and recover μ from $L_f - \lfloor \mathbf{c}_f^\intercal \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil \sim \mu \lfloor p/2 \rceil$.

Figure 6.3. Sketch of secret sharing scheme for KP-ABE.

$(\lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil, \lfloor p/2 \rceil)$ and the IPFE ciphertext ict encodes $(\mathbf{s}^\intercal, \mu)$. Together, they decrypt to exactly $[\![L_f]\!]_{\mathrm{T}}$. Since both vectors live in $\mathbb{Z}_p$, the KP-ABE key, consisting of only isk, is of size independent of d.

Our secret sharing scheme is summarized in Figure 6.3. It turns out that arguing security is actually tricky and requires additional modification.

Entropic Shares. However, using AB-LFE creates a further complication, as its security relies on flooding the e_f, e_s terms (which may contain information of $\mathbf{s}$) with e, in order to prove pseudorandomness of L_f. By Equations (6.3, 6.4), when $f(\mathbf{x}) = 1$,

$$L_f = \mu \lfloor p/2 \rceil + \lfloor \mathsf{EvalCX}(\mathbf{L}^{\mathbf{x}}, f, \mathbf{x})^\intercal \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil - \lfloor (\mathbf{s}^\intercal \mathbf{a} + e_a) \cdot p/q \rceil + (e - e_f + e_s).$$

Observe that in the above, for convenience later, an additional LWE noise e_a is introduced in $\lfloor (\mathbf{s}^\intercal \mathbf{a} + e_a) \cdot p/q \rceil$ (which, by rounding, simply equals to $\lfloor \mathbf{s}^\intercal \mathbf{a} \cdot p/q \rceil$).

At this point, in order to show that L_f is pseudorandom, if $\mathbf{x}$ is chosen before Setup, one could program the public matrices by $\mathbf{A}_i = \mathbf{A}'_i + \mathbf{x}[i]\mathbf{G}$ according to $\mathbf{x}$, where $\mathbf{B}' = (\mathbf{A}_0 - \mathbf{G}, \mathbf{A}'_1, \ldots, \mathbf{A}'_\ell)$ is sampled at random. And one would hope to apply LWE to argue that

$$\mathbf{L}^{\mathbf{x}} = \mathbf{s}^\intercal \big(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}\big) + \mathbf{e}^\intercal = \mathbf{s}^\intercal \mathbf{B}' + \mathbf{e}^\intercal \qquad \text{and} \qquad \mathbf{s}^\intercal \mathbf{a} + e_a$$

are jointly pseudorandom. However, the noise terms e_f, e_s may leak information about $\mathbf{e}$ (hence $\mathbf{s}$).

The solution in [QWW18] is noise flooding. By setting e to be super-polynomially larger than $(e_f - e_s)$, we have $e - e_f + e_s \approx_s e$. By LWE, we can now switch $\mathbf{L}^{\mathbf{x}}$ and $(\mathbf{s}^\intercal \mathbf{a} + e_a)$ to random and conclude that L_f is pseudorandom.

However, the unique challenge here is that L_f is going to be computed in the exponent of the pairing group and decryption only recovers $(\mu \lfloor p/2 \rceil + e)$ in the exponent. When e is super-polynomial, we can no longer extract μ out of the exponent. Our solution is avoiding flooding altogether (and as a by-product, forgoing the noise e in L_f). As such, we cannot prove pseudorandomness of L_f, but only a weaker security notion that we call *non-annihilability* (for L_f). This notion captures that L_f is still entropic.

NON-ANNIHILABILITY. Non-annihilability requires that no adversary, after seeing $\mathbf{L}^{\mathbf{x}}$ (but not L_f), can come up with a non-zero affine function γ such that $\gamma(L_f) = 0$. As we will see, this security notion, combined with GGM, suffices for our proof.

The intuition behind non-annihilability is that the noise e contributes so little to the entropy of L_f that it does not hurt to remove it. The noise $(e_f - e_s)$ is a leakage of the randomness (in particular, $\mathbf{e}$) that generates $\mathbf{L}^{\mathbf{x}}$. The results of [JP14,CCL18] (leakage simulation) suggest that any polynomial-range leakage can be simulated to arbitrary polynomial precision in polynomial time, which fits our case of $(e_f - e_s)$. However, even with leakage simulation, L_f cannot be proven pseudorandom, because the simulated is still $(e_f - e_s)$ is dependent on (the random values switched from) LWE samples — consider an imaginary case where the leakage simulation forces the least significant bit of L_f to be zero. In summary, L_f is something pseudorandom plus a small value, hence its non-annihilability. When formalized, it turns out that the proof can be done using a simple union bound. For the original proof using leakage simulation, see [LLL22a].

Multi-Key Security of KP-ABE in GGM. Our KP-ABE scheme combines an IPFE scheme with the secret sharing scheme described above. In our KP-ABE scheme, we only compute the L_f part of secret sharing using IPFE, and leave the $\mathbf{L}^{\mathbf{x}}$ part in the clear

so that rounding can be performed. To achieve multi-key security, we further employ the idea from [AY20,AWY20] to "isolate" each ABE secret key in GGM by multiplying it with a fresh random element δ. Equation (6.1) expands to

$$\left.\begin{array}{lll} \mathsf{kpsk}_f: & [\![\delta]\!]_2, & \mathsf{isk}([\![\delta\lfloor \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a})\cdot p/q\rceil,\ \delta\lfloor p/2\rceil]\!]_2) \\ \mathsf{kpct}_{\mathbf{x}}: & \mathbf{L}^{\mathbf{x}}, & \mathsf{ict}([\![\quad\quad \mathbf{s}^\top \quad\quad ,\quad \mu\quad]\!]_1) \end{array}\right\} \quad [\![\delta L_f]\!]_T \text{ and } \mathbf{L}^{\mathbf{x}}.$$

The decryption algorithm first computes IPFE decryption to recover $[\![\delta L_f]\!]_T$. It then computes (homomorphically in the exponent of $\mathbb{G}_T$)

$$[\![\delta L_f]\!]_T - [\![\delta]\!]_2 \cdot [\![\lfloor \mathbf{c}_f^\top \mathbf{G}^{-1}(\mathbf{a})\cdot p/q\rceil]\!]_1 = [\![\delta]\!]_2 \cdot [\![\mu\lfloor p/2\rceil + (-e_f + e_s)]\!]_1.$$

Since the noise $(-e_f + e_s)$ is in a polynomial range, the decryption algorithm enumerates all its possible values to recover μ.

Multi-key security, at a high level, relies on the fact that in GGM, an adversary can only learn information about $[\![\delta L_f]\!]_T$ by submitting zero-test queries of affine functions. When the adversary attacks multiple keys, it is submitting zero-test queries over $\{\delta_j, \delta_j L_{f_j}\}_j$. Let $\gamma(\{\delta_j, \delta_j L_{f_j}\}_j)$ be any adversarial zero-test query, we can view it as a degree-1 polynomial over δ_j's:

$$\gamma(\{\delta_j, \delta_j L_{f_j}\}_j) = \gamma_0 + \sum_j \gamma_j(L_{f_j})\cdot\delta_j,$$

where $\gamma_j(L_{f_j})$ is the coefficient of δ_j. Since each δ_j is sampled independently at random, by Schwartz–Zippel, with all but negligible probability, γ evaluates to zero only if all γ_j's evaluate to zero. In other words, the adversary is effectively constrained to annihilate each L_{f_j} individually. By the non-annihilability for L_f, if any γ_j is not the zero function, it evaluates to non-zero with overwhelming probability. Therefore, the adversary learns no information about any L_{f_j} and hence the message μ encoded in them.

Summary of Our KP-ABE. Combining the above secret sharing scheme with an IPFE scheme, we obtain a KP-ABE scheme for bounded-depth circuits as summarized in Figure 6.4.

Our KP-ABE scheme achieves the same asymptotic ciphertext compactness as the BGG$^+$ scheme. Let d be an upper bound on the depth of the policy f, then

$|\mathsf{ct}| = (|\mathbf{x}| + 1)\,\mathrm{poly}(\lambda, d)$. The secret keys of our scheme contains only $\mathrm{O}(1)$ group elements, in fact only *three* using the IPFE scheme of [ALS16]. We set $\log p = \mathrm{poly}(\lambda)$ and obtain constant-size keys.

$\mathsf{Setup}()$:	$\mathsf{mpk} = (\mathsf{pp}, \mathsf{impk})$ and $\mathsf{msk} = \mathsf{imsk}$.
$\mathsf{KeyGen}(\mathsf{msk}, f)$:	Sample random non-zero δ. Compute $\mathbf{B}_f = \mathsf{EvalC}(\mathbf{B}, f)$. Output $\mathsf{sk} = (\llbracket \delta \rrbracket_2, \mathsf{isk}(\llbracket \delta \lfloor \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil, \delta \lfloor p/2 \rceil \rrbracket_2))$.
$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$:	Run $(\mathbf{L}_0, \{\mathbf{L}_i^b\}_i^b, \mathbf{s}) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp})$. Output $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict}(\llbracket \mathbf{s}^\intercal, \mu \rrbracket_1))$.
$\mathsf{Dec}(\mathsf{mpk}, f, \mathsf{sk}, \mathbf{x}, \mathsf{ct})$:	Run IPFE decryption to get $\llbracket \delta L_f \rrbracket_\mathrm{T}$. Compute $\mathbf{c}_f^\intercal = \mathsf{EvalCX}(\mathbf{B}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$. Find μ from $\llbracket \delta L_f \rrbracket_\mathrm{T} - \llbracket \delta \rrbracket_2 \llbracket \lfloor \mathbf{c}_f^\intercal \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil \rrbracket_1 \sim \llbracket \delta \rrbracket_2 \llbracket \mu \lfloor p/2 \rceil \rrbracket_1$.

Figure 6.4. Sketch of KP-ABE.

6.2.2 CP-ABE with Double Succinctness

Building Doubly Succinct CP-ABE. To build a CP-ABE scheme we need a different secret sharing construction. As described in Equation (6.2), in the CP case, we use IPFE to compute $\mathbf{L}^{\mathbf{x}}$ in the exponent and cannot perform rounding on it. Without rounding, the $\mathbf{e}_f$ term, as a result of EvalCX, in Equation (6.4) becomes super-polynomial. This again makes ABE decryption inefficient.

Fortunately, for Boolean formulae, the work of [GV15] develops specialized homomorphic evaluation procedures $\mathsf{EvalF}, \mathsf{EvalFX}$ ensuring that the evaluation noise $\mathbf{e}_f$ is in a polynomial range. Therefore, our secret sharing scheme for CP-ABE removes the rounding and replaces $\mathsf{EvalC}, \mathsf{EvalCX}$ by $\mathsf{EvalF}, \mathsf{EvalFX}$. We summarize our modified secret sharing scheme in Figure 6.5 ($\mathsf{Setup}, \mathsf{ShareX}$ mostly remain the same).

As noted before, in our CP-ABE scheme we use IPFE to compute $\mathbf{L}^{\mathbf{x}}$. We put an IPFE key for a "selector" vector into sk for $\mathbf{x}$, and arrange $\mathbf{L}$'s in IPFE ciphertexts in ct for f carefully so that they decrypt to $\llbracket \delta \mathbf{L}^{\mathbf{x}} \rrbracket_\mathrm{T}$.

Given $\mathbf{x} \in \{0,1\}^\ell$, its selection vector is

$$\mathbf{v} = (1,\ 1 - \mathbf{x}[1],\ \mathbf{x}[1],\ \ldots,\ 1 - \mathbf{x}[\ell],\ \mathbf{x}[\ell])^\intercal.$$

The shares are encoded as the columns of

$$\begin{pmatrix} L_f & \mathbf{L}_0[1] & \mathbf{L}_0[2] & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{L}_1^0[1] & \mathbf{L}_1^0[2] & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{L}_1^1[1] & \mathbf{L}_1^1[2] & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

The inner product between $\mathbf{v}$ and each column is L_f, or a component of $\mathbf{L}_0$, or a component of $\mathbf{L}_i^{\mathbf{x}[i]}$, or zero (from zero times a component of $\mathbf{L}_i^{\neg\mathbf{x}[i]}$). In other words, exactly the shares $L_f, \mathbf{L}^{\mathbf{x}}$.

Our CP-ABE scheme for Boolean formulae is summarized in Figure 6.6. It is doubly succinct — the key size is $|\mathsf{sk}| = \mathrm{poly}(\lambda)$ and the ciphertext size is $|\mathsf{ct}| = (\ell+1)^2\,\mathrm{poly}(\lambda)$.

Simulation Security for IPFE in GGM. Similar to the security proof for KP-ABE, our security proof for CP-ABE requires simulation security of IPFE. Suppose the adversary submits the challenge after a key query, then we need

$$\mathsf{isk}([\![\delta\mathbf{v}]\!]_2), \{\mathsf{ict}([\![\mathbf{u}\text{'s}]\!]_1)\} \approx \widetilde{\mathsf{isk}}([\![\delta\mathbf{v}]\!]_2; [\![\delta L_f, \delta\mathbf{L}^{\mathbf{x}}]\!]_2), \{\widetilde{\mathsf{ict}}(\perp)\}.$$

In the above, we need to simulate *multiple* IPFE ciphertexts and program all their decryption outcome $\delta L_f, \delta\mathbf{L}^{\mathbf{x}}$ in the secret key. This is possible using existing IPFE schemes (see [ALS16] and Chapter 5), but at the cost of having the secret key size

$\mathsf{ShareF}'(\mathsf{pp}, f, \mu, \mathbf{s})$:	Compute $\mathbf{B}_f = \mathsf{EvalF}(\mathbf{B}, f)$, and output $L_f = \mathbf{s}^\intercal\mathbf{B}_f\mathbf{G}^{-1}(\mathbf{a}) + e + \mu\lfloor p/2\rceil$.
$\mathsf{Recon}'(\mathsf{pp}, f, \mathbf{x}, L_f, \mathbf{L}^{\mathbf{x}})$:	If $f(\mathbf{x}) = 1$, output $\perp$; otherwise, compute $\mathbf{c}_f^\intercal = \mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$, and recover μ from $L_f - \mathbf{c}_f^\intercal\mathbf{G}^{-1}(\mathbf{a}) \sim \mu\lfloor p/2\rceil$.

Figure 6.5. Modified secret sharing scheme for CP-ABE.

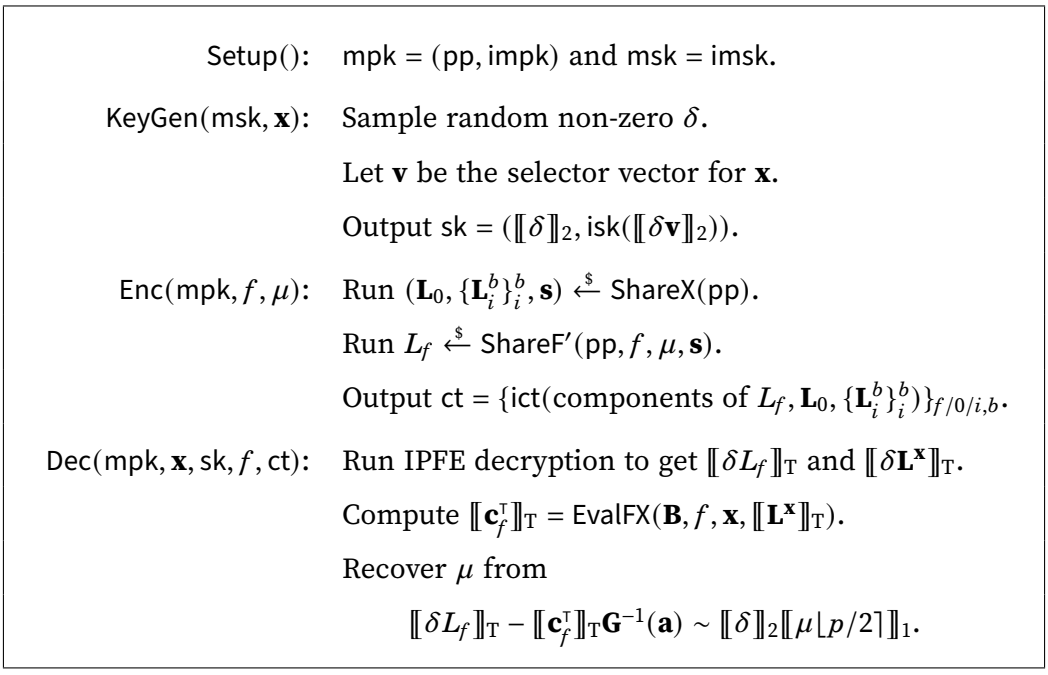


Figure 6.6. Sketch of CP-ABE.

proportional to the number of ciphertexts to be simulated, which is at least ℓ for us. However, we aim for constant-size secret keys (independent of ℓ). Unfortunately, in the standard model, it is impossible to achieve simulation security for k ciphertexts if the secret key is shorter than k bits by an incompressibility argument [BSW11]. We show that simulation security for unbounded polynomially many ciphertexts can nevertheless be achieved with constant-size secret keys in the GGM. In particular, the IPFE scheme of [ABDP15], whose secret key is a single group element, satisfies it. Roughly speaking, in the GGM, an adversary only learns information about values in the exponent through zero-test queries over the pairings of keys and ciphertexts, which the simulator can answer by translating them into zero-test queries over the inner products. We in fact prove *adaptive* simulation security for the [ABDP15] IPFE scheme.

6.2.3 Adaptive Security

Examining the security sketch for KP-ABE, we observe that in our construction, the $\mathsf{ict}([\![\mathbf{s}, \mu]\!]_1)$ component does not depend on the challenge attribute $\mathbf{x}$. This means that

even in the adaptive KP-ABE game, where $\mathbf{x}$ is decided after some key queries, the ict component of ABE ct can be fixed at the beginning of the game, before any key queries. Therefore, we can still rely on selective simulation security of IPFE for the first proof step.

However, when we next need to invoke non-annihilability for L_f, we run into a problem. The security only holds when $\mathbf{x}$ is chosen before the LWE public matrix $\mathbf{B}$ is revealed in the public parameter pp of the secret sharing scheme. To achieve adaptive security for ABE, we need adaptive non-annihilability, which allows $\mathbf{x}$ to be chosen adaptively, dependent on pp. We show that this is implied by the *adaptive* LWE assumption formulated in [QWW18].

In summary, we obtain adaptively secure KP-ABE for circuits and CP-ABE for Boolean formulae both with constant-size keys from GGM and adaptive LWE. (Our adaptive CP-ABE still enjoys ciphertext smaller than policy description size.) However, see Section 6.8 for a follow-up discussion on adaptive LWE.

6.3 Preliminaries

Table 6.1 explains select single-letter symbols used in this chapter.

The following lemmas will come handy.

Lemma 6.1 (Schwartz–Zippel). *Let $P(\mathbf{z})$ be a non-zero polynomial with Z indeterminates of degree at most d over $\mathbb{Z}_p$, then* $\Pr\left[\mathbf{z} \xleftarrow{\$} \mathbb{Z}_p^Z : P(\mathbf{z}) = 0\right] \leq d/p$.

Lemma 6.2 (splitting rounding). *It holds that $\left|\lfloor x + y \rceil - \lfloor x \rceil - \lfloor y \rceil\right| \leq 1$ for all $x, y \in \mathbb{R}$.*

Lemma 6.3 (factoring rounding). *Let $n \in \mathbb{N}$, $\mathbf{s} \in \mathbb{Z}^n$, $\mathbf{b} \in \mathbb{R}^n$, then $\left|\lfloor \mathbf{s}^\intercal \mathbf{b} \rceil - \mathbf{s}^\intercal \lfloor \mathbf{b} \rceil\right| \leq n\|\mathbf{s}\|$.*

Lemma 6.4 (rounding and modulus switching). *Let $p, q \in \mathbb{N}_+$ and X, Y be uniformly distributed over $\mathbb{Z}_q, \mathbb{Z}_p$, respectively, then the statistical distance between $\lfloor X \cdot p/q \rceil$ and Y is bounded by p/q, where the multiplication is done over $\mathbb{R}$ by taking any representative and the rounded value is interpreted as an element of $\mathbb{Z}_p$.*

Computation Models. In this chapter, we construct KP-ABE for circuits and CP-ABE for formulae. For technical reasons, when working with CP-ABE, we use permutation

Table 6.1. Select single-letter symbols in this chapter.

symbol	meaning
b, μ	challenge bits
p, q	modulus of pairing, modulus of LWE
$\gamma, \boldsymbol{\gamma}, \boldsymbol{\Gamma}$	zero-test query (or batches of them; GGM/pairing)
$\mathbf{G}, \mathbf{H}$	gadget matrix, homomorphic evaluation matrix
$\mathbf{A}, \mathbf{a}, \mathbf{s}, \mathbf{e}$	public matrix, public vector, secret, noise (LWE)
$\mathbf{B}, \mathbf{Q}$	public matrix, matrix introduced by reduction (adaptive LWE)
$f, \mathbf{x}, L, \mathbf{L}$	function, input, share, shares
r / $\mathbf{s}$	secret sharing randomness (abstract/concrete)
δ	random scalar (GGM/pairing)
$\mathbf{w}, a$	IPFE master secret key, public key randomness (DDH-style)
s, a	IPFE encryption randomness

branching programs of width five (5-permutation branching programs, or 5-PBP) in place of Boolean formulae. They are inter-convertible in polynomial time [Bar86]. The precise definition of 5-PBP is standard in the literature, for which we refer the readers to [GV15].

6.3.1 Lattice Tools

Homomorphic Evaluation. We use the following abstraction of homomorphic evaluation for ABE over lattices, developed in a series of works [GSW13,BGG+14,GV15] with the syntax in [BV15,BTVW17]. The actual algorithms we use are that for ABE for circuits in [BGG+14] (slightly changed), and that for ABE for 5-PBP in [GV15]. The former handles general polynomial-size circuits, but the noise grows exponentially in the circuit depth. The latter is specialized for 5-PBP, and the noise grows polynomially in the branching program size.

In our version of the algorithms from [BGG+14], instead of using $\mathbf{G}$ as the gadget matrix, we consider $\mathbf{QG}$ for any invertible $\mathbf{Q}$. Note that $\mathbf{G}^{-1}(\mathbf{Q}^{-1} \cdot \textvisiblespace)$ is a right inverse

of **QG** with binary output. We replace any invocation of $\mathbf{G}^{-1}(\sqcup)$ in the original algorithms by $\mathbf{G}^{-1}(\mathbf{Q}^{-1} \cdot \sqcup)$ to obtain the following.

Lemma 6.5 (homomorphic evaluation for circuits, adapted from [BGG+14]). *There are two efficient deterministic algorithms* $\mathsf{EvalC}, \mathsf{EvalCX}$ *satisfying the following conditions. Let* n, ℓ, q *be positive integers,* $m = n\lceil \log_2 q \rceil$, $\mathbf{G}$ *the gadget matrix,* $\mathbf{B}$ *a matrix over* $\mathbb{Z}_q$ *of shape* $n \times (\ell+1)m$, $\mathbf{Q}$ *an invertible matrix over* $\mathbb{Z}_q$ *of shape* $n \times n$, $\mathbf{x}$ *an* ℓ*-bit string (row vector), and* f *a circuit of depth* d *with input length* ℓ.

- $\mathsf{EvalC}(\mathbf{B}, \mathbf{Q}, f)$ *outputs* $\mathbf{H}_f \in \mathbb{Z}^{(\ell+1)m \times m}$.
- $\mathsf{EvalCX}(\mathbf{B}, \mathbf{Q}, f, \mathbf{x})$ *outputs* $\mathbf{H}_{f,\mathbf{x}} \in \mathbb{Z}^{(\ell+1)m \times m}$.

They satisfy

$$\|\mathbf{H}_f^\intercal\|, \|\mathbf{H}_{f,\mathbf{x}}^\intercal\| \le (m+2)^d, \qquad \big(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{QG}\big)\mathbf{H}_{f,\mathbf{x}} = \mathbf{B}\mathbf{H}_f - f(\mathbf{x})\mathbf{QG}.$$

Lemma 6.6 (homomorphic evaluation for 5-PBP [GV15]). *There exist two efficient deterministic algorithms* $\mathsf{EvalF}, \mathsf{EvalFX}$ *satisfying the following conditions. Let* n, ℓ, q *be positive integers,* $m = n\lceil \log_2 q \rceil$, $\mathbf{G}$ *the gadget matrix,* $\mathbf{B}$ *a matrix over* $\mathbb{Z}_q$ *of shape* $n \times (\ell+1)m$, $\mathbf{x}$ *an* ℓ*-bit string (row vector), and* f *a 5-PBP of length* ℓ_{BP} *with input length* ℓ.

- $\mathsf{EvalF}(\mathbf{B}, f)$ *outputs* $\mathbf{H}_f \in \mathbb{Z}^{(\ell+1)m \times m}$.
- $\mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x})$ *outputs* $\mathbf{H}_{f,\mathbf{x}} \in \mathbb{Z}^{(\ell+1)m \times m}$.

They satisfy

$$\|\mathbf{H}_f^\intercal\|, \|\mathbf{H}_{f,\mathbf{x}}^\intercal\| \le 3m\ell_{\mathrm{BP}} + 1, \qquad \big(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}\big)\mathbf{H}_{f,\mathbf{x}} = \mathbf{B}\mathbf{H}_f - f(\mathbf{x})\mathbf{G}.$$

Assumption. We rely on the adaptive learning with errors (LWE) assumption, a natural variant of LWE first proposed in [QWW18]. (However, see Section 6.8 for a follow-up discussion on this assumption.)

Assumption 6.1 (adaptive LWE, adapted from [QWW18]). Let $n \le \mathrm{poly}(\lambda)$, $q \le 2^{\mathrm{poly}(\lambda)}$, $\sigma = \sigma(\lambda)$, $m = n\lceil \log_2 q \rceil$, and $\mathbf{G}$ be the gadget matrix. (The argument λ for those functions are omitted below.) The *adaptive LWE assumption* $\mathsf{ALWE}_{n,q,\sigma}$ states that $\mathsf{Exp}^0_{\mathrm{ALWE}} \approx \mathsf{Exp}^1_{\mathrm{ALWE}}$, where $\mathsf{Exp}^b_{\mathrm{ALWE}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $(1^\ell, 1^{m'})$ from it. Sample

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m'}, \qquad \mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times(\ell+1)m},$$

  and send $(\mathbf{A}, \mathbf{B})$ to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ submits $\mathbf{x} \in \{0,1\}^\ell$. Depending on b,

$$\begin{aligned}
&(\text{if } b = 0) \quad \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \quad \mathbf{e} \xleftarrow{\$} \mathcal{D}^{m'}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}, \quad \mathbf{f} \xleftarrow{\$} \mathcal{D}^{(\ell+1)m}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}},\\
&\qquad\qquad \mathbf{c}^\intercal \leftarrow \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal, \quad \mathbf{d}^\intercal \leftarrow \mathbf{s}^\intercal\big(\mathbf{B} - (1, \mathbf{x}^\intercal)\otimes\mathbf{G}\big) + \mathbf{f}^\intercal;\\
&(\text{if } b = 1) \quad \mathbf{c}^\intercal \xleftarrow{\$} \mathbb{Z}_q^{1\times m'}, \quad \mathbf{d}^\intercal \xleftarrow{\$} \mathbb{Z}_q^{1\times(\ell+1)m}.
\end{aligned}$$

  Send $(\mathbf{c}, \mathbf{d})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit, which is the outcome of the experiment.

For our KP-ABE, we need a small-secret variant of the adaptive LWE assumption.

Assumption 6.2 (small-secret adaptive LWE). Using the symbols in Assumption 6.1, the *small-secret adaptive LWE assumption* $\mathsf{sALWE}_{n,q,\sigma}$ states that $\mathsf{Exp}^0_{\mathsf{sALWE}} \approx \mathsf{Exp}^1_{\mathsf{sALWE}}$, where $\mathsf{Exp}^b_{\mathsf{sALWE}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $(1^\ell, 1^{m'})$ from it. Sample

$$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m'}, \qquad \mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times(\ell+1)m}, \qquad \boxed{\mathbf{Q} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}},$$

  conditioned on $\mathbf{Q}$ being invertible. Send $(\mathbf{A}, \mathbf{B}, \mathbf{Q})$ to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ submits $\mathbf{x} \in \{0,1\}^\ell$. Depending on b,

$$\begin{aligned}
&(\text{if } b = 0) \quad \mathbf{s} \xleftarrow{\$} \boxed{\mathcal{D}^n_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}}, \quad \mathbf{e} \xleftarrow{\$} \mathcal{D}^{m'}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}, \quad \mathbf{f} \xleftarrow{\$} \mathcal{D}^{(\ell+1)m}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}},\\
&\qquad\qquad \mathbf{c}^\intercal \leftarrow \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal, \quad \mathbf{d}^\intercal \leftarrow \mathbf{s}^\intercal\big(\mathbf{B} - (1, \mathbf{x}^\intercal)\otimes\boxed{\mathbf{Q}}\mathbf{G}\big) + \mathbf{f}^\intercal;\\
&(\text{if } b = 1) \quad \mathbf{c}^\intercal \xleftarrow{\$} \mathbb{Z}_q^{1\times m'}, \quad \mathbf{d}^\intercal \xleftarrow{\$} \mathbb{Z}_q^{1\times(\ell+1)m}.
\end{aligned}$$

  Send $(\mathbf{c}, \mathbf{d})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit, which is the outcome of the experiment.

Lemma 6.7 (¶). *Small-secret adaptive LWE* $\mathsf{sALWE}_{n,q,\sigma}$ *(Assumption 6.2) holds if so does adaptive LWE* $\mathsf{ALWE}_{n,q,\sigma}$ *(Assumption 6.1).*

Parameters. We rely on the hardness of adaptive LWE with subexponential modulus-to-noise ratio. For some $0 < \rho < \frac{1}{2}$, the adaptive LWE assumption is assumed to be hard when the dimension is $n = \mathrm{poly}(\lambda)$, the *prime* modulus is $q = 2^{\Theta(n^\rho)}$, and the noise width is $\sigma = B_{\mathrm{init}}/\sqrt{\lambda}$ for some $B_{\mathrm{init}} = \mathrm{poly}(\lambda)$.[57] Similarly to Section 2.3, those parameters should be chosen by the adversary subject to those constraints — we require a prime q, it is currently unknown [TCH12] how to pick $\mathrm{poly}(\lambda)$-bit primes in deterministic polynomial time, and the adversary's choice of scheme parameters affect the ranges of LWE parameters.

Proof (Lemma 6.7). We follow the ideas in [ACPS09] to transform LWE samples into small-secret LWE samples. For now, suppose q is prime (this restriction can be removed later). Let $\mathcal{A}$ be an efficient adversary against $\mathsf{sALWE}_{n,q,\sigma}$. Consider the following $\mathcal{B}$ against $\mathsf{ALWE}_{n,q,\sigma}$.

1. The adversary $\mathcal{B}$ launches $\mathcal{A}()$, receives $(1^\ell, 1^{m'})$ from $\mathcal{A}$, and sends $(1^\ell, 1^{6n\lambda+m'})$ to its own experiment.

2. It receives back $(\mathbf{A}, \mathbf{B})$ of shapes $n \times (6n\lambda + m')$ and $n \times (\ell+1)m$. The adversary $\mathcal{B}$ goes through the first $6n\lambda$ columns of $\mathbf{A}$ to accumulate n columns that form an invertible matrix. If no such invertible matrix is found, $\mathcal{B}$ aborts. Otherwise, it sets $\mathbf{Q}^{-1}$ to be this invertible matrix, and $\mathbf{A}'$ to be the last m' columns of $\mathbf{A}$. The adversary $\mathcal{B}$ sends $(\mathbf{QA}', \mathbf{QB}, \mathbf{Q})$ to $\mathcal{A}$.

3. $\mathcal{B}$ receives an input $\mathbf{x} \in \{0,1\}^\ell$ from $\mathcal{A}$, and forwards $\mathbf{x}$ to its own experiment.

4. $\mathcal{B}$ receives back $(\mathbf{c}, \mathbf{d})$. It defines $\mathbf{q}$ to be the entries of $\mathbf{c}$ corresponding to the columns that form $\mathbf{Q}^{-1}$, and $\mathbf{c}'$ to be the last m' entries of $\mathbf{c}$. The adversary $\mathcal{B}$ sets

$$\widetilde{\mathbf{c}}^\intercal = \mathbf{q}^\intercal\mathbf{QA}' - (\mathbf{c}')^\intercal, \qquad \widetilde{\mathbf{d}}^\intercal = \mathbf{q}^\intercal\mathbf{QB} - (1, \mathbf{x}^\intercal) \otimes \mathbf{q}^\intercal\mathbf{QG} - \mathbf{d}^\intercal,$$

and sends $(\widetilde{\mathbf{c}}, \widetilde{\mathbf{d}})$ to $\mathcal{A}$.

[57] Unlike in Section 2.3, where the initial noise can be super-polynomial in λ, we crucially rely on B_{init} being polynomial because later, we will brute-force discrete logarithm in a range polynomially related to B_{init}.

5. The adversary $\mathcal{B}$ receives a bit from $\mathcal{A}$. It outputs the same bit.

Since q is prime, $\mathcal{B}$ can complete Step 2 by greedily taking each column as long as it is independent of the already chosen ones. Therefore, $\mathcal{B}$ is efficient. If $\mathcal{B}$ does not abort in Step 2, then $\mathcal{B}$ in $\mathsf{Exp}^b_{\mathrm{ALWE}}$ emulates $\mathsf{Exp}^b_{\mathrm{sALWE}}$ for $\mathcal{A}$ (the case of $b = 0$ can be verified with tedious yet routine calculation). It remains to show the aborting probability is negligible. We split the $6n\lambda$ columns into 6λ blocks of n columns, then

$$\begin{aligned}\Pr[\text{first block is invertible}] &= (1-q^{-n})\cdots(1-q^{-1}) \geq \prod_{k=1}^{+\infty}(1-2^{-n}) \\ &= \exp\left(\sum_{k=1}^{+\infty}\log(1-2^{-n})\right) \geq \exp\left(-2\sum_{k=1}^{+\infty}2^{-n}\right) = e^{-2}.\end{aligned}$$

When $\mathcal{B}$ aborts, no block can be invertible. This gives

$$\begin{aligned}\Pr[\mathcal{B}\text{ aborts}] &\leq \Pr[\text{all blocks are non-invertible}] \\ &\leq (\Pr[\text{first block is non-invertible}])^{6\lambda} \\ &\leq (1-e^{-2})^{6\lambda} \leq \left((1-e^{-2})^{e^2}\right)^{\lambda/\log_2 e} \leq (e^{-1})^{\lambda/\log_2 e} = 2^{-\lambda}.\end{aligned}$$

This completes the reduction for prime q.

It can be tweaked to work with arbitrary q, without knowing its prime factors.[58] For this, we need nt columns (split into t blocks of n columns) for finding $\mathbf{Q}^{-1}$ and let $\mathcal{B}$ pick the first invertible block. Calculation shows that $t = \Theta(\lambda \log^2 \log q)$ ensures a failure probability of at most $2^{-\Theta(\lambda)}$. □

6.3.2 Generic Asymmetric Pairing Group Model

We construct our ABE using pairing groups and prove its security in the generic (asymmetric) pairing group model (GGM), where the pairing groups can only be accessed via handles representing group elements and oracles for operating the handles. There are several different yet equivalent definitions of the model. We use a minimalist definition simplified from [Ps16], which suffices for security proofs.

[58] When q contains two prime factors, the greedy algorithm fails. When q contains three known prime factors, deciding whether $\mathbf{A}$ contains an invertible $n \times n$ matrix is NP-hard by reduction from 3-dimensional matching. All hope is not lost (read on). The tweak can be ported back to the setting of [ACPS09] (the latter only considers the case when q is a prime power).

Definition 6.1 (GGM [Ps16]). Let $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of distributions over prime numbers. The *generic (asymmetric pairing) group model (GGM)* of order $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four stateful oracles. The state consists of a sample $p \xleftarrow{\$} p_\lambda$ and three lists $\{\mathsf{Vals}_i\}_{i\in\{1,2,\mathrm{T}\}}$ storing the encoded values, which are initially singleton lists of 1 (representing the generators). The oracles are as follows.

- The encoding oracle $\mathsf{GrpEnc}_i(v)$, where $i \in \{1, 2, \mathrm{T}\}$, takes $v \in \mathbb{Z}_p$ as input. It appends v to Vals_i and outputs the current length of Vals_i (as a handle for v).
- The zero-test oracle $\mathsf{GrpZT}(\gamma)$ takes as input a linear function γ. It evaluates

$$\gamma\left(\{\mathsf{Vals}_1[j_1]\cdot\mathsf{Vals}_2[j_2]\}_{j_1\in[J_1],j_2\in[J_2]},\ \{\mathsf{Vals}_\mathrm{T}[j_\mathrm{T}]\}_{j_\mathrm{T}\in[J_\mathrm{T}]}\right),$$

  where J_i is the current length of Vals_i for $i \in \{1, 2, \mathrm{T}\}$. The oracle outputs whether the evaluation yields 0 or not.

Both the security experiment and the adversary is given $(1^\lambda, p)$ at the beginning. The security experiment can call all the four oracles, yet the adversary is only allowed to call GrpZT.

We remark two major differences between our formulation and that in [Ps16]. First, instead of random bit-strings, we use sequential indices as the handles, which can be used to position the coefficients of linear functions passed into GrpZT. Second, for zero-test queries, the function is linear over all values that can be computed in the target group, which simplifies the interface.

As noted in [Ps16], the definition does not explicitly allow algebraic operations over the encodings, yet this is without loss of generality. The security experiment provides the oracles and maintains their states, so it *knows* each value encoded inside each handle, and can operate over the values and call GrpEnc_i on the result.[59] The adversary can replace all (allowed) algebraic operations by keeping track of appropriate linear functions over the values encoded by the security experiment, and zero-test them.

[59] Due to the sequential numbering of handles, the handles returned from GrpEnc_i's are "leaked" to the adversary, even if the security experiment intends to keep it as an intermediate computation result not given to the adversary. Therefore, it is important to model the security experiment as not making extra calls.

6.3.3 Inner-Product Functional Encryption

Inner-product functional encryption schemes enable generating keys and ciphertexts tied to vectors. Decryption reveals the inner product and nothing more about the plaintext vector. In this chapter, we consider IPFE schemes based on pairing, where keys and ciphertexts are encoded in the two source groups and decryption recovers inner products encoded in the target group.

Definition 6.2 (IPFE). Let GroupGen be a pairing group sampler (Definition 2.5). An *inner-product functional encryption scheme* based on GroupGen is a PHFE (Definition 2.1) with

$$\begin{aligned} \mathsf{Params}_{\lambda,\mathsf{gp}} &= \{\, 1^N \mid N \in \mathbb{N} \,\}, & [\![\mathbf{v}]\!]_2 &\in F_{\lambda,\mathsf{gp},1^N} = \mathbb{G}_2^N, \\ X_{\lambda,\mathsf{gp},1^N} &= \{\bot\}, & [\![\mathbf{u}]\!]_1 &\in Y_{\lambda,\mathsf{gp},1^N} = \mathbb{G}_1^N, \\ Z_{\lambda,\mathsf{gp},1^N} &= \mathbb{G}_T, & \varphi_{\lambda,\mathsf{gp},1^N}([\![\mathbf{v}]\!]_2, \bot, [\![\mathbf{u}]\!]_1) &= [\![\mathbf{u}^\top \mathbf{v}]\!]_T. \end{aligned}$$

The scheme is *succinct* if the length of isk[60] is independent of N and only depends on λ, gp.

Definition 6.3 (selective simulation [Wee17]). A *simulator* for an IPFE scheme (Definition 6.2) consists of three efficient algorithms.

- $\mathsf{SimSetup}(1^\lambda, \mathsf{gp}, 1^N)$ takes the same input as Setup, and outputs simulated keys $(\mathsf{impk}, \mathsf{imsk})$.
- $\mathsf{SimKeyGen}(\mathsf{imsk}, [\![\mathbf{v}]\!]_2, [\![z]\!]_2)$ takes as input the simulated imsk, a vector encoded in $\mathbb{G}_2$, and an inner product encoded in $\mathbb{G}_2$. It outputs a simulated key isk.
- $\mathsf{SimEnc}(\mathsf{imsk})$ takes the simulated master *secret* key as input, and outputs a simulated ciphertext ict.

The IPFE scheme is *selectively simulation-secure* if there exists a simulator such that $\mathsf{Exp}_{\mathsf{real}} \approx \mathsf{Exp}_{\mathsf{sim}}$, where $\mathsf{Exp}_{\mathsf{real}}(1^\lambda, \mathcal{A})$ or $\mathsf{Exp}_{\mathsf{sim}}(1^\lambda, \mathcal{A})$ proceeds as follows.

[60] Although this kind of IPFE is formally defined as a PHFE, by convention, we still write impk, imsk, isk, ict for its components. Compare with Definition 4.4, which directly defines another kind of IPFE and specifies its component names.

- **Challenge.** Launch $\mathcal{A}(1^\lambda, \mathsf{gp})$ and receive from it the vector dimension 1^N and the challenge vector $\mathbf{u} \in \mathbb{Z}_p^N$.
- **Setup.** Run

$$\begin{aligned}
&(\text{in } \mathsf{Exp}_{\mathsf{real}}) \quad (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, 1^N), \qquad \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{impk}, [\![\mathbf{u}]\!]_1);\\
&(\text{in } \mathsf{Exp}_{\mathsf{sim}}) \quad (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{SimSetup}(1^\lambda, \mathsf{gp}, 1^N), \quad \mathsf{ict} \xleftarrow{\$} \mathsf{SimEnc}(\mathsf{imsk}).
\end{aligned}$$

  Send $\mathsf{impk}, \mathsf{ict}$ to $\mathcal{A}$.

- **Query.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ submits a vector $[\![\mathbf{v}_j]\!]_2$ encoded in $\mathbb{G}_2$. Upon receiving the query, run

$$\begin{aligned}
&(\text{in } \mathsf{Exp}_{\mathsf{real}}) \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, [\![\mathbf{v}_j]\!]_2);\\
&(\text{in } \mathsf{Exp}_{\mathsf{sim}}) \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{SimKeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2, \mathbf{u}^\intercal [\![\mathbf{v}_j]\!]_2).
\end{aligned}$$

  Send isk_j to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit, which is the outcome of the experiment.

Lemma 6.8 ([ALS16,Wee17]). *Assuming the MDDH assumption (Assumption 2.1; true in GGM per Definition 6.1), there exists a succinct* selectively simulation-secure *IPFE scheme (Definitions 6.2 and 6.3). Its components have sizes*

$$\begin{aligned}
|\mathsf{impk}| &= k(k+1+N)|g_1|, & |\mathsf{imsk}| &= (k+1)N \cdot \Theta(\log_2 p),\\
|\mathsf{isk}| &= (k+1)|g_2|, & |\mathsf{ict}| &= (k+1+N)|g_1|,
\end{aligned}$$

where p is the modulus, k is the MDDH parameter (can be 1 in GGM), N is the dimension, and $|g_i|$ is the bit-length of an element of $\mathbb{G}_i$.

6.4 Computational Secret Sharing

Secret sharing schemes have been used extensively to construct ABE schemes. The seminal work of [GPSW06] and a long line of follow-up works ([LOS+10,OT10,LW12, Att16,KW19] to name a few) used *linear* secret sharing schemes to construct ABE

schemes in pairing groups. Its security notion is information-theoretic. The share size of such a scheme is equal to the smallest monotone span program [KW93] computing the policy. It is also known that functions computed by polynomial-size span programs are in NC [Bei96,Ber84,BDHM92,KW93,Mul87].

The works of [AY20,AWY20] introduced the notion of *nearly linear* secret sharing with *computational* security. The relaxations enabled greater expressiveness and better efficiency. Assuming LWE, such a scheme exists for all polynomial-size circuits [BGG+14,GV15,AY20,AWY20] and the shares are *succinct*, i.e., they only grow with the circuit depth, but not the circuit size. However, the scheme is only *selectively* secure. Furthermore, due to technical reasons, when combined with pairing to obtain ABE, it only applies to Boolean formulae (equivalent to 5-PBP).

This chapter follows the blueprint of [AY20,AWY20] for the notions of secret sharing schemes, but departs from them in three important aspects. First, we consider a different security notion, *adaptive non-annihilability*, which is incomparable[61] to selective pseudorandomness considered in [AY20,AWY20] and enables us to prove adaptive security of ABE. Second, we further relax the linearity requirement so that it could apply to KP-ABE for polynomial-size circuits. Third, we refine the syntax to separate encodings of input and function. This separation helps proving adaptive security for our CP-ABE scheme, in which we need to simulate input encodings before the function is known. Our secret sharing schemes for both variants achieve succinct share sizes.

Definition 6.4 (secret sharing). Let $\mathcal{F} = \{\mathcal{F}_{\lambda,\ell,\mathsf{param}}\}_{\lambda,\ell\in\mathbb{N},\mathsf{param}\in\mathsf{Params}_{\lambda,\ell}}$ be an ensemble of Boolean function families such that for all $\lambda, \ell \in \mathbb{N}$ and $\mathsf{param} \in \mathsf{Params}_{\lambda,\ell}$, every $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$ is a function mapping $\{0,1\}^\ell$ to $\{0,1\}$. A *secret sharing scheme* for $\mathcal{F}$ consists of four efficient algorithms.

- $\mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param})$ takes as input the input length and param. It outputs some public parameter pp.
- $\mathsf{ShareX}(\mathsf{pp})$ takes pp as input. It outputs $(1+2\ell)$ shares, $L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}$, and

[61] It is stronger in that it is adaptive, but weaker in that the shares are not necessarily pseudorandom.

some shared randomness r. For $\mathbf{x} \in \{0,1\}^\ell$, denote by $L^{\mathbf{x}}$ the set of shares $L_0, \{L_i^{\mathbf{x}[i]}\}_{i\in[\ell]}$.

- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, r)$ takes as input pp, some $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, a secret $\mu \in \{0,1\}$, and the shared randomness r (output by ShareX). It outputs a share L_f.
- $\mathsf{Recon}(\mathsf{pp}, f, \mathbf{x}, L_f, L^{\mathbf{x}})$ takes as input pp, f, the input $\mathbf{x} \in \{0,1\}^\ell$ to f, and the shares $L_f, L^{\mathbf{x}}$. It is supposed to recover μ if $f(\mathbf{x}) = 0$.[62]

The scheme must be *correct*, i.e., for all $\lambda, \ell \in \mathbb{N}$, $\mathsf{param} \in \mathsf{Params}_{\lambda,\ell}$, $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, $\mathbf{x} \in \{0,1\}^\ell$, $\mu \in \{0,1\}$ such that $f(\mathbf{x}) = 0$, it holds that

$$\Pr\left[\begin{array}{ll} & \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param}) \\ (L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} & \mathsf{ShareX}(\mathsf{pp}) \qquad : \mathsf{Recon}(\mathsf{pp}, f, \mathbf{x}, L_f, L^{\mathbf{x}}) = \mu \\ & L_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r) \end{array}\right] = 1.$$

Definition 6.5 (succinct shares). A secret sharing scheme (Definition 6.4) is *succinct* if the size of each share output by ShareX, ShareF is a fixed polynomial in $\lambda, |\mathsf{param}|$, independent of the length of $\mathbf{x}$ or the description size of f, i.e., $|L_f|, |L_0|, |L_i^b|$ are all $\mathrm{poly}(\lambda, |\mathsf{param}|)$, where $i \in [\ell]$, $b \in \{0,1\}$.[63]

While correctness and succinctness (Definitions 6.4 and 6.5) are defined similarly to those in [AWY20], our linearity and security notions are different. Furthermore, our secret sharing schemes for KP-ABE and CP-ABE require slightly different linearity and security properties, so we introduce their definitions separately with the respective constructions.

6.5 KP-ABE for Circuits with Constant-Size Keys

6.5.1 More Definitions for Secret Sharing

In our KP-ABE construction, we need a secret sharing scheme with two linearity properties. The first is a relaxation of the nearly linear reconstruction requirement

[62] We use $f(\mathbf{x}) = 0$ to express authorization.

[63] There are $(2 + 2|\mathbf{x}|)$ shares, so the total share size is linear in the length of $\mathbf{x}$.

in [AWY20]. Our relaxed version (Definition 6.6) only stipulates it to be linear in $\mathbf{L}_f$ (and possibly non-linear in $L^{\mathbf{x}}$). The second is an additional linearity requirement on ShareF.

Definition 6.6 (weakly nearly linear reconstruction). A secret sharing scheme (Definition 6.4) is *weakly nearly linear* if it satisfies the following requirements.

- The parameter has the format of $\mathsf{param} = (p, \dots)$ with p being a prime.
- $L_f = \mathbf{L}_f$ is a vector over $\mathbb{Z}_p$.
- There is an efficient coefficient-finding algorithm $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, L^{\mathbf{x}})$, taking as input pp, $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, some $\mathbf{x} \in \{0,1\}^\ell$ to f, and the shares $L^{\mathbf{x}}$. It outputs an affine function γ over $\mathbb{Z}_p$ and a noise bound 1^B. For all $\lambda, \ell \in \mathbb{N}$, $(p, \dots) = \mathsf{param} \in \mathsf{Params}_{\lambda,\ell}$, $\mathbf{x} \in \{0,1\}^\ell$, $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, $\mu \in \{0,1\}$ with $f(\mathbf{x}) = 0$, it holds that

$$\Pr\left[\begin{array}{rc} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param}) & \\ (L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}) & 4B+1 < p \text{ and} \\ \mathbf{L}_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r) & : \quad \gamma(\mathbf{L}_f) - \mu\lfloor p/2 \rceil \in [-B, B] \\ (\gamma, 1^B) \xleftarrow{\$} \mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, L^{\mathbf{x}}) & \end{array}\right] = 1.$$

A weakly nearly linear scheme is always correct. Given FindCoef, we let Recon call FindCoef to obtain γ, B and output the unique $\mu \in \{0,1\}$ satisfying

$$\gamma(\mathbf{L}_f) - \mu\lfloor p/2 \rceil \in [-B, B].$$

The constructed Recon is efficient and correct. Since Recon is implied by FindCoef, we will only specify FindCoef and omit Recon when constructing weakly nearly linear secret sharing schemes.

Definition 6.7 (linear function sharing). A secret sharing scheme (Definition 6.4) has *linear function sharing* if it satisfies the following requirements.

- The parameter has the format of $\mathsf{param} = (p, \dots)$ with p being a prime.
- The public parameter has the format of $\mathsf{pp} = (1^n, \dots)$.

- $r = \mathbf{s}$ is an n-dimensional vector over $\mathbb{Z}_p$.
- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{s})$ is deterministic and linear in $(\mu, \mathbf{s})$.

Security. We consider a new security notion called *non-annihilability*. Unlike [AWY20], which fixes the choice of policy f before Setup is run, we allow the adversary to adaptively choose f after seeing the public parameter pp and the input shares $L^{\mathbf{x}}$. Another difference is that instead of requiring all shares $(\mathbf{L}_f, L^{\mathbf{x}})$ to look random, we only require that efficient adversaries cannot find a non-trivial affine function (potentially dependent on $L^{\mathbf{x}}$) that evaluates to zero on $\mathbf{L}_f$. This notion suffices for the security proofs of our KP-ABE scheme.

Definition 6.8 (non-annihilability for $\mathbf{L}_f$). Given a secret sharing scheme (Definition 6.4) such that $\mathsf{param} = (p, \dots)$ with p being prime and $\mathbf{L}_f$ is a vector over $\mathbb{Z}_p$, the scheme is *adaptively non-annihilable for* $\mathbf{L}_f$ if every efficient adversary wins $\mathsf{Exp}^{\mathrm{f}}_{\mathrm{ann}}$ with only negligible probability, where $\mathsf{Exp}^{\mathrm{f}}_{\mathrm{ann}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $(1^\ell, \mathsf{param})$ from it, where $\mathsf{param} = (p, \dots)$. Run

$$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param}), \qquad \left(L_0, \{L_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r\right) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}),$$

  and send pp to $\mathcal{A}$.

- **Share.** Optionally, the adversary $\mathcal{A}$ submits an input $\mathbf{x} \in \{0,1\}^\ell$. Upon receiving it, send $L^{\mathbf{x}}$ to $\mathcal{A}$.

- **Challenge.** The adversary $\mathcal{A}$ outputs some $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, a message bit $\mu \in \{0,1\}$, and an affine function γ. Upon receiving them, run

$$\mathbf{L}_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r)$$

  to determine the outcome of the experiment. The adversary $\mathcal{A}$ wins if *i)* $f(\mathbf{x}) = 1$ or $\mathbf{x}$ was not submitted[64] and *ii)* γ is not the zero function and *iii)* $\gamma(\mathbf{L}_f) = 0$. Otherwise, $\mathcal{A}$ loses.

[64] In the conference version [LLL22a], the case without $\mathbf{x}$ is missing. However, it only affects the intermediate abstraction, not the security of ABE (see Footnote 65).

A secret sharing scheme is *selectively non-annihilable for* $\mathbf{L}_f$ if it satisfies the above with the change that the adversary must choose the input $\mathbf{x}$ before receiving pp.

6.5.2 Secret Sharing for Bounded-Depth Circuits from Adaptive LWE

We now construct a succinct secret sharing scheme, satisfying the above linearity and adaptive annihilability for bounded-depth circuits from small-secret adaptive LWE. Our construction is based on the attribute-based laconic function evaluation scheme of [QWW18], which itself is based on [BGG+14].

Construction 6.1 (secret sharing for circuits). Let (see Definition 6.4)

$$\mathsf{Params}_\ell = \left\{ (p, 1^d) \;\middle|\; p \text{ prime},\ p \geq p_0(\lambda),\ d \leq d_+(p) \right\},$$
$$\mathcal{F}_{\ell,p,1^d} = \left\{ \text{Boolean circuit } C : \{0,1\}^\ell \to \{0,1\} \text{ of depth at most } d \right\},$$

where p_0 is a minimum smaller modulus function and d_+ is a maximum depth bound function (both specified later). Let $(\mathsf{EvalC}, \mathsf{EvalCX})$ be the algorithms in Lemma 6.5. Our weakly nearly linear secret sharing scheme for $\mathcal{F}$ with succinct shares works as follows.

- $\mathsf{Setup}(1^\ell, p, 1^d)$ takes as input the input length ℓ, the smaller modulus p, and the maximum depth d. The algorithm picks appropriate n, B_{init}, q (specified later) and sets $m = n\lceil \log_2 q \rceil$ and $\sigma = B_{\mathsf{init}}/\sqrt{\lambda}$. It samples and sets

  $$\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n, \qquad \mathbf{A}_0, \mathbf{A}_{-1}, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \qquad \mathbf{B} = (\mathbf{A}_0, \mathbf{A}_{-1}, \mathbf{A}_1, \ldots, \mathbf{A}_\ell).$$

  The algorithm samples a random invertible matrix $\mathbf{Q} \in \mathbb{Z}_q^{n\times n}$ and outputs

  $$\mathsf{pp} = (1^n, p, 1^d, q, m, B_{\mathsf{init}}, \sigma, \mathbf{a}, \mathbf{B}, \mathbf{Q}).$$

- $\mathsf{ShareX}(\mathsf{pp})$ takes pp as input. It samples and sets[65]

  $$\mathbf{s} \xleftarrow{\$} \mathcal{D}^n_{\mathbb{Z},\sigma,\leq B_{\mathsf{init}}}, \qquad \mathbf{e}_0, \mathbf{e}_{-1}, \mathbf{e}_1, \ldots, \mathbf{e}_\ell \xleftarrow{\$} \mathcal{D}^m_{\mathbb{Z},\sigma,\leq B_{\mathsf{init}}},$$

[65] We prepend an extra bit $x_{-1} = 0$ to the input, which will be set to $x_{-1} = 1$ if $\mathbf{x}$ is not submitted during $\mathsf{Exp}^{\mathsf{f}}_{\mathsf{ann}}$ (see Footnote 64), ensuring that f' always evaluates to 1. The conference version [LLL22a] simply uses $\mathbf{H}_f$, which, for the trivial circuit of $f(\mathbf{x}) \equiv 0$, is the zero matrix, making $\mathbf{L}_f$ easily annihilable. However, such f is not a useful choice for the adversary against ABE security (as no $\mathbf{x}$ can be chosen without trivializing the experiment). This is merely an artifact of abstraction.

$$\mathbf{L}_0 = \left(\mathbf{s}^\intercal(\mathbf{A}_0 - \mathbf{QG}) + \mathbf{e}_0^\intercal,\ \mathbf{s}^\intercal\mathbf{A}_{-1} + \mathbf{e}_{-1}^\intercal\right),$$
$$\mathbf{L}_i^b = \mathbf{s}^\intercal(\mathbf{A}_i - b\mathbf{QG}) + \mathbf{e}_i^\intercal \qquad (\text{for } i \in [\ell], b \in \{0,1\}),$$

and outputs $\left(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, \mathbf{s}\right)$.

- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{s})$ takes as input pp, some $f \in \mathcal{F}_{\ell,p,1^d}$, a secret bit $\mu \in \{0,1\}$, and the shared randomness $\mathbf{s}$. The algorithm defines

$$f'(x_{-1}, \mathbf{x}) = x_{-1} \vee f(\mathbf{x}).$$

It runs $\mathbf{H}_{f'} \leftarrow \mathsf{EvalC}(\mathbf{B}, \mathbf{Q}, f')$ and sets and outputs

$$\mathbf{L}_f = \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil + \mu\lfloor p/2 \rceil.$$

The entries of $\mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a})$ are over $\mathbb{Z}_q$, the multiplication by p/q is done over $\mathbb{Z}$ by taking any representative, and $\mathbf{L}_f$ is over $\mathbb{Z}_p$ (invariant under choice of representative).

- $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}})$ takes as input pp, some $f \in \mathcal{F}_{\ell,p,1^d}$, some input $\mathbf{x} \in \{0,1\}^\ell$ to f, and the shares $\mathbf{L}^{\mathbf{x}}$. If $f(\mathbf{x}) = 1$, the algorithm outputs $\perp$ and terminates. Otherwise, it runs $\mathbf{H}_{f',0,\mathbf{x}} \leftarrow \mathsf{EvalCX}(\mathbf{B}, \mathbf{Q}, f', 0, \mathbf{x})$, where f' is as in ShareF. The algorithm defines

$$\gamma(\mathbf{L}_f) = \mathbf{L}_f - \lfloor \mathbf{L}^{\mathbf{x}}\mathbf{H}_{f',0,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil, \qquad B = (n+1)B_{\text{init}} + 1,$$

and outputs $(\gamma, 1^B)$.

Linearity Properties. The scheme indeed has linear function sharing (Definition 6.7) because ShareF is a deterministic linear function over $\mu, \mathbf{s}$ with coefficients $\lfloor p/2 \rceil$, $\lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil$.

Parameters. Recall that $0 < \rho < \frac{1}{2}$ is the exponent of LWE hardness (Assumption 6.2). In our application, we let $p_0 = 2^{\lambda^{O(1)} \cap \omega(\log \lambda)}$ be a super-polynomial lower bound of the

order of pairing groups, and $p \geq p_0$ is the actual order. We require (for correctness)[66]

$$q \geq (m+2)^{d+2} p, \qquad 4((n+1)B_{\text{init}} + 1) + 1 < p,$$

for which we can set (also considering security)

$$n = B_{\text{init}} = \left((d+2)(\rho^{-1}+1) + \log_2 p + \mathrm{O}(d)\right)^{2/\rho} \leq \mathrm{poly}(\lambda, d, \log p),$$
$$q = 2^{\Theta(n^\rho)}, \qquad d_+ \sim \frac{p^{\rho/4} - \log_2 p}{1+\rho^{-1}} \cdot \Theta(1).$$

Since p is super-polynomial, d_+ is super-polynomial in λ.

Efficiency and Succinctness. In Construction 6.1, the algorithm FindCoef is indeed efficient as B is polynomially bounded. The public parameter pp primarily consists of matrices $\mathbf{a}, \mathbf{B}, \mathbf{Q}$, whose total bit-length is $(\ell+1) \cdot \mathrm{poly}(\lambda, d)$. The share $\mathbf{L}_0$ is a vector in $\mathbb{Z}_q^{2m}$ and each of $\mathbf{L}_i^b$ is a vector in $\mathbb{Z}_q^m$, so each share is of size $\mathrm{poly}(\lambda, d)$. Lastly, $\mathbf{L}_f$ is a single element of $\mathbb{Z}_p$, so its size is $\mathrm{poly}(\lambda)$. Therefore, the scheme has succinct shares (Definition 6.5).

Furthermore, the size of $\mathbf{L}_f$ does not depend on d — this is stronger than required in Definition 6.5, because $|\mathsf{param}| \geq d$ and d_+ (the upper bound of d) is super-polynomial in λ.

Theorem 6.9 (¶). *Construction 6.1 is* weakly nearly linear *(Definition 6.6).*

Proof (Theorem 6.9). This amounts to showing that if $f(\mathbf{x}) = 0$, then $4B+1 < p$ and $\gamma(\mathbf{L}_f) = \mu\lfloor p/2 \rceil + e$ for some $e \in [-B, B]$. By the choice of n, B_{init}, we have

$$4B + 1 = 4((n+1)B_{\text{init}} + 1) + 1 < p.$$

By construction we have

$$\begin{aligned}
\gamma(\mathbf{L}_f) &= \mathbf{L}_f - \lfloor \mathbf{L}^{\mathbf{x}} \mathbf{H}_{f',0,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil \\
&= \mu\lfloor p/2 \rceil + \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_{f'} \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil \\
&\qquad - \left\lfloor \left(\mathbf{s}^\intercal(\mathbf{B} - (1, 0, \mathbf{x}^\intercal) \otimes \mathbf{Q}\mathbf{G}) + (\mathbf{e}_0^\intercal, \mathbf{e}_{-1}^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\right) \mathbf{H}_{f',0,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \right\rceil
\end{aligned}$$

[66] The bounds in [LLL22a] are different (and have typos, fixed here). In the conference version, it was implicit that the basic gates in a circuit are addition, subtraction, and multiplication constrained to $\{0, \pm 1\}$. In this dissertation, they are AND, OR, NOT. In addition, the construction is changed to accommodate Footnote 64 (see also Footnote 65).

$$\begin{aligned}
\text{(Lemma 6.5)} \quad &= \mu\lfloor p/2\rceil + \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q\rceil \\
&\quad - \Big\lfloor \big(\mathbf{s}^\intercal(\mathbf{B}\mathbf{H}_{f'} - \underbrace{f'(\mathbf{x}')}_{=0 \vee f(\mathbf{x})=0}\mathbf{Q}\mathbf{G})\mathbf{G}^{-1}(\mathbf{a}) + \underbrace{(\mathbf{e}_0^\intercal, \dots)\mathbf{H}_{f',0,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})}_{=e_f}\big) \cdot p/q \Big\rceil \\
&= \mu\lfloor p/2\rceil + \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q\rceil - \lfloor (\mathbf{s}^\intercal \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) + e_f) \cdot p/q\rceil.
\end{aligned}$$

Since $\mathbf{G}^{-1}(\mathbf{a}) \in \{0,1\}^m$, by Lemma 6.5, we have (note that f' is one level deeper than f)

$$|e_f| \le m \cdot \|\mathbf{H}_{f',0,\mathbf{x}}^\intercal\| \cdot \|(\mathbf{e}_0^\intercal, \mathbf{e}_{-1}^\intercal, \mathbf{e}_1^\intercal, \dots, \mathbf{e}_\ell^\intercal)^\intercal\| \le (m+2)^{d+2} B_{\text{init}}.$$

We can break a rounded sum into a sum of individually rounded terms, at the expense of some rounding errors:

$$\begin{aligned}
\lfloor (\mathbf{s}^\intercal \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) + e_f) \cdot p/q\rceil &= \lfloor \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q\rceil + \lfloor e_f \cdot p/q\rceil + e'', \quad \text{where } |e''| \le 1, \\
\lfloor \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q\rceil &= \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q\rceil + e_s, \quad \text{where } |e_s| \le n \cdot \|\mathbf{s}\| \le nB_{\text{init}}.
\end{aligned}$$

The bounds of e'', e_s are due to Lemmas 6.2 and 6.3. Combining the above,

$$\begin{aligned}
\gamma(\mathbf{L}_f) &= \mu\lfloor p/2\rceil + \mathbf{s}^\intercal \lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q\rceil - \lfloor (\mathbf{s}^\intercal \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) + e_f) \cdot p/q\rceil \\
&= \mu\lfloor p/2\rceil - (\underbrace{\lfloor e_f \cdot p/q\rceil + e'' + e_s}_{=e}).
\end{aligned}$$

By our choice of q,

$$|e| \le \lceil (m+2)^{d+2} B_{\text{init}} \cdot p/q\rceil + 1 + nB_{\text{init}} \le B. \qquad \square$$

6.5.3 Security Proof of Secret Sharing Scheme

Theorem 6.10 (¶). *Assuming small-secret adaptive LWE (Assumption 6.2), Construction 6.1 is* adaptively non-annihilable for $\mathbf{L}_f$ *(Definition 6.8).*

If we only assume plain (non-adaptive) LWE (implying small-secret LWE [ACPS09]), then Construction 6.1 (with $\mathbf{Q}$ always set to $\mathbf{I}$) is selectively non-annihilable for $\mathbf{L}_f$.

Proof (Theorem 6.10). Let $\mathcal{A}$ be an efficient adversary against the non-annihilability experiment for $\mathbf{L}_f$ of the scheme. Recall that $\mathcal{A}$ chooses $1^\ell, p, 1^d$, sees pp, chooses $\mathbf{x}$ (or not), and lastly f, μ, γ such that $f(\mathbf{x}) = 1$ (if $\mathbf{x}$ is chosen) and γ is not the zero

function. It wins if $\gamma(\mathbf{L}_f) = 0$. Let

$$\mathbf{x}' = \begin{cases} (0, \mathbf{x}^\intercal)^\intercal, & \text{if } \mathbf{x} \text{ is chosen;} \\ (1, \mathbf{0})^\intercal, & \text{otherwise;} \end{cases} \qquad \overline{\mathbf{L}}^\intercal = \mathbf{s}^\intercal(\mathbf{B} - (1, (\mathbf{x}')^\intercal) \otimes \mathbf{QG}) + (\mathbf{e}_0^\intercal, \mathbf{e}_{-1}^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal),$$

then $f'(\mathbf{x}') = 1$ because either $f(\mathbf{x}) = 1$ or $\mathbf{x}$ is not chosen, and $\overline{\mathbf{L}}^\intercal = \mathbf{L}^{\mathbf{x}}$ whenever $\mathbf{x}$ is chosen. We proceed to rewrite $\mathbf{L}_f$ in $\mathsf{Exp}^{\mathsf{f}}_{\mathsf{ann}}$. By Lemma 6.5,

$$\begin{aligned}
\mathbf{s}^\intercal \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) &= \mathbf{s}^\intercal(\mathbf{QG} + \mathbf{B}\mathbf{H}_{f'} - f'(\mathbf{x}')\mathbf{QG})\mathbf{G}^{-1}(\mathbf{a}) \\
&= \mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + \mathbf{s}^\intercal(\mathbf{B}\mathbf{H}_{f'} - f'(\mathbf{x}')\mathbf{QG})\mathbf{G}^{-1}(\mathbf{a}) \\
&= \mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + \mathbf{s}^\intercal(\mathbf{B} - (1, (\mathbf{x}')^\intercal) \otimes \mathbf{QG})\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a}) \\
&= (\mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e') + \big(\mathbf{s}^\intercal(\mathbf{B} - (1, (\mathbf{x}')^\intercal) \otimes \mathbf{QG}) + (\mathbf{e}_0^\intercal, \mathbf{e}_{-1}^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\big)\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a}) \\
&\qquad - e' - (\mathbf{e}_0^\intercal, \mathbf{e}_{-1}^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a}) \\
&= (\mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e') + \overline{\mathbf{L}}^\intercal\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a}) - e' - e_f,
\end{aligned}$$

where $e' \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma,\le B_{\mathsf{init}}}$ is a newly introduced noise and $e_f = (\mathbf{e}_0^\intercal, \ldots)\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a})$ is the homomorphic noise. Note that $|e_f| \le B_{\mathsf{init}} \cdot (m+2)^{d+1} \cdot m \le (m+2)^{d+2} B_{\mathsf{init}}$. We have

$$\begin{aligned}
\mathbf{L}_f &= \mu\lfloor p/2 \rceil + \mathbf{s}^\intercal\lfloor \mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil = \mu\lfloor p/2 \rceil + e_s + \lfloor \mathbf{s}^\intercal\mathbf{B}\mathbf{H}_{f'}\mathbf{G}^{-1}(\mathbf{a}) \cdot p/q \rceil \\
&= \mu\lfloor p/2 \rceil + e_s + \left\lfloor \left((\mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e') + \overline{\mathbf{L}}^\intercal\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a}) - e' - e_f\right) \cdot p/q \right\rceil \\
&= \mu\lfloor p/2 \rceil + \underbrace{e_s + e'' + \lfloor (-e' - e_f) \cdot p/q \rceil}_{=\widetilde{e}} + \left\lfloor \left((\mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e') + \overline{\mathbf{L}}^\intercal\mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a})\right) \cdot p/q \right\rceil
\end{aligned}$$

where $|e_s| \le n \cdot \|\mathbf{s}\| \le nB_{\mathsf{init}}$ (Lemma 6.3) and $|e''| \le 1$ (Lemma 6.2). Furthermore,

$$\left|\lfloor (-e' - e_f) \cdot p/q \rceil\right| \le (B_{\mathsf{init}} + |e_f|) \cdot p/q + 1 \le B_{\mathsf{init}} \cdot 1 + |e_f| \cdot p/q + 1 \le 2B_{\mathsf{init}} + 1$$

by our choice of q. Therefore, $|\widetilde{e}| \le (n+2)B_{\mathsf{init}} + 2$.

Let $B' = (n+2)B_{\mathsf{init}} + 2$. With the above, we are ready to proceed to the hybrids.[67] For hybrid H_i, we write ε_i for $\Pr[\mathcal{A} \text{ wins in } \mathsf{H}_i]$.

- H_1 is $\mathsf{Exp}^{\mathsf{f}}_{\mathsf{ann}}$ (Definition 6.8). Define

$$\gamma_{\widetilde{e}',\mu}(z) = \gamma(z + \widetilde{e}' + \mu\lfloor p/2 \rceil) \qquad \text{for } \widetilde{e}' \in \mathbb{Z},\ \mu \in \{0,1\},$$

[67] The proof is simplified from [LLL22a]. We no longer rely on the leakage simulation lemma.

then $\gamma_{\widetilde{e}',\mu}$ is not the zero function if and only if neither is γ. In H_1, the adversary wins if and only if $f(\mathbf{x}) = 1$ (or $\mathbf{x}$ is not chosen) and $\gamma_{\widetilde{e},\mu}$ is not the zero function and

$$\gamma_{\widetilde{e},\mu}\left(\left\lfloor \left((\mathbf{s}^\intercal \mathbf{Q}\mathbf{a} + e') + \overline{\mathbf{L}}^\intercal \mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a})\right) \cdot p/q \right\rceil\right) = 0.$$

The condition above is a rewritten form of $\gamma(\mathbf{L}_f) = 0$.

- H_2 proceeds identically to H_1 except that $\mathcal{A}$ wins if and only if
  - $f(\mathbf{x}) = 1$ or $\mathbf{x}$ is not chosen,
  - for all $\widetilde{e}' \in [-B', B']$, the function $\gamma_{\widetilde{e}',\mu}$ is not the zero function, and
  - there exists $\widetilde{e}' \in [-B', B']$ such that

$$\gamma_{\widetilde{e}',\mu}\left(\left\lfloor \left((\mathbf{s}^\intercal \mathbf{Q}\mathbf{a} + e') + \overline{\mathbf{L}}^\intercal \mathbf{H}_{f',\mathbf{x}'}\mathbf{G}^{-1}(\mathbf{a})\right) \cdot p/q \right\rceil\right) = 0.$$

  The first condition is the same in H_1 and H_2, the second condition equivalent in both, and the third is relaxed in H_2. Therefore, $\varepsilon_2 \geq \varepsilon_1$.

- H_3 proceeds identically to H_2, except that $\overline{\mathbf{L}}^\intercal$ and $(\mathbf{s}^\intercal \mathbf{Q}\mathbf{a} + e')$ are replaced by random.

  Claim 6.11 (¶). Under small-secret adaptive LWE (Assumption 6.2), $\mathsf{H}_2 \approx \mathsf{H}_3$.

- H_4 proceeds identically to H_3, except that the third condition is changed as follows. Sample $c' \xleftarrow{\$} \mathbb{Z}_p$ and check that there exists $\widetilde{e}' \in [-B', B']$ such that

$$\gamma_{\widetilde{e}',\mu}(c') = 0.$$

  Claim 6.12 (¶). $\mathsf{H}_3 \approx_s \mathsf{H}_4$.

We claim the following.

Claim 6.13 (¶). ε_4 is negligible.

By hybrid argument, ε_2 is also negligible, and hence so is ε_1 (due to $0 \leq \varepsilon_1 \leq \varepsilon_2$). □

Proof (Claim 6.11). Consider the following adversary $\mathcal{B}$ against small-secret adaptive LWE (Assumption 6.2).

- $\mathcal{B}$ launches $\mathcal{A}()$ and receives $(1^\ell, p, 1^d)$ from it. It chooses $n, B_{\text{init}}, q, m, \sigma$ as in Setup, sets $m' = 1$, and sends $(1^{\ell+1}, 1^{m'})$ to the small-secret adaptive LWE experiment.

- $\mathcal{B}$ receives $(\mathbf{A}, \mathbf{B}, \mathbf{Q})$. It sets $\mathbf{a} = \mathbf{Q}^{-1}\mathbf{A}$ and sends the assembled pp to $\mathcal{A}$.

- If $\mathcal{B}$ receives $\mathbf{x} \in \{0,1\}^\ell$ from $\mathcal{A}$, it sends $\mathbf{x}' = (1, \mathbf{x}^\intercal)^\intercal$ to its own experiment, and receives

$$\begin{aligned} \mathbf{c}^\intercal &= \text{either} \quad \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal \quad \text{or random,} \\ \mathbf{d}^\intercal &= \text{either } \mathbf{s}^\intercal(\mathbf{B} - (1, (\mathbf{x}')^\intercal) \otimes \mathbf{Q}\mathbf{G}) + \mathbf{f}^\intercal \text{ or random.} \end{aligned}$$

  The adversary $\mathcal{B}$ sets $\overline{\mathbf{L}} = \mathbf{d}$ (implicitly, $\mathbf{f}^\intercal = (\mathbf{e}_0^\intercal, \mathbf{e}_{-1}^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)$). It sends $\mathbf{L}^{\mathbf{x}} = \overline{\mathbf{L}}^\intercal$ to $\mathcal{A}$.

- $\mathcal{B}$ receives f, μ, γ from $\mathcal{A}$. If $\mathcal{A}$ did not choose $\mathbf{x}$, the adversary $\mathcal{B}$ sends $\mathbf{x}' = (1, \mathbf{0})^\intercal$ to its own experiment, receives $(\mathbf{c}, \mathbf{d})$, and sets $\overline{\mathbf{L}} = \mathbf{d}$. It sets f' from f, then checks whether $f(\mathbf{x}) = 1$ (or $\mathbf{x}$ is not chosen) and γ is not the zero function and there exists $\widetilde{e}' \in [-B', B']$ such that

$$\gamma_{\widetilde{e}',\mu}\left(\left\lfloor (\mathbf{c}^\intercal + \overline{\mathbf{L}}^\intercal \mathbf{H}_{f',\mathbf{x}'} \mathbf{G}^{-1}(\mathbf{a})) \cdot p/q \right\rceil\right) = 0.$$

  If so, $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

The adversary $\mathcal{B}$ is efficient, thanks to B' being polynomially bounded. Moreover, $\mathcal{B}$ simulates $\mathcal{A}$ in H_{2+b} when it is in $\mathsf{Exp}^b_{\text{sALWE}}$. In particular, when $b = 0$, it implicitly sets $e' = \mathbf{e}$ and indeed $\mathbf{c}^\intercal = \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal = \mathbf{s}^\intercal\mathbf{Q}\mathbf{a} + e'$. This completes the reduction. □

Proof (Claim 6.12). Conditioned on pp, which fixes p, q, we have that $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_q$ is (conditionally) independent of everything else in the check whether there exists $\widetilde{e}' \in [-B', B']$ such that

$$\gamma_{\widetilde{e}',\mu}\left(\left\lfloor (\mathbf{c}^\intercal + \overline{\mathbf{L}}^\intercal \mathbf{H}_{f',\mathbf{x}'} \mathbf{G}^{-1}(\mathbf{a})) \cdot p/q \right\rceil\right) = 0.$$

Therefore, so is the quantity $(\mathbf{c}^\intercal + \overline{\mathbf{L}}^\intercal \mathbf{H}_{f',\mathbf{x}'} \mathbf{G}^{-1}(\mathbf{a}))$. By Lemma 6.4, replacing

$$\left\lfloor (\mathbf{c}^\intercal + \overline{\mathbf{L}}^\intercal \mathbf{H}_{f',\mathbf{x}'} \mathbf{G}^{-1}(\mathbf{a})) \cdot p/q \right\rceil$$

by an independent $c' \xleftarrow{\$} \mathbb{Z}_p$ incurs a statistical error no more than p/q. Our choice of $q = 2^{\Theta(n^\rho)}$ with $n \geq \log_2^{2/\rho} p$ satisfies $q \geq p^2$, so $p/q \leq 1/p \leq 1/p_0$. By the law of total expectation (to remove the conditioning of pp), it follows that $|\varepsilon_3 - \varepsilon_4| \leq 1/p_0$. □

Proof (Claim 6.13). Conditioned on pp (fixing p), by a union bound (enumerating $\widetilde{e}'$) over Schwartz–Zippel (Lemma 6.1) for degree-1 polynomials,

$$\Pr[\mathcal{A} \text{ wins in } \mathsf{H}_4 \mid \mathsf{pp}] \leq \frac{2B'+1}{p} \leq \frac{2B'+1}{p_0}.$$

By the law of total expectation, $\varepsilon_4 \leq (2B'+1)/p_0$, which is negligible. □

6.5.4 Construction of KP-ABE Scheme

Ingredients of Construction 6.2. Let

- GroupGen be a pairing group sampler (Definition 2.5),
- KPSS = (KPSS.Setup, KPSS.ShareX, KPSS.ShareF, KPSS.FindCoef) the weakly nearly linear secret sharing scheme in Construction 6.1, with succinct shares and linear function sharing (Definitions 6.4, 6.5, 6.6, and 6.7), and
- IPFE = (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec) an IPFE scheme based on GroupGen (Definition 6.2).

Construction 6.2 (KP-ABE). Let (see Definition 2.3)

$$\begin{aligned}
\mathsf{Params} &= \left\{ (1^\ell, 1^d) \;\middle|\; \ell, d \in \mathbb{N}_+,\ d \leq d_+(\lambda) \right\},\\
F_{1^\ell,1^d} &= \left\{ \text{circuit } f \;\middle|\; f : \{0,1\}^\ell \to \{0,1\} \text{ is of depth at most } d \right\},\\
X_{1^\ell,1^d} &= \{0,1\}^\ell, \qquad\qquad P_{1^\ell,1^d}(f, \mathbf{x}) = \neg f(\mathbf{x}),
\end{aligned}$$

where d_+ is a super-polynomial upper bound on the maximum depth.[68] Our KP-ABE scheme works as follows.

- $\mathsf{Setup}(1^\ell, 1^d)$ takes as input the attribute length ℓ and the maximum depth d. It runs and sets

$$(p, \dots) = \mathsf{gp} \xleftarrow{\$} \mathsf{GroupGen}(),$$

[68] $d_+(\lambda) = \min_p d'_+(p)$, where p runs through all possible primes produced by GroupGen and d'_+ is the d_+ of KPSS (Construction 6.1).

$$(1^n, \dots) = \mathsf{KPSS.pp} \xleftarrow{\$} \mathsf{KPSS.Setup}(1^\ell, p, 1^d),$$
$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^N) \qquad \text{for dimension } N = n+1,$$
$$\mathsf{mpk} = (\mathsf{gp}, \mathsf{KPSS.pp}, \mathsf{impk}), \qquad \mathsf{msk} = (\mathsf{mpk}, \mathsf{imsk}).$$

The algorithm outputs $(\mathsf{mpk}, \mathsf{msk})$.

- $\mathsf{KeyGen}(\mathsf{msk}, f)$ takes as input msk and a circuit f. Since KPSS has linear function sharing (Definition 6.7), the function $\mathsf{KPSS.ShareF}(\mathsf{KPSS.pp}, f, \sqcup, \sqcup)$ is linear in $(\mu, \mathbf{s})$, where $\mathbf{s}$ is n-dimensional. The algorithm KeyGen efficiently finds its coefficients $\mathbf{c} \in \mathbb{Z}_p^{n+1}$. It samples and runs

$$\delta \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}, \qquad \mathsf{isk} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta \mathbf{c}]\!]_2).$$

  The algorithm outputs $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk})$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ takes as input mpk, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$. It runs

$$(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i \in [\ell]}^{b \in \{0,1\}}, \mathbf{s}) \xleftarrow{\$} \mathsf{KPSS.ShareX}(\mathsf{KPSS.pp}), \quad \mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![(\mu, \mathbf{s}^\intercal)^\intercal]\!]_1).$$

  The algorithm outputs $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict})$.

- $\mathsf{Dec}(\mathsf{mpk}, f, \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ takes as input mpk, sk with its policy f, and ct with its attribute $\mathbf{x}$. If $f(\mathbf{x}) \neq 0$, the algorithm outputs $\perp$ and terminates. Otherwise, it parses sk, ct as defined in KeyGen, Enc, recomputes $\mathbf{c}$ as in KeyGen, and runs

$$L' \xleftarrow{\$} \mathsf{IPFE.Dec}(\mathsf{impk}, [\![\delta]\!]_2 \cdot \mathbf{c}, \mathsf{isk}, \mathsf{ict}),$$
$$(\gamma, 1^B) \xleftarrow{\$} \mathsf{KPSS.FindCoef}(\mathsf{KPSS.pp}, f, \mathbf{x}, \mathbf{L}^{\mathbf{x}}).$$

  The algorithm reads the coefficients $\gamma_1, \gamma_0 \in \mathbb{Z}_p$ of γ (so that $\gamma(\mathbf{L}_f) = \gamma_1 \mathbf{L}_f + \gamma_0$). It finds the unique $\mu' \in \{0,1\}$ and $e \in [-B, B]$ such that

$$\gamma_1 L' + [\![\gamma_0]\!]_1 [\![\delta]\!]_2 = [\![\mu' \lfloor p/2 \rceil + e]\!]_1 [\![\delta]\!]_2,$$

  The algorithm outputs this μ'.

Correctness. By the correctness of IPFE and the linear function sharing of KPSS,

$$L' = [\![(\mu, \mathbf{s}^\top) \cdot \delta\mathbf{c}]\!]_T = [\![\delta \mathsf{KPSS.ShareF}(\mathsf{KPSS.pp}, f, \mu, \mathbf{s})]\!]_T = [\![\delta \mathbf{L}_f]\!]_T.$$

Therefore,

$$\gamma_1 L' + [\![\gamma_0]\!]_1 [\![\delta]\!]_2 = \gamma_1 [\![\delta \mathbf{L}_f]\!]_T + [\![\gamma_0]\!]_1 [\![\delta]\!]_2 = [\![\gamma(\mathbf{L}_f)]\!]_1 [\![\delta]\!]_2.$$

By the correctness of KPSS and due to $\delta \neq 0$, the decryption algorithm outputs $\mu' = \mu$.

Efficiency. By Lemma 6.8, when instantiated with MDDH_1 and $N = n + 1 = \mathrm{poly}(\lambda, d)$, the IPFE components have bit-lengths

$$|\mathsf{impk}|, |\mathsf{imsk}|, |\mathsf{ict}| = \mathrm{poly}(\lambda, d), \qquad |\mathsf{isk}| = \mathrm{poly}(\lambda).$$

Notably, the pairing group is sampled independently of d, and isk consists of two elements of $\mathbb{G}_2$. Recall that KPSS components have bit-lengths

$$|\mathsf{KPSS.pp}| = (\ell + 1)\,\mathrm{poly}(\lambda, d), \qquad |\mathbf{L}_0|, |\mathbf{L}_i^b| = \mathrm{poly}(\lambda, d), \qquad |\mathbf{L}_f| = \mathrm{poly}(\lambda).$$

In Construction 6.2,

- mpk consists of KPSS.pp and impk, of total size $|\mathsf{mpk}| = (\ell + 1)\,\mathrm{poly}(\lambda, d)$,
- sk consists of a single isk and $[\![\delta]\!]_2$, of total size $|\mathsf{sk}| = \mathrm{poly}(\lambda)$ (independent of d, i.e., constant-size; concretely, three elements of $\mathbb{G}_2$), and
- ct consists of a single ict and $(\ell + 1)$ shares, of total size $|\mathsf{ct}| = (\ell + 1)\,\mathrm{poly}(\lambda, d)$.

6.5.5 Security of KP-ABE Scheme

Security. We state and prove the security of Construction 6.2.

Theorem 6.14 (¶). *Suppose* IPFE *is* selectively simulation-secure *(Definition 6.3) and* KPSS *is* adaptively non-annihilable for $\mathbf{L}_f$ *(Definition 6.8), then Construction 6.2 is* secure *(Definition 2.3) in GGM (Definition 6.1).*

We remark that if KPSS is selectively non-annihilable for $\mathbf{L}_f$, then the resultant KP-ABE achieves selective security in GGM.

Proof (Theorem 6.14). Recall that in $\mathsf{Exp}^{\mu}_{\mathrm{ABE}}$, the adversary adaptively chooses f_j's to receive sk_j's, both before and after submitting the challenge attribute $\mathbf{x}$ and being given ct encrypting $\mu \in \{0,1\}$. Let $(\mathsf{IPFE.SimSetup}, \mathsf{IPFE.SimKeyGen}, \mathsf{IPFE.SimEnc})$ be the simulator of IPFE. Consider the following hybrids.

- H^{μ}_1 is $\mathsf{Exp}^{\mu}_{\mathrm{ABE}}$. More concretely,

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^{n+1}), \qquad \mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![(\mu, \mathbf{s}^\intercal)^\intercal]\!]_1),$$
$$\delta_j \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}, \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta_j \mathbf{c}_j]\!]_2),$$

  and $\mathsf{sk}_j = ([\![\delta_j]\!]_2, \mathsf{isk}_j)$ and $\mathsf{ct} = (\mathbf{L}^{\mathbf{x}}, \mathsf{ict})$.

- In H^{μ}_2, the δ_j's are no longer conditioned to be non-zero, i.e.,

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^{n+1}), \qquad \mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![(\mu, \mathbf{s}^\intercal)^\intercal]\!]_1),$$
$$\delta_j \xleftarrow{\$} \boxed{\mathbb{Z}_p}, \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta_j \mathbf{c}_j]\!]_2).$$

  We have $\mathsf{H}^{\mu}_1 \approx_{\mathrm{s}} \mathsf{H}^{\mu}_2$.

- In H^{μ}_3, the IPFE components become simulated, i.e.,

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \boxed{\mathsf{IPFE.SimSetup}}(\mathsf{gp}, 1^{n+1}), \qquad \mathsf{ict} \xleftarrow{\$} \boxed{\mathsf{IPFE.SimEnc}(\mathsf{imsk})},$$
$$\delta_j \xleftarrow{\$} \mathbb{Z}_p, \qquad \mathsf{isk}_j \xleftarrow{\$} \boxed{\mathsf{IPFE.SimKeyGen}}(\mathsf{imsk}, [\![\delta_j \mathbf{c}_j]\!]_2, \boxed{[\![\delta_j \mathbf{L}_{f_j}]\!]_2}).$$

  Since the IPFE plaintext vector is $(\mu, \mathbf{s}^\intercal)$, independent of any adversarial choice (except n), the selective simulation security (Definition 6.3) of IPFE applies. Moreover, the purported inner products are indeed correct,

$$(\mu, \mathbf{s}^\intercal) \cdot \delta_j \mathbf{c}_j = \delta_j \mathbf{L}_{f_j},$$

  because by definition, $\mathbf{c}_j$ is the coefficients of $(\mu, \mathbf{s}^\intercal) \mapsto \mathbf{L}_{f_j}$. Therefore, $\mathsf{H}^{\mu}_2 \approx \mathsf{H}^{\mu}_3$.

Let's consider the following experiment G^{μ}_1 in GGM (Definition 6.1).

- **Setup.** Launch $\mathcal{A}()$ and receive from it $(1^\ell, 1^d)$. Let p be the pairing group order. Run

$$\mathsf{KPSS.pp} \xleftarrow{\$} \mathsf{KPSS.Setup}(1^\ell, p, 1^d), \quad (\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, \mathbf{s}) \xleftarrow{\$} \mathsf{KPSS.ShareX}(\mathsf{KPSS.pp}),$$

  and send $(p, \mathsf{KPSS.pp})$ to $\mathcal{A}$.

- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ submits a circuit $f_j : \{0,1\}^\ell \to \{0,1\}$ of depth no more than d. Sample and run

$$\delta_j \xleftarrow{\$} \mathbb{Z}_p, \qquad \mathbf{L}_{f_j} \xleftarrow{\$} \mathsf{KPSS.ShareF}(\mathsf{pp}, f_j, \mu, \mathbf{s}),$$

  and send handles representing $(\delta_j, \delta_j \mathbf{L}_{f_j})$ in $\mathbb{G}_2$ to $\mathcal{A}$.

- **Challenge.** The adversary $\mathcal{A}$ chooses $\mathbf{x} \in \{0,1\}^\ell$. Send $\mathbf{L}^{\mathbf{x}}$ to $\mathcal{A}$.

- **Query II.** Same as Query I.

- **Guess.** The adversary outputs a bit μ'. The outcome of the experiment is μ' if $f_j(\mathbf{x}) = 1$ for all queried f_j's. Otherwise, the outcome is set to 0.

Clearly, H_3^μ in GGM is just G_1^μ. Let γ_k denote the k^{th} GGM query, then

$$\gamma_k(\{\delta_j, \delta_j \mathbf{L}_{f_j}\}_j) = \gamma_{k,0} + \sum_j \delta_j \gamma_{k,j}(\mathbf{L}_{f_j}),$$

where $\gamma_{k,j}$'s are affine functions whose coefficients can be read out from those of γ and $\gamma_{k,0}$ is the constant term of γ. Consider the following hybrids.

- In G_1^μ, each GGM query is answered as whether it evaluates to zero.

- In G_2^μ, each GGM query γ_k is answered as zero if and only if $\gamma_{k,j}(\mathbf{L}_{f_j})$ evaluates to zero for all j and $\gamma_{k,0} = 0$. By Schwartz–Zippel (Lemma 6.1) for degree-1 polynomials (δ_j's being the indeterminates), we have $\mathsf{G}_1^\mu \approx_{\mathsf{s}} \mathsf{G}_2^\mu$. (The statistical error is K/p, where K is the maximum number of zero-test queries.)

- In G_3^μ, each GGM query γ_k is answered as zero if and only if $\gamma_{k,j}(\mathbf{L}_{f_j})$ is the zero function for all j and $\gamma_{k,0} = 0$.

We make the following claims.

Claim 6.15 (¶). $\mathsf{G}_2^\mu \approx \mathsf{G}_3^\mu$ *by the non-annihilability for* $\mathbf{L}_f$ *of* KPSS.

Claim 6.16 (¶). $\mathsf{G}_3^0 \equiv \mathsf{G}_3^1$.

By hybrid argument, we have $\mathsf{Exp}^0_{\mathrm{ABE}} \equiv \mathsf{H}_1^0 \approx \mathsf{H}_1^1 \equiv \mathsf{Exp}^1_{\mathrm{ABE}}$ in GGM. □

Proof (Claim 6.15). We use a guessing and coupling reduction. Fix $\mu \in \{0,1\}$. Let K, J be polynomial upper bounds of the number of zero-test queries and secret key queries made by $\mathcal{A}$, respectively. Couple one trial with $\mathcal{A}$ of G_2^μ and G_3^μ and augment them with a flag of bad event, which is set when $\gamma_{k,j}$ with the lowest (k,j), in lexicographical order, evaluates to zero (in G_2^μ) but is not the zero function (in G_3^μ). Clearly, the two trials are identical until the bad flag is set. It remains to prove $\Pr[\mathsf{bad}]$ is negligible.

Let $\mathcal{B}$ be the following efficient adversary against the (adaptive) non-annihilability for $\mathbf{L}_f$ of KPSS.

- $\mathcal{B}$ prepares a GGM oracle and samples $(k,j) \xleftarrow{\$} [K] \times [J]$. It launches $\mathcal{A}()$, receives $(1^\ell, 1^d)$ from $\mathcal{A}$, and sends $(1^\ell, p, 1^d)$ to its own security experiment, where p is the pairing group order of the GGM oracle prepared by $\mathcal{B}$ for $\mathcal{A}$.
- $\mathcal{B}$ receives KPSS.pp and sends $(p, \mathsf{KPSS.pp})$ to $\mathcal{A}$.
- $\mathcal{B}$ interacts with $\mathcal{A}$, until $\mathcal{A}$ submits the k^{th} zero-test query, as follows.
  - If $\mathcal{A}$ queries for a secret key, $\mathcal{B}$ records the function and hands out GGM handles.
  - If $\mathcal{A}$ submits a challenge $\mathbf{x}$, the adversary $\mathcal{B}$ forwards it to its own security experiment.
  - If $\mathcal{A}$ submits a zero-test query, $\mathcal{B}$ responds as in G_3^μ.
  - If $\mathcal{A}$ makes a guess, $\mathcal{B}$ gives up and terminates.
- If $\mathcal{A}$ submits the k^{th} zero-test query, $\mathcal{B}$ checks whether $\gamma_{k,j}$ has an out-of-range index. If not, it outputs $(f_j, \mu, \gamma_{k,j})$. Otherwise, it gives up and terminates.

Routine calculation shows $\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\mathsf{bad}]/KJ$, completing the reduction. □

Proof (Claim 6.16). We use induction to show that after t rounds of interaction in H_3^μ, the view of $\mathcal{A}$ does not depend on μ. The base case of $t = 0$ is clear, since the view of $\mathcal{A}$ prior to any interaction is just its random tape. For the inductive case, by the induction hypothesis, the next query submitted by $\mathcal{A}$ do not depend on μ, since it is computed from the view of $\mathcal{A}$ prior to this query. This fact is used in all of the cases analyzed below.

- If the interaction is **Setup**, **Challenge**, or **Guess**, then by definition the response does not depend on μ.
- If the interaction is **Query I/II**, then only handles are returned, which do not reveal any information about the encoded values (including μ). The number of handles returned is also independent of μ.
- If the interaction is a zero-test query, since the response is computed solely from the coefficients of γ_k, not encoded values inside the handles (including μ), the response also does not depend on μ.

This completes the induction. □

6.6 IPFE in GGM with Stronger Security

For our CP-ABE, we rely on IPFE with very strong simulation security — the only information that can be inferred from keys and ciphertexts is whether known affine functions over the inner products and the key vectors evaluate to zero or not. We define and realize such security in GGM.

Definition 6.9 (adaptive simulation of IPFE in GGM). A *simulator* for an IPFE scheme (Definition 6.2) in GGM consists of four efficient algorithms.

- $\mathsf{SimSetup}(1^\lambda, p, 1^N)$ takes as input the prime group order p and the dimension N. It outputs a simulated master public key impk and an initial state st.
- $\mathsf{SimKeyGen}(\mathsf{st})$ outputs a simulated key isk and an updated state st'.
- $\mathsf{SimEnc}(\mathsf{st})$ outputs a simulated ciphertext ict and an updated state st'.
- $\mathsf{SimGrpZT}^{\mathsf{IPZT}}(\mathsf{st}, \gamma)$ takes as input a (GGM) zero-test query γ, which is an affine function over all possible parings of the group handles created by earlier calls to the other simulation algorithms. It has oracle access to IPZT, which performs zero-tests of affine functions of the inner products and the vectors in the keys (described in more details later). The algorithm outputs an answer z to the query γ and an updated state st'.

The scheme is *adaptively simulation-secure in GGM* of order $\{p_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.1) if there exists a simulator such that $\mathsf{Exp}_{\mathrm{sim}}$ halts in polynomial time whenever $\mathsf{Exp}_{\mathrm{real}}$ does so and $\mathsf{Exp}_{\mathrm{real}} \approx \mathsf{Exp}_{\mathrm{sim}}$, where the experiments $\mathsf{Exp}_{\mathrm{real}}(1^\lambda, \mathcal{A})$ and $\mathsf{Exp}_{\mathrm{sim}}(1^\lambda, \mathcal{A})$ proceed as follows.

- **Setup.** Sample the group order $\mathsf{gp} = p \xleftarrow{\$} p_\lambda$, launch $\mathcal{A}(1^\lambda, p)$, and receive the dimension 1^N from it. Run

$$\begin{aligned} &(\text{in } \mathsf{Exp}_{\mathrm{real}}) && (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{gp}, 1^N), \\ &(\text{in } \mathsf{Exp}_{\mathrm{sim}}) && (\mathsf{impk}, \mathsf{st}) \xleftarrow{\$} \mathsf{SimSetup}(1^\lambda, p, 1^N), \end{aligned}$$

  and send impk to $\mathcal{A}$.

- **Challenge.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ has three choices.

  - It can submit a vector $\mathbf{v}_j$ encoded in $\mathbb{G}_2$ for a key. Upon receiving such a query, run

$$\begin{aligned} &(\text{in } \mathsf{Exp}_{\mathrm{real}}) && \mathsf{isk}_j \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}_j]\!]_2), \\ &(\text{in } \mathsf{Exp}_{\mathrm{sim}}) && (\mathsf{isk}_j, \mathsf{st}') \xleftarrow{\$} \mathsf{SimKeyGen}(\mathsf{st}), \end{aligned}$$

    and send isk_j to $\mathcal{A}$. In addition, in $\mathsf{Exp}_{\mathrm{sim}}$, update the state $\mathsf{st} \leftarrow \mathsf{st}'$.

  - It can submit a vector $\mathbf{u}_i$ encoded in $\mathbb{G}_1$ for a ciphertext. Upon receiving such a query, run

$$\begin{aligned} &(\text{in } \mathsf{Exp}_{\mathrm{real}}) && \mathsf{ict}_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}_i]\!]_1), \\ &(\text{in } \mathsf{Exp}_{\mathrm{sim}}) && (\mathsf{ict}_i, \mathsf{st}') \xleftarrow{\$} \mathsf{SimEnc}(\mathsf{st}), \end{aligned}$$

    and send ict_i to $\mathcal{A}$. In addition, in $\mathsf{Exp}_{\mathrm{sim}}$, update the state $\mathsf{st} \leftarrow \mathsf{st}'$.

  - It can submit a GGM zero-test query γ. In $\mathsf{Exp}_{\mathrm{real}}$, it is responded by the real GGM oracle. In $\mathsf{Exp}_{\mathrm{sim}}$, it is done as follows. Let $\{\mathbf{u}_i\}_{i\in[I]}, \{\mathbf{v}_j\}_{j\in[J]}$ be the lists of vectors in ciphertexts and keys queried up to this point. Let $\mathsf{IPZT}(\gamma')$ be a function that on input an affine function γ', outputs whether

$$\gamma'\left(\{\mathbf{v}_j\}_{j\in[J]}, \{\mathbf{u}_i^\mathsf{T}\mathbf{v}_j\}_{i\in[I],j\in[J]}\right)$$

evaluates to zero. Run

$$(z, \mathsf{st}') \xleftarrow{\$} \mathsf{SimGrpZT}^{\mathsf{IPZT}}(\mathsf{st}, \gamma),$$

update the state $\mathsf{st} \leftarrow \mathsf{st}'$, and return z to $\mathcal{A}$.

- **Guess.** The adversary $\mathcal{A}$ outputs a bit, which is the outcome of the experiment.

6.6.1 The ABDP Scheme

The IPFE scheme in [ABDP15] is only known to be selectively IND-CPA secure in the standard model. It turns out that the very same scheme satisfies the (much stronger) adaptive simulation security in GGM. We recall the construction below.

Construction 6.3 ([ABDP15]). The ABDP scheme works as follows.

- $\mathsf{Setup}(\mathsf{gp}, 1^N)$ reads the group order p from gp. It samples $a \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^N$, and outputs $\mathsf{impk} = [\![a, a\mathbf{w}^\top]\!]_1$, $\mathsf{imsk} = \mathbf{w}^\top$.
- $\mathsf{KeyGen}(\mathsf{imsk}, [\![\mathbf{v}]\!]_2)$ outputs $\mathsf{isk} = \mathbf{w}^\top [\![\mathbf{v}]\!]_2$.
- $\mathsf{Enc}(\mathsf{impk}, [\![\mathbf{u}]\!]_1)$ samples $s \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\mathsf{ict} = (s[\![a]\!]_1, s[\![a\mathbf{w}^\top]\!]_1 + [\![\mathbf{u}^\top]\!]_1)$.
- $\mathsf{Dec}(\mathsf{impk}, [\![\mathbf{v}]\!]_2, \mathsf{isk}, \mathsf{ict})$ outputs $\mathsf{ict} \cdot \begin{pmatrix} -\mathsf{isk} \\ [\![\mathbf{v}]\!]_2 \end{pmatrix}$.

Correctness is readily verified by $-sa \cdot \mathbf{w}^\top\mathbf{v} + (sa\mathbf{w}^\top + \mathbf{u}^\top) \cdot \mathbf{v} = \mathbf{u}^\top\mathbf{v}$. The scheme is succinct, as each secret key consists of only one group element, independent of N.

We remark that the above construction has two minor differences compared to the original scheme of [ABDP15]. One is that in the original version, key vectors are encoded in $\mathbb{Z}_p$ instead of any group $\mathbb{G}_2$ that can pair with $\mathbb{G}_1$. The other is that the master public key in [ABDP15] is simply $[\![\mathbf{w}]\!]_1$, without the random scalar a. As we will see later, the introduction of random scalar a makes our security analysis easier.

6.6.2 Stronger Security of the ABDP Scheme

Theorem 6.17 (¶). *Construction 6.3 is* adaptively simulation-secure in GGM *(Definition 6.9).*

Proof (Theorem 6.17). Since $\mathsf{impk}, \mathsf{isk}, \mathsf{ict}$ all consist of a fixed number of group elements, the simulator simply hands out the handles for them. Our simulator will keep counters $\mathsf{st} = (I, J)$ of the numbers of ciphertext and key queries as its state. Instead of specifying $\mathsf{SimGrpZT}$ upfront, we list the hybrids, and the simulator is just what the final hybrid does.

- H_1 is $\mathsf{Exp}_{\mathsf{real}}$, where the components are

$$\mathsf{impk} = [\![a, a\mathbf{w}^\intercal]\!]_1, \qquad \mathsf{isk}_j = [\![\mathbf{w}^\intercal \mathbf{v}_j]\!]_2, \qquad \mathsf{ict}_i = [\![s_i a, s_i a\mathbf{w}^\intercal + \mathbf{u}_i^\intercal]\!]_1.$$

  The zero-test queries are answered by the GGM oracle.

- In H_2, we change how impk and ict's are generated into

$$\mathsf{impk} = [\![\boxed{a_0}, \boxed{a_0}\,\mathbf{w}^\intercal]\!]_1, \qquad \mathsf{isk}_j = [\![\mathbf{w}^\intercal \mathbf{v}_j]\!]_2, \qquad \mathsf{ict}_i = [\![\boxed{a_i}, \boxed{a_i}\,\mathbf{w}^\intercal + \mathbf{u}_i^\intercal]\!]_1,$$

  where a_i's are independent and uniformly random over $\mathbb{Z}_p$. We have $\mathsf{H}_1 \approx_{\mathsf{s}} \mathsf{H}_2$. For convenience, we write $\mathbf{u}_0 = \mathbf{0}$, then impk is in the form of an ict for $\mathbf{u}_0$, and each zero-test query is (equivalently)

$$\gamma\Big(\big(1, \{a_i\}_{i\in[0,I]}, \{a_i\mathbf{w}^\intercal + \mathbf{u}_i^\intercal\}_{i\in[0,I]}\big) \otimes \big(1, \{\mathbf{v}_j\}_{j\in[J]}, \{\mathbf{w}^\intercal\mathbf{v}_j\}_{j\in[J]}\big)\Big),$$

  where γ is a *linear* function (note that 1 is included in the arguments). We omit the range of i, j below.

- In H_3, the zero-test queries are answered differently. Each zero-test query is regarded as a degree-3 polynomial with $\{a_i\}_i, \mathbf{w}$ being the indeterminates, i.e.,

$$\begin{aligned}
&\gamma\Big(\big(1, \{a_i\}_i, \{a_i\mathbf{w}^\intercal + \mathbf{u}_i^\intercal\}_i\big) \otimes \big(1, \{\mathbf{v}_j\}_j, \{\mathbf{w}^\intercal\mathbf{v}_j\}_j\big)\Big) \\
&\quad = 1 \cdot \gamma_0(\{\mathbf{u}_i\}_i, \{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}) \quad + \mathbf{w}^\intercal \cdot \boldsymbol{\gamma}_1(\{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}) \\
&\qquad + \sum_i a_i \cdot \gamma_{2,i}(\{\mathbf{v}_j\}_j) \quad + \sum_i a_i\mathbf{w}^\intercal \cdot \boldsymbol{\gamma}_{3,i}(\{\mathbf{v}_j\}_j) \quad + \sum_i a_i\mathbf{w}^\intercal \cdot \boldsymbol{\Gamma}_{4,i}(\{\mathbf{v}_j\}_j) \cdot \mathbf{w},
\end{aligned}$$

  where each entry of $\gamma_0, \boldsymbol{\gamma}_1, \{\gamma_{2,i}, \boldsymbol{\gamma}_{3,i}, \boldsymbol{\Gamma}_{4,i}\}_i$ is affine in their respective arguments and (see below) their coefficients can be read out from those of γ. The zero-test query is answered as "zero" if and only if $\gamma_0, \boldsymbol{\gamma}_1, \{\gamma_{2,i}, \boldsymbol{\gamma}_{3,i}\}_i$ all evaluate to zero and $\boldsymbol{\Gamma}_{4,i}^\intercal(\{\mathbf{v}_j\}_j) + \boldsymbol{\Gamma}_{4,i}(\{\mathbf{v}_j\}_j) = \mathbf{0}$ for all i.[69]

[69] Due to the symmetry of $\mathbf{w}$ and $\mathbf{w}^\intercal$ (quadratic form), the condition $\boldsymbol{\Gamma}_{4,i}^\intercal(\{\mathbf{v}_j\}_j) + \boldsymbol{\Gamma}_{4,i}(\{\mathbf{v}_j\}_j) = \mathbf{0}$ exactly says all coefficients of $a_i\mathbf{w}[u]\mathbf{w}[v]$ are zero (when $p \neq 2$, of course).

Since $a_i, \mathbf{w}$ are sampled at random, $\mathsf{H}_2 \approx_s \mathsf{H}_3$ by Schwartz–Zippel (Lemma 6.1) for degree-3 polynomials.

- H_4 is $\mathsf{Exp}_{\mathsf{sim}}$, which is mostly identical to H_3 except that the GGM zero-test queries are answered as follows. Each query γ is first dissected into $\gamma_0, \boldsymbol{\gamma}_1, \{\gamma_{2,i}, \boldsymbol{\gamma}_{3,i}, \boldsymbol{\Gamma}_{4,i}\}_i$ as in H_3. Next, consult IPZT to check whether $\{\gamma_{2,i}, \boldsymbol{\gamma}_{3,i}, \boldsymbol{\Gamma}_{4,i}^\intercal + \boldsymbol{\Gamma}_{4,i}\}_i$ evaluate to zero — note that these functions are affine over $\{\mathbf{v}_j\}_j$. If any of them evaluates to non-zero, the response to γ is "non-zero". Otherwise, convert $\gamma_0, \boldsymbol{\gamma}_1$ into equivalent queries affine over $\{\mathbf{v}_j\}_j, \{\mathbf{u}_i^\intercal \mathbf{v}_j\}_{i,j}$ (in particular, not over $\mathbf{u}_i$ or $\mathbf{u}_i \otimes \mathbf{v}_j$ like $\gamma_0, \boldsymbol{\gamma}_1$ themselves), then use IPZT again to determine the final response to γ.

  The efficient conversion is explained below (hence specifying $\mathsf{SimGrpZT}$), which yield answers identical to those in H_3, implying $\mathsf{H}_3 \equiv \mathsf{H}_4$.

Now, the conversion of $\gamma_0, \boldsymbol{\gamma}_1$. Writing out a zero-test query γ explicitly,

$$\begin{aligned}
&\gamma\Big(\big(1, \{a_i\}_i, \{a_i \mathbf{w}^\intercal + \mathbf{u}_i^\intercal\}_i\big) \otimes \big(1, \{\mathbf{v}_j\}_j, \{\mathbf{w}^\intercal \mathbf{v}_j\}_j\big)\Big) \\
&\quad = \alpha + \sum_i \big(b_i \cdot a_i + \mathbf{c}_i^\intercal \cdot (a_i \mathbf{w} + \mathbf{u}_i)\big) + \sum_j \big(d_j \cdot \mathbf{w}^\intercal \mathbf{v}_j + \mathbf{f}_j^\intercal \cdot \mathbf{v}_j\big) \\
&\qquad + \sum_{i,j} \big(g_{i,j} \cdot a_i \cdot \mathbf{w}^\intercal \mathbf{v}_j + \mathbf{h}_{i,j}^\intercal \cdot a_i \cdot \mathbf{v}_j + \mathbf{m}_{i,j}^\intercal \cdot (a_i \mathbf{w} + \mathbf{u}_i) \cdot \mathbf{w}^\intercal \mathbf{v}_j + (a_i \mathbf{w} + \mathbf{u}_i)^\intercal \cdot \mathbf{N}_{i,j} \cdot \mathbf{v}_j\big),
\end{aligned}$$

where $\alpha, b_i, \mathbf{c}_i, d_j, \mathbf{f}_j, g_{i,j}, \mathbf{h}_{i,j}, \mathbf{m}_{i,j}, \mathbf{N}_{i,j}$ are the coefficients. We remark that the last term $(a_i \mathbf{w} + \mathbf{u}_i)^\intercal \cdot \mathbf{N}_{i,j} \cdot \mathbf{v}_j$ enables arbitrary cross-pairing of $(a_i \mathbf{w} + \mathbf{u}_i)$ with $\mathbf{v}_j$ and combining the results. Each entry in $\mathbf{N}_{i,j}$ corresponds to the coefficient of one such pairing result. From these coefficients, we read out $\gamma_0, \boldsymbol{\gamma}_1, \{\gamma_{2,i}, \boldsymbol{\gamma}_{3,i}, \boldsymbol{\Gamma}_{4,i}\}_i$, which are

$$\gamma_0(\{\mathbf{u}_i\}_i, \{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}) = \alpha + \sum_i \mathbf{u}_i^\intercal \mathbf{c}_i + \sum_j \mathbf{f}_j^\intercal \mathbf{v}_j + \sum_{i,j} \mathbf{u}_i^\intercal \mathbf{N}_{i,j} \mathbf{v}_j, \tag{6.5}$$

$$\boldsymbol{\gamma}_1(\{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}) = \sum_j d_j \mathbf{v}_j + \sum_{i,j} \mathbf{v}_j \mathbf{m}_{i,j}^\intercal \mathbf{u}_i, \tag{6.6}$$

$$\gamma_{2,i}(\{\mathbf{v}_j\}_j) = b_i + \sum_j \mathbf{h}_{i,j}^\intercal \mathbf{v}_j, \tag{6.7}$$

$$\boldsymbol{\gamma}_{3,i}(\{\mathbf{v}_j\}_j) = \mathbf{c}_i + \sum_j \big(g_{i,j} \mathbf{v}_j + \mathbf{N}_{i,j} \mathbf{v}_j\big), \tag{6.8}$$

$$\boldsymbol{\Gamma}_{4,i}(\{\mathbf{v}_j\}_j) = \sum_j \mathbf{m}_{i,j} \mathbf{v}_j^\intercal. \tag{6.9}$$

Since γ_0 is tested only when $\{\boldsymbol{\gamma}_{3,i}\}_i$ all evaluate to zero, we substitute (6.8) into (6.5) and obtain

$$\begin{aligned}\gamma_0(\{\mathbf{u}_i\}_i, \{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}) &= \alpha + \sum_j \mathbf{f}_j^\intercal \mathbf{v}_j + \sum_i \mathbf{u}_i^\intercal \left(\mathbf{c}_i + \sum_j \mathbf{N}_{i,j}\mathbf{v}_j\right) \\ &= \alpha + \sum_j \mathbf{f}_j^\intercal \mathbf{v}_j + \sum_i \mathbf{u}_i^\intercal \left(\boldsymbol{\gamma}_{3,i}(\{\mathbf{v}_j\}_j) - \sum_j g_{i,j}\mathbf{v}_j\right) \\ &= \alpha + \sum_j \mathbf{f}_j^\intercal \mathbf{v}_j - \sum_{i,j} g_{i,j}\mathbf{u}_i^\intercal \mathbf{v}_j,\end{aligned}$$

which is affine in $\{\mathbf{v}_j\}_j, \{\mathbf{u}_i^\intercal \mathbf{v}_j\}_{i,j}$. Similarly, since $\boldsymbol{\gamma}_1$ is tested only when $\{\boldsymbol{\Gamma}_{4,i}^\intercal + \boldsymbol{\Gamma}_{4,i}\}_i$ all evaluate to zero, we substitute (6.9) into (6.6) and obtain

$$\begin{aligned}\boldsymbol{\gamma}_1(\{\mathbf{v}_j\}_j, \{\mathbf{u}_i \otimes \mathbf{v}_j\}_{i,j}) &= \sum_j d_j \mathbf{v}_j + \sum_i \left(\sum_j \mathbf{v}_j \mathbf{m}_{i,j}^\intercal\right)\mathbf{u}_i \\ &= \sum_j d_j \mathbf{v}_j + \sum_i \boldsymbol{\Gamma}_{4,i}^\intercal(\{\mathbf{v}_j\}_j) \cdot \mathbf{u}_i \\ &= \sum_j d_j \mathbf{v}_j - \sum_i \boldsymbol{\Gamma}_{4,i}(\{\mathbf{v}_j\}_j) \cdot \mathbf{u}_i \\ &= \sum_j d_j \mathbf{v}_j - \sum_i \left(\sum_j \mathbf{m}_{i,j}\mathbf{v}_j^\intercal\right)\mathbf{u}_i = \sum_j d_j \mathbf{v}_j - \sum_{i,j} \mathbf{m}_{i,j} \cdot \mathbf{v}_j^\intercal \mathbf{u}_i,\end{aligned}$$

which is again affine in $\{\mathbf{v}_j\}_j, \{\mathbf{u}_i^\intercal \mathbf{v}_j\}_{i,j}$. □

6.7 Doubly Succinct CP-ABE for Boolean Formulae

6.7.1 More Definitions for Secret Sharing

In our CP-ABE construction, we need a secret sharing scheme with the following linearity property, defined similarly to that of [AWY20].

Definition 6.10 (nearly linear reconstruction). A secret sharing scheme (Definition 6.4) is *nearly linear* if it satisfies the following requirements.

- The parameter has the format of $\mathsf{param} = (p, \dots)$ with p being a prime.
- $L_0 = \mathbf{L}_0$, $L_i^b = \mathbf{L}_i^b$, $L_f = \mathbf{L}_f$ are vectors over $\mathbb{Z}_p$.
- There is an efficient coefficient-finding algorithm $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x})$, taking as input pp, $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, and some $\mathbf{x} \in \{0,1\}^\ell$ to f. It outputs an affine function

γ over $\mathbb{Z}_p$ and a noise bound 1^B. For all $\lambda, \ell \in \mathbb{N}$, $(p, \dots) = \mathsf{param} \in \mathsf{Params}_{\lambda,\ell}$, $\mathbf{x} \in \{0,1\}^\ell$, $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, $\mu \in \{0,1\}$ with $f(\mathbf{x}) = 0$, it holds that

$$\Pr\left[\begin{array}{rl} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param}) & \\ (\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}) & 4B+1 < p \text{ and} \\ \mathbf{L}_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r) & : \quad \gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) - \mu\lfloor p/2 \rceil \in [-B, B] \\ (\gamma, 1^B) \xleftarrow{\$} \mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x}) & \end{array}\right] = 1.$$

Security. Unlike the case of KP-ABE, our CP-ABE requires non-annihilability jointly for $\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}$.

Definition 6.11 (joint non-annihilability). Given a secret sharing scheme (Definition 6.4) such that $\mathsf{param} = (p, \dots)$ with p being prime and $\mathbf{L}_0, \mathbf{L}_i^b, \mathbf{L}_f$ are vectors over $\mathbb{Z}_p$, the scheme is *adaptively jointly non-annihilable* if every efficient adversary wins $\mathsf{Exp}^{\mathrm{fx}}_{\mathrm{ann}}$ with only negligible probability, where $\mathsf{Exp}^{\mathrm{fx}}_{\mathrm{ann}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $(1^\ell, \mathsf{param})$ from it, where $\mathsf{param} = (p, \dots)$. Run

  $$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^\ell, \mathsf{param})$$

  and send pp to $\mathcal{A}$.

- **Challenge.** The adversary $\mathcal{A}$ outputs $f \in \mathcal{F}_{\lambda,\ell,\mathsf{param}}$, an input $\mathbf{x} \in \{0,1\}^\ell$, a message bit $\mu \in \{0,1\}$, and an affine function γ. Upon the challenge, run

  $$(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, r) \xleftarrow{\$} \mathsf{ShareX}(\mathsf{pp}), \qquad \mathbf{L}_f \xleftarrow{\$} \mathsf{ShareF}(\mathsf{pp}, f, \mu, r),$$

  to determine the outcome of the experiment. The adversary $\mathcal{A}$ wins if *i)* $f(\mathbf{x}) = 1$ and *ii)* γ is not the zero function and *iii)* $\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) = 0$. Otherwise, $\mathcal{A}$ loses.

A secret sharing scheme is *selectively jointly non-annihilable* if it satisfies the above with the change that the adversary must choose the input $\mathbf{x}$ before receiving pp.

6.7.2 Secret Sharing for Boolean Formulae from Adaptive LWE

We now construct a succinct secret sharing scheme, satisfying the above linearity and security definitions, for 5-PBP from adaptive LWE. Our scheme is based on the

attribute-based laconic function evaluation scheme in [QWW18] and we employ the techniques in [BGG+14,GV15] to control LWE noises.

Construction 6.4 (secret sharing for 5-PBP). Let (see Definition 6.4)

$$\mathsf{Params}_\ell = \left\{ p \,\middle|\, p \text{ prime},\ p \geq p_0(\lambda) \right\},$$
$$\mathcal{F}_{\ell,p} = \left\{ \text{5-PBP } f : \{0,1\}^\ell \to \{0,1\} \text{ of length at most } s_+(\lambda,\ell,p) \right\},$$

where p_0 is a minimum modulus function and s_+ is a maximum length bound function (both specified later). Let $(\mathsf{EvalF}, \mathsf{EvalFX})$ be the algorithms in Lemma 6.6. Our nearly linear secret sharing scheme for $\mathcal{F}$ with succinct shares works as follows.

- $\mathsf{Setup}(1^\ell, p)$ takes as input the input length ℓ and the modulus p. The algorithm picks appropriate n, B_{init} (specified later) and sets $m = n\lceil \log_2 p \rceil$ and $\sigma = B_{\mathrm{init}}/\sqrt{\lambda}$. It samples and sets

$$\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^n, \qquad \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \xleftarrow{\$} \mathbb{Z}_p^{n\times m}, \qquad \mathbf{B} = (\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell).$$

  It outputs $\mathsf{pp} = (1^n, p, m, B_{\mathrm{init}}, \sigma, \mathbf{a}, \mathbf{B})$.

- $\mathsf{ShareX}(\mathsf{pp})$ takes pp as input. It samples and sets

$$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^n, \qquad \mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_\ell \xleftarrow{\$} \mathcal{D}^m_{\mathbb{Z},\sigma,\leq B_{\mathrm{init}}},$$
$$\mathbf{L}_0 = \mathbf{s}^\intercal(\mathbf{A}_0 - \mathbf{G}) + \mathbf{e}_0^\intercal, \qquad \{\mathbf{L}_i^b = \mathbf{s}^\intercal(\mathbf{A}_i - b\mathbf{G}) + \mathbf{e}_i^\intercal\}_{i\in[\ell]}^{b\in\{0,1\}},$$

  and outputs $(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i\in[\ell]}^{b\in\{0,1\}}, \mathbf{s})$.

- $\mathsf{ShareF}(\mathsf{pp}, f, \mu, \mathbf{s})$ takes as input pp, some $f \in \mathcal{F}_{\ell,p}$, a secret bit $\mu \in \{0,1\}$, and the shared randomness $\mathbf{s}$. It runs $\mathbf{H}_f \leftarrow \mathsf{EvalF}(\mathbf{B}, f)$, and samples and sets

$$\bar{e} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma,\leq B_{\mathrm{init}}}, \qquad \mathbf{L}_f = \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) + \bar{e} + \mu\lfloor p/2 \rceil.$$

  The algorithm outputs $\mathbf{L}_f$.

- $\mathsf{FindCoef}(\mathsf{pp}, f, \mathbf{x})$ takes as input pp, some $f \in \mathcal{F}_{\ell,p}$, and some input $\mathbf{x} \in \{0,1\}^\ell$ to f. If $f(\mathbf{x}) = 1$, the algorithm outputs $\perp$ and terminates. Otherwise, it runs $\mathbf{H}_{f,\mathbf{x}} \leftarrow \mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x})$ and defines

$$\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) = \mathbf{L}_f - \mathbf{L}^{\mathbf{x}}\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}), \qquad B = (3m^2\ell_{\mathrm{BP}} + m + 1)B_{\mathrm{init}},$$

  where ℓ_{BP} is the length of f. The algorithm outputs $(\gamma, 1^B)$.

Parameters. Recall that $0 < \rho < \frac{1}{2}$ is the exponent of LWE hardness (Assumption 6.2). In our application, we let $p_0 = 2^{\lambda^{O(1)} \cap \omega(\log \lambda)}$ be a super-polynomial lower bound of the order of pairing groups, $p \geq p_0$ the actual order, and $q = p$ the LWE modulus. We require (for correctness)

$$4(3m^2 s_+ + m + 1)B_{\mathsf{init}} + 1 < p,$$

for which we can set (also considering security)

$$n, B_{\mathsf{init}} = \mathrm{poly}(\lambda), \qquad m = n\lceil \log_2 p \rceil, \qquad s_+ = \left\lfloor \frac{p - 2 - 4(m+1)B_{\mathsf{init}}}{12m^2 B_{\mathsf{init}}} \right\rfloor.$$

Note that $s_+ = \lambda^{\omega(1)}$, so the scheme is unbounded in terms of 5-PBP length (i.e., Boolean formula size).

Efficiency and Succinctness. In Construction 6.4, the algorithm $\mathsf{FindCoef}$ is indeed efficient as B is polynomially bounded. The public parameter pp primarily consists of matrices $\mathbf{a}, \mathbf{B}$, whose total bit-length is $(\ell + 1) \cdot \mathrm{poly}(\lambda)$. Each share is a vector in $\mathbb{Z}_p^m$ and $\mathbf{L}_f$ is a single element of $\mathbb{Z}_p$, so their sizes are $\mathrm{poly}(\lambda)$. This shows that the scheme has succinct shares (Definition 6.5).

Theorem 6.18 (¶). *Construction 6.4 is* nearly linear *(Definition 6.10).*

Proof (Theorem 6.18). This amounts to showing that if $f(\mathbf{x}) = 0$, then $4B + 1 < p$ and $\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) = \mu\lfloor p/2 \rceil + e$ for some $e \in [-B, B]$. By the choice of s_+, we have

$$4B + 1 = 4(3m^2 \ell_{\mathsf{BP}} + m + 1)B_{\mathsf{init}} + 1 \leq 4(3m^2 s_+ + m + 1)B_{\mathsf{init}} + 1 \leq p - 1 < p.$$

Letting $\mathbf{e}^\intercal = (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)$, by construction,

$$\begin{aligned}
\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) &= \mathbf{L}_f - \mathbf{L}^{\mathbf{x}} \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) \\
&= \left(\mathbf{s}^\intercal \mathbf{B} \mathbf{H}_f \mathbf{G}^{-1}(\mathbf{a}) + \bar{e} + \mu\lfloor p/2 \rceil\right) - \left((\mathbf{s}^\intercal(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal)\mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a})\right) \\
&= \mu\lfloor p/2 \rceil - \mathbf{e}^\intercal \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) + \bar{e} + \mathbf{s}^\intercal \underbrace{\left(\mathbf{B}\mathbf{H}_f - (\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{f,\mathbf{x}}\right)}_{= \mathbf{0}, \text{ since } f(\mathbf{x}) = 0} \mathbf{G}^{-1}(\mathbf{a}) \\
\text{(Lemma 6.6)} \quad &= \mu\lfloor p/2 \rceil - \mathbf{e}^\intercal \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) + \bar{e}.
\end{aligned}$$

Write $e = -\mathbf{e}^\intercal \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{a}) + \bar{e}$. Since $\mathbf{G}^{-1}(\mathbf{a}) \in \{0,1\}^m$, by Lemma 6.6,

$$|e| \leq B_{\mathsf{init}}(3m\ell_{\mathsf{BP}} + 1)m + B_{\mathsf{init}} = B. \qquad \square$$

6.7.3 Security Proof of Secret Sharing Scheme

Theorem 6.19 (¶). *Assuming adaptive LWE (Assumption 6.1), Construction 6.4 is* adaptively jointly non-annihilable *(Definition 6.11).*

If we only assume plain (non-adaptive) LWE, then Construction 6.4 is selectively jointly non-annihilable.

Proof (Theorem 6.19). The proof is similar to that of Theorem 6.10. Let $\mathcal{A}$ be an efficient adversary against the joint non-annihilability experiment of the scheme. Recall that $\mathcal{A}$ chooses $1^\ell, p,$ sees pp, chooses $f, \mathbf{x}, \mu, \gamma$ such that $f(\mathbf{x}) = 1$ and γ is not the zero function. It wins if $\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) = 0$.

Write $\mathbf{e}^\intercal = (\mathbf{e}_0^\intercal, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_\ell^\intercal)$, then

$$\begin{aligned}
\mathbf{L}_f &= \mathbf{s}^\intercal \mathbf{B}\mathbf{H}_f\mathbf{G}^{-1}(\mathbf{a}) + \overline{e} + \mu\lfloor p/2\rceil \\
\text{(since } f(\mathbf{x}) = 1) \quad &= \mathbf{s}^\intercal(\mathbf{B}\mathbf{H}_f - f(\mathbf{x})\mathbf{G})\mathbf{G}^{-1}(\mathbf{a}) + (\mathbf{s}^\intercal\mathbf{a} + \overline{e}) + \mu\lfloor p/2\rceil \\
\text{(Lemma 6.6)} \quad &= \mathbf{s}^\intercal(\mathbf{B} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) + (\mathbf{s}^\intercal\mathbf{a} + \overline{e}) + \mu\lfloor p/2\rceil \\
&= \mathbf{L}^{\mathbf{x}}\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) - \mathbf{e}^\intercal\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) + (\mathbf{s}^\intercal\mathbf{a} + \overline{e}) + \mu\lfloor p/2\rceil.
\end{aligned}$$

Let $\widetilde{e} = -\mathbf{e}^\intercal\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a})$ and $B' = B_{\mathsf{init}}(3m\ell_{\mathsf{BP}} + 1)m,$ then it follows from Lemma 6.6 that $|\widetilde{e}| \le B'$ is polynomially bounded. We are ready to proceed to the hybrids.[70] For hybrid H_i, we write ε_i for $\Pr[\mathcal{A} \text{ wins in } \mathsf{H}_i]$.

- H_1 is $\mathsf{Exp}^{\mathsf{fx}}_{\mathsf{ann}}$ (Definition 6.11). Define

  $$\gamma_{\widetilde{e}',\mu}(z, \mathbf{L}^{\mathbf{x}}) = \gamma(z + \widetilde{e}' + \mu\lfloor p/2\rceil, \mathbf{L}^{\mathbf{x}}) \qquad \text{for } \widetilde{e}' \in \mathbb{Z},\ \mu \in \{0,1\},$$

  then $\gamma_{\widetilde{e}',\mu}$ is not the zero function if and only if neither is γ. In H_1, the adversary wins if and only if $f(\mathbf{x}) = 1$ and $\gamma_{\widetilde{e},\mu}$ is not the zero function and

  $$\gamma_{\widetilde{e},\mu}\big(\mathbf{L}^{\mathbf{x}}\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) + (\mathbf{s}^\intercal\mathbf{a} + \overline{e}),\ \mathbf{L}^{\mathbf{x}}\big) = 0.$$

  The condition above is a rewritten form of $\gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}}) = 0$.

- H_2 proceeds identically to H_1 except that $\mathcal{A}$ wins if and only if

[70] The proof is simplified from [LLL22a]. We no longer rely on the leakage simulation lemma.

- $f(\mathbf{x}) = 1$,
- for all $\widetilde{e}' \in [-B', B']$, the function $\gamma_{\widetilde{e}',\mu}$ is not the zero function, and
- there exists $\widetilde{e}' \in [-B', B']$ such that

$$\gamma_{\widetilde{e}',\mu}\big(\mathbf{L}^{\mathbf{x}}\mathbf{H}_{f,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{a}) + (\mathbf{s}^{\mathsf{T}}\mathbf{a} + \overline{e}),\ \mathbf{L}^{\mathbf{x}}\big) = 0.$$

The first condition is the same in H_1 and H_2, the second condition equivalent in both, and the third is relaxed in H_2. Therefore, $\varepsilon_2 \geq \varepsilon_1$.

- H_3 proceeds identically to H_2, except that $\mathbf{L}^{\mathbf{x}}$ and $(\mathbf{s}^{\mathsf{T}}\mathbf{a} + \overline{e})$ are replaced by random. Under adaptive LWE (Assumption 6.1), $\mathsf{H}_2 \approx \mathsf{H}_3$.

- H_4 proceeds identically to H_3, except that the third condition is changed as follows. Sample $c' \xleftarrow{\$} \mathbb{Z}_p$ and check that there exists $\widetilde{e}' \in [-B', B']$ such that

$$\gamma_{\widetilde{e}',\mu}(c', \mathbf{L}^{\mathbf{x}}) = 0.$$

Clearly, $\mathsf{H}_3 \equiv \mathsf{H}_4$.

Lastly, by a union bound (enumerating $\widetilde{e}'$) over Schwartz–Zippel (Lemma 6.1) for degree-1 polynomials, ε_4 is negligible. By hybrid argument, ε_2 is also negligible, and hence so is ε_1 (due to $0 \leq \varepsilon_1 \leq \varepsilon_2$). □

6.7.4 Construction of CP-ABE Scheme

Ingredients of Construction 6.5. Let

- GroupGen be a pairing group sampler (Definition 2.5),
- CPSS = (CPSS.Setup, CPSS.ShareX, CPSS.ShareF, CPSS.FindCoef) the nearly linear secret sharing scheme in Construction 6.4, with succinct shares (Definitions 6.4, 6.5, and 6.10), and
- IPFE = (IPFE.Setup, IPFE.KeyGen, IPFE.Enc, IPFE.Dec) an IPFE scheme based on GroupGen with adaptive simulation security in GGM (Definitions 6.2 and 6.9).

Construction 6.5 (CP-ABE). Let (see Definition 2.3)

$$\begin{aligned}\mathsf{Params} &= \left\{ 1^\ell \;\middle|\; \ell \in \mathbb{N}_+ \right\}, & F_{1^\ell} &= \{0,1\}^\ell,\\ X_{1^\ell} &= \left\{ \text{5-PBP } f \;\middle|\; f : \{0,1\}^\ell \to \{0,1\} \text{ is of length at most } s_+ \right\},\\ P_{1^\ell}(\mathbf{x}, f) &= \neg f(\mathbf{x}),\end{aligned}$$

where s_+ is the super-polynomial bound on 5-PBP length of Construction 6.4. Our CP-ABE scheme works as follows.

- $\mathsf{Setup}(1^\ell)$ takes the attribute length ℓ as input. It runs and sets

  $$\begin{aligned}(p, \dots) = \mathsf{gp} &\xleftarrow{\$} \mathsf{GroupGen}(), & \mathsf{CPSS.pp} &\xleftarrow{\$} \mathsf{CPSS.Setup}(1^\ell, p),\\ (\mathsf{impk}, \mathsf{imsk}) &\xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^N) & &\text{for dimension } N = 2\ell + 1,\\ \mathsf{mpk} &= (\mathsf{gp}, \mathsf{CPSS.pp}, \mathsf{impk}), & \mathsf{msk} &= (\mathsf{mpk}, \mathsf{imsk}).\end{aligned}$$

  The algorithm outputs $(\mathsf{mpk}, \mathsf{msk})$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ takes as input msk and $\mathbf{x} \in \{0,1\}^\ell$. It samples $\delta \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}$ and defines the "selector vector" $\mathbf{v} \in \mathbb{Z}_p^{2\ell+1}$ as (recall that $\boldsymbol{\iota}$'s are the standard basis vectors)

  $$\mathbf{v} = \boldsymbol{\iota}_1 + \sum_{i \in [\ell]} \boldsymbol{\iota}_{2i + \mathbf{x}[i]}.$$

  The algorithm runs

  $$\mathsf{isk} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta \mathbf{v}]\!]_2)$$

  and outputs $\mathsf{sk} = ([\![\delta]\!]_2, \mathsf{isk})$.

- $\mathsf{Enc}(\mathsf{mpk}, f, \mu)$ takes as input mpk, a 5-PBP f, and a message μ. It runs

  $$\begin{aligned}(\mathbf{L}_0, \{\mathbf{L}_i^b\}_{i \in [\ell]}^{b \in \{0,1\}}, r) &\xleftarrow{\$} \mathsf{CPSS.ShareX}(\mathsf{CPSS.pp}),\\ \mathbf{L}_f &\xleftarrow{\$} \mathsf{CPSS.ShareF}(\mathsf{CPSS.pp}, f, \mu, r).\end{aligned}$$

  Let $m_0, m_{i,b}, m_f$ be the dimensions of $\mathbf{L}_0, \mathbf{L}_i^b, \mathbf{L}_f$, respectively. The algorithm sets and runs

  $$\begin{aligned}(k \in [m_f]) \qquad & \mathsf{ict}_{f,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_f[k] \cdot \boldsymbol{\iota}_1]\!]_2),\\ (k \in [m_0]) \qquad & \mathsf{ict}_{0,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_0[k] \cdot \boldsymbol{\iota}_1]\!]_2),\end{aligned}$$

$$(i \in [\ell], b \in \{0,1\}, k \in [m_{i,b}]) \qquad \mathsf{ict}_{i,b,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_i^b[k] \cdot \boldsymbol{\iota}_{2i+b}]\!]_2).$$

It outputs $\mathsf{ct} = \left(\{\mathsf{ict}_{f,k}\}_{k\in[m_f]}, \{\mathsf{ict}_{0,k}\}_{k\in[m_0]}, \{\mathsf{ict}_{i,b,k}\}_{i\in[\ell],b\in\{0,1\},k\in[m_{i,b}]}\right)$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}, \mathsf{sk}, f, \mathsf{ct})$ takes as input mpk, $\mathbf{x}$, sk, f, ct. If $f(\mathbf{x}) \neq 0$, the algorithm outputs $\perp$ and terminates. Otherwise, it parses sk, ct as input $\mathsf{KeyGen}, \mathsf{Enc}$, recomputes $\mathbf{v}$, and runs

$$\begin{aligned}
&(\gamma, 1^B) \xleftarrow{\$} \mathsf{CPSS.FindCoef}(\mathsf{pp}, f, \mathbf{x}),\\
(k \in [m_f]) \qquad &\Lambda_{f,k} \xleftarrow{\$} \mathsf{IPFE.Dec}(\mathsf{impk}, [\![\delta]\!]_2\mathbf{v}, \mathsf{isk}, \mathsf{ict}_{f,k}),\\
(k \in [m_0]) \qquad &\Lambda_{0,k} \xleftarrow{\$} \mathsf{IPFE.Dec}(\mathsf{impk}, [\![\delta]\!]_2\mathbf{v}, \mathsf{isk}, \mathsf{ict}_{0,k}),\\
(i \in [\ell], k \in [m_{i,\mathbf{x}[i]}]) \qquad &\Lambda_{i,\mathbf{x}[i],k} \xleftarrow{\$} \mathsf{IPFE.Dec}(\mathsf{impk}, [\![\delta]\!]_2\mathbf{v}, \mathsf{isk}, \mathsf{ict}_{i,\mathbf{x}[i],k}).
\end{aligned}$$

The algorithm defines

$$\boldsymbol{\Lambda}_f = \left(\Lambda_{f,1}, \ldots, \Lambda_{f,m_f}\right)^\mathsf{T}, \qquad \boldsymbol{\Lambda}^{\mathbf{x}} = \left(\{\Lambda_{0,k}\}_{k\in[m_0]}, \{\Lambda_{i,\mathbf{x}[i],k}\}_{i\in[\ell],k\in[m_{i,\mathbf{x}[i]}]}\right)^\mathsf{T},$$

and finds the unique $\mu' \in \{0,1\}$ such that there exists $e \in [-B, B]$ satisfying

$$\gamma(\boldsymbol{\Lambda}_f, \boldsymbol{\Lambda}^{\mathbf{x}}) = [\![\mu' \lfloor p/2 \rceil + e]\!]_1 [\![\delta]\!]_2.$$

It outputs this μ'.

Correctness. By the correctness of IPFE and how the vectors are set,

$$\boldsymbol{\Lambda}_f = [\![\delta \mathbf{L}_f]\!]_\mathrm{T}, \qquad \boldsymbol{\Lambda}^{\mathbf{x}} = [\![\delta \mathbf{L}^{\mathbf{x}}]\!]_\mathrm{T},$$

and therefore,

$$\gamma(\boldsymbol{\Lambda}_f, \boldsymbol{\Lambda}^{\mathbf{x}}) = [\![\delta \cdot \gamma(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}})]\!]_\mathrm{T}.$$

Since $\delta \neq 0$, by the correctness of CPSS, we have $\mu' = \mu$.

Efficiency. By Construction 6.3, the IPFE components have bit-lengths

$$|\mathsf{impk}|, |\mathsf{imsk}|, |\mathsf{ict}| = (N+1)\,\mathrm{poly}(\lambda), \qquad |\mathsf{isk}| = \mathrm{poly}(\lambda),$$

where $N = 2\ell + 1$. Notably, isk consists of a single element of $\mathbb{G}_2$. Recall that the bit-length of CPSS.pp is $(N+1)\,\mathrm{poly}(\lambda)$. Together with the fact that CPSS has succinct shares, in Construction 6.5,

- mpk consists of CPSS.pp and impk, of total size $|\mathsf{mpk}| = (\ell + 1)\,\mathrm{poly}(\lambda, d)$,
- sk consists of a single isk and $[\![\delta]\!]_2$, of total size $|\mathsf{sk}| = \mathrm{poly}(\lambda)$ (i.e., constant-size; concretely, two elements of $\mathbb{G}_2$), and
- ct consists of $\Theta(\ell^2)$ many ict's, of total size $|\mathsf{ct}| = (\ell^2 + 1)\,\mathrm{poly}(\lambda)$ (notably, potentially much smaller than the description size of f, with which ct is associated).

Therefore, the scheme is doubly succinct.

6.7.5 Security of CP-ABE Scheme

Security. We state and prove the security of Construction 6.5.

Theorem 6.20 (¶). *Suppose* IPFE *is* adaptively simulation-secure in GGM *(Definition 6.9) and* CPSS *is* adaptively jointly non-annihilable *(Definition 6.11), then Construction 6.5 is* secure *(Definition 2.3) in GGM (Definition 6.1).*

We remark that if CPSS is selectively jointly non-annihilable, then the resultant CP-ABE achieves selective security in GGM.

Proof (Theorem 6.20). Recall that in $\mathsf{Exp}^{\mu}_{\mathrm{ABE}}$, the adversary adaptively chooses $\mathbf{x}_j$'s to receive sk_j's, both before and after submitting the challenge policy f and being given ct encrypting $\mu \in \{0,1\}$. Let (IPFE.SimSetup, IPFE.SimKeyGen, IPFE.SimEnc, IPFE.SimGrpZT) be the simulator of IPFE. Consider the following hybrids (always in GGM).

- H^{μ}_1 is $\mathsf{Exp}^{\mu}_{\mathrm{ABE}}$. More concretely,

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^{2\ell+1}), \quad \mathsf{ict}_{f,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_f[k] \cdot \boldsymbol{\iota}_1]\!]_1),$$
$$\mathsf{ict}_{0,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_0[k] \cdot \boldsymbol{\iota}_1]\!]_1), \quad \mathsf{ict}_{i,b,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}^b_i[k] \cdot \boldsymbol{\iota}_{2i+b}]\!]_1),$$
$$\delta_j \xleftarrow{\$} \mathbb{Z}_p \setminus \{0\}, \quad \mathbf{v}_j = \sum_{i \in [\ell]} \boldsymbol{\iota}_{2i+\mathbf{x}_j[i]}, \quad \mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta_j \mathbf{v}_j]\!]_2).$$

- In H_2^μ, the δ_j's are no longer conditioned to be non-zero, i.e.,

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathsf{gp}, 1^{2\ell+1}), \quad \mathsf{ict}_{f,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_f[k] \cdot \boldsymbol{\iota}_1]\!]_1),$$
$$\mathsf{ict}_{0,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_0[k] \cdot \boldsymbol{\iota}_1]\!]_1), \quad \mathsf{ict}_{i,b,k} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, [\![\mathbf{L}_i^b[k] \cdot \boldsymbol{\iota}_{2i+b}]\!]_1),$$
$$\delta_j \xleftarrow{\$} \boxed{\mathbb{Z}_p}, \qquad \mathbf{v}_j = \sum_{i\in[\ell]} \boldsymbol{\iota}_{2i+\mathbf{x}_j[i]}, \qquad \mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, [\![\delta_j \mathbf{v}_j]\!]_2).$$

  We have $\mathsf{H}_1^\mu \approx_{\mathrm{s}} \mathsf{H}_2^\mu$.

- In H_3^μ, the IPFE is simulated, i.e.,

$$(\mathsf{impk}, \boxed{\mathsf{st}}) \xleftarrow{\$} \boxed{\mathsf{IPFE.SimSetup}}(\boxed{p}, 1^{2\ell+1}), \quad (\mathsf{ict}_{f,k}\boxed{, \mathsf{st}'}) \xleftarrow{\$} \boxed{\mathsf{IPFE.SimEnc(st)}},$$
$$(\mathsf{ict}_{0,k}\boxed{, \mathsf{st}'}) \xleftarrow{\$} \boxed{\mathsf{IPFE.SimEnc(st)}}, \quad (\mathsf{ict}_{i,b,k}\boxed{, \mathsf{st}'}) \xleftarrow{\$} \boxed{\mathsf{IPFE.SimEnc(st)}},$$
$$\delta_j \xleftarrow{\$} \mathbb{Z}_p, \qquad \mathbf{v}_j = \sum_{i\in[\ell]} \boldsymbol{\iota}_{2i+\mathbf{x}_j[i]}, \qquad (\mathsf{isk}_j\boxed{, \mathsf{st}'}) \xleftarrow{\$} \boxed{\mathsf{IPFE.SimKeyGen(st)}}.$$

  The simulator's state is appropriately updated. In addition, the GGM queries are answered by IPFE.SimGrpZT using IPZT. The inner products are

$(\mathsf{ict}_{f,k}$ and $\mathsf{isk}_j)$	$\delta_j \mathbf{L}_f[k]$,	$(\mathsf{ict}_{i,\mathbf{x}_j[i],k}$ and $\mathsf{isk}_j)$	$\delta_j \mathbf{L}_i^{\mathbf{x}[i]}[k]$,
$(\mathsf{ict}_{0,k}$ and $\mathsf{isk}_j)$	$\delta_j \mathbf{L}_0[k]$,	$(\mathsf{ict}_{i,\neg\mathbf{x}_j[i],k}$ and $\mathsf{isk}_j)$	0.

  Therefore, the t^{th} query to IPZT tests whether some

$$\sum_j \delta_j \gamma_{t,j}(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}_j})$$

  evaluates to zero, where each $\gamma_{t,j}$ is an affine function. The constant term of $\gamma_{t,j}$ comes from $[\![\delta_j]\!]_2$ or the fact that vectors in keys can always be combined for testing. The coefficients of always-zero inner products are ignored.

  By the security of IPFE, we have $\mathsf{H}_2^\mu \approx \mathsf{H}_3^\mu$.

- In H_4^μ, the oracle IPZT is changed. For each test $\sum_j \delta_j \gamma_{t,j}(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}_j})$, it returns whether all of $\gamma_{t,j}$'s evaluate to zero (without combining using δ_j's).

  By Schwartz–Zippel (Lemma 6.1) for degree-1 polynomials, we have $\mathsf{H}_3^\mu \approx_{\mathrm{s}} \mathsf{H}_4^\mu$.

- In H_5^μ, the oracle IPZT is changed again. For each test $\sum_j \delta_j \gamma_{t,j}(\mathbf{L}_f, \mathbf{L}^{\mathbf{x}_j})$, it returns whether all of $\gamma_{t,j}$'s are the zero function.

  Claim 6.21 (¶). $\mathsf{H}_4^\mu \approx \mathsf{H}_5^\mu$ *by the joint non-annihilability of* CPSS.

Clearly, $\mathsf{H}_5^0 \equiv \mathsf{H}_5^1$ — in both hybrids, μ only (nominally) appears inside group handles and answers to zero-test queries do not depend on μ.

By hybrid argument, $\mathsf{Exp}_{\mathrm{ABE}}^0 \equiv \mathsf{H}_1^0 \approx \mathsf{H}_1^1 \equiv \mathsf{Exp}_{\mathrm{ABE}}^1$. □

Proof (Claim 6.21). We use a guessing and coupling reduction. Fix $\mu \in \{0,1\}$. Let T, J be polynomial upper bounds of IPZT queries and secret key queries, respectively. Couple one trial of H_4^μ and H_5^μ and augment them with a flag of bad event, which is set when $\gamma_{t,j}$ with the lowest (t,j), in lexicographical order, evaluates to zero (in H_4^μ) but is not the zero function (in H_5^μ). The two trials are identical until the bad flag is set. It remains to prove $\Pr[\mathsf{bad}]$ is negligible.

Consider the following efficient adversary $\mathcal{B}$ against the (adaptive) joint non-annihilability of CPSS.

- $\mathcal{B}$ prepares a GGM oracle and samples $(t,j) \xleftarrow{\$} [T] \times [J]$. It launches $\mathcal{A}()$, receives 1^ℓ from $\mathcal{A}$, and sends $(1^\ell, p)$ to its own security experiment, where p is the pairing group order of the GGM oracle prepared by $\mathcal{B}$ for $\mathcal{A}$.
- $\mathcal{B}$ receives CPSS.pp and sends $(p, \mathsf{CPSS.pp}, \mathsf{impk})$ to $\mathcal{A}$, where impk consists of an appropriate number of GGM handles.
- $\mathcal{B}$ interacts with $\mathcal{A}$, until it encounters the t^{th} IPZT query, as follows.
  - If $\mathcal{A}$ queries for a secret key for $\mathbf{x}_j$ or submits a challenge for f, then $\mathcal{B}$ records $\mathbf{x}_j$ or f and hands out GGM handles.
  - If $\mathcal{A}$ submits a GGM zero-test query, $\mathcal{B}$ internally converts it to (potentially many) IPZT queries using IPFE.SimGrpZT. Each IPZT query is then processed as in H_5^μ (except if the t^{th} one is reached, in which case $\mathcal{B}$ breaks out of this loop). If all the converted IPZT queries are processed here, $\mathcal{B}$ responds to $\mathcal{A}$ by finishing IPFE.SimGrpZT.
  - If $\mathcal{A}$ makes a guess, $\mathcal{B}$ gives up and terminates.
- If $\mathcal{B}$ encounters the t^{th} IPZT query, it checks whether $\gamma_{t,j}$ has an out-of-range index. If not, it outputs $(f, \mathbf{x}_j, \mu, \gamma_{t,j})$. Otherwise, it gives up and terminates.

Routine calculation shows $\Pr[\mathcal{B}\text{ wins}] \geq \Pr[\mathsf{bad}]/TJ$, completing the reduction. □

6.8 Further Discussions

During the writing of this dissertation, adaptive LWE (Assumption 6.1) is proven to fail when ℓ is large enough ($\ell \sim n^2 \log^2 q$ suffices for a complete break). The attack is due to [AMR25b] and easily extends to the small-secret variant (Assumption 6.2). This does not affect the efficiency parameters of our KP- and CP-ABE schemes, since they can be based on the usual LWE. If we insist on adaptive security and assume LWE with subexponential advantage, we can use complexity leveraging, which makes λ scale polynomially with ℓ. At first sight, it appears that our KP-ABE no longer has constant-size keys and our CP-ABE is no longer doubly succinct. Closer inspection shows that *this is not the case* and the most important efficiency characteristics are preserved.

For our KP-ABE, the key size depends on the pairing group order. We apply complexity leveraging, not to lift ABE from selective to adaptive, but to prove adaptive LWE from LWE. Consequently, the pairing group order is not bumped up, and our KP-ABE still has constant-size keys.

For our CP-ABE, the pairing group order p is the same as the LWE modulus q and affects both key and ciphertext sizes. Although complexity leveraging could make q grow exponentially with ℓ, we can fix it. Suppose $\mathsf{ALWE}_{n,q,\sigma}$ (Assumption 6.1) holds, then $\mathsf{ALWE}_{n,q/z,\sigma}$ also holds for any super-polynomial z by a simple reduction. Thanks to the polynomial noise growth (see Construction 6.4 and the parameters after it), we can artificially tune down q (to a smaller super-polynomial) while keeping the size upper bound of supported formulae super-polynomial. In effect, relying on complexity leveraging only enlarges n, m, σ, not q, used in Constructions 6.4 and 6.5. It makes the ciphertext size of CP-ABE larger, but still $\mathrm{poly}(\lambda, \ell)$, but does not affect the key size, which can still be $\mathrm{poly}(\lambda)$. Therefore, the scheme is still doubly succinct.

Chapter 7

Succinct CP-ABE for Circuits and ABE for DFA from Lattices via Evasive IPFE

(Lattice-Based IPFE, Noisy Linear Garbling)

This chapter is an adaptation of [HLL24b,HLL24a].[71]

In this chapter, we show how to instantiate the general paradigm solely using lattice-based components. We obtain a new candidate of lattice-based CP-ABE for circuits as well as the *first* provably secure lattice-based public-key ABE for uniform models of computation (concretely, DFA and L).

7.1 Introduction

The area of ABE witnessed substantial progress over the past decade, thanks to the insights on leveraging the mathematical structures present in integer lattices and the learning with errors (LWE) assumption. Notably, Gorbunov, Vaikuntanathan, and Wee [GVW13] achieved a significant milestone by constructing the first KP-ABE scheme for circuits of unbounded size, but with *a priori* bounded depth, relying on the LWE assumption. Subsequent work by Boneh *et al.* [BGG+14] further enhanced KP-ABE efficiency, delivering the first scheme with short ciphertexts and secret keys, both scaling polynomially with the depth bound and independent of the circuit size.

Despite these remarkable advances, several challenges have persisted for over a decade in the lattice-based ABE landscape. They include *i)* the search for ABE schemes accommodating circuits of unbounded depth, *ii)* the construction of CP-ABE schemes

[71] Yao-Ching Hsieh, Huijia Lin, Ji Luo:
A General Framework for Lattice-Based ABE Using Evasive Inner-Product Functional Encryption,
 DOI 10.1007/978-3-031-58723-8_15.

supporting all polynomial-size circuits, even confined to predetermined depth bounds, and *iii)* extending ABE to uniform models of computation, such as deterministic finite automata (DFA), logspace Turing machines, and RAM, which have much more succinct descriptions than circuits and can process arbitrarily long inputs. Until recently, the only lattice-based solution addressing challenges *i)* and *iii)* were due to Goldwasser *et al.* [GKP+13a], who presented a KP-ABE scheme for RAM. However, it relies on the heavy tools of SNARK and extractable witness encryption, which can be instantiated using lattices, but require strong knowledge-type assumptions. Beyond the scope of lattice-based constructions, one could attain these objectives using indistinguishability obfuscation ($i\mathcal{O}$) as shown in [GGH+13a] and Chapter 9.

Recently, the landscape has evolved with the introduction of a novel class of assumptions known as *evasive LWE* [Wee22,Tsa22]. At a high level, evasive LWE captures a "generic-model view" on lattices. When presented with LWE samples $(\mathbf{s}^\intercal\mathbf{B}+\mathbf{e}_1^\intercal, \mathbf{s}^\intercal\mathbf{A}+\mathbf{e}_0^\intercal)$ and a short (i.e., low-norm) trapdoor preimages $\mathbf{K}$ such that $\mathbf{BK}=\mathbf{P}$, denoted as $\mathbf{K}=\mathbf{B}^{-1}(\mathbf{P})$, where all terms are relative to a randomly sampled matrix $\mathbf{B}$ and jointly sampled matrices $\mathbf{A}$ and $\mathbf{P}$, it postulates that if the LWE samples $(\mathbf{s}^\intercal\mathbf{P}+\mathbf{e}_2^\intercal, \mathbf{s}^\intercal\mathbf{B}+\mathbf{e}_1^\intercal, \mathbf{s}^\intercal\mathbf{A}+\mathbf{e}_0^\intercal)$ are pseudorandom when $\mathbf{e}_2^\intercal$ is freshly generated, then so are $(\mathbf{s}^\intercal\mathbf{B}+\mathbf{e}_1^\intercal, \mathbf{s}^\intercal\mathbf{A}+\mathbf{e}_0^\intercal)$ given $\mathbf{B}^{-1}(\mathbf{P})$. In simpler terms, adversaries are limited to utilizing the trapdoor to acquire LWE samples $(\mathbf{s}^\intercal\mathbf{P}+\mathbf{e}_2^\intercal)$ and treating them as LWE samples with fresh noises. The work by [Wee22] argued that if this type of generic attacks fail, then all known cryptanalytic techniques, including the family of zeroizing attacks (e.g., [CHL+15,CVW18,HJL21,JLLS23]), would also be ineffective.

The introduction of the evasive LWE assumption has catalyzed new progress on multiple fronts, including *i)* and *ii)* mentioned above. Wee [Wee22] presented a CP-ABE scheme for bounded-depth circuits under evasive LWE and another new assumption called tensor LWE. More recently, in Chapter 8, we achieved the first KP-ABE scheme for circuits of unbounded depth. Two other works [VWW22,Tsa22] constructed witness encryption schemes, which can be viewed as CP-ABE where the secret key for an attribute is the attribute itself. Furthermore, Water, Wee, and Wu [WWW22] proposed a multi-authority ABE for subset policies without resorting to random oracles.

These accomplishments underscore the potential of evasive LWE. However, each work uses evasive LWE in a way tailored to each construction, relying on a customized variant of evasive LWE following the generic-model principle. This often requires intricate handling of algebraic details. We ask:

Can we formulate a general framework for
constructing ABE schemes based on evasive LWE?

Such a framework would provide several advantages, including *i)* deepening our understanding, specifically by elucidating the role of evasive LWE, *ii)* facilitating the development of new ABE constructions, by making them more modular and conceptually simpler, and *iii)* streamlining security proofs, by encapsulating low-level algebraic intricacies within the framework and providing natural and useful tools.

Our Results. To this end, we propose a new notion called *evasive* inner-product functional encryption (IPFE), and present a general framework for constructing CP- and KP-ABE schemes for a class of functions from evasive IPFE and *noisy* linear secret sharing for the same class. Our framework can be seen as the lattice-world counterpart of the general framework for constructing ABE from pairing-based IPFE and (noiseless or noisy) linear secret sharing, as described in Chapters 4, 5, and 6. Similar ideas have been employed in various previous pairing-based ABE constructions, which have been realized through different methods, including dual system encryption and hash proof systems.

Evasive Inner-Product Functional Encryption (IPFE). Let's start by introducing evasive IPFE. Similar to a standard IPFE [ABDP15], evasive IPFE allows encrypting a vector $\mathbf{u} \in \mathbb{Z}_q^Z$ into a ciphertext $\mathsf{ict}(\mathbf{u})$, and generating secret keys $\{\mathsf{isk}(\mathbf{v}_j)\}_j$ tied to vectors $\mathbf{v}_j \in \mathbb{Z}_q^Z$, except that decryption reveals *noisy* (as opposed to exact) inner products $\{\mathbf{u}^\intercal \mathbf{v}_j + e_j\}_j$. The more substantial difference lies in the security requirements. Standard IPFE ensures that only the inner products are revealed and no other information of the encrypted vector $\mathbf{u}$ is leaked (the key vectors $\mathbf{v}_j$ are public). Evasive IPFE tries to capture a "generic-model view" similar to evasive LWE, but at the higher abstraction level of IPFE. It postulates that if the noisy inner products $\{\mathbf{u}^\intercal \mathbf{v}_j + e_j\}_j$ with

"idealized", i.e., freshly generated (hence independent), noises are pseudorandom, then the ciphertext is pseudorandom. More formally,

$$\begin{aligned} \text{if} \quad & \{\mathbf{v}_j,\ \mathbf{u}^\intercal \mathbf{v}_j + e_j\}_j \approx \{\mathbf{v}_j,\ \$\}_j, \\ \text{then} \quad & \mathsf{impk},\ \mathsf{ict}(\mathbf{u}),\ \{\mathbf{v}_j,\ \mathsf{isk}(\mathbf{v}_j)\}_j \approx \mathsf{impk},\ \$,\ \{\mathbf{v}_j,\ \mathsf{isk}(\mathbf{v}_j)\}_j, \end{aligned}$$

where e_j's are independent Gaussian noises. Evasive IPFE is a convenient tool for generating LWE samples. Consider a simple example where the encrypted vector is an LWE secret $\mathbf{u} = \mathbf{s}$ and the key vectors correspond to an LWE public matrix $\mathbf{A}$, which may or may not be random. Security ensures that as long as the noisy outputs $(\mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal)$ with *fresh* noises are pseudorandom, the ciphertext encrypting $\mathbf{s}$ is pseudorandom.

Jumping ahead, in Section 7.4, we present candidate lattice-based evasive IPFE schemes, and show their security for restricted distributions of plaintext vectors (observed by all our ABE constructions) based on evasive LWE.

NOISY LINEAR SECRET SHARING SCHEME (LSSS). The other ingredient of our general framework is noisy LSSS. A noisy LSSS over $\mathbb{Z}_q$ for a function f (say, of Boolean input and output), denoted as LSSS_f, consists of a non-zero vector $\mathbf{w}_{\text{out}}$ and matrices $\mathbf{T}, \{\mathbf{W}_{\ell,b}\}_{\ell,b}$. Given randomness $\mathbf{s}$, the secret and the shares for an L-bit input $\mathbf{x}$ are created from those matrices.

$$\begin{aligned} \mathsf{LSSS}_f\text{:} \quad & \mathbf{w}_{\text{out}}, \mathbf{T}, \{\mathbf{W}_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}; \\ \text{Secret:} \quad & \mathbf{s}^\intercal \mathbf{w}_{\text{out}} + e_{\text{out}}; \qquad \text{Shares:} \quad \mathbf{s}^\intercal \mathbf{T} + \mathbf{e}_{\mathsf{t}}^\intercal, \{\mathbf{s}^\intercal \mathbf{W}_{\ell,\mathbf{x}[\ell]} + \mathbf{e}_\ell^\intercal\}_{\ell\in[L]}. \end{aligned}$$

If $f(\mathbf{x}) = 1$, the secret $(\mathbf{s}^\intercal \mathbf{w}_{\text{out}} + e_{\text{out}})$ can be approximately recovered from the shares, whereas if $f(\mathbf{x}) = 0$, the secret and the shares must be jointly pseudorandom. In addition, we say that LSSS_f is *short* if its $\mathbf{w}_{\text{out}}, \mathbf{T}, \{\mathbf{W}_{\ell,b}\}_{\ell,b}$ have small norms.

Noisy LSSS is a significantly weaker primitive than ABE in that it only considers a single function f and a single input $\mathbf{x}$, and that the shares for $\mathbf{x}$ can depend on f. In contrast, ABE supports publishing secret keys and ciphertexts for many functions and many inputs, and both must be generated independent of the data on the other side. Interestingly, almost every known LWE-based ABE scheme implicitly defines an

LSSS scheme. In the literature, noiseless and information-theoretically secure LSSS are known for low-depth functions [KW93,Ber84,BDHM92,Mul87]. Based on LWE, prior works have constructed a similar primitive, called nearly linear secret sharing scheme, for arbitrary circuits (see [QWW18,AY20,AWY20] and Chapter 6). See related works for a comparison.

General Framework for Constructing ABE. Our general framework uses evasive IPFE to generically "lift" noisy LSSS into full-fledged ABE schemes. For technical reasons elaborated in Section 7.2, we need to start from a *short* noisy LSSS.

Construction 7.5. *Given a short noisy LSSS for any family of functions over $\mathbb{Z}_q$, KP- and CP-ABE schemes for the same family can be constructed from an evasive IPFE scheme over $\mathbb{Z}_q$.*

When instantiated with our evasive IPFE scheme (Section 7.4.4), the KP-ABE scheme has

$$|\mathsf{mpk}| = \mathrm{poly}(\log q), \qquad |\mathsf{sk}_f| = |\mathsf{LSSS}_f|\,\mathrm{poly}(\log q), \qquad |\mathsf{ct}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(\log q),$$

and the CP-ABE scheme has

$$|\mathsf{mpk}| = \mathrm{poly}(\log q), \qquad |\mathsf{sk}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(\log q), \qquad |\mathsf{ct}_f| = |\mathsf{LSSS}_f|\,\mathrm{poly}(\log q).$$

Here, $|\mathsf{LSSS}_f|$ is the length of its defining vectors and matrices, and polynomial factors in the security parameter λ are hidden.

Using our general framework, the task of building ABE for an arbitrary function class reduces to the simpler task of constructing noisy LSSS for it. This has led to the following new ABE constructions.

- In Section 7.6, we design new *short* noisy LSSS for all arithmetic circuits based on LWE. The schemes are adapted from the arithmetic garbling proposed by [AIK11]. Combining them with our evasive IPFE scheme yields a new CP-ABE scheme for bounded-depth bounded-arithmetic circuits from evasive LWE. Compared with Wee's construction [Wee22], our scheme eliminates the tensor LWE assumption, but does not have succinct ciphertext.

- For deterministic finite automata (DFA) and logspace Turing machines (L), noiseless LSSS are known, which have been used to construct pairing-based ABE for DFA and L in Chapter 4. In lattice-based ABE landscape, handling uniform models of computation has been a challenge. The state-of-the-art ABE schemes for DFA only tolerate bounded collusion [AS17a,Wee21], are in the secret-key setting [AMY19a] (this also supports NFA), or do not come with a security proof [Wee21]. In Section 7.8, we apply our general framework to those LSSS with simple tweaks for uniform computation, obtaining both KP- and CP-ABE for DFA and L based on evasive LWE, the first provably secure lattice-based public-key ABE schemes tolerating unbounded collusion for these uniform models.

Corollaries 7.17 and 7.25 (CP-ABE for circuits and ABE for DFA). *Assuming LWE and evasive LWE, there exists a CP-ABE scheme for arithmetic circuits with*

$$|\mathsf{mpk}| = \mathrm{poly}(d, \log M), \quad |\mathsf{sk}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(d, \log M), \quad |\mathsf{ct}_C| = |C|\,\mathrm{poly}(d, \log M),$$

where $|C|$ is the circuit size, d is a bound on the depth of C, and M is a bound on the magnitude of intermediate computation values.

Under the same set of assumptions, there exist ABE schemes for DFA.

$$\begin{aligned} &\text{KP-ABE for DFA:} && |\mathsf{mpk}| = \mathrm{O}(1), && |\mathsf{sk}_{\mathbf{x}}| = \mathrm{O}(|\mathbf{x}|), && |\mathsf{ct}_{\Gamma}| = \mathrm{O}(|\Gamma|);\\ &\text{CP-ABE for DFA:} && |\mathsf{mpk}| = \mathrm{O}(1), && |\mathsf{sk}_{\Gamma}| = \mathrm{O}(|\Gamma|), && |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(|\mathbf{x}|). \end{aligned}$$

Here, $|\mathbf{x}|$ is the input length and $|\Gamma|$ is the number of states of the DFA. Each $\mathrm{O}(\cdot)$ hides a polynomial factor in λ.

Upgrading Our General Framework. An unsatisfactory aspect of the above general framework is that it requires *short* noisy LSSS. On the other hand, there are noisy LSSS that are not *short*. For instance, the noisy LSSS underlying the ABE scheme of [BGG+14], as described below.

$$\begin{aligned} &\mathsf{LSSS}_f\colon \quad \mathbf{w}_{\mathrm{out}} = \mathbf{A}_f\mathbf{G}^{-1}(\mathbf{u}),\ \mathbf{T} = \mathbf{A}_0,\ \{\mathbf{W}_{\ell,b} = \mathbf{A}_\ell - b\mathbf{G}\}_{\ell\in[L],b\in\{0,1\}};\\ &\text{Secret:}\quad \mathbf{s}^\intercal\mathbf{A}_f\mathbf{G}^{-1}(\mathbf{u}); \qquad \text{Shares:}\quad \mathbf{s}^\intercal\mathbf{A}_0 + \mathbf{e}_0^\intercal, \{\mathbf{s}^\intercal(\mathbf{A}_\ell - \mathbf{x}[\ell]\mathbf{G}) + \mathbf{e}_\ell^\intercal\}_{\ell\in[L]}. \end{aligned}$$

The vector $\mathbf{u}$ and matrices $\mathbf{A}_0$, $\mathbf{A}_\ell$'s are random, hence they have large entries. (So are the entries of the gadget matrix $\mathbf{G}$.) We would like to upgrade our general framework to accommodate the BGG-based LSSS. Beyond the consideration of generality, that LSSS is the only known LSSS for circuits of succinct size, in particular, $|\mathsf{LSSS}_f| = \mathrm{poly}(d)$ grows polynomially with the maximum depth of the computation, instead of size. Once combined with our framework, we immediately obtain a CP-ABE scheme supporting bounded-depth circuits with *succinct* ciphertexts.

We achieve this by introducing a stronger variant of evasive IPFE, where the noisy inner products returned from decryption additionally contains structured noises in the form of $e'\mathbf{g}$, where $\mathbf{g}$ is the gadget vector. More precisely, in an evasive IPFE scheme with structured noises, a ciphertext $\mathsf{ict}(\mathbf{u})$ still encrypts a vector $\mathbf{u}$, but a secret key $\mathsf{isk}(\mathbf{V})$ now encodes a matrix $\mathbf{V}$ whose number of columns is the dimension of $\mathbf{g}$. Decryption produces $(\mathbf{u}^\intercal\mathbf{V} + e'\mathbf{g}^\intercal + \mathbf{e}^\intercal)$, where $\mathbf{e}$ is a small noise vector as before and $e'\mathbf{g}$ is the structured noise. Security is enhanced correspondingly. If the noisy outputs with idealized fresh noises $e', \mathbf{e}$ are pseudorandom, so is the ciphertext, i.e.,

$$\begin{aligned} &\text{if} && \{\mathbf{V}_j,\ \mathbf{u}^\intercal\mathbf{V}_j + e'_j\mathbf{g}^\intercal + \mathbf{e}_j^\intercal\} \approx \{\mathbf{V}_j,\ \$\}, \\ &\text{then} && \mathsf{impk},\ \mathsf{ict}(\mathbf{u}),\ \{\mathbf{V}_j,\ \mathsf{isk}(\mathbf{V}_j)\}_j \approx \mathsf{impk},\ \$,\ \{\mathbf{V}_j,\ \mathsf{isk}(\mathbf{V}_j)\}_j. \end{aligned}$$

Using evasive IPFE with structured noises, we can now "lift" arbitrary noisy LSSS, without any norm restriction, into full-fledged ABE schemes.

Construction 7.10 and Corollary 7.30. *Given a (not necessarily short) noisy LSSS for any family of functions over $\mathbb{Z}_q$, KP- and CP-ABE schemes for the same family can be constructed from evasive IPFE with structured noises over $\mathbb{Z}_q$.*

When instantiated with our evasive IPFE with structured noises (Section 7.4.5), the resultant ABE schemes have the same asymptotic sizes as those in Construction 7.5 — compared with which, it is now possible to use LSSS with smaller sizes. Assuming LWE and a variant of evasive LWE, there exists a CP-ABE scheme for arithmetic circuits with

$$|\mathsf{mpk}| = \mathrm{poly}(d, \log M), \qquad |\mathsf{sk}_{\mathbf{x}}| = |\mathbf{x}|\,\mathrm{poly}(d, \log M), \qquad |\mathsf{ct}_f| = \mathrm{poly}(d, \log M).$$

Constructions of Evasive IPFE. We propose candidate lattice-based evasive IPFE schemes, both with and without structured noises in Section 7.4, and extensively study

them. We show that when the input vectors $\mathbf{u}_i$ have certain form, namely $\mathbf{u}_i^\intercal = \mathbf{s}^\intercal \mathbf{A}_i$ for arbitrary $\mathbf{A}_i$'s sampled using public coins and independent and uniformly random $\mathbf{s}$ (e.g., $\mathbf{u}^\intercal = \mathbf{s}^\intercal = \mathbf{s}^\intercal \mathbf{I}$), then we can base the security of evasive IPFE on variants of evasive LWE. All ABE constructions in this chapter uses evasive IPFE to encrypt vectors of this form. For evasive IPFE without structured noises, we rely on the usual type of evasive LWE assumption, where all noises involved are small (Assumption 7.2). For the version with structured noises, we rely on a variant of evasive LWE with structured noises (Assumption 7.1).

In order to reduce to evasive LWE, the ABE constructions must use a single IPFE instance, instead of many instances with independently sampled master public key. However, we often want to impose some restriction on what inner products are produced, e.g., only computing $\mathbf{u}_1^\intercal \mathbf{v}_1$ and $\mathbf{u}_2^\intercal \mathbf{v}_2$, without revealing $\mathbf{u}_1^\intercal \mathbf{v}_2$ or $\mathbf{u}_2^\intercal \mathbf{v}_1$. To achieve this using a single instance of evasive IPFE, we extend the interface to be identity-based (see Section 7.4.4), so that a pair of ciphertext $\mathsf{ict}_{\mathsf{id}}(\mathbf{u})$ and secret key $\mathsf{isk}_{\mathsf{id}'}(\mathbf{v})$ only reveals the noisy inner product if $\mathsf{id} = \mathsf{id}'$.

A nice addition is that the study of evasive IPFE already employs several pairing-based techniques. Nevertheless, the constructions and security proofs are intricate and require ironing out many details. We hope our evasive IPFE schemes serve as useful tools in future works.

Related Works. We discuss related works by aspects.

Lattice-Based Linear Secret Sharing Schemes for Circuits. The works of [AY20, AWY20] constructed CP-ABE with succinct ciphertext for NC^1 relying on LWE and the generic (pairing) group model (GGM). In Chapter 6, we constructed CP-ABE for circuits with constant-size keys (i.e., of size $\mathrm{poly}(\lambda)$) and succinct ciphertext (of size $|\mathbf{x}|\,\mathrm{poly}(\lambda)$, independent of depth). Both schemes combine a nearly linear secret sharing scheme with a pairing-based IPFE.

Nearly LSSS is very similar to noisy LSSS. Both are noisy and if $f(\mathbf{x}) = 1$, evaluation reveals a random secret s perturbed by a small noise e. The major differences are *i)* nearly LSSS requires linear reconstruction and the noise e to be small, which are not required by noisy LSSS, and *ii)* noisy LSSS shares must have the specific form $\mathbf{s}^\intercal \mathbf{T} + \mathbf{e}_{\mathsf{t}}^\intercal$,

$\{\mathbf{s}^\top \mathbf{W}_{\ell,\mathbf{x}[\ell]} + \mathbf{e}_\ell^\top\}_{\ell\in[L]}$, where the noises are small. These requirements are tailored for composition with pairing-based IPFE and evasive IPFE, respectively. We compare their secret sharing scheme and techniques with ours.

Based on the laconic function evaluation (LFE) protocol of [QWW18], which in turn is based on the circuit KP-ABE scheme of [BGG+14], the works of [AY20,AWY20] proposed a nearly linear secret sharing scheme for NC^1 and Chapter 6 one for circuits. The nearly LSSS schemes are then lifted to ABE schemes using pairing-based IPFE — those ABE schemes use pairing-based IPFE to compute rerandomized nearly LSSS shares. As a result, the computed shares reside in the exponent of the pairing group. By the fact that reconstruction is linear, the perturbed secret $(s + e)$ can be recovered in the exponent if $f(\mathbf{x}) = 1$. However, to recover the secret s in the clear, the schemes rely on exhaustive search of the noise e. Therefore, e must be polynomially bounded, which is particularly difficult to achieve when evaluating high-depth circuits (see Chapter 6). The fact that e must be small also makes the security proof more challenging, as one cannot rely on noise smudging.

Differently, in our general framework and hence CP-ABE schemes, the secret shares are computed in the clear and the evaluation noise e just need to be sufficiently small compared to the modulus, and evaluation can be non-linear. However, we require randomization of the shares in the form of $\mathbf{s}^\top \mathbf{T} + \mathbf{e}_t^\top, \{\mathbf{s}^\top \mathbf{W}_{\ell,\mathbf{x}[\ell]} + \mathbf{e}_\ell^\top\}_{\ell\in[L]}$ with small noises, in order to match the precondition of evasive LWE, where these small noises are "idealized" to be fresh.

A technical relation is that those ABE schemes rely on GGM or knowledge-type assumption (the Knowledge of OrthogonALity Assumption, or "KOALA") on the pairing group to argue that correlated instances of secret sharing for different computation $\{(f_i, \mathbf{x}_j)\}_{i,j}$ where $f_i(\mathbf{x}_j) = 0$ for all i, j (resulting from decrypting all pairs of secret keys and ciphertexts) are jointly secure. In our ABE scheme, we rely on evasive IPFE (and by reduction, evasive LWE) to assert the security of the many correlated instances of secret sharing. In folklore, evasive LWE is regarded as a lattice counterpart of GGM, although there has been no formal known connections between them.

The works of [AY20,AWY20] as well as Chapter 6 *combine* pairing techniques with

lattice ones, relying on pairing-based IPFE as well as LWE. This chapter *translates* pairing techniques to the lattice world, creating a lattice implementation of IPFE, suitable for ABE when combined with LWE. Evasive IPFE is reminiscent to the very strong simulation security of IPFE in GGM proven in Chapter 6, which can be seen as a step towards understanding the connection between GGM and evasive LWE.

NOISY IPFE. The work of [Agr19], followed by [AP20], proposed a primitive called noisy IPFE, which evaluates noisy inner products, i.e., $(\mathbf{u}^\top \mathbf{v} + e)$. It is defined with strong security property. For any two plaintext vectors $\mathbf{u}, \mathbf{u}'$ and key vectors $\mathbf{v}_1, \ldots, \mathbf{v}_J$ such that their inner products are approximately equal, i.e., $(\mathbf{u} - \mathbf{u}')^\top \mathbf{v}_j$ is small for all j, the ciphertext encrypting $\mathbf{u}$ and that encrypting $\mathbf{u}'$ must be indistinguishable. The rationale is that the noises resulting from decryption is sufficient to "smudge" the difference between the inner products. This primitive is strong enough to imply indistinguishability obfuscation when combined with other well-studied assumptions.

Our evasive IPFE has a different security requirement. It states that if the inner products plus fresh noises are pseudorandom, i.e., $\{\mathbf{u}_i^\top \mathbf{v}_j + e_{ij}\} \approx \{\$\}$, then the ciphertexts encrypting $\mathbf{u}_i$'s are pseudorandom given the keys. We achieve this under evasive LWE for a general class of distributions of inputs.

ABE FOR DFA FROM LATTICES. The work of [AS17a] presented a public-key FE for DFA from LWE against bounded collusion. A bounded-collusion ABE for DFA, with better efficiency and also based on LWE, was presented in [Wee21]. In [AMY19a], the authors devised a lattice-based secret-key ABE for NFA. The work of [Wee21] also proposed a candidate public-key KP-ABE for DFA based on lattices. In terms of methods, it relies on the DFA garbling implicit in [Wat12] (made explicit in Chapter 4), which we also use for our scheme. While many known ABE for DFA (including [Wat12,GWW19, GW20a,Wee21], Chapter 4, and this chapter) are roughly based on the idea of *securely* computing a pseudorandom garbling for DFA, to our knowledge, only Chapter 4 and this chapter explicitly use *functional encryption* (concretely, IPFE) to do so. In terms of results, in [Wee21], there was no reductionist proof of security but some preliminary cryptanalysis, in which an idea in the line of evasive LWE is used (replacing a trapdoor preimage by the intended usage result, with fresh noises). The development of evasive

LWE is later to [Wee21]. While we do not immediately see a proof from the version of evasive LWE in our work or [Wee22,Tsa22], it is plausible that another variant of evasive LWE could be formulated to prove its security (or a slightly modified variant thereof).

7.2 Technical Overview

We give an overview of our techniques.

7.2.1 Review and Renew

Our starting point is a well-established method of constructing ABE using linear secret sharing schemes (LSSS) and pairing.

A traditional LSSS is noiseless and information-theoretic. An LSSS for an arithmetic function $f : \mathbb{Z}^L \to \{0,1\}$ is specified by vector $\mathbf{w}_{\mathsf{out}} \neq \mathbf{0}$ and matrices $\mathbf{T}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}$ over $\mathbb{Z}_q$, each of which has n_f rows. Given randomness $\mathbf{s} \in \mathbb{Z}_q^{n_f}$, the secret is $\mathbf{s}^\intercal\mathbf{w}_{\mathsf{out}}$ and the shares are

$$\mathbf{s}^\intercal\mathbf{T} \quad \text{and} \quad \{\mathbf{s}^\intercal(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times})\}_{\ell\in[L]}.$$

For correctness, if $f(\mathbf{x}) = 1$, the shares $\mathbf{s}^\intercal\mathbf{T}, \{\mathbf{s}^\intercal\mathbf{W}_{\ell,\mathbf{x}[\ell]}\}_{\ell\in[L]}$ must determine the secret $\mathbf{s}^\intercal\mathbf{w}_{\mathsf{out}}$, where $\mathbf{W}_{\ell,\mathbf{x}[\ell]} = \mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}$. For security, if $f(\mathbf{x}) = 0$ and $\mathbf{s}$ is uniformly random, the shares must be independent of $\mathbf{s}^\intercal\mathbf{w}_{\mathsf{out}}$.

To obtain (KP- or CP-) ABE from LSSS, pairing is used so that when decrypting a ciphertext using a key, tied to $f, \mathbf{x}$, it will first compute the LSSS_f shares of $\mathbf{x}$ encoded in the exponent of the target group, which can then be used to recover the secret (again, in the exponent) if $f(\mathbf{x}) = 1$. For the construction to be secure, the computed shares should use good randomness, which must be kept hidden. The mechanism of computing the shares vary across known constructions, and oftentimes it can be regarded as inner-product functional encryption (IPFE) [ABDP15]. Since the shares are computed in the exponent, pseudorandomness can often be obtained from DDH-style rerandomization, e.g., in CP-ABE, keys contain random r, ciphertexts contain $\mathbf{s}^\intercal\mathbf{T}, \{\mathbf{s}^\intercal\mathbf{W}_{\ell,\mathbf{x}[\ell]}\}_{\ell\in[L]}$ with random $\mathbf{s}$, and the computed shares are $r\mathbf{s}^\intercal\mathbf{T}, \{r\mathbf{s}^\intercal\mathbf{W}_{\ell,\mathbf{x}[\ell]}\}_{\ell\in[L]}$

in the exponent of the target group. Intuitively, by the decisional Diffie–Hellman assumption, the computed shares, using $r\mathbf{s}$ as its randomness, is indistinguishable from being created with fresh randomness. This intuition has been implemented in multiple ways, e.g., dual system encryption, hash proof systems, or using slotted IPFE.

ABE from LSSS and Lattices. Pairing-based ABE only supports low-depth computations, because the expressive power of polynomial-size LSSS is limited in NC [KW93, Ber84,BDHM92,Mul87]. To circumvent this lower bound, we must relax the notion of LSSS to support richer classes of functions. We consider LSSS with noises, termed *noisy linear secret sharing* in this chapter. Such a scheme for f is again specified by $\mathbf{w}_{\text{out}}, \mathbf{T}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}$, but the secret and the shares now contain noises.

$$\begin{aligned} &\text{Secret:} && \mathbf{s}^\top \mathbf{w}_{\text{out}} + e_{\text{out}}; \\ &\text{Shares:} && \mathbf{s}^\top \mathbf{T} + \mathbf{e}_{\text{t}}^\top \quad \text{and} \quad \{\mathbf{s}^\top(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_\ell^\top\}_{\ell\in[L]}. \end{aligned}$$

We additionally allow the secret sharing to have a reusable part $\mathbf{R}$ (always given and not randomized). If $f(\mathbf{x}) = 1$, correctness only guarantees recovering the secret approximately. For security, if $f(\mathbf{x}) = 0$, the secret and the shares should be jointly pseudorandom even given $\mathbf{R}, \mathbf{w}_{\text{out}}, \mathbf{T}, \mathbf{W}$'s. Following the previous pairing-based general framework, we now need an appropriate IPFE that can compute and rerandomize the noisy secret shares.

Insufficiency of Existing IPFE. When using pairing-based IPFE, since the shares are computed in the exponent, one can only reconstruct the secret plus noise in the exponent. To be able to recover the secret in the clear, the noise must be polynomially small to allow for exhaustive search. However, keeping the noise small is difficult when evaluating arbitrary depth circuits.

To get around this limitation, we ask whether lattice-based IPFE schemes such as [ABDP15,ALS16] can help. They yield decryption results in the clear, dispensing the limitation on the magnitude of noise. However, it is unclear how to use these scheme to generate the noises needed in each rerandomized secret shares. In fact, Agrawal [Agr19] together with Pellet-Mary [AP20] showed that a noisy IPFE scheme

capable of computing inner products added with (computationally) good Gaussian noises implies $i\mathcal{O}$ when combined with other standard assumption.

Evasive IPFE. We solve the aforementioned challenges with a new primitive called *evasive IPFE*. For convenience, we directly define the identity-based variant. Each key is for a vector $\mathbf{v}$ under some identity id', and each ciphertext, $\mathbf{u}$ and id. Decryption yields an approximation of $\mathbf{u}^\intercal\mathbf{v}$ if and only if $\mathsf{id} = \mathsf{id}'$.

$$\left.\begin{matrix}\mathsf{isk}_{\mathsf{id}'}(\mathbf{v})\\ \mathsf{ict}_{\mathsf{id}}(\mathbf{u}^\intercal)\end{matrix}\right] \xrightarrow{\mathsf{Dec}} \begin{cases}\mathbf{u}^\intercal\mathbf{v} + e, & \text{if } \mathsf{id} = \mathsf{id}';\\ \bot, & \text{if } \mathsf{id} \neq \mathsf{id}'.\end{cases}$$

We also denote batches of keys and ciphertexts as $\mathsf{isk}_{\mathsf{id}}(\mathbf{V})$ and $\mathsf{ict}_{\mathsf{id}}(\mathbf{U}^\intercal)$ so that their pairwise decryptions approximate $\mathbf{U}^\intercal\mathbf{V}$.

For security, given arbitrarily many keys, the ciphertexts hide $\mathbf{u}$'s if all the inner products that can be computed are jointly pseudorandom, when they are added with fresh Gaussian noises. More precisely, given secret keys generated for $\mathbf{v}_j$ under identity id'_j and ciphertexts for $\mathbf{u}_i$ under id_i, the security property stipulates that

$$\begin{aligned}\text{if}\quad & \mathsf{aux}, \{\mathbf{u}_i^\intercal\mathbf{v}_j + e_{i,j}\}_{\mathsf{id}_i=\mathsf{id}'_j} \approx \mathsf{aux}, \{\$\}_{\mathsf{id}_i=\mathsf{id}'_j},\\ \text{then}\quad & \mathsf{aux}, \{\mathsf{isk}_{\mathsf{id}'_j}(\mathbf{v}_j)\}_j, \{\mathsf{ict}_{\mathsf{id}_i}(\mathbf{u}_i)\}_i \approx \mathsf{aux}, \{\mathsf{isk}_{\mathsf{id}'_j}(\mathbf{v}_j)\}_j, \{\mathsf{ict}_{\mathsf{id}_i}(\$)\}_i.\end{aligned}$$

Here, $e_{i,j}$'s are fresh (Gaussian) noises, and aux is auxiliary information about $\mathbf{v}, \mathbf{u}$'s — think of it as the public coins for sampling $\mathbf{v}$'s.

7.2.2 ABE from Noisy Linear Secret Sharing and Evasive IPFE

Given evasive IPFE and noisy LSSS, our general framework for constructing KP- and CP-ABE is modular and simple. For example, to construct CP-ABE, each ciphertext for a function f consists of a series of IPFE ciphertexts encoding $\mathbf{S}^\intercal\mathbf{W}$'s, where $\mathbf{S}$ is an LWE secret matrix, and each key consists of IPFE keys encoding LWE public vector $\mathbf{r}$ and input $\mathbf{x}$. Their decryption produces rerandomized secret shares of the form $(\mathbf{r}^\intercal\mathbf{S}^\intercal\mathbf{W} + \mathbf{e}^\intercal)$. Below is an example.

$$\begin{aligned}
\mathsf{sk}_{\mathbf{x}}: &\quad \mathsf{isk}_0(\ \lfloor q/2 \rceil,\ \mathbf{r}\ ),\quad \{\mathsf{isk}_\ell(\ \mathbf{r},\ \mathbf{x}[\ell]\mathbf{r}\ )\}_{\ell\in[L]};\\
\mathsf{ct}_f: &\quad \mathsf{ict}_0(\ \mathbf{0},\ \mathbf{T}^\intercal\mathbf{S}\ ),\quad \{\mathsf{ict}_\ell(\ \mathbf{W}_{\ell,0}^\intercal\mathbf{S},\ \mathbf{W}_{\ell,\times}^\intercal\mathbf{S}\ )\}_{\ell\in[L]},\\
&\quad \mathsf{ict}_0(\ \mu,\ \mathbf{w}_{\mathrm{out}}^\intercal\mathbf{S}\ ).
\end{aligned}$$

The decryption yields a noisy version of $\mathbf{W}^\intercal\mathbf{S}\mathbf{r}$'s as desired.

The security proof of ABE involves invoking the security of evasive IPFE to argue that all IPFE ciphertexts are hiding, which only applies if all noisy inner products are pseudorandom. Recall that in the ABE security game, there are multiple keys, and the inner products are

$$\begin{aligned}
(\text{for } \mathbf{x}_j)\qquad & (\mathbf{S}\mathbf{r}_j)^\intercal\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal,\quad \{(\mathbf{S}\mathbf{r}_j)^\intercal(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^\intercal\}_{\ell\in[L]},\\
& (\mathbf{S}\mathbf{r}_j)^\intercal\mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \mu\cdot\lfloor q/2\rceil.
\end{aligned}$$

Those are exactly the shares and secrets generated using randomness $\mathbf{S}\mathbf{r}_j$, but the components as-is are not pseudorandom, because $\mathbf{r}_j$'s are public, and the same secret matrix $\mathbf{S}$ and matrices $\mathbf{T}$, $\mathbf{W}$'s are reused in all shares for different $\mathbf{x}_j$'s. Hopefully, if we could somehow change the randomness from $\mathbf{S}\mathbf{r}_j$ to $(\mathbf{S}\mathbf{r}_j + \mathbf{e}_{\mathrm{r},j}^\intercal)$ with independent Gaussian $\mathbf{e}_{\mathrm{r},j}$'s, then by LWE (for secret $\mathbf{S}$), the rerandomization generates pseudorandomness, i.e., $\mathbf{S}\mathbf{r}_j + \mathbf{e}_{\mathrm{r},j}^\intercal \approx \$$, and then so are the shares. We show that how to achieve this in two ways.

Short Secret Sharing. If we require that the secret sharing be short, i.e., the matrices $\mathbf{w}_{\mathrm{out}}$, $\mathbf{T}$, $\mathbf{W}_{\ell,\mathbf{x}[\ell]}$'s have sufficiently small norm, then thanks to flooding by $\mathbf{e}_{\mathrm{t},j}$'s,

$$\{(\mathbf{S}\mathbf{r}_j)^\intercal\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal\}_{j\in[J]} \approx_{\mathrm{s}} \{(\mathbf{S}\mathbf{r}_j + \mathbf{e}'_{\mathrm{t},j})^\intercal\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal\}_{j\in[J]}.$$

The same holds for all the shares and secrets, achieving rerandomization.

Evasive IPFE with Structured Noises. Alternatively, we can augment the notion of evasive IPFE to include structured noises to work with arbitrary noisy LSSS. Instead of having only small noises, the noisy inner products now look like $(\mathbf{u}^\intercal\mathbf{V} + \mathbf{e}^\intercal\mathbf{G} + (\mathbf{e}')^\intercal)$, where $\mathbf{V}$ is associated with the secret key. Correspondingly, security holds if the noisy inner products with fresh noises $\mathbf{e}'$ and $\mathbf{e}$ are pseudorandom.

To use such evasive IPFE with structured noises to rerandomize secret shares, we need an additional trick. The problem is that the structured noises are of the

form $\mathbf{e}^\intercal\mathbf{G}$ instead of $\mathbf{e}^\intercal\mathbf{W}_{\ell,\mathbf{x}_j[\ell]}$, and the $\mathbf{e}$ here is different for each decryption (e.g., for computing the shares corresponding to $\mathbf{x}_j[1]$ and $\mathbf{x}_j[2]$ for the *same* j). To get consistent $\mathbf{e}_j^\intercal\mathbf{W}_{\ell,\mathbf{x}[\ell]}$ for each $\mathbf{x}_j$ (key) independent of ℓ (input bit index), we perform a noisy secret sharing

$$\underbrace{(\mathbf{r}_j^\intercal\mathbf{S}+\mathbf{e}_j^\intercal)\mathbf{W}_{\ell,\mathbf{x}_j[\ell]}}=\underbrace{\mathbf{r}_j^\intercal\mathbf{S}\mathbf{W}_{\ell,\mathbf{x}_j[\ell]}-(\mathbf{r}'_j)^\intercal\mathbf{Q}\mathbf{G}^{-1}(\mathbf{W}_{\ell,\mathbf{x}_j[\ell]})}+((\underbrace{\mathbf{r}'_j)^\intercal\mathbf{Q}+\mathbf{e}_j^\intercal\mathbf{G}})\mathbf{G}^{-1}(\mathbf{W}_{\ell,\mathbf{x}_j[\ell]}),$$

where the wavy underlines indicate small noises (without $\mathbf{G}$ structure) and each noisy term on the right-hand side can be computed using evasive IPFE, without and with structured noises respectively. Importantly, each ABE key only uses one structured noise $\mathbf{e}_j$, and the shares for the L input bits use the same $\mathbf{e}_j$ in $\mathbf{e}_j^\intercal\mathbf{W}_{\ell,\mathbf{x}_j[\ell]}$.

Short Noisy Linear Secret Sharing for Polynomial-Size Circuits. Our construction of short noisy linear secret sharing scheme for circuits is inspired by the arithmetic garbling scheme of [AIK11] and deals with the circuit gate by gate. In this paragraph, we will use "labels" for "shares" as it is customary for circuits (noisy linear secret sharing is a partially hiding garbling).

To begin, each gate i is associated with two random low-norm matrices $\mathbf{W}_{i,0},\mathbf{W}_{i,\times}$, which we call the label function of gate i. The label (before randomization) encoding some value x is $\mathbf{W}_{i,x}=\mathbf{W}_{i,0}+x\mathbf{W}_{i,\times}$, and its randomized version is $(\mathbf{s}^\intercal\mathbf{W}_{i,x}+\mathbf{e}_{i,x}^\intercal)$. Suppose gate i has its input wires connected to gates i_1,i_2 and their output values are x,x_1,x_2, we can sample a random low-norm matrix $\mathbf{U}_i$ and rewrite

$$\begin{aligned}\mathbf{W}_{i,0}+(x_1+x_2)\mathbf{W}_{i,\times}&=(\mathbf{U}_i+x_1\mathbf{W}_{i,\times})+(\mathbf{W}_{i,0}-\mathbf{U}_i+x_2\mathbf{W}_{i,\times}),\\ \mathbf{W}_{i,0}+x_1x_2\mathbf{W}_{i,\times}&=x_2(\mathbf{U}_i+x_1\mathbf{W}_{i,\times})+(\mathbf{W}_{i,0}-x_2\mathbf{U}_i),\end{aligned}$$

depending on whether gate i is an addition or multiplication gate. We denote the decomposed terms that are affine in x_1 as $\widetilde{\mathbf{W}}_{i_1,x_1}$, and those affine in x_2 as $\widetilde{\mathbf{W}}_{i_2,x_2}$ — note that $\widetilde{\mathbf{W}}_{i,x}$ is contributed by all the fan-outs of gate i. The randomized, noisy $\mathbf{s}^\intercal\mathbf{W}_{i,x}$ can be obtained from the noisy versions of $\mathbf{s}^\intercal\widetilde{\mathbf{W}}_{i_1,x_1}$ and $\mathbf{s}^\intercal\widetilde{\mathbf{W}}_{i_2,x_2}$. We call $\widetilde{\mathbf{W}}$'s and $\mathbf{s}^\intercal\widetilde{\mathbf{W}}$'s the expanded labels, and consequently, $\mathbf{W}$'s and $\mathbf{s}^\intercal\mathbf{W}$'s the shrunken labels.

For each gate i, we also sample and publish a low-norm matrix $\mathbf{R}_i$, and publish

in $\mathbf{s}^\intercal\mathbf{T}$ (the "garbled table entries" for gate i)

$$\mathbf{s}^\intercal(\mathbf{W}_{i,0}\mathbf{R}_i + \widetilde{\mathbf{W}}_{i,0}) + (\mathbf{e}'_{i,0})^\intercal, \qquad \mathbf{s}^\intercal(\mathbf{W}_{i,\times}\mathbf{R}_i + \widetilde{\mathbf{W}}_{i,\times}) + (\mathbf{e}'_{i,\times})^\intercal.$$

The values of $\mathbf{R}_i$'s and noisy $\mathbf{s}^\intercal\mathbf{T}$ are given regardless of the input values. Note that these values enable computing the expanded label $\mathbf{s}^\intercal\widetilde{\mathbf{W}}_{i,x}$ from the shrunken label $\mathbf{s}^\intercal\mathbf{W}_{i,x}$ for any value x that gate i happens to take. We also include $\mathbf{s}^\intercal(\mathbf{W}_{|C|,0}\mathbf{R}_{|C|} + \mathbf{w}_{\text{out}})$ to facilitate the recovery of secret when $C(\mathbf{x}) = 0$. The shares consist of the input labels as well as the garbled table entries for all gates.

Evaluation also proceeds gate by gate, starting from the input gates. For each gate i with value x, the evaluation procedure recovers the shrunken label, then use $\mathbf{R}_i$ and the garbled table to recover the expanded label, which is later used to recover the shrunken labels for gates taking gate i as input. Eventually, if the output is 0, we approximately recover $\mathbf{s}^\intercal\mathbf{w}_{\text{out}}$ from $\mathbf{s}^\intercal\mathbf{W}_{|C|,0}$ and $\mathbf{s}^\intercal(\mathbf{W}_{|C|,0}\mathbf{R}_{|C|} + \mathbf{w}_{\text{out}})$ and $\mathbf{R}_{|C|}$.

We sketch the security proof. In the first step, we replace all the noisy $\mathbf{s}^\intercal\mathbf{W}_{i,0}$'s and $\mathbf{s}^\intercal\mathbf{W}_{i,\times}$'s by random $\boldsymbol{\delta}_{i,0}^\intercal$ and $\boldsymbol{\delta}_{i,\times}^\intercal$, using the flipped LWE assumption. Then, for a particular input $\mathbf{x}$, we simulate $\boldsymbol{\delta}_{i,0}$ and $\boldsymbol{\delta}_{i,\times}$ using $\boldsymbol{\delta}_{i,x_i}$ (call this "active") and $\boldsymbol{\delta}_{i,\times}$, where x_i is the value of gate i when evaluated on input $\mathbf{x}$. The key observation here is that in such hybrids, the active expanded labels and garbled table entries no longer depend on the "$\times$" ones. To argue the input labels and the garbled table are jointly pseudorandom, we deal with active labels/table and "$\times$" ones in two passes. In the first pass, we work on the "$\times$" labels/table bottom-up (output gate to input gate). This step is computational and involves flipped LWE for public matrix $\mathbf{R}$ — for each "$\times$" table $(\boldsymbol{\delta}_{i,\times}^\intercal\mathbf{R}_i + \widetilde{\boldsymbol{\delta}}_{i,\times}^\intercal)$, the secret $\boldsymbol{\delta}_{i,\times}^\intercal$ will be absent everywhere else when the proof comes to this table, so flipped LWE ensures that the first term is pseudorandom, hiding $\widetilde{\boldsymbol{\delta}}_{i,\times}$, thus removing information about the "$\times$" labels of its fan-outs. In the second pass, we work on the active labels/table top-down. This step is statistical using the information-theoretic garbling security (so technically security is already in place when we finish the first pass), and the top-down order is just a pedagogy device for understanding the proof.

Succinct CP-ABE for Circuits. A (non-short) noisy linear secret sharing scheme can be distilled from the celebrated KP-ABE for circuits due to [BGG+14], as observed

by [AWY20]. The scheme is succinct — the sizes of $\mathbf{R}$, $\mathbf{T}$, $\mathbf{W}$'s only depend on the depth, not the size of the circuit. Combined with evasive IPFE with structured noises, we immediately obtain succinct CP-ABE.

Secret Sharing and ABE for DFA. The existing linear secret sharing scheme for DFA (see [Wat12] or Construction 4.8) can be cast as a noisy linear secret sharing and is short. The definition of noisy linear secret sharing inherently considers fixed input length, while in (say, CP-) ABE for uniform computation like DFA, we would like a ciphertext (tied to DFA) to potentially authorize a key (tied to input) of arbitrary length, and conversely a key to be accepted by a ciphertext of arbitrarily many states. In the previous scheme, ct_f determines the input length it accepts, so the same construction cannot be directly used.

We follow the paradigm in [Wat12] and Section 4.6 and exploit the locality of DFA computation. Namely, each DFA step is simply reading an input bit from a fixed location and transitioning the previous state to the next. At a very high level, the most important shares (related to state transitions) are of the form $(\mathbf{s}_\ell^\intercal \mathbf{\Gamma}_{\mathbf{x}[\ell]} - \mathbf{s}_{\ell-1}^\intercal)$, where $\mathbf{\Gamma}_0, \mathbf{\Gamma}_1$ are determined by the DFA state transition function, and $\mathbf{s}_0, \ldots, \mathbf{s}_L$, each of dimension $\mathfrak{Q}$ (number of states of DFA), together constitutes the secret sharing randomness $\mathbf{s}$ of length $L\mathfrak{Q}$. The locality of shares in $\mathbf{s}$ is inherited from the locality of DFA computation, and the subtraction expresses the state transition. Evaluation is a telescoping sum that chains the state transitions.

In (again, say CP-) ABE, we set $\mathbf{s}$ to be LWE samples jointly generated by the keys (size dependent on L) and the ciphertexts (size dependent on $\mathfrak{Q}$), $\mathbf{s}_\ell = \underbrace{\mathbf{S}\mathbf{r}_\ell}$ for $\mathbf{r}_\ell$'s in each key and $\mathbf{S}$ in each ciphertext. Continuing with the high level form of the most important shares, we set

$$\begin{aligned} \mathsf{sk}_{\mathbf{x}} : \quad & \{\mathsf{isk}(\quad (1-\mathbf{x}[\ell])\mathbf{r}_\ell, \quad \mathbf{x}[\ell]\mathbf{r}_\ell, \quad \mathbf{r}_{\ell-1} \quad)\}_{\ell\in[L]}, \\ \mathsf{ct}_{\mathrm{DFA}} : \quad & \mathsf{ict}(\qquad\quad \mathbf{\Gamma}_0^\intercal \mathbf{S}, \qquad \mathbf{\Gamma}_1^\intercal \mathbf{S}, \quad -\mathbf{S} \quad). \end{aligned}$$

Upon decryption, they compute all the shares, which reveal the secret approximately if the computation is accepting. For security, we first argue that $\underbrace{\mathbf{S}\mathbf{r}}$'s are pseudorandom, hence the shares are indistinguishable to created using fresh randomness, then we

invoke the (information-theoretic) security of DFA secret sharing, and conclude ABE security from evasive IPFE security and pseudorandomness of the shares.

7.2.3 Instantiation of Evasive IPFE

Our candidate evasive IPFE schemes draw inspiration heavily from known IPFE constructions, and we proceed in three steps.

Basic Scheme. The basic scheme without identities nor structured noises is reminiscent to the schemes due to [ABDP15,ALS16]. The master public key consists of two matrices $\mathbf{B}, \mathbf{A}$, the master secret key is a trapdoor of $\mathbf{B}$, and

$$\mathsf{isk}(\mathbf{v}) = \mathbf{B}^{-1}(\mathbf{A}\mathbf{G}^{-1}(\mathbf{v})), \qquad \mathsf{ict}(\mathbf{u}) = (\mathbf{d}^\intercal\mathbf{B}, \mathbf{d}^\intercal\mathbf{A} + \mathbf{u}^\intercal\mathbf{G}) = (\mathbf{c}_1^\intercal, \mathbf{c}_2^\intercal),$$

where $\mathbf{B}^{-1}(\mathbf{p})$ is a low-norm vector $\mathbf{k}$ satisfying $\mathbf{B}\mathbf{k} = \mathbf{p}$, which can be sampled using the master secret key. For correctness, observe that $\mathbf{c}_2^\intercal\mathbf{G}^{-1}(\mathbf{v}) - \mathbf{c}_1^\intercal\mathbf{k}$ approximates $\mathbf{u}^\intercal\mathbf{v}$. We study the security of this simple scheme extensively in Section 7.4. Most importantly, its security can be reduced to a variant of evasive LWE if the plaintext vectors $\mathbf{u}$ are uniformly random over a publicly known subspace.

Identity-Based Scheme. We borrow techniques from pairing to generically achieve identity-based schemes. In pairing-based IPFE, identity binding can be achieved using a single-identity scheme (denoted with primes) by adding a blinding factor in the decryption result. Let φ, ψ be random, then

$$\begin{aligned} \mathsf{isk}_{\mathsf{id}'}(\mathbf{v}) &: \quad \mathsf{isk}'(\ \mathbf{v},\ \psi,\ \mathsf{id}' \cdot \psi\ ), \\ \mathsf{ict}_{\mathsf{id}}(\mathbf{u}) &: \quad \mathsf{ict}'(\ \mathbf{u},\ \mathsf{id} \cdot \varphi,\ -\varphi\ ). \end{aligned}$$

The decryption outcome is $(\mathbf{u}^\intercal\mathbf{v} + (\mathsf{id} - \mathsf{id}')\varphi\psi)$. When the identities match, the result is the desired. When they do not match, the result is blinded by $\varphi\psi$, which is pseudorandom in the exponent (e.g., by DDH). The similar idea can be applied to evasive IPFE, except the blinding factor is $\boldsymbol{\varphi}^\intercal\boldsymbol{\psi}$, which is pseudorandom by LWE. To see security, observe that non-matching identities, due to $\boldsymbol{\varphi}^\intercal\boldsymbol{\psi}$, yield pseudorandom inner products concerned by the underlying scheme, whereas matching inner products are already pseudorandom by premise.

Scheme with Structured Noises. The structured noise $\mathbf{g}^\intercal \otimes \mathbf{e}^\intercal$ is simply $\mathbf{e}$ multiplied by powers of two. To obtain them, we change the format of $\mathbf{d}^\intercal\mathbf{B}$ in ict into

$$\mathbf{d}^\intercal\mathbf{B}_0 + 2^0\mathbf{e}^\intercal + (\mathbf{e}'_0)^\intercal, \quad \ldots, \quad \mathbf{d}^\intercal\mathbf{B}_{K-1} + 2^{K-1}\mathbf{e}^\intercal + (\mathbf{e}'_{K-1})^\intercal,$$

where $\mathbf{e}, \mathbf{e}'_0, \ldots, \mathbf{e}'_{K-1}$ are Gaussian noises. For $\mathbf{u}^\intercal\mathbf{V} + \mathbf{g}^\intercal \otimes e''$, we would like to transform the previous blocks into

$$\mathbf{d}^\intercal\mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_0) + 2^0 e'' + e'''_0, \quad \ldots, \quad \mathbf{d}^\intercal\mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_{K-1}) + 2^{K-1}e'' + e'''_{K-1},$$

so that they will cancel $\mathbf{d}^\intercal\mathbf{A}\mathbf{G}^{-1}(\mathbf{V})$ from $(\mathbf{d}^\intercal\mathbf{A} + \mathbf{u}^\intercal\mathbf{G})\mathbf{G}^{-1}(\mathbf{V})$ while attaching $\mathbf{g}^\intercal \otimes e''$ to it. This can be done by finding low-norm $\mathbf{K}$ such that $\mathbf{B}_k\mathbf{K} = \mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_k)$ for all k, or equivalently $\widetilde{\mathbf{B}}\mathbf{K} = \widetilde{\mathbf{P}}$ for

$$\widetilde{\mathbf{B}} = \begin{pmatrix} \mathbf{B}_0 \\ \vdots \\ \mathbf{B}_{K-1} \end{pmatrix}, \qquad \widetilde{\mathbf{P}} = \begin{pmatrix} \mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_1) \\ \vdots \\ \mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_k) \end{pmatrix}.$$

We sample $\widetilde{\mathbf{B}}$ with trapdoor to be able to sample $\mathbf{K}$. Some more care is required to support decryption with either structured noises or not. Relying on a modified version of evasive LWE, termed *evasive learning with structured errors* assumption, our candidate is again secure for $\mathbf{u}$ random over public subspace, sufficient for our application of succinct ABE.

7.3 Preliminaries

Table 7.1 explains select single-letter symbols used in this chapter.

Auxiliary Gadget Vector/Matrix. Recall that $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\intercal$ for the usual gadget, where $\mathbf{g}$ is of dimension m/n. Given $K \in \mathbb{N}$, we let (note the different order in the Kronecker product)

$$\widetilde{\mathbf{g}} = (2^0, \ldots, 2^{K-1})^\intercal \qquad \text{and} \qquad \widetilde{\mathbf{G}} = \widetilde{\mathbf{g}}^\intercal \otimes \mathbf{I}.$$

When $2^K \geq q$, the vector $\widetilde{\mathbf{G}}^{-1}(\mathbf{p})$ is again the bit decomposition of $\mathbf{p} \in \mathbb{Z}_q^\star$, arranged so that $\widetilde{\mathbf{G}}\widetilde{\mathbf{G}}^{-1}(\mathbf{p}) = \mathbf{p}$. The notation $\widetilde{\mathbf{G}}^{-1}(\cdot)$ extends to matrices column-wise.

Table 7.1. Select single-letter symbols in this chapter.

context	symbol	meaning
generic	λ, κ	computational/statistical security parameter
	$\beta, \delta, \boldsymbol{\delta}, \boldsymbol{\Delta}$	challenge bit, random scalar/vector/matrix
ABE	J, j	key count, index
	$\mathbf{r}, \mathbf{s}$	rerandomizer, garbling randomness
lattices	n, m, q, ρ	dimension, dimension, modulus, hardness exponent
	$\mathbf{g}, \mathbf{G}$	gadget vector/matrix
	$\mathbf{p}, \mathbf{P}, \mathbf{k}, \mathbf{K}$	image vector/matrix, preimage vector/matrix
	$\mathbf{A}, \mathbf{B}, \tau$	matrix, matrix with trapdoor, trapdoor
	$\mathcal{D}, \sigma$	discrete Gaussian distribution, width
	$e, \mathbf{e}, \mathbf{E}, B$	noise scalar/vector/matrix, noise bound
	$\mathbf{d}, r$	(evasive) LWE secret, sampler randomness
	K, k	structured noise length/index
IPFE	$\mathcal{I}, q, Z, z$	identity space (family), modulus, dimension, index
	B, σ	error bound, noise width
	$\mathbf{v}, \mathbf{u}^\intercal$	key/plaintext vector
	J, I, j, i	key/ciphertext count, key/ciphertext index
	$\boldsymbol{\psi}, \boldsymbol{\varphi}^\intercal$	key/ciphertext randomness for identity binding
garbling	$\mathcal{F}, f, \mathbf{x}, L, \ell$	function family, function, input, length, index
	$q, n, \mathbf{s}$	modulus, randomness dimension, garbling randomness
	$w, \mathbf{w}, \mathbf{W}$	secret scalar/vector, label (function) vector/matrix
	$\mathbf{t}, \mathbf{T}$	garbled table vector/matrix
	$\mathbf{R}$	reusable information
	$B, \mathbf{e}$	noise bound, garbling noise
circuits	C, d	circuit, depth (bound)
	x, M	wire value, wire value bound
	$\mathbf{x}, L, \ell$	input, length, index
DFA	$\Gamma, \mathfrak{Q}, \mathfrak{q}, \boldsymbol{\Gamma}$	DFA, number of states, state, transition matrix
	$\boldsymbol{\iota}_1, \boldsymbol{\xi}, \overline{L}$	initial/rejection state vector, input length bound
	$\mathbf{x}, L, \ell$	input, length, index

Evasive LWE Assumption. Conceptually, we consider two knowledge assumptions — evasive learning with errors (evasive LWE) and evasive learning with *structured* errors (not abbreviated). The former is a special case of the latter, and we formulate the definitions as such.

Assumption 7.1 (evasive learning with structured errors). Let $\mathcal{S}(1^\lambda; r)$ be an algorithm that, given randomness r, outputs

$$1^n, 1^m, 1^K, q, 1^{n'}, 1^J, 1^I, 1^{m'}, \qquad \sigma_{-1}, \quad \sigma_{\mathsf{post}} \geq \sigma_{\mathsf{pre}} \geq 0,$$
$$\widetilde{\mathbf{P}} \in \mathbb{Z}_q^{n(K+1)\times J}, \qquad \{\mathbf{A}_{1,i}, \mathbf{A}_{0,i}\}_{i\in[I]} \quad (\mathbf{A}_{1,i} \in \mathbb{Z}_q^{n\times m'},\ \mathbf{A}_{0,i} \in \mathbb{Z}_q^{n'\times m'}),$$

where $m \geq m_0(n(K+1), q)$ and $\sigma_{-1} \geq \sigma_0(n(K+1), m)$ are constrained by Lemma 2.3. Suppose

$$(\widetilde{\mathbf{B}}, \tau) \xleftarrow{\$} \mathsf{TrapGen}(1^{n(K+1)}, 1^m, q), \qquad \mathbf{K} \xleftarrow{\$} \mathsf{SampD}(\widetilde{\mathbf{B}}, \tau, \widetilde{\mathbf{P}}, \sigma_{-1}),$$
$$\mathbf{d}_0 \xleftarrow{\$} \mathbb{Z}_q^{n'}; \quad \mathbf{e}_{\mathsf{pre},i,0} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}^{m'}, \quad \mathbf{e}_{\mathsf{post},i,0} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{post}}}^{m'}, \quad \boldsymbol{\delta}_{i,0} \xleftarrow{\$} \mathbb{Z}_q^{m'},$$
$$\mathbf{d}_i \xleftarrow{\$} \mathbb{Z}_q^{n}, \quad \mathbf{e}_{\mathsf{pre},i,1} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}^{m(K+1)}, \quad \mathbf{e}_{\mathsf{post},i,1} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{post}}}^{m(K+1)}, \quad \boldsymbol{\delta}_{i,1} \xleftarrow{\$} \mathbb{Z}_q^{m(K+1)},$$
$$\mathbf{e}_{\mathsf{pre},i,2} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}^{J(K+1)}, \quad \mathbf{e}_{\mathsf{pre},i,4} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}^{J}, \quad \boldsymbol{\delta}_{i,2} \xleftarrow{\$} \mathbb{Z}_q^{J(K+1)},$$
$$\mathbf{e}_{\mathsf{pre},i,3} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}^{m}, \quad \mathbf{e}_{\mathsf{post},i,3} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{post}}}^{m}, \quad \text{for all } i \in [I].$$

Let $\widetilde{\mathbf{g}}^\intercal = (2^0, 2^1, \ldots, 2^{K-1})^\intercal$ and

$$\widetilde{\mathbf{B}} = (\mathbf{B}_{-1}^\intercal, \mathbf{B}_0^\intercal, \ldots, \mathbf{B}_{K-1}^\intercal)^\intercal, \qquad \mathbf{B} = (\mathbf{B}_{-1}, \mathbf{B}_0, \ldots, \mathbf{B}_{K-1}),$$
$$\widetilde{\mathbf{P}} = (\mathbf{P}_{-1}^\intercal, \mathbf{P}_0^\intercal, \ldots, \mathbf{P}_{K-1}^\intercal)^\intercal, \qquad \mathbf{P} = (\mathbf{P}_{-1}, \mathbf{P}_0, \ldots, \mathbf{P}_{K-1}),$$

where $\mathbf{B}_k \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{P}_k \in \mathbb{Z}_q^{n\times J}$. The precondition $\mathsf{evLWsE}_{\mathsf{pre}}^{\mathcal{S}}$ is

$$\left\{\begin{pmatrix} 1^\lambda, \\ r, \\ \mathbf{B}, \end{pmatrix}, \begin{Bmatrix} \mathbf{d}_i^\intercal \mathbf{P} + (\mathbf{0}_{1\times J}, \widetilde{\mathbf{g}}^\intercal \otimes \mathbf{e}_{\mathsf{pre},i,4}^\intercal) + \mathbf{e}_{\mathsf{pre},i,2}^\intercal, \\ \mathbf{d}_i^\intercal \mathbf{B} + (\mathbf{0}_{1\times m}, \widetilde{\mathbf{g}}^\intercal \otimes \mathbf{e}_{\mathsf{pre},i,3}^\intercal) + \mathbf{e}_{\mathsf{pre},i,1}^\intercal, \\ \mathbf{d}_i^\intercal \mathbf{A}_{1,i} + \mathbf{d}_0^\intercal \mathbf{A}_{0,i} + \mathbf{e}_{\mathsf{pre},i,0}^\intercal \end{Bmatrix}_{i\in[I]}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \\ r, \\ \mathbf{B}, \end{pmatrix}, \begin{Bmatrix} \boldsymbol{\delta}_{i,2}^\intercal, \\ \boldsymbol{\delta}_{i,1}^\intercal, \\ \boldsymbol{\delta}_{i,0}^\intercal \end{Bmatrix}_{i\in[I]}\right\}_{\lambda\in\mathbb{N}}.$$

The postcondition $\mathsf{evLWsE}_{\mathsf{post}}^{\mathcal{S}}$ is

$$\left\{\begin{pmatrix} 1^\lambda, r, \\ \mathbf{B}, \mathbf{K}, \end{pmatrix}, \begin{Bmatrix} \mathbf{d}_i^\intercal \mathbf{B} + (\mathbf{0}_{1\times m}, \widetilde{\mathbf{g}}^\intercal \otimes \mathbf{e}_{\mathsf{post},i,3}^\intercal) + \mathbf{e}_{\mathsf{post},i,1}^\intercal, \\ \mathbf{d}_i^\intercal \mathbf{A}_{1,i} + \mathbf{d}_0^\intercal \mathbf{A}_{0,i} + \mathbf{e}_{\mathsf{post},i,0}^\intercal \end{Bmatrix}_{i\in[I]}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, r, \\ \mathbf{B}, \mathbf{K}, \end{pmatrix}, \begin{Bmatrix} \boldsymbol{\delta}_{i,1}^\intercal, \\ \boldsymbol{\delta}_{i,0}^\intercal \end{Bmatrix}_{i\in[I]}\right\}_{\lambda\in\mathbb{N}}.$$

The *evasive learning with structured errors assumption* states that $\mathsf{evLWsE}^{\mathcal{S}}_{\mathrm{pre}}$ implies $\mathsf{evLWsE}^{\mathcal{S}}_{\mathrm{post}}$ for all efficient $\mathcal{S}$.

Assumption 7.2 (evasive LWE). A sampler $\mathcal{S}$ (Assumption 7.1) does not use structured noises if $\Pr[K = 0] = 1$, for which $\mathsf{evLWE}^{\mathcal{S}}_{\mathrm{pre}}$ (resp. $\mathsf{evLWE}^{\mathcal{S}}_{\mathrm{post}}$) is defined to be $\mathsf{evLWsE}^{\mathcal{S}}_{\mathrm{pre}}$ (resp. $\mathsf{evLWsE}^{\mathcal{S}}_{\mathrm{post}}$). The *evasive LWE assumption* states that $\mathsf{evLWE}^{\mathcal{S}}_{\mathrm{pre}}$ implies $\mathsf{evLWE}^{\mathcal{S}}_{\mathrm{post}}$ for all efficient $\mathcal{S}$ not using structured noises.

Note that for evasive LWE, $\widetilde{\mathbf{B}} = \mathbf{B} = \mathbf{B}_{-1}$ and $\widetilde{\mathbf{P}} = \mathbf{P} = \mathbf{P}_{-1}$.

Remark 7.1 (public-coin samplers). As in [Wee22,WWW22], our evasive assumptions are only for public-coin samplers, which we enforce by providing the sampler randomness to the distinguisher. It is weaker than the versions [Tsa22,VWW22] used for witness encryption, which considers private-coin samplers and the distinguisher, instead of the sampler randomness, gets the auxiliary information, an additional output produced by the sampler. The public-coin assumption avoids obfuscation-based counterexamples, where $\mathbf{P}$ is sampled with a trapdoor and the auxiliary information contains an obfuscated program with the trapdoor hardwired.

Remark 7.2 (comparison with [Wee22,WWW22]). As suggested in [Wee22], the noise magnitude in the postcondition can be made larger than that in the precondition for a more conservative assumption. We can implement it by requiring the assumptions to hold only for samplers with a gap between σ_{post} and σ_{pre}. We additionally allow a worse Gaussian preimage $\mathbf{K}$. Other than the magnitudes of samples, our version of evasive LWE is stronger than the version in [Wee22], and is similar to but incomparable with that in [WWW22].

While appearing much more complex, our evasive learning with structured errors assumption follows the same rationale as all existing versions of public-coin evasive LWE. In evasive LWE (without structured noises), the basic idea is that given

$$\mathbf{K} \xleftarrow{\$} \mathbf{B}^{-1}(\mathbf{P}), \qquad \mathbf{d}^{\intercal}\mathbf{B} + \mathbf{e}_1^{\intercal}, \qquad \mathbf{d}^{\intercal}\mathbf{A} + \mathbf{e}_0^{\intercal},$$

the trapdoor preimages can only be meaningfully used to multiply the corresponding LWE samples, i.e., $(\mathbf{d}^{\intercal}\mathbf{B} + \mathbf{e}_1^{\intercal})\mathbf{K} = \mathbf{d}^{\intercal}\mathbf{P} + \mathbf{e}_1^{\intercal}\mathbf{K}$, and that moreover, the correlation

between $\mathbf{e}_1$ and $\mathbf{e}_1^\intercal\mathbf{K}$ is not useful to the attacker if the samples have no apparent relations when $\mathbf{e}_1^\intercal\mathbf{K}$ is replaced by fresh $\mathbf{e}_2^\intercal$. Therefore, the precondition requires that

$$\mathbf{d}^\intercal\mathbf{P} + \mathbf{e}_2^\intercal, \qquad \mathbf{d}^\intercal\mathbf{B} + \mathbf{e}_1^\intercal, \qquad \mathbf{d}^\intercal\mathbf{A} + \mathbf{e}_0^\intercal$$

be pseudorandom, and evasive LWE asserts that $(\mathbf{d}^\intercal\mathbf{B} + \mathbf{e}_1^\intercal, \mathbf{d}^\intercal\mathbf{A} + \mathbf{e}_0^\intercal)$ is pseudorandom in the presence of $\mathbf{K}$.

For evasive learning with structured errors, suppose we are given

$$\mathbf{K}, \qquad \{\mathbf{d}^\intercal\mathbf{B}_k + 2^k\mathbf{e}_3^\intercal + \mathbf{e}_{1,k}^\intercal\}_{0\le k\le K-1}, \qquad \mathbf{d}^\intercal\mathbf{A} + \mathbf{e}_0^\intercal,$$

where $\mathbf{B}_k\mathbf{K} = \mathbf{P}_k$ for all k. Again, the trapdoor preimages can only be meaningfully used to multiply the corresponding LWE samples to obtain

$$(\mathbf{d}^\intercal\mathbf{B}_k + 2^k\mathbf{e}_3^\intercal + \mathbf{e}_{1,k}^\intercal)\mathbf{K} = \mathbf{d}^\intercal\mathbf{P}_k + 2^k\mathbf{e}_3^\intercal\mathbf{K} + \mathbf{e}_{1,k}^\intercal\mathbf{K}.$$

Following the same rationale of "correlation among noises not helpful if samples with fresh noises are pseudorandom", we replace $\{\mathbf{e}_{1,k}^\intercal\mathbf{K}\}_k, \mathbf{e}_3^\intercal\mathbf{K}$ by independent $\{\mathbf{e}_{2,k}^\intercal\}_k, \mathbf{e}_4^\intercal$. This gives rise to the conditions in our assumption. It remains to figure out how to sample $\mathbf{K}$ simultaneously satisfying $\mathbf{B}_k\mathbf{K} = \mathbf{P}_k$ for all k. The equations can be rewritten as

$$\begin{pmatrix}\mathbf{B}_1\\ \vdots \\ \mathbf{B}_k\end{pmatrix}\mathbf{K} = \begin{pmatrix}\mathbf{P}_1\\ \vdots \\ \mathbf{P}_k\end{pmatrix},$$

so it suffices to sample the vertically blocked $\mathbf{B}_k$'s with trapdoor. We further incorporate a block ($\mathbf{B}_{-1}$ and $\mathbf{P}_{-1}$) without structured noises and arrive at Assumption 7.1.

7.4 Evasive Inner-Product Functional Encryption

We define identity-based IPFE with structured noises and its security. A scheme might not support every combination of moduli, identities, and noise structures, which can be signaled by the algorithms aborting.

Definition 7.1 ((IB-)evIPFE(-w/sn)). Let $\mathcal{I} = \{\mathcal{I}_{\lambda,q,K,Z}\}_{\lambda,q,K,Z\in\mathbb{N}}$ be a family of sets. An *(identity-based) evasive IPFE (with structured noises) scheme* for $\mathcal{I}$ consists of four efficient algorithms.

- $\mathsf{Setup}(1^\lambda, q, 1^K, 1^Z)$ takes as input the modulus q, the noise parameter K, and the dimension Z. It outputs a pair $(\mathsf{impk}, \mathsf{imsk})$ of master public/secret keys.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g})$ takes as input imsk, an identity $\mathsf{id} \in \mathcal{I}_{\lambda,q,K,Z}$, a key vector $\mathbf{v}_\mathfrak{o} \in \mathbb{Z}_q^Z$, and a key matrix $\mathbf{V}_\mathfrak{g} \in \mathbb{Z}_q^{Z\times K}$. It outputs a secret key isk for $(\mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g})$ under identity id.
- $\mathsf{Enc}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{u}^\intercal)$ takes as input impk, id, a noise structure $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$, and a plaintext vector $\mathbf{u} \in \mathbb{Z}_q^Z$. It outputs a ciphertext ict of $\mathbf{u}$ under id for noise structure $\mathfrak{s}$.
- $\mathsf{Dec}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}, \mathsf{isk}, \mathfrak{s}, \mathsf{ict})$ takes as input impk, id, $\mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}$, isk, $\mathfrak{s}$, ict. If ict is for noise structure $\mathfrak{s}$ and both isk and ict are under id, the decryption result is in $\mathbb{Z}_q$ and is supposed to approximate $\mathbf{u}^\intercal\mathbf{v}_\mathfrak{o}$ (when $\mathfrak{s} = \mathfrak{o}$) or $\mathbf{u}^\intercal\mathbf{V}_\mathfrak{g}$ (when $\mathfrak{s} = \mathfrak{g}$).

Let B be a function mapping (λ, q, K, Z) to natural numbers. The scheme is *B-correct for* $\mathfrak{o}$ if for all $\lambda, q, K, Z \in \mathbb{N}$, $\mathsf{id} \in \mathcal{I}_{\lambda,q,K,Z}$, $\mathbf{V}_\mathfrak{g} \in \mathbb{Z}_q^{Z\times K}$, $\mathbf{v}_\mathfrak{o}, \mathbf{u} \in \mathbb{Z}_q^Z$, it holds that

$$\Pr\left[\begin{array}{ll} (\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, q, 1^K, 1^Z) & \mathsf{Dec}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}, \mathsf{isk}, \mathfrak{o}, \mathsf{ict}) \\ \mathsf{isk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}) : & = \mathbf{u}^\intercal\mathbf{v}_\mathfrak{o} + e_\mathfrak{o} \quad \text{for some} \\ \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathfrak{o}, \mathbf{u}^\intercal) & e_\mathfrak{o} \in [-B(\lambda,q,K,Z), B(\lambda,q,K,Z)] \end{array}\right] = 1.$$

For *B-correctness for* $\mathfrak{g}$, the requirement is

$$\Pr\left[\begin{array}{ll} \text{—}''\text{—} \ \text{for impk, imsk, isk} & \mathsf{Dec}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}, \mathsf{isk}, \mathfrak{g}, \mathsf{ict}) \\ \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathfrak{g}, \mathbf{u}^\intercal) : & = \mathbf{u}^\intercal\mathbf{V}_\mathfrak{g} + \widetilde{\mathbf{g}}^\intercal \otimes e_{\mathfrak{g}\mathfrak{g}} + \mathbf{e}_{\mathfrak{g}\mathfrak{o}}^\intercal \quad \text{for some} \\ & e_{\mathfrak{g}\mathfrak{g}}, \mathbf{e}_{\mathfrak{g}\mathfrak{o}} \in [-B(\lambda,q,K,Z), B(\lambda,q,K,Z)]^\star \end{array}\right] = 1.$$

The scheme is *B-correct* if both are satisfied.

The correctness property is trivial when $\mathcal{I} = \emptyset$ or $B \geq q/2$. We are interested in large enough $\mathcal{I}$ and reasonably small B, for useful choices of q, Z. Jumping ahead, Constructions 7.2 and 7.3 support exponentially large $|\mathcal{I}|$ and q/B, which suffice for our ABE schemes.

When referring to specific schemes, if $|\mathcal{I}|$ is small, we remove "identity-based" (IB), and if they only support $K = 0$, we remove "with structured noises" (w/sn).

Batch Notations. It is convenient to consider KeyGen, Enc, Dec in batches. Let

$$\mathbf{V}_{\mathfrak{o}} = (\mathbf{v}_{1,\mathfrak{o}}, \ldots, \mathbf{v}_{J,\mathfrak{o}}) \in \mathbb{Z}_q^{Z\times J}, \quad \mathbf{V}_{\mathfrak{g}} = (\mathbf{V}_{1,\mathfrak{g}}, \ldots, \mathbf{V}_{J,\mathfrak{g}}) \in \mathbb{Z}_q^{Z\times KJ}, \quad \mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_I) \in \mathbb{Z}_q^{Z\times I}$$

be matrices (batches of vectors/matrices) and $\mathsf{id} \in \mathcal{I}$, $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$. We write

$$\mathsf{isk} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, \mathsf{id}, \mathbf{V}_{\mathfrak{o}}, \mathbf{V}_{\mathfrak{g}}) \qquad \text{and} \qquad \mathsf{ict} \xleftarrow{\$} \mathsf{Enc}(\mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{U}^\intercal)$$

for $\mathsf{isk} \leftarrow \{\mathsf{isk}_j\}_{j\in[J]}$ and $\mathsf{ict} \leftarrow \{\mathsf{ict}_i\}_{i\in[I]}$, where

$$\begin{aligned} \mathsf{isk}_j &\xleftarrow{\$} \mathsf{KeyGen}(\mathsf{imsk}, \mathsf{id}, \mathbf{v}_{j,\mathfrak{o}}, \mathbf{V}_{j,\mathfrak{g}}) && \text{for all } j \in [J], \\ \mathsf{ict}_i &\xleftarrow{\$} \mathsf{Enc}(\mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{u}_i^\intercal) && \text{for all } i \in [I], \end{aligned}$$

and $\mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{V}_{\mathfrak{o}}, \mathbf{V}_{\mathfrak{g}}, \mathsf{isk}, \mathfrak{s}, \mathsf{ict})$ means

$$\begin{pmatrix} \mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_{1,\mathfrak{o}}, \mathbf{V}_{1,\mathfrak{g}}, \mathsf{isk}_1, \mathfrak{s}, \mathsf{ict}_1) & \cdots & \mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_{J,\mathfrak{o}}, \mathbf{V}_{J,\mathfrak{g}}, \mathsf{isk}_J, \mathfrak{s}, \mathsf{ict}_1) \\ \vdots & \ddots & \vdots \\ \mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_{1,\mathfrak{o}}, \mathbf{V}_{1,\mathfrak{g}}, \mathsf{isk}_1, \mathfrak{s}, \mathsf{ict}_I) & \cdots & \mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_{J,\mathfrak{o}}, \mathbf{V}_{J,\mathfrak{g}}, \mathsf{isk}_J, \mathfrak{s}, \mathsf{ict}_I) \end{pmatrix},$$

which is approximately $\mathbf{U}^\intercal\mathbf{V}_{\mathfrak{s}}$. The mnemonic is that the decryption result is the plaintext matrix $\mathbf{U}^\intercal$ multiplied by the key matrix $\mathbf{V}_{\mathfrak{s}}$.

Security. We consider *very selective* (also known as *static*) security with respect to a sampler producing pseudorandom structurally noisy outputs.

Definition 7.2 (IB-evIPFE-w/sn security). Fix an IB-evIPFE-w/sn scheme (Definition 7.1). Let $\mathcal{S} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}})$ be two algorithms with the following syntax.

- $\mathcal{S}_{\mathrm{V}}(1^\lambda; r_{\mathrm{pub}})$, given randomness r_{pub}, outputs

$$\begin{aligned} &q, 1^K, 1^Z, \mathsf{ID} \subseteq \mathcal{I}_{\lambda,q,K,Z}, \{1^{J_{\mathsf{id}}}, 1^{I_{\mathsf{id},\mathfrak{o}}}, 1^{I_{\mathsf{id},\mathfrak{g}}}\}_{\mathsf{id}\in\mathsf{ID}}, && \sigma_{\mathrm{pre}} \geq 0, \\ &\{\mathbf{V}_{\mathsf{id},\mathfrak{o}}, \mathbf{V}_{\mathsf{id},\mathfrak{g}}\}_{\mathsf{id}\in\mathsf{ID}} && (\mathbf{V}_{\mathsf{id},\mathfrak{o}} \in \mathbb{Z}_q^{Z\times J_{\mathsf{id}}}, \mathbf{V}_{\mathsf{id},\mathfrak{g}} \in \mathbb{Z}_q^{Z\times KJ_{\mathsf{id}}}). \end{aligned}$$

- $\mathcal{S}_{\mathrm{U}}(1^\lambda, r_{\mathrm{pub}}; r_{\mathrm{priv}})$, given r_{pub} and additional randomness r_{priv}, outputs

$$\{\mathbf{U}_{\mathsf{id},\mathfrak{s}}^\intercal\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}} \qquad (\mathbf{U}_{\mathsf{id},\mathfrak{s}} \in \mathbb{Z}_q^{Z\times I_{\mathsf{id},\mathfrak{s}}}),$$

  where $q, Z, \mathsf{ID}, \{I_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$ agree with the output of $\mathcal{S}_{\mathrm{V}}(1^\lambda; r_{\mathrm{pub}})$.

Suppose $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, q, 1^K, 1^Z)$ and for all $\mathsf{id} \in \mathsf{ID}$, $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$,

$$\begin{aligned} &\mathbf{E}_{\mathsf{id},\mathfrak{o}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{J_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{o}}}, && \mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{g}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{J_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{g}}}, && \mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{o}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{KJ_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{g}}}, \\ &\mathbf{\Delta}_{\mathrm{pre},\mathsf{id},\mathfrak{o}} \xleftarrow{\$} \mathbb{Z}_q^{J_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{o}}}, && \mathbf{\Delta}_{\mathrm{pre},\mathsf{id},\mathfrak{g}} \xleftarrow{\$} \mathbb{Z}_q^{KJ_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{g}}}, \end{aligned}$$

$$\mathsf{isk}_{\mathsf{id}} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{imsk}, \mathsf{id}, \mathbf{V}_{\mathsf{id},\mathfrak{o}}, \mathbf{V}_{\mathsf{id},\mathfrak{g}}),$$
$$\mathsf{ict}^*_{\mathsf{id},\mathfrak{s}} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{U}^\intercal_{\mathsf{id},\mathfrak{s}}),$$
$$\mathbf{U}_{\$,\mathsf{id},\mathfrak{s}} \xleftarrow{\$} \mathbb{Z}_q^{Z \times I_{\mathsf{id},\mathfrak{s}}}, \qquad \mathsf{ict}^{\$}_{\mathsf{id},\mathfrak{s}} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{U}^\intercal_{\$,\mathsf{id},\mathfrak{s}}).$$

$\mathcal{S}$ has *pseudorandom structurally noisy inner products*, denoted $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$, if

$$\left\{\begin{pmatrix} 1^\lambda, \\ r_{\mathrm{pub}}, \end{pmatrix} \begin{Bmatrix} \mathbf{U}^\intercal_{\mathsf{id},\mathfrak{o}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} & + \mathbf{E}^\intercal_{\mathsf{id},\mathfrak{o}}, \\ \mathbf{U}^\intercal_{\mathsf{id},\mathfrak{g}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^\intercal \otimes \mathbf{E}^\intercal_{\mathsf{id},\mathfrak{g}\mathfrak{g}} & + \mathbf{E}^\intercal_{\mathsf{id},\mathfrak{g}\mathfrak{o}} \end{Bmatrix}_{\mathsf{id}\in\mathsf{ID}}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \\ r_{\mathrm{pub}}, \end{pmatrix} \begin{Bmatrix} \boldsymbol{\Delta}^\intercal_{\mathrm{pre},\mathsf{id},\mathfrak{o}}, \\ \boldsymbol{\Delta}^\intercal_{\mathrm{pre},\mathsf{id},\mathfrak{g}} \end{Bmatrix}_{\mathsf{id}\in\mathsf{ID}}\right\}_{\lambda\in\mathbb{N}}.$$

The scheme is *$\mathcal{S}$-secure*, denoted $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{post}}$, if

$$\left\{\begin{pmatrix} 1^\lambda, \\ r_{\mathrm{pub}}, \\ \mathsf{impk}, \end{pmatrix} \begin{Bmatrix} \mathsf{ict}^*_{\mathsf{id},\mathfrak{o}}, \\ \mathsf{ict}^*_{\mathsf{id},\mathfrak{g}}, \\ \mathsf{isk}_{\mathsf{id}} \end{Bmatrix}_{\mathsf{id}\in\mathsf{ID}}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \\ r_{\mathrm{pub}}, \\ \mathsf{impk}, \end{pmatrix} \begin{Bmatrix} \mathsf{ict}^{\$}_{\mathsf{id},\mathfrak{o}}, \\ \mathsf{ict}^{\$}_{\mathsf{id},\mathfrak{g}}, \\ \mathsf{isk}_{\mathsf{id}} \end{Bmatrix}_{\mathsf{id}\in\mathsf{ID}}\right\}_{\lambda\in\mathbb{N}}.$$

The scheme is *$\sigma_{\mathrm{pre},0}$-secure* if it is $\mathcal{S}$-secure for all efficient $\mathcal{S}$ such that $\sigma_{\mathrm{pre}} \le \sigma_{\mathrm{pre},0}$ and $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$ holds.

Remark 7.3 (public and private coins). It is necessary to make the sampling of $\mathbf{U}$ private-coin. Since IPFE does not protect $\mathbf{V}$ (the key vectors/matrices), the adversary is assumed to have full knowledge of $\mathbf{V}$. If $\mathbf{U}$ were sampled with public coins, the distinguisher against the ciphertexts (in $\mathsf{IPFEsec}_{\mathrm{post}}$) would know both $\mathbf{V}$ and $\mathbf{U}$, thus $\mathbf{U}^\intercal\mathbf{V}$. In the first distribution (containing ciphertexts of $\mathbf{U}$), decryption will yield the correct output, whereas it is not the case for the second distribution (containing ciphertexts of random vectors).

We formulate $\mathcal{S}_{\mathrm{V}}$ as public-coin and $\mathcal{S}_{\mathrm{U}}$ as fully private-coin, to avoid obvious counterexamples (e.g., obfuscation-based).

7.4.1 Basic Construction — Single-Identity, No Structured Noises

We first present a simple candidate for one identity without structured noises. The scheme is reminiscent to many existing IPFE constructions, e.g., in [ABDP15,ALS16, Wee17,Agr19,AP20] and Chapters 4 and 5.

Ingredients of Construction 7.1. We rely on Lemma 2.3 for correctness. The security of the scheme is elaborated in Sections 7.4.2 and 7.4.3.

Construction 7.1 (evIPFE). Our scheme works as follows.

- $\mathsf{Setup}(q, 1^K, 1^Z)$ takes as input q, K, Z. If $K \neq 0$, it sets $\mathcal{I}_{q,K,Z} = \emptyset$ and terminates (aborting case). Otherwise, the algorithm sets $\mathcal{I}_{q,K,Z} = \{1\}$, and *deterministically* picks n, m and $\sigma_{-1} \geq \sigma_0(n, m)$ appropriate for Lemma 2.3, as well as $\sigma_{\mathrm{post}}, \kappa$. It samples and sets

$$(\mathbf{B}, \tau) \xleftarrow{\$} \mathsf{TrapGen}(1^n, 1^m, q), \qquad \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times Z\lceil \log_2 q \rceil},$$
$$\mathsf{impk} = (1^n, 1^m, q, 1^Z, \sigma_{-1}, \sigma_{\mathrm{post}}, 1^\kappa, \mathbf{A}, \mathbf{B}), \qquad \mathsf{imsk} = (\mathsf{impk}, \tau).$$

  The algorithm outputs $(\mathsf{impk}, \mathsf{imsk})$.

- $\mathsf{KeyGen}(\mathsf{imsk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g})$ takes as input imsk, $\mathsf{id} = 1$, $\mathbf{v}_\mathfrak{o} \in \mathbb{Z}_q^Z$, and $\mathbf{V}_\mathfrak{g} = \perp$. It runs

$$\mathbf{k} \xleftarrow{\$} \mathsf{SampD}(\mathbf{B}, \tau, \mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_\mathfrak{o}), \sigma_{-1}).$$

  Here, $\mathbf{G}$ is of shape $Z \times Z\lceil \log_2 q \rceil$. The algorithm outputs $\mathsf{isk} = \mathbf{k}$.

- $\mathsf{Enc}(\mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{u}^\intercal)$ takes as input impk, $\mathsf{id} = 1$, $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$, and $\mathbf{u} \in \mathbb{Z}_q^Z$. If $\mathfrak{s} = \mathfrak{g}$, the algorithm outputs $\perp$ and terminates (the scheme does not produce any key capable of decrypting such a ciphertext). Otherwise, it samples

$$\mathbf{d} \xleftarrow{\$} \mathbb{Z}_q^n, \qquad \mathbf{e}_1 \xleftarrow{\$} \mathcal{D}^m_{\mathbb{Z}, \sigma_{\mathrm{post}}, \leq \sigma_{\mathrm{post}}\sqrt{\kappa}}, \qquad \mathbf{e}_0 \xleftarrow{\$} \mathcal{D}^{Z\lceil \log_2 q \rceil}_{\mathbb{Z}, \sigma_{\mathrm{post}}, \leq \sigma_{\mathrm{post}}\sqrt{\kappa}}.$$

  The algorithm outputs $\mathsf{ict} = (\mathbf{d}^\intercal\mathbf{B} + \mathbf{e}_1^\intercal,\ \mathbf{d}^\intercal\mathbf{A} + \mathbf{e}_0^\intercal + \mathbf{u}^\intercal\mathbf{G})$.

- $\mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}, \mathsf{isk}, \mathfrak{s}, \mathsf{ict})$ computes and outputs

$$\mathsf{ict} \cdot \begin{pmatrix} -\mathsf{isk} \\ \mathbf{G}^{-1}(\mathbf{v}_\mathfrak{o}) \end{pmatrix}.$$

Correctness. By Lemma 2.3, we have $\mathbf{B}\mathbf{k} = \mathbf{A}\mathbf{G}^{-1}(\mathbf{v}_\mathfrak{o})$ and $\|\mathbf{k}\| \leq \sigma_{-1}\sqrt{m}$, so

$$\mathsf{ict} \cdot \begin{pmatrix} -\mathsf{isk} \\ \mathbf{G}^{-1}(\mathbf{v}_\mathfrak{o}) \end{pmatrix} - \mathbf{u}^\intercal\mathbf{v}_\mathfrak{o} = \left(\mathbf{d}^\intercal\mathbf{B} + \mathbf{e}_1^\intercal,\ \mathbf{d}^\intercal\mathbf{A} + \mathbf{e}_0^\intercal + \mathbf{u}^\intercal\mathbf{G}\right) \begin{pmatrix} -\mathbf{k} \\ \mathbf{G}^{-1}(\mathbf{v}_\mathfrak{o}) \end{pmatrix} - \mathbf{u}^\intercal\mathbf{v}_\mathfrak{o}$$
$$= -\mathbf{e}_1^\intercal\mathbf{k} + \mathbf{e}_0^\intercal\mathbf{G}^{-1}(\mathbf{v}_\mathfrak{o}),$$

$$|-\mathbf{e}_1^\intercal \mathbf{k} + \mathbf{e}_0^\intercal \mathbf{G}^{-1}(\mathbf{v}_\mathfrak{v})| \le m \cdot \|\mathbf{e}_1\| \cdot \|\mathbf{k}\| + Z\lceil \log_2 q \rceil \cdot \|\mathbf{e}_0\| \cdot \|\mathbf{G}^{-1}(\mathbf{v}_\mathfrak{v})\|$$
$$\le \kappa^{1/2} \sigma_{\text{post}} (m^{3/2} \sigma_{-1} + Z\lceil \log_2 q \rceil).$$

Let B be the right-hand side, then Construction 7.1 is B-correct.

7.4.2 Security as an Assumption

Inspired by the evasive LWE assumption, we propose a new assumption that is closely related and translates to the security of Construction 7.1.

The precondition of evasive LWE rules out a combination of two attacking strategies [Wee22], attacks against LWE and zeroizing attacks. Alternatively, the evasive LWE assumption can be seen as a belief that *i)* the only meaningful way of using $\mathbf{B}^{-1}(\mathbf{P})$ is to multiply it to a noisy version of $\mathbf{d}^\intercal \mathbf{B}$, and that *ii)* the skewed noise attached to $\mathbf{d}^\intercal \mathbf{P}$ in $(\mathbf{d}^\intercal \mathbf{B} + \mathbf{e}^\intercal) \cdot \mathbf{B}^{-1}(\mathbf{P})$ does not give adversaries more advantage as long as there is *no apparent relation* among quantities related to $\mathbf{d}$ when the noises are not skewed. The pseudorandomness in the precondition of evasive LWE exactly captures the absence of apparent relation.

Similar to the belief captured by evasive LWE, we propose the following.

Assumption 7.3 (evasive IPFE). Let $\mathcal{S} = (\mathcal{S}_{\text{V}}, \mathcal{S}_{\text{U}})$ be two algorithms with the following syntax.

- $\mathcal{S}_{\text{V}}(1^\lambda; r_{\text{pub}})$, given randomness r_{pub}, outputs

  $$1^n, 1^m, q, 1^Z, 1^J, 1^I, \qquad \sigma_{-1}, \qquad \sigma_{\text{post}} \ge \sigma_{\text{pre}} \ge 0, \qquad \mathbf{V} \in \mathbb{Z}_q^{Z \times J},$$

  where $m \ge m_0(n, q)$ and $\sigma_{-1} \ge \sigma_0(n, m)$ are constrained by Lemma 2.3.

- $\mathcal{S}_{\text{U}}(1^\lambda, r_{\text{pub}}; r_{\text{priv}})$, given r_{pub} and additional randomness r_{priv}, outputs $\mathbf{U}^\intercal$, where $\mathbf{U} \in \mathbb{Z}_q^{Z \times I}$ and q, Z, I agree with the output of $\mathcal{S}_{\text{V}}(1^\lambda; r_{\text{pub}})$.

Suppose

$$(\mathbf{B}, \tau) \xleftarrow{\$} \mathsf{TrapGen}(1^n, 1^m, q), \qquad \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times Z\lceil \log_2 q \rceil}, \qquad \mathbf{D} \xleftarrow{\$} \mathbb{Z}_q^{n \times I},$$
$$\mathbf{\Delta}_0 \xleftarrow{\$} \mathbb{Z}_q^{Z\lceil \log_2 q \rceil \times I}, \qquad \mathbf{\Delta}_1 \xleftarrow{\$} \mathbb{Z}_q^{m \times I}, \qquad \mathbf{\Delta}_2 \xleftarrow{\$} \mathbb{Z}_q^{J \times I},$$

$$\mathbf{E}_{\mathrm{pre},0} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{Z\lceil\log_2 q\rceil\times I}, \qquad \mathbf{E}_{\mathrm{pre},1} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{m\times I}, \qquad \mathbf{E}_{\mathrm{pre},2} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{J\times I},$$
$$\mathbf{E}_{\mathrm{post},0} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{post}}}^{Z\lceil\log_2 q\rceil\times I}, \qquad \mathbf{E}_{\mathrm{post},1} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{post}}}^{m\times I},$$
$$\mathbf{K} \xleftarrow{\$} \mathsf{SampD}(\mathbf{B}, \tau, \mathbf{A}\mathbf{G}^{-1}(\mathbf{V}), \sigma_{-1}).$$

The *inner-product* precondition $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{ipre}}$ is

$$\{(1^\lambda, r_{\mathrm{pub}}, \mathbf{U}^\intercal\mathbf{V} + \mathbf{E}_{\mathrm{pre},2}^\intercal)\}_{\lambda\in\mathbb{N}} \approx \{(1^\lambda, r_{\mathrm{pub}}, \boldsymbol{\Delta}_2^\intercal)\}_{\lambda\in\mathbb{N}}.$$

The *stricter* precondition $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$ is

$$\{(1^\lambda, r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{D}^\intercal\mathbf{A}\mathbf{G}^{-1}(\mathbf{V}) + \mathbf{E}_{\mathrm{pre},2}^\intercal, \mathbf{D}^\intercal\mathbf{B} + \mathbf{E}_{\mathrm{pre},1}^\intercal, \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}_{\mathrm{pre},0}^\intercal + \mathbf{U}^\intercal\mathbf{G})\}_{\lambda\in\mathbb{N}}$$
$$\approx \{(1^\lambda, r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \quad \boldsymbol{\Delta}_2^\intercal \quad , \quad \boldsymbol{\Delta}_1^\intercal \quad , \quad \boldsymbol{\Delta}_0^\intercal \quad)\}_{\lambda\in\mathbb{N}}.$$

The postcondition $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{post}}$ is

$$\{(1^\lambda, r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \mathbf{D}^\intercal\mathbf{B} + \mathbf{E}_{\mathrm{post},1}^\intercal, \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}_{\mathrm{post},0}^\intercal + \mathbf{U}^\intercal\mathbf{G})\}_{\lambda\in\mathbb{N}}$$
$$\approx \{(1^\lambda, r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \quad \boldsymbol{\Delta}_1^\intercal \quad , \quad \boldsymbol{\Delta}_0^\intercal \quad)\}_{\lambda\in\mathbb{N}}.$$

The $(n_0, \sigma_{\mathrm{post},0})$-*evasive IPFE assumption* states that $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{ipre}}$ implies $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{post}}$ for all efficient $\mathcal{S}$ such that $n \geq n_0$ and $\sigma_{\mathrm{post}} \geq \sigma_{\mathrm{post},0}$. The *conservative evasive IPFE assumption* states that $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$ implies $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{post}}$ for all efficient $\mathcal{S}$.[72]

The rationale of public and private coins in the above formulation is similar to that (Remark 7.3) of evasive IPFE security definition.

Evasive IPFE Security from Evasive IPFE Assumption. There are only minor syntactical differences between the samplers of evasive IPFE (Assumption 7.3) and IPFEsec (Definition 7.2) instantiated for Construction 7.1. The evasive IPFE assumption implies the security of our basic scheme:

Theorem 7.1 (¶). *Construction 7.1 is* σ_{post}-secure *(Definition 7.2) under* $(n, \sigma_{\mathrm{post}})$*-evasive IPFE (Assumption 7.3), where* $n, \sigma_{\mathrm{post}}$ *are those picked by* Setup.

[72] For the first part, explicitly requiring lower bounds on $n, \sigma_{\mathrm{post}}$ is necessary, because $\mathsf{evIPFE}_{\mathrm{ipre}}$ could be unconditional when $J < Z$ yet $\mathsf{evIPFE}_{\mathrm{post}}$ is always computational. For the second part, $\mathsf{evIPFE}_{\mathrm{spre}}$ implies LWE, so the lower bounds on $n, \sigma_{\mathrm{post}}$ are implicit.

Proof (Theorem 7.1). Let $\mathcal{S} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}})$ be an efficient sampler (Definition 7.2) satisfying $\sigma_{\mathrm{pre}} \le \sigma_{\mathrm{post}}$ and $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$. Consider the following $\mathcal{S}' = (\mathcal{S}'_{\mathrm{V}}, \mathcal{S}'_{\mathrm{U}})$ for Assumption 7.3.

- $\mathcal{S}'_{\mathrm{V}}(r_{\mathrm{pub}})$ runs

$$(q, 1^0, 1^Z, \{1\}, 1^{J_1}, 1^{I_{1,\mathfrak{o}}}, 1^0, \sigma_{\mathrm{pre}}, \mathbf{V}_{1,\mathfrak{o}}, \bot) \leftarrow \mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}}),$$

  picks $n, m, \sigma_{-1}, \sigma_{\mathrm{post}}$ according to Construction 7.1, sets $J, I, \mathbf{V}$ to $J_1, I_{1,\mathfrak{o}}, \mathbf{V}_{1,\mathfrak{o}}$, and outputs[73]

$$1^n, 1^m, q, 1^Z, 1^J, 1^I, \qquad \sigma_{-1}, \qquad \sigma_{\mathrm{post}}, \sigma_{\mathrm{pre}}, \qquad \mathbf{V}.$$

- $\mathcal{S}'_{\mathrm{U}}(r_{\mathrm{pub}}; r_{\mathrm{priv}})$ runs $(\mathbf{U}^\intercal_{1,\mathfrak{o}}, \bot) \leftarrow \mathcal{S}_{\mathrm{U}}(r_{\mathrm{pub}}; r_{\mathrm{priv}})$ and outputs $\mathbf{U}^\intercal = \mathbf{U}^\intercal_{1,\mathfrak{o}}$.

Clearly, $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{ipre}}$ is simply $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$, which is assumed to hold. Therefore, by the $(n, \sigma_{\mathrm{post}})$-evasive IPFE assumption, $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{post}}$ holds, implying

$$\begin{aligned}
&(r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \ \mathbf{D}^\intercal\mathbf{B} + \mathbf{E}^\intercal_{\mathrm{post},1}, \ \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}^\intercal_{\mathrm{post},0} + \mathbf{U}^\intercal\mathbf{G} \qquad) \\
\approx\ &(r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \qquad \boldsymbol{\Delta}^\intercal_1 \qquad , \qquad \boldsymbol{\Delta}^\intercal_0 \qquad) \\
\equiv\ &(r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \qquad \boldsymbol{\Delta}^\intercal_1 \qquad , \qquad \boldsymbol{\Delta}^\intercal_0 + \mathbf{U}^\intercal_{\$}\mathbf{G} \qquad) \\
\approx\ &(r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \ \mathbf{D}^\intercal\mathbf{B} + \mathbf{E}^\intercal_{\mathrm{post},1}, \ \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}^\intercal_{\mathrm{post},0} + (\mathbf{U} + \mathbf{U}_{\$})^\intercal\mathbf{G}) \\
\equiv\ &(r_{\mathrm{pub}}, \mathbf{A}, \mathbf{B}, \mathbf{K}, \ \mathbf{D}^\intercal\mathbf{B} + \mathbf{E}^\intercal_{\mathrm{post},1}, \ \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}^\intercal_{\mathrm{post},0} + \mathbf{U}^\intercal_{\$}\mathbf{G} \qquad),
\end{aligned}$$

where $\mathbf{U}_{\$} = \mathbf{U}_{\$,1,\mathfrak{o}} \xleftarrow{\$} \mathbb{Z}_q^{Z \times I}$. Opening up $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{post}}$ for Construction 7.1, we see that

$$\underbrace{r_{\mathrm{pub}}}_{\mathcal{S}} = \underbrace{r_{\mathrm{pub}}}_{\mathcal{S}'}, \qquad \begin{aligned}
\mathsf{impk} &= \overbrace{(1^n, 1^m, q, 1^Z, \sigma_{-1}, \sigma_{\mathrm{post}}, 1^\kappa, \mathbf{A}, \mathbf{B})}^{\text{efficiently computable from } r_{\mathrm{pub}}}, \qquad \mathsf{isk}_1 = \mathbf{K}, \\
\mathsf{ict}^*_{1,\mathfrak{o}} &= (\mathbf{D}^\intercal\mathbf{B} + \mathbf{E}^\intercal_{\mathrm{post},1}, \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}^\intercal_{\mathrm{post},0} + \mathbf{U}^\intercal\mathbf{G}), \\
\mathsf{ict}^{\$}_{1,\mathfrak{o}} &= (\mathbf{D}^\intercal\mathbf{B} + \mathbf{E}^\intercal_{\mathrm{post},1}, \mathbf{D}^\intercal\mathbf{A} + \mathbf{E}^\intercal_{\mathrm{post},0} + \mathbf{U}^\intercal_{\$}\mathbf{G}),
\end{aligned}$$

except for noise truncation only up to a negligible statistical error by Lemma 2.1. Therefore, $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{post}}$ follows from the previous indistinguishability. □

[73] Without loss of generality, we may assume that $\mathcal{S}_{\mathrm{V}}$ always outputs $\mathsf{ID} = \{1\}$ and $I_{1,\mathfrak{g}} = 0$. The other cases can be converted to $\mathsf{ID} = \{1\}$ and $I_{1,\mathfrak{o}} = J_1 = 0$, with $\mathbf{U}^\intercal_{1,\mathfrak{g}}$ discarded, or both. The values of $n, m, \sigma_{-1}, \sigma_{\mathrm{post}}$ are determined by q, Z, so $\mathcal{S}'_{\mathrm{V}}$ does not rely on the randomness of Setup of Construction 7.1.

Full and Conservative Versions. The evasive IPFE assumption might appear custom-made for the scheme, rendering the security proof almost tautological, and it looks far-fetched from and bears little resemblance to the evasive LWE assumption. We justify it below.

Intuitively, the only meaningful way of using $\mathbf{B}^{-1}(\mathbf{A}\mathbf{G}^{-1}(\mathbf{V}))$ is to compute

$$-\underset{\sim}{\mathbf{D}^\intercal\mathbf{B}} \cdot \mathbf{B}^{-1}(\mathbf{A}\mathbf{G}^{-1}(\mathbf{V})) + (\underset{\sim}{\mathbf{D}^\intercal\mathbf{A}} + \mathbf{U}^\intercal\mathbf{G})\mathbf{G}^{-1}(\mathbf{V}) = \underset{\sim}{\mathbf{U}^\intercal\mathbf{V}},$$

where wavy underlines indicate noises. In this computation, it appears that *some* cancellation has happened so that $\mathbf{U}^\intercal\mathbf{V}$ is approximately recovered, which is different from evasive LWE, where *no* cancellation happens. But the precondition requires that the inner products with noise be pseudorandom, i.e., the "cancellation" only leads to pseudorandom results, morally equivalent to that there is no *actual* cancellation. It excludes the obvious zeroizing attacks.

The *conservative* evasive IPFE assumption is more along the lines of the evasive LWE assumption, for which the only addition is the ability to combine privately sampled value $\mathbf{U}$ with LWE sample $\mathbf{D}^\intercal\mathbf{A}$. It appears to be a smaller leap of faith than the full version, yet this is actually superfluous.

In fact, the full version does not demand more than the conservative version in suitable parameter regimes. We show that $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{ipre}}$ implies $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$ under the LWE assumption with noise super-polynomially smaller than σ_{pre}:

Theorem 7.2 (¶). *If $\mathcal{S}$ (Assumption 7.3) is efficient and $\mathsf{LWE}_{n,m+Z\lceil\log_2 q\rceil,q,\sigma_{\mathrm{help}}}$ (Assumption 2.2) holds for some $\sigma_{\mathrm{help}} \le \frac{\sigma_{\mathrm{pre}}}{2^{\kappa+6}\sqrt{\kappa}\cdot Z\lceil\log_2 q\rceil}$, where $q, Z, \sigma_{\mathrm{pre}}$ are those picked by $\mathcal{S}$, then $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{ipre}}$ implies $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$.*

We obtain a conservative version of Theorem 7.1 as a corollary.

Corollary 7.3 (¶). *Construction 7.1 is σ_{post}-secure (Definition 7.2) under* conservative *evasive IPFE (Assumption 7.3) and $\mathsf{LWE}_{n,m+Z\lceil\log_2 q\rceil,q,\sigma_{\mathrm{help}}}$ (Assumption 2.2) for some $\sigma_{\mathrm{help}} \le \frac{\sigma_{\mathrm{post}}}{(2^{\kappa+6}\sqrt{\kappa})^2 Z\lceil\log_2 q\rceil}$, where q, Z are those picked by $\mathcal{S}$ and $n, m, \sigma_{\mathrm{post}}, \kappa$ are those picked by* Setup.

The idea for the proof of Theorem 7.2 is

$$\begin{aligned}
& \left(\quad \underline{\underline{\mathbf{D}^\intercal \mathbf{A} \mathbf{G}^{-1}(\mathbf{V})}} \quad , \ \underline{\mathbf{D}^\intercal \mathbf{B}}, \ \underline{\mathbf{D}^\intercal \mathbf{A}} + \mathbf{U}^\intercal \mathbf{G} \right) \\
\text{(noise flooding)} \quad \approx_s & \left((\underline{\mathbf{D}^\intercal \mathbf{A}} + \mathbf{U}^\intercal \mathbf{G}) \mathbf{G}^{-1}(\mathbf{V}) - \underline{\underline{\mathbf{U}^\intercal \mathbf{V}}}, \ \underline{\mathbf{D}^\intercal \mathbf{B}}, \ \underline{\mathbf{D}^\intercal \mathbf{A}} + \mathbf{U}^\intercal \mathbf{G} \right) \\
\text{(LWE)} \quad \approx & \left(\quad \mathbf{\Delta}_0^\intercal \mathbf{G}^{-1}(\mathbf{V}) - \underline{\underline{\mathbf{U}^\intercal \mathbf{V}}}, \ \mathbf{\Delta}_1^\intercal , \quad \mathbf{\Delta}_0^\intercal \quad \right) \\
\text{(precondition)} \quad \approx & \left(\quad \mathbf{\Delta}_2^\intercal \quad , \ \mathbf{\Delta}_1^\intercal , \quad \mathbf{\Delta}_0^\intercal \quad \right),
\end{aligned}$$

where wavy underlines indicate noises. The rewriting in the first step is essentially IPFE simulation (see [Wee17,ACF+18,ALMT20] and Chapter 5), i.e., $\underline{\underline{\mathbf{D}^\intercal \mathbf{A} \mathbf{G}^{-1}(\mathbf{V})}}$ contains exactly the information of $\underline{\underline{\mathbf{U}^\intercal \mathbf{V}}}$, when given $\underline{\mathbf{D}^\intercal \mathbf{B}}$ and $(\underline{\mathbf{D}^\intercal \mathbf{A}} + \mathbf{U}^\intercal \mathbf{G})$.

Proof (Theorem 7.2). We consider the following hybrids.

- H_0 is the first distribution in $\mathsf{evIPFE}^{\mathcal{S}}_{\mathsf{spre}}$, i.e.,

$$\begin{aligned}
& r_{\mathsf{pub}}, \mathbf{A}, \mathbf{B}, \qquad \mathbf{D}^\intercal \mathbf{A} \mathbf{G}^{-1}(\mathbf{V}) + \mathbf{E}_{\mathsf{pre},2}^\intercal, \\
& \mathbf{D}^\intercal \mathbf{B} + \mathbf{E}_{\mathsf{pre},1}^\intercal, \qquad\qquad\qquad \mathbf{D}^\intercal \mathbf{A} + \mathbf{E}_{\mathsf{pre},0}^\intercal + \mathbf{U}^\intercal \mathbf{G}.
\end{aligned}$$

- In H_1, the matrix $\mathbf{B}$ is sampled uniformly at random, without a known trapdoor. By Lemma 2.3, we have $\mathsf{H}_0 \approx_s \mathsf{H}_1$.

- In H_2, helper noises are inserted and flooded by $\mathbf{E}_{\mathsf{pre}}$'s, i.e.,

$$\begin{aligned}
& r_{\mathsf{pub}}, \mathbf{A}, \mathbf{B}, \qquad \mathbf{D}^\intercal \mathbf{A} \mathbf{G}^{-1}(\mathbf{V}) + (\mathbf{E}_{\mathsf{pre},2}^\intercal + \mathbf{E}_{\mathsf{help},0}^\intercal \mathbf{G}^{-1}(\mathbf{V})), \\
& \mathbf{D}^\intercal \mathbf{B} + (\mathbf{E}_{\mathsf{pre},1} + \mathbf{E}_{\mathsf{help},1})^\intercal, \qquad \mathbf{D}^\intercal \mathbf{A} + \mathbf{U}^\intercal \mathbf{G} + (\mathbf{E}_{\mathsf{pre},0} + \mathbf{E}_{\mathsf{help},0})^\intercal,
\end{aligned}$$

  where the entries of $\mathbf{E}_{\mathsf{help}}$'s are sampled independently from $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{help}},\leq \sigma_{\mathsf{help}}\sqrt{\kappa}}$. Since $\|(\mathbf{G}^{-1}(\mathbf{V}))^\intercal\| \leq Z\lceil \log_2 q \rceil$, the entries of ($\mathbf{E}_{\mathsf{help}}$'s and) $\mathbf{E}_{\mathsf{help},0}^\intercal \mathbf{G}^{-1}(\mathbf{V})$ are bounded by $\sigma_{\mathsf{help}}\sqrt{\kappa} \cdot Z\lceil \log_2 q \rceil$. Together with $\sigma_{\mathsf{pre}} \geq 2^{\kappa+6}\sigma_{\mathsf{help}}\sqrt{\kappa} \cdot Z\lceil \log_2 q \rceil$, it follows from Lemma 2.2 that $\mathsf{H}_1 \approx_s \mathsf{H}_2$.

- In H_3, we change $\mathbf{E}_{\mathsf{pre},2}$ to $(-\mathbf{E}_{\mathsf{pre},2})$, rearrange some terms, and derive the noisy $\mathbf{D}^\intercal \mathbf{A} \mathbf{G}^{-1}(\mathbf{V})$ from the IPFE decryption relation, i.e.,

$$\begin{aligned}
& r_{\mathsf{pub}}, \mathbf{A}, \mathbf{B}, \qquad (\mathbf{D}^\intercal \mathbf{A} + \mathbf{E}_{\mathsf{help},0}^\intercal + \mathbf{U}^\intercal \mathbf{G}) \mathbf{G}^{-1}(\mathbf{V}) - (\mathbf{U}^\intercal \mathbf{V} + \mathbf{E}_{\mathsf{pre},2}^\intercal), \\
& (\mathbf{D}^\intercal \mathbf{B} + \mathbf{E}_{\mathsf{help},1}^\intercal) + \mathbf{E}_{\mathsf{pre},1}^\intercal, \qquad (\mathbf{D}^\intercal \mathbf{A} + \mathbf{E}_{\mathsf{help},0}^\intercal + \mathbf{U}^\intercal \mathbf{G}) + \mathbf{E}_{\mathsf{pre},0}^\intercal.
\end{aligned}$$

  Since $\mathbf{E}_{\mathsf{pre},2}$ follows a symmetric distribution, $\mathsf{H}_2 \equiv \mathsf{H}_3$.

- In H_4, the $\mathbf{E}_{\mathrm{help}}$'s are no longer truncated, i.e., their entries are sampled independently from $\mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{help}}}$. By Lemma 2.1, it holds that $\mathsf{H}_3 \approx_{\mathrm{s}} \mathsf{H}_4$.
- In H_5, we invoke $\mathsf{LWE}_{n,m+Z\lceil\log_2 q\rceil,q,\sigma_{\mathrm{help}}}$ for secrets[74] $\mathbf{D}$ and public matrix $(\mathbf{B},\mathbf{A})$, and the distribution becomes

$$r_{\mathrm{pub}},\mathbf{A},\mathbf{B},\quad (\boldsymbol{\Delta}_0^\intercal+\mathbf{U}^\intercal\mathbf{G})\mathbf{G}^{-1}(\mathbf{V})-(\mathbf{U}^\intercal\mathbf{V}+\mathbf{E}_{\mathrm{pre},2}^\intercal),$$
$$\boldsymbol{\Delta}_1^\intercal+\mathbf{E}_{\mathrm{pre},1}^\intercal,\qquad (\boldsymbol{\Delta}_0^\intercal+\mathbf{U}^\intercal\mathbf{G})+\mathbf{E}_{\mathrm{pre},0}^\intercal.$$

  By the LWE assumption, $\mathsf{H}_4 \approx \mathsf{H}_5$.[75]
- In H_6, we make $\boldsymbol{\Delta}$'s absorb terms added to them, i.e.,

$$r_{\mathrm{pub}},\mathbf{A},\mathbf{B},\quad (\boldsymbol{\Delta}_0^\intercal-\mathbf{E}_{\mathrm{pre},0}^\intercal)\mathbf{G}^{-1}(\mathbf{V})-(\mathbf{U}^\intercal\mathbf{V}+\mathbf{E}_{\mathrm{pre},2}^\intercal),\quad \boldsymbol{\Delta}_1^\intercal,\quad \boldsymbol{\Delta}_0^\intercal.$$

  Clearly, $\mathsf{H}_5 \equiv \mathsf{H}_6$.
- In H_7, we invoke $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{ipre}}$ so the distribution becomes

$$r_{\mathrm{pub}},\mathbf{A},\mathbf{B},\quad (\boldsymbol{\Delta}_0^\intercal-\mathbf{E}_{\mathrm{pre},0}^\intercal)\mathbf{G}^{-1}(\mathbf{V})-\boldsymbol{\Delta}_2^\intercal,\quad \boldsymbol{\Delta}_1^\intercal,\quad \boldsymbol{\Delta}_0^\intercal.$$

  By the assumption, $\mathsf{H}_6 \approx \mathsf{H}_7$.
- H_8 is the second distribution in $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$, i.e.,

$$r_{\mathrm{pub}},\mathbf{A},\mathbf{B},\quad \boldsymbol{\Delta}_2^\intercal,\quad \boldsymbol{\Delta}_1^\intercal,\quad \boldsymbol{\Delta}_0^\intercal.$$

  Clearly, $\mathsf{H}_7 \equiv \mathsf{H}_8$.

By a hybrid argument, $\mathsf{H}_0 \approx \mathsf{H}_8$, i.e., $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$ holds. □

Proof (Corollary 7.3). Given an efficient $\mathcal{S}$ (Definition 7.2) satisfying $\sigma_{\mathrm{pre}} \le \sigma_{\mathrm{post}}$ and $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$, we do the following:

1. construct $\mathcal{S}'$ (Assumption 7.3) *similarly* to the proof of Theorem 7.1 and show that $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{ipre}}$ holds,

[74] Technically, there are a series of hybrids over the columns of $\mathbf{D}$.

[75] This step requires that $\mathcal{S}$ (part of the reduction algorithm) be efficient.

2. apply Theorem 7.2 to $\mathcal{S}'$ for proving $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{spre}}$,
3. derive $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{post}}$ from the conservative evasive IPFE assumption, and
4. conclude $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{post}}$ in the same way as the final parts of the proof of Theorem 7.1.

Let $\sigma^{\mathcal{S}}_{\mathrm{pre}}, \sigma^{\mathcal{S}'}_{\mathrm{pre}}$ be the σ_{pre} of $\mathcal{S}, \mathcal{S}'$. We set

$$\sigma^{\mathcal{S}'}_{\mathrm{pre}} = \begin{cases} \sigma^{\mathcal{S}}_{\mathrm{pre}}, & \text{if } \sigma^{\mathcal{S}}_{\mathrm{pre}} \geq \frac{\sigma_{\mathrm{post}}}{2^{\kappa+6}\sqrt{\kappa}}; \\ \sigma_{\mathrm{post}}, & \text{otherwise.} \end{cases}$$

In the first case, $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{ipre}}$ is just $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$ and Theorem 7.2 applies because

$$\sigma_{\mathrm{help}} \leq \frac{\sigma_{\mathrm{post}}}{(2^{\kappa+6}\sqrt{\kappa})^2 Z \lceil \log_2 q \rceil} \leq \frac{\sigma^{\mathcal{S}'}_{\mathrm{pre}}}{2^{\kappa+6}\sqrt{\kappa} \cdot Z \lceil \log_2 q \rceil}.$$

In the second case, we cannot set $\sigma^{\mathcal{S}'}_{\mathrm{pre}}$ to $\sigma^{\mathcal{S}}_{\mathrm{pre}}$, which might not be large enough compared to σ_{help} per the premise of Theorem 7.2. Instead, with our choice, $\mathsf{evIPFE}^{\mathcal{S}'}_{\mathrm{ipre}}$ follows from $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$ by noise truncation and flooding (Lemmas 2.1 and 2.2), and Theorem 7.2 again applies. □

We remark that for security against $\mathcal{S}$ with particular σ_{pre}'s (e.g., $\sigma_{\mathrm{pre}} = \sigma_{\mathrm{post}}$), the bound on σ_{help} can be tuned accordingly (e.g., without the quadratic blow-up).

7.4.3 Security for Restricted Samplers from Evasive LWE

We prove that Construction 7.1 is $\mathcal{S}$-secure for a restricted class of samplers under the evasive LWE assumption. These samplers are efficient, satisfy $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$, and their $\mathcal{S}_{\mathrm{U}}$'s output a uniformly random vector in a publicly known subspace. This will suffice for our ABE applications.

Definition 7.3 (restricted IB-evIPFE-w/sn security). A sampler $\mathcal{S} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}})$ per Definition 7.2 is *restricted* if $\mathcal{S}_{\mathrm{U}}(1^\lambda, r_{\mathrm{pub}}; r_{\mathrm{priv}})$ works as follows.

1. It uses a fixed deterministic algorithm $\mathcal{S}_{\mathrm{A0}}$ to compute

$$(1^{n'}, \{\mathbf{A}_{\mathsf{id},\mathfrak{s},0,i}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{v},\mathfrak{g}\},i\in[I_{\mathsf{id},\mathfrak{s}}]}) \leftarrow \mathcal{S}_{\mathrm{A0}}(1^\lambda, r_{\mathrm{pub}}),$$

where $\mathbf{A}_{\mathsf{id},\mathfrak{s},0,i} \in \mathbb{Z}_q^{n' \times Z}$ for all $\mathsf{id} \in \mathsf{ID}$, $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$, $i \in [I_{\mathsf{id},\mathfrak{s}}]$, and the values of q, Z, ID, $\{I_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$ agree with the output of $\mathcal{S}_{\mathrm{V}}(1^\lambda; r_{\mathrm{pub}})$.

2. It uses r_{priv} to sample $\mathbf{d}_0 \xleftarrow{\$} \mathbb{Z}_q^{n'}$. The algorithm outputs $\{\mathbf{U}_{\mathsf{id},\mathfrak{s}}^\intercal\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$

$$\text{for} \quad \mathbf{U}_{\mathsf{id},\mathfrak{s}} = (\mathbf{u}_{\mathsf{id},\mathfrak{s},1}, \ldots, \mathbf{u}_{\mathsf{id},\mathfrak{s},I_{\mathsf{id},\mathfrak{s}}})$$
$$\text{with} \quad \mathbf{u}_{\mathsf{id},\mathfrak{s},i}^\intercal = \mathbf{d}_0^\intercal \mathbf{A}_{\mathsf{id},\mathfrak{s},0,i} \quad \text{for all } \mathsf{id} \in \mathsf{ID},\ \mathfrak{s} \in \{\mathfrak{o},\mathfrak{g}\},\ i \in [I_{\mathsf{id},\mathfrak{s}}].$$

An IB-evIPFE-w/sn scheme (Definition 7.1) is *restricted-$\sigma_{\mathrm{pre},0}$-secure* if it is $\mathcal{S}$-secure (Definition 7.2) for all efficient restricted $\mathcal{S}$ such that $\sigma_{\mathrm{pre}} \le \sigma_{\mathrm{pre},0}$ and $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}}$ holds.

Parallel to restricted security, we formulate the following assumption.

Assumption 7.4 (restricted conservative evasive IPFE). A sampler $\mathcal{S} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}})$ for Assumption 7.3 is *restricted* if $\mathcal{S}_{\mathrm{U}}(1^\lambda, r_{\mathrm{pub}}; r_{\mathrm{priv}})$ works as follows.

1. It uses a fixed deterministic algorithm $\mathcal{S}_{\mathrm{A0}}$ to compute

$$(1^{n'}, \{\mathbf{A}'_{0,i}\}_{i\in[I]}) \leftarrow \mathcal{S}_{\mathrm{A0}}(1^\lambda, r_{\mathrm{pub}}) \quad \text{with} \quad \mathbf{A}'_{0,i} \in \mathbb{Z}_q^{n' \times Z} \text{ for all } i \in [I],$$

   where q, Z, I agree with the output of $\mathcal{S}_{\mathrm{V}}(1^\lambda; r_{\mathrm{pub}})$.

2. It uses r_{priv} to sample $\mathbf{d}_0 \xleftarrow{\$} \mathbb{Z}_q^{n'}$. The algorithm outputs $\mathbf{U}^\intercal$ for

$$\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_I) \quad \text{and} \quad \mathbf{u}_i^\intercal = \mathbf{d}_0^\intercal \mathbf{A}'_{0,i} \text{ for all } i \in [I].$$

The *restricted conservative evasive IPFE assumption* states that $\mathsf{evIPFE}_{\mathrm{spre}}^{\mathcal{S}}$ implies $\mathsf{evIPFE}_{\mathrm{post}}^{\mathcal{S}}$ for all efficient restricted $\mathcal{S}$.

Lemma 7.4. *The sampler (Assumption 7.3) constructed in the proof of Corollary 7.3 is* restricted *(Assumption 7.4) if so is the original sampler (Definitions 7.2 and 7.3).*

Lemma 7.4 can be easily verified. The following implication holds:

Theorem 7.5 (¶). *Evasive LWE (Assumption 7.2) implies restricted conservative evasive IPFE (Assumption 7.4).*

Combining Theorem 7.2, the proof of Corollary 7.3, Lemma 7.4, and Theorem 7.5:

Corollary 7.6. *Construction 7.1 is* restricted-σ_{post}-secure *(Definition 7.3) under evasive LWE (Assumption 7.2) and* $\mathsf{LWE}_{n,m+Z\lceil\log_2 q\rceil,q,\sigma_{\mathrm{help}}}$ *(Assumption 2.2) for some*

$$\sigma_{\mathrm{help}} \leq \frac{\sigma_{\mathrm{post}}}{(2^{\kappa+6}\sqrt{\kappa})^2 Z\lceil\log_2 q\rceil},$$

where q, Z *are those picked by* $\mathcal{S}$ *and* $n, m, \sigma_{\mathrm{post}}, \kappa$ *are those picked by* Setup.

Proof (Theorem 7.5). Let $\mathcal{S} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}})$ be any efficient restricted sampler with $\mathcal{S}_{\mathrm{A0}}$ (Assumptions 7.3 and 7.4) such that $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$ holds. Construct the following $\mathcal{S}'(r)$ for Assumption 7.2. It parses $r = (r_{\mathrm{pub}}, r'_{\mathrm{pub}})$ and runs

$$(1^n, 1^m, q, 1^Z, 1^J, 1^I, \sigma_{-1}, \sigma_{\mathrm{post}}, \sigma_{\mathrm{pre}}, \mathbf{V}) \leftarrow \mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}}),$$
$$(1^{n'}, \{\mathbf{A}'_{0,i}\}_{i\in[I]}) \leftarrow \mathcal{S}_{\mathrm{A0}}(r_{\mathrm{pub}}).$$

The algorithm samples $\mathbf{A}$ uniformly random over $\mathbb{Z}_q^{n\times Z\lceil\log_2 q\rceil}$ using r'_{pub} in the straightforward manner,[76] and sets $K = 0$ and $m' = Z\lceil\log_2 q\rceil$. It outputs

$$1^n, 1^m, 1^K, q, 1^{n'}, 1^J, 1^I, 1^{m'}, \qquad \sigma_{-1}, \quad \sigma_{\mathrm{post}}, \quad \sigma_{\mathrm{pre}},$$
$$\widetilde{\mathbf{P}} = \mathbf{A}\mathbf{G}^{-1}(\mathbf{V}), \quad \{\mathbf{A}_{1,i}, \mathbf{A}_{0,i}\}_{i\in[I]} \quad (\text{with } \mathbf{A}_{1,i} = \mathbf{A} \text{ and } \mathbf{A}_{0,i} = \mathbf{A}'_{0,i}\mathbf{G}),$$

By setting

$$\mathbf{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_I), \qquad \mathbf{\Delta}_\alpha = (\boldsymbol{\delta}_{1,\alpha}, \ldots, \boldsymbol{\delta}_{I,\alpha}) \text{ for all } \alpha \in \{0,1,2\},$$
$$\mathbf{E}_{p,\alpha} = (\mathbf{e}_{p,1,\alpha}, \ldots, \mathbf{e}_{p,I,\alpha}) \text{ for all } (p,\alpha) \in \left\{\begin{array}{r}(\mathrm{pre},2),(\mathrm{pre},1),(\mathrm{pre},0),\\(\mathrm{post},1),(\mathrm{post},0)\end{array}\right\},$$

the components in $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre/post}}, \mathsf{evLWE}^{\mathcal{S}'}_{\mathrm{pre/post}}$ are perfectly lined up, except that $\mathbf{A}$ corresponds to r'_{pub}. It is readily verified that $\mathsf{evLWE}^{\mathcal{S}'}_{\mathrm{pre}}$ follows from $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{spre}}$. By the evasive LWE assumption, $\mathsf{evLWE}^{\mathcal{S}'}_{\mathrm{post}}$ holds, implying $\mathsf{evIPFE}^{\mathcal{S}}_{\mathrm{post}}$.[77] □

[76] Precisely speaking, the distribution of r'_{pub} conditioned on $\mathbf{A}$ should be efficiently sampleable from $\mathbf{A}$ so that it does not contain a trapdoor of $\mathbf{A}$ or otherwise impedes security. If we insist that r'_{pub} be a binary string, we can make r'_{pub} at least $\max_{r_{\mathrm{pub}}}((\kappa + \lceil\log_2 q\rceil)\cdot nZ\lceil\log_2 q\rceil)$ bits long and set $\mathbf{A}[i,j]$ to be its $((i-1)+(j-1)n+1)^{\mathrm{st}}$ chunk of $(\kappa+\lceil\log_2 q\rceil)$ bits interpreted as an integer in binary then reduced modulo q, incurring a statistical error of $2^{-\kappa}nZ\lceil\log_2 q\rceil$.

[77] The reduction algorithms reshape components between vectors and matrices, and either reversely sample r'_{pub} from $\mathbf{A}$ or computes $\mathbf{A}$ from r'_{pub}.

7.4.4 Identity-Based Scheme

Construction 7.1 only supports a single identity. We employ pairing-based techniques to generically obtain an exponentially large identity space. The core idea is to generate identity-based keys and ciphertexts using a single-identity IPFE (denoted with primes), e.g., when not considering structured noises,

$$\left.\begin{array}{lllll} \mathsf{isk}_{\mathsf{id}'}(\mathbf{v}): & \mathsf{isk}'(& \mathbf{v}, & \psi, & \mathsf{id}'\cdot\psi \;\;) \\ \mathsf{ict}_{\mathsf{id}}\,(\mathbf{u}): & \mathsf{ict}'(& \mathbf{u}, & \mathsf{id}\cdot\varphi, & -\varphi \;\;) \end{array}\right\} \xrightarrow{\mathsf{Dec}'} \mathbf{u}^\intercal\mathbf{v} + (\mathsf{id}-\mathsf{id}')\varphi\psi$$

so that decryption yields $\mathbf{u}^\intercal\mathbf{v}$ if $\mathsf{id}=\mathsf{id}'$ and the result is masked by a blinding factor $(\mathsf{id}-\mathsf{id}')\varphi\psi$ if $\mathsf{id}\neq\mathsf{id}'$. Intuitively, the masks are pseudorandom by the DDH assumption (this idea can be materialized in various ways). In our adaptation, we use the LWE assumption to create the pseudorandom blinding factors.

We note that the transformation of [GKW16] compressing exponentially many copies of a single-identity scheme into an identity-based scheme is not as desirable as the upcoming construction. Security of evIPFE(-w/sn) is knowledge-type. Since Definition 7.2 requires fully private-coin sampling of plaintext vectors (see Remark 7.3), it cannot be applied to two copies unless the two copies only encrypt independent plaintext vectors — but in our applications, plaintext vectors under different identities are correlated. Moreover, even if we consider an alternative definition with arbitrary (efficiently computable) auxiliary information about the plaintext vectors, we would have to assume evasive LWE with very tight pre/postcondition distinguishing gap for it to be applied super-constantly many times (corresponding to the number of identities; used in CP-ABE for circuits).

Ingredients of Construction 7.2. Let $\mathsf{IPFE}'=(\mathsf{Setup}',\mathsf{KeyGen}',\mathsf{Enc}',\mathsf{Dec}')$ be an underlying evIPFE(-w/sn) such that $1\in\mathcal{I}'_{q,K,Z}$ (e.g., Constructions 7.1 and 7.3).

Construction 7.2 (IB-evIPFE(-w/sn)). Our scheme works as follows.

- $\mathsf{Setup}(q,1^K,1^Z)$ takes as input q,K,Z. If q is not a prime or $1\notin\mathcal{I}'_{q,K,Z}$, the algorithm sets $\mathcal{I}_{q,K,Z}=\varnothing$ and terminates (aborting case). Otherwise, it sets

$\mathcal{I}_{q,K,Z} = [q]$[78] and *deterministically* picks n.[79] The algorithm runs

$$(\mathsf{impk}', \mathsf{imsk}') \xleftarrow{\$} \mathsf{Setup}'(q, 1^K, 1^{Z+2n})$$

and outputs $\mathsf{impk} = (q, 1^K, 1^Z, 1^n, \mathsf{impk}')$ and $\mathsf{imsk} = (q, 1^K, 1^Z, 1^n, \mathsf{imsk}')$.

- $\mathsf{KeyGen}(\mathsf{imsk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g})$ takes as input imsk, $\mathsf{id} \in [q]$, $\mathbf{v}_\mathfrak{o} \in \mathbb{Z}_q^Z$, and $\mathbf{V}_\mathfrak{g} \in \mathbb{Z}_q^{Z \times K}$. It samples $\boldsymbol{\psi}_\mathfrak{o} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\boldsymbol{\Psi}_\mathfrak{g} \xleftarrow{\$} \mathbb{Z}_q^{n \times K}$, runs

$$\mathsf{isk}' \xleftarrow{\$} \mathsf{KeyGen}'\left(\mathsf{imsk}', 1, \begin{pmatrix} \mathbf{v}_\mathfrak{o} \\ \boldsymbol{\psi}_\mathfrak{o} \\ \mathsf{id} \cdot \boldsymbol{\psi}_\mathfrak{o} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_\mathfrak{g} \\ \boldsymbol{\Psi}_\mathfrak{g} \\ \mathsf{id} \cdot \boldsymbol{\Psi}_\mathfrak{g} \end{pmatrix}\right),$$

  and outputs $\mathsf{isk} = (\boldsymbol{\psi}_\mathfrak{o}, \boldsymbol{\Psi}_\mathfrak{g}, \mathsf{isk}')$.

- $\mathsf{Enc}(\mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{u}^\mathsf{T})$ takes as input imsk, $\mathsf{id} \in [q]$, $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$, and $\mathbf{u} \in \mathbb{Z}_q^Z$. It samples $\boldsymbol{\varphi} \xleftarrow{\$} \mathbb{Z}_q^n$, and runs and outputs

$$\mathsf{ict} = \mathsf{ict}' \xleftarrow{\$} \mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, (\mathbf{u}^\mathsf{T}, \mathsf{id} \cdot \boldsymbol{\varphi}^\mathsf{T}, -\boldsymbol{\varphi}^\mathsf{T})).$$

- $\mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}, \mathsf{isk}, \mathfrak{s}, \mathsf{ict})$ runs and outputs

$$\mathsf{Dec}'\left(\mathsf{impk}', 1, \begin{pmatrix} \mathbf{v}_\mathfrak{o} \\ \boldsymbol{\psi}_\mathfrak{o} \\ \mathsf{id} \cdot \boldsymbol{\psi}_\mathfrak{o} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_\mathfrak{g} \\ \boldsymbol{\Psi}_\mathfrak{g} \\ \mathsf{id} \cdot \boldsymbol{\Psi}_\mathfrak{g} \end{pmatrix}, \mathsf{isk}', \mathfrak{s}, \mathsf{ict}'\right).$$

Correctness. Suppose IPFE′ is $B'(q', K', Z')$-correct, then for noise structure $\mathfrak{o}$,

$$\begin{aligned} \mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g}, \mathsf{isk}, \mathfrak{o}, \mathsf{ict}) &= \mathbf{u}^\mathsf{T} \cdot \mathbf{v}_\mathfrak{o} + (\mathsf{id} \cdot \boldsymbol{\varphi}^\mathsf{T}) \cdot \boldsymbol{\psi}_\mathfrak{o} + (-\boldsymbol{\varphi}^\mathsf{T}) \cdot (\mathsf{id} \cdot \boldsymbol{\psi}_\mathfrak{o}) + e_\mathfrak{o} \\ &= \mathbf{u}^\mathsf{T} \mathbf{v}_\mathfrak{o} + e_\mathfrak{o} \end{aligned}$$

for some $e_\mathfrak{o} \in [-B'(q, K, Z + 2n), B'(q, K, Z + 2n)]$, and similarly for the case of $\mathfrak{g}$. In summary, Construction 7.2 is B-correct for $B(q, K, Z) = B'(q, K, Z + 2n)$.

---

[78] Due to homomorphic noise growth and noise flooding, q is super-polynomial in this chapter. When q is polynomially large, the identity space can be enlarged using full-rank difference hash [ABB10].

[79] This construction is generic. The quantity n here is unrelated to those in Construction 7.1 or 7.3, although they can be set to the same value.

Security. We state and prove the security of Construction 7.2.

Theorem 7.7 (¶). *Suppose the underlying* IPFE′ *is* $\sigma_{\mathrm{pre},0}$-secure *(Definition 7.2) and* $\mathsf{LWE}_{n,J_1',q,\sigma_{\mathrm{help}}}$ *(Assumption 2.2) holds for some* $\sigma_{\mathrm{help}} \le \frac{\sigma_{\mathrm{pre},0}}{(2^{\kappa+6}\sqrt{\kappa})^2}$,[80] *then Construction 7.2 is also* $\sigma_{\mathrm{pre},0}$-secure, *where* q *and* $J_1' = \sum_{\mathsf{id}\in\mathsf{ID}} J_{\mathsf{id}}$ *are picked by* $\mathcal{S}$ *and* n *is picked by* Setup.

Suppose instead IPFE′ *is* restricted-$\sigma_{\mathrm{pre},0}$-secure *(Definition 7.3) while keeping the other conditions intact, then Construction 7.2 is also* restricted-$\sigma_{\mathrm{pre},0}$-secure.

The proof works by arguing that when the noisy inner products for the matching identities are pseudorandom, so are the inner products concerned by IPFE′, e.g., for $\mathfrak{o}$,

$$\begin{cases} \mathsf{id} \neq \mathsf{id}' : & \mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id}',\mathfrak{o}} + (\mathsf{id}-\mathsf{id}')\mathbf{\Phi}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{\Psi}_{\mathsf{id}',\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}} \approx \mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id}',\mathfrak{o}} + \$ \equiv \$; \\ \mathsf{id} = \mathsf{id}' : & \mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^{\mathsf{T}} \end{cases} \quad \begin{array}{l} \uparrow \\ \mathrm{LWE} \end{array} \quad \begin{array}{l} \mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}} \\ \downarrow \\ \approx \$. \end{array}$$

When the noise widths of $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$ and LWE do not match, we resolve the issue similarly to the proof of Corollary 7.3.

Proof (Theorem 7.7). Let $\mathcal{S} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}})$ be an efficient sampler (Definition 7.2) such that $\sigma_{\mathrm{pre}} \le \sigma_{\mathrm{pre},0}$ and $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$ holds against Construction 7.2. Consider the following efficient $\mathcal{S}' = (\mathcal{S}'_{\mathrm{V}}, \mathcal{S}'_{\mathrm{U}})$.

- $\mathcal{S}'_{\mathrm{V}}(r''_{\mathrm{pub}})$ parses $r''_{\mathrm{pub}} = (r_{\mathrm{pub}}, r'_{\mathrm{pub}})$ and runs

  $$(q, 1^K, 1^Z, \mathsf{ID}, \{1^{J_{\mathsf{id}}}, 1^{I_{\mathsf{id},\mathfrak{o}}}, 1^{I_{\mathsf{id},\mathfrak{g}}}\}_{\mathsf{id}\in\mathsf{ID}}, \sigma_{\mathrm{pre}}, \{\mathbf{V}_{\mathsf{id},\mathfrak{o}}, \mathbf{V}_{\mathsf{id},\mathfrak{g}}\}_{\mathsf{id}\in\mathsf{ID}}) \leftarrow \mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}}).$$

  The algorithm aborts if q is not a prime or $1 \notin \mathcal{I}'_{q,K,Z}$. Otherwise, it computes n used by Setup of Construction 7.2, and samples $\{\mathbf{\Psi}_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$, with $\mathbf{\Psi}_{\mathsf{id},\mathfrak{o}}$ (resp. $\mathbf{\Psi}_{\mathsf{id},\mathfrak{g}}$) uniformly random over $\mathbb{Z}_q^{n\times J_{\mathsf{id}}}$ (resp. $\mathbb{Z}_q^{n\times KJ_{\mathsf{id}}}$) for all $\mathsf{id}\in\mathsf{ID}$, using r''_{pub} in the straight-forward manner.[81] The algorithm sets

  $$Z' = Z + 2n, \qquad \mathsf{ID}' = \{1\}, \qquad J'_1 = \sum_{\mathsf{id}\in\mathsf{ID}} J_{\mathsf{id}}, \qquad I'_{1,\mathfrak{s}} = \sum_{\mathsf{id}\in\mathsf{ID}} I_{\mathsf{id},\mathfrak{s}},$$

  $$\sigma'_{\mathrm{pre}} = \begin{cases} \sigma_{\mathrm{pre}}, & \text{if } \sigma_{\mathrm{pre}} \ge \frac{\sigma_{\mathrm{pre},0}}{2^{\kappa+6}\sqrt{\kappa}}; \\ \sigma_{\mathrm{pre},0}, & \text{otherwise;} \end{cases} \qquad \mathbf{V}'_{1,\mathfrak{s}} = \begin{pmatrix} \cdots & \mathbf{V}_{\mathsf{id},\mathfrak{s}} & \cdots \\ \cdots & \mathbf{\Psi}_{\mathsf{id},\mathfrak{s}} & \cdots \\ \cdots & \mathsf{id}\cdot\mathbf{\Psi}_{\mathsf{id},\mathfrak{s}} & \cdots \end{pmatrix}_{\mathsf{id}\in\mathsf{ID}},$$

[80] Here, κ is unrelated to that in Construction 7.1, although they can be set to the same value.

[81] The distribution of r''_{pub} conditioned on $\{\mathbf{\Psi}_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$ must be efficiently sampleable from $\{\mathbf{\Psi}_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$. See Footnote 76.

and outputs $(q, 1^K, 1^Z, \mathsf{ID}', 1^{J'_1}, 1^{I'_{1,\mathfrak{o}}}, 1^{I'_{1,\mathfrak{g}}}, \sigma'_{\mathrm{pre}}, \mathbf{V}'_{1,\mathfrak{o}}, \mathbf{V}'_{1,\mathfrak{g}})$.

- $\mathcal{S}'_{\mathrm{U}}(r''_{\mathrm{pub}}; r''_{\mathrm{priv}})$ parses $r''_{\mathrm{pub}} = (r_{\mathrm{pub}}, r'_{\mathrm{pub}})$ and $r''_{\mathrm{priv}} = (r_{\mathrm{priv}}, r'_{\mathrm{priv}})$. It recomputes q, ID, $\{I_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$ and n as in $\mathcal{S}'_{\mathrm{V}}$. The algorithm runs

$$\{\mathbf{U}^{\mathsf{T}}_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}} \leftarrow \mathcal{S}_{\mathrm{U}}(r_{\mathrm{pub}}; r_{\mathrm{priv}}),$$

and samples $\mathbf{\Phi}_{\mathsf{id},\mathfrak{s}}$ uniformly random over $\mathbb{Z}_q^{n\times I_{\mathsf{id},\mathfrak{s}}}$ for all $\mathsf{id}\in\mathsf{ID}$ and $\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}$ using r'_{priv}. It sets

$$(\mathbf{U}'_{1,\mathfrak{s}})^{\mathsf{T}} = \begin{pmatrix} \vdots & \vdots & \vdots \\ \mathbf{U}^{\mathsf{T}}_{\mathsf{id},\mathfrak{s}} & \mathsf{id}\cdot\mathbf{\Phi}^{\mathsf{T}}_{\mathsf{id},\mathfrak{s}} & -\mathbf{\Phi}^{\mathsf{T}}_{\mathsf{id},\mathfrak{s}} \\ \vdots & \vdots & \vdots \end{pmatrix}_{\mathsf{id}\in\mathsf{ID}}$$

and outputs $\{(\mathbf{U}'_{1,\mathfrak{s}})^{\mathsf{T}}\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$.

Claim 7.8 (¶). $\mathsf{IPFEsec}^{\mathcal{S}'}_{\mathrm{pre}}$ *(Definition 7.2) holds.*

From the $\sigma_{\mathrm{pre},0}$-security of IPFE', it follows that

$$\begin{aligned} &(r''_{\mathrm{pub}}, \mathsf{impk}', \mathsf{isk}'_1, \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, (\mathbf{U}'_{1,\mathfrak{s}})^{\mathsf{T}})\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}) \\ \approx\ &(r''_{\mathrm{pub}}, \mathsf{impk}', \mathsf{isk}'_1, \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, \quad \mathbf{\Delta}^{\mathsf{T}}_{\mathfrak{s}} \quad)\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}), \end{aligned}$$

where $\mathbf{\Delta}_{\mathfrak{s}} \xleftarrow{\$} \mathbb{Z}_q^{Z'\times I'_{1,\mathfrak{s}}}$. (This is the "initial part" of this proof.)

Now consider the following efficient $\mathcal{S}_{\$} = (\mathcal{S}_{\$,\mathrm{V}}, \mathcal{S}_{\$,\mathrm{U}})$.

- $\mathcal{S}_{\$,\mathrm{V}}$ is just $\mathcal{S}_{\mathrm{V}}$.
- $\mathcal{S}_{\$,\mathrm{U}}(r_{\mathrm{pub}}; r_{\$,\mathrm{priv}})$ recomputes $q, Z, \mathsf{ID}, \{I_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$ as in $\mathcal{S}_{\$,\mathrm{V}}$, then uses $r_{\$,\mathrm{priv}}$ to sample $\mathbf{U}_{\$,\mathsf{id},\mathfrak{s}} \xleftarrow{\$} \mathbb{Z}_q^{Z\times I_{\mathsf{id},\mathfrak{s}}}$ for all $\mathsf{id}\in\mathsf{ID}$ and $\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}$, and lastly outputs $\{\mathbf{U}^{\mathsf{T}}_{\$,\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$.

It can be readily verified that $\mathsf{IPFEsec}^{\mathcal{S}_{\$}}_{\mathrm{pre}}$ follows from $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{pre}}$:

$$\begin{aligned} &\left(r_{\mathrm{pub}}, \left\{ \begin{matrix} \mathbf{U}^{\mathsf{T}}_{\$,\mathsf{id},\mathfrak{o}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} & + \mathbf{E}^{\mathsf{T}}_{\mathsf{id},\mathfrak{o}}, \\ \mathbf{U}^{\mathsf{T}}_{\$,\mathsf{id},\mathfrak{g}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}}\otimes\mathbf{E}^{\mathsf{T}}_{\mathsf{id},\mathfrak{g}\mathfrak{g}} & + \mathbf{E}^{\mathsf{T}}_{\mathsf{id},\mathfrak{g}\mathfrak{o}} \end{matrix} \right\}_{\mathsf{id}\in\mathsf{ID}}\right) \\ \equiv\ &\left(r_{\mathrm{pub}}, \left\{ \begin{matrix} (\mathbf{U}_{\$,\mathsf{id},\mathfrak{o}} + \mathbf{U}_{\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} & + \mathbf{E}^{\mathsf{T}}_{\mathsf{id},\mathfrak{o}}, \\ (\mathbf{U}_{\$,\mathsf{id},\mathfrak{g}} + \mathbf{U}_{\mathsf{id},\mathfrak{g}})^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}}\otimes\mathbf{E}^{\mathsf{T}}_{\mathsf{id},\mathfrak{g}\mathfrak{g}} & + \mathbf{E}^{\mathsf{T}}_{\mathsf{id},\mathfrak{g}\mathfrak{o}} \end{matrix} \right\}_{\mathsf{id}\in\mathsf{ID}}\right) \end{aligned}$$

$$\approx \left(r_{\mathrm{pub}}, \left\{\begin{matrix}\mathbf{U}_{\$,\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + \boldsymbol{\Delta}_{\mathsf{pre},\mathsf{id},\mathfrak{o}}^{\mathsf{T}},\\ \mathbf{U}_{\$,\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \boldsymbol{\Delta}_{\mathsf{pre},\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\end{matrix}\right\}_{\mathsf{id}\in\mathsf{ID}}\right) \equiv \left(r_{\mathrm{pub}}, \left\{\begin{matrix}\boldsymbol{\Delta}_{\mathsf{pre},\mathsf{id},\mathfrak{o}}^{\mathsf{T}},\\ \boldsymbol{\Delta}_{\mathsf{pre},\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\end{matrix}\right\}_{\mathsf{id}\in\mathsf{ID}}\right).$$

Applying the initial part of the proof to $\mathcal{S}_{\$}$, we obtain

$$\begin{aligned}&\left(r''_{\mathrm{pub}}, \mathsf{impk}', \mathsf{isk}'_1, \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, (\mathbf{U}'_{\$,1,\mathfrak{s}})^{\mathsf{T}})\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}\right)\\ \approx\ &\left(r''_{\mathrm{pub}}, \mathsf{impk}', \mathsf{isk}'_1, \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, \quad \boldsymbol{\Delta}_{\mathfrak{s}}^{\mathsf{T}} \quad)\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}\right),\end{aligned}$$

where $\mathbf{U}'_{\$,1,\mathfrak{s}}$ is to $\mathbf{U}_{\$,\mathsf{id},\mathfrak{s}}$'s as $\mathbf{U}'_{1,\mathfrak{s}}$ is to $\mathbf{U}_{\mathsf{id},\mathfrak{s}}$'s for each $\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}$. Therefore,

$$\begin{aligned}&\left(r''_{\mathrm{pub}}, \mathsf{impk}', \mathsf{isk}'_1, \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, (\mathbf{U}'_{\$,1,\mathfrak{s}})^{\mathsf{T}})\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}\right)\\ \approx\ &\left(r''_{\mathrm{pub}}, \mathsf{impk}', \mathsf{isk}'_1, \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, \ (\mathbf{U}'_{1,\mathfrak{s}})^{\mathsf{T}} \ )\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}\right).\end{aligned}$$

Opening up $\mathsf{IPFEsec}^{\mathcal{S}}_{\mathrm{post}}$, we see that

$$\begin{aligned}\overbrace{r_{\mathrm{pub}}}^{\text{part of } r''_{\mathrm{pub}}}, \qquad \mathsf{impk} &= (\overbrace{q, 1^K, 1^Z, 1^n}^{\text{from } r_{\mathrm{pub}}}, \mathsf{impk}'),\\ \{\mathsf{ict}^{*}_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}} &= \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, \ (\mathbf{U}'_{1,\mathfrak{s}})^{\mathsf{T}} \ )\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}},\\ \{\mathsf{ict}^{\$}_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}} &= \{\mathsf{Enc}'(\mathsf{impk}', 1, \mathfrak{s}, (\mathbf{U}'_{\$,1,\mathfrak{s}})^{\mathsf{T}})\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}},\\ \{\mathsf{isk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathsf{ID}} &= (\{\underbrace{\boldsymbol{\Psi}_{\mathsf{id},\mathfrak{o}}, \boldsymbol{\Psi}_{\mathsf{id},\mathfrak{g}}}_{\text{from } r'_{\mathrm{pub}} \text{ in } r''_{\mathrm{pub}}}\}_{\mathsf{id}\in\mathsf{ID}}, \mathsf{isk}'_1),\end{aligned}$$

so it follows from the previous indistinguishability.

For the part regarding restricted security, it remains to verify that $\mathcal{S}'$ is restricted whenever $\mathcal{S}$ is restricted (below), and that $\mathcal{S}_{\$}$ is always restricted (clear by inspection). Let $\mathcal{S}_{\mathrm{A0}}$ be the algorithm in Definition 7.3 for $\mathcal{S}$. The $\mathcal{S}'_{\mathrm{A0}}(r''_{\mathrm{pub}})$ for $\mathcal{S}'$ parses r''_{pub} as $(r_{\mathrm{pub}}, r'_{\mathrm{pub}})$, recomputes ID, $\{I_{\mathsf{id},\mathfrak{s}}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\}}$, $I'_{1,\mathfrak{o}}, I'_{1,\mathfrak{g}}, n$ as in $\mathcal{S}_{\mathrm{V}}$, runs and sets

$$(1^{n'}, \{\mathbf{A}_{\mathsf{id},\mathfrak{s},0,i}\}_{\mathsf{id}\in\mathsf{ID},\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\},i\in[I_{\mathsf{id},\mathfrak{s}}]}) \leftarrow \mathcal{S}_{\mathrm{A0}}(r_{\mathrm{pub}}), \qquad n'' = n' + n(I'_{1,\mathfrak{o}} + I'_{1,\mathfrak{g}}),$$

$$\mathbf{A}'_{1,\mathfrak{o},0,i'_{\mathfrak{o}}} = \begin{pmatrix}\mathbf{A}_{\mathsf{id},\mathfrak{o},0,i} & & \\ & \mathbf{0}_{n(i'_{\mathfrak{o}}-1)\times n} & \\ & \mathsf{id}\cdot\mathbf{I}_n \quad -\mathbf{I}_n & \\ & & \mathbf{0}_{n(I'_{1,\mathfrak{o}}-i'_{\mathfrak{o}}+I'_{1,\mathfrak{g}})\times n}\end{pmatrix}, \quad \mathbf{A}'_{1,\mathfrak{g},0,i'_{\mathfrak{g}}} = \begin{pmatrix}\mathbf{A}_{\mathsf{id},\mathfrak{g},0,i} & & \\ & \mathbf{0}_{n(I'_{1,\mathfrak{o}}+i'_{\mathfrak{g}}-1)\times n} & \\ & \mathsf{id}\cdot\mathbf{I}_n \quad -\mathbf{I}_n & \\ & & \mathbf{0}_{n(I'_{1,\mathfrak{g}}-i'_{\mathfrak{g}})\times n}\end{pmatrix},$$

with (id, i) being the $(i'_{\mathfrak{s}})^{\text{th}}$ item among all such that $\mathsf{id} \in \mathsf{ID}$ and $i \in [I_{\mathsf{id},\mathfrak{s}}]$. It outputs $(1^{n''}, \{\mathbf{A}'_{1,\mathfrak{s},0,i'_{\mathfrak{s}}}\}_{\mathfrak{s}\in\{\mathfrak{o},\mathfrak{g}\},i'_{\mathfrak{s}}\in[I'_{1,\mathfrak{s}}]})$. Conceptually, let

$$(\mathbf{d}'_0)^\top = (\mathbf{d}_0^\top, \underbrace{\ldots, \boldsymbol{\varphi}_{\mathsf{id},\mathfrak{o},i}^\top, \ldots}_{\mathsf{id}\in\mathsf{ID},\, i\in[I_{\mathsf{id},\mathfrak{o}}]}, \underbrace{\ldots, \boldsymbol{\varphi}_{\mathsf{id},\mathfrak{g},i}^\top, \ldots}_{\mathsf{id}\in\mathsf{ID},\, i\in[I_{\mathsf{id},\mathfrak{g}}]}),$$

then

$$\begin{aligned}(\mathbf{u}'_{1,\mathfrak{s},i'_{\mathfrak{s}}})^\top &= (\mathbf{u}_{\mathsf{id},\mathfrak{s},i}^\top, \mathsf{id}\cdot\boldsymbol{\varphi}_{\mathsf{id},\mathfrak{s},i}^\top, -\boldsymbol{\varphi}_{\mathsf{id},\mathfrak{s},i}^\top)\\ &= (\mathbf{d}_0^\top\mathbf{A}_{\mathsf{id},\mathfrak{s},0,i}, \mathsf{id}\cdot\boldsymbol{\varphi}_{\mathsf{id},\mathfrak{s},i}^\top, -\boldsymbol{\varphi}_{\mathsf{id},\mathfrak{s},i}^\top) = (\mathbf{d}'_0)^\top\mathbf{A}'_{1,\mathfrak{s},0,i'_{\mathfrak{s}}}.\end{aligned}$$

□

Proof (Claim 7.8). Consider the following hybrids.

- H_0 is the first distribution in $\mathsf{IPFEsec}^{\mathcal{S}'}_{\mathrm{pre}}$, i.e.,

  $$\boldsymbol{r}''_{\mathrm{pub}}, \qquad (\mathbf{U}'_{1,\mathfrak{o}})^\top\mathbf{V}'_{1,\mathfrak{o}} + (\mathbf{E}'_{1,\mathfrak{o}})^\top, \qquad (\mathbf{U}'_{1,\mathfrak{g}})^\top\mathbf{V}'_{1,\mathfrak{g}} + \widetilde{\mathbf{g}}^\top \otimes (\mathbf{E}'_{1,\mathfrak{g}\mathfrak{g}})^\top + (\mathbf{E}'_{1,\mathfrak{g}\mathfrak{o}})^\top.$$

  The components can be rearranged into

  $$\begin{aligned}&r_{\mathrm{pub}}, \qquad \boldsymbol{r}'_{\mathrm{pub}},\\ &\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^\top\mathbf{V}_{\mathsf{id}',\mathfrak{o}} + (\mathsf{id}-\mathsf{id}')\boldsymbol{\Phi}_{\mathsf{id},\mathfrak{o}}^\top\boldsymbol{\Psi}_{\mathsf{id}',\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^\top\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\\ &\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^\top\mathbf{V}_{\mathsf{id}',\mathfrak{g}} + (\mathsf{id}-\mathsf{id}')\boldsymbol{\Phi}_{\mathsf{id},\mathfrak{g}}^\top\boldsymbol{\Psi}_{\mathsf{id}',\mathfrak{g}} + \widetilde{\mathbf{g}}^\top \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{g}})^\top + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{o}})^\top\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\\ &\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^\top\mathbf{V}_{\mathsf{id},\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^\top\right\}_{\mathsf{id}\in\mathsf{ID}},\\ &\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^\top\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^\top \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{g}})^\top + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{o}})^\top\right\}_{\mathsf{id}\in\mathsf{ID}},\end{aligned}$$

  where

  $$\mathbf{E}'_{1,\mathfrak{e}} = \begin{pmatrix}\mathbf{E}'_{*,*,\mathfrak{e}} & \mathbf{E}'_{\mathsf{id},*,\mathfrak{e}} & \cdots\\ \mathbf{E}'_{*,\mathsf{id}',\mathfrak{e}} & \mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{e}} & \cdots\\ \vdots & \vdots & \ddots\end{pmatrix}_{\mathsf{id}',\mathsf{id}\in\mathsf{ID}} \qquad \text{for all } \mathfrak{e}\in\{\mathfrak{o},\mathfrak{g}\mathfrak{g},\mathfrak{g}\mathfrak{o}\}$$

  with $\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}} \in \mathbb{Z}^{J_{\mathsf{id}'}\times I_{\mathsf{id},\mathfrak{o}}}$, $\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{g}} \in \mathbb{Z}^{J_{\mathsf{id}'}\times I_{\mathsf{id},\mathfrak{g}}}$, $\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{o}} \in \mathbb{Z}^{KJ_{\mathsf{id}'}\times I_{\mathsf{id},\mathfrak{g}}}$ for all $\mathsf{id}, \mathsf{id}' \in \mathsf{ID}$.

- In H_1, we insert noises $\mathbf{E}^{\mathrm{help}}$ into the inner products for $\mathfrak{o}$ between non-matching identities, i.e.,

  $$\begin{aligned}&r_{\mathrm{pub}}, \qquad \boldsymbol{r}'_{\mathrm{pub}},\\ &\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^\top\mathbf{V}_{\mathsf{id}',\mathfrak{o}} + (\mathsf{id}-\mathsf{id}')\boldsymbol{\Phi}_{\mathsf{id},\mathfrak{o}}^\top\boldsymbol{\Psi}_{\mathsf{id}',\mathfrak{o}} + (\mathbf{E}^{\mathrm{help}}_{\mathsf{id},\mathsf{id}'})^\top + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^\top\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\\ &\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^\top\mathbf{V}_{\mathsf{id}',\mathfrak{g}} + (\mathsf{id}-\mathsf{id}')\boldsymbol{\Phi}_{\mathsf{id},\mathfrak{g}}^\top\boldsymbol{\Psi}_{\mathsf{id}',\mathfrak{g}} + \widetilde{\mathbf{g}}^\top \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{g}})^\top + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{o}})^\top\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\end{aligned}$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{g}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$

where $\mathbf{E}^{\mathrm{help}}_{\mathsf{id},\mathsf{id}'} \xleftarrow{\$} \mathcal{D}^{J_{\mathsf{id}'}\times I_{\mathsf{id},\mathfrak{o}}}_{\mathbb{Z},\sigma_{\mathrm{help}},\le\sigma_{\mathrm{help}}\sqrt{\kappa}}$, each of which is flooded by $\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}}$.

By Lemma 2.2, we have $\mathsf{H}_0 \approx_{\mathrm{s}} \mathsf{H}_1$.

- In $\mathsf{H}_{2,\alpha}$ (with $0 \le \alpha \le |\mathsf{ID}|$), we no longer truncate $\mathbf{E}^{\mathrm{help}}$, and change the non-matching inner products with the ciphertexts under the first α identities for $\mathfrak{o}$ to random, i.e.,

$$r_{\mathrm{pub}}, \qquad r'_{\mathrm{pub}},$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id}',\mathfrak{o}} + (\mathbf{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}^{\mathsf{id}\neq\mathsf{id}',\mathsf{id}\le\mathsf{id}_\alpha}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}},$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id}',\mathfrak{o}} + (\mathsf{id}-\mathsf{id}')\mathbf{\Phi}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{\Psi}_{\mathsf{id}',\mathfrak{o}} + (\mathbf{E}^{\mathrm{help}}_{\mathsf{id},\mathsf{id}'})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}^{\mathsf{id}\neq\mathsf{id}',\mathsf{id}>\mathsf{id}_\alpha}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}},$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id}',\mathfrak{g}} + (\mathsf{id}-\mathsf{id}')\mathbf{\Phi}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{\Psi}_{\mathsf{id}',\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{g}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{o}})^{\mathsf{T}}\right\}^{\mathsf{id}\neq\mathsf{id}'}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}},$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{g}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$

where $\mathbf{E}^{\mathrm{help}}_{\mathsf{id},\mathsf{id}'} \xleftarrow{\$} \mathcal{D}^{J_{\mathsf{id}'}\times I_{\mathsf{id},\mathfrak{o}}}_{\mathbb{Z},\sigma_{\mathrm{help}}}$ and $\mathbf{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}} \xleftarrow{\$} \mathbb{Z}_q^{J_{\mathsf{id}'}\times I_{\mathsf{id},\mathfrak{o}}}$ for all $\mathsf{id},\mathsf{id}' \in \mathsf{ID}$. Here, $\mathsf{id}_\alpha \in \mathsf{ID}$ is the α^{th}-smallest among all $\mathsf{id} \in \mathsf{ID}$ in some fixed efficiently comparable total order for all $\alpha \in [|\mathsf{ID}|]$, and id_0 is a sentinel value that compares less than all $\mathsf{id} \in \mathsf{ID}$.

By Lemma 2.1, we have $\mathsf{H}_1 \approx_{\mathrm{s}} \mathsf{H}_{2,0}$. By a hybrid argument, from $\mathsf{LWE}_{n,J'_1,q,\sigma_{\mathrm{help}}}$ it follows that $\mathsf{H}_{2,\alpha} \approx \mathsf{H}_{2,\alpha+1}$ for all $0 \le \alpha < |\mathsf{ID}|$, with the LWE public matrix being

$$\left(\ldots,\ (\mathsf{id}_{\alpha+1}-\mathsf{id}')\mathbf{\Psi}_{\mathsf{id}',\mathfrak{o}},\ \ldots\right)^{\mathsf{id}'\neq\mathsf{id}_{\alpha+1}}_{\mathsf{id}'\in\mathsf{ID}}$$

and the LWE secret being each column of $\mathbf{\Phi}_{\mathsf{id},\mathfrak{o}}$. Roughly speaking, the reduction algorithm computes[82] $\mathbf{\Psi}_{\mathsf{id}',\mathfrak{o}}$ for all $\mathsf{id}' \in \mathsf{ID}$ from the public matrices, and samples the other components by itself.

- In H_3, we change all non-matching inner products for $\mathfrak{o}$ to random, i.e.,

$$r_{\mathrm{pub}}, \qquad r'_{\mathrm{pub}}, \qquad \left\{(\mathbf{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}^{\mathsf{id}\neq\mathsf{id}'}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}},$$

[82] This step relies on q being a prime.

$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id}',\mathfrak{g}} + (\mathsf{id}-\mathsf{id}')\boldsymbol{\Phi}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\boldsymbol{\Psi}_{\mathsf{id}',\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{g}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{g}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}}.$$

Comparing $\mathsf{H}_{2,|\mathsf{ID}|}$ and H_3, we see that $(\boldsymbol{\Delta}')$'s absorb the terms added to them without changing the distribution, so $\mathsf{H}_{2,|\mathsf{ID}|} \equiv \mathsf{H}_3$.

- In H_4, we change all non-matching inner products for $\mathfrak{g}$ to random, i.e.,

$$r_{\mathrm{pub}},\quad \boldsymbol{r}'_{\mathrm{pub}},\quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{g}})^{\mathsf{T}} + (\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}}.$$

The transition from H_3 to H_4 is analogous to that from H_0 to H_3, and $\mathsf{H}_3 \approx \mathsf{H}_4$.

- In H_5, we alter the noises attached to the matching inner products into

$$r_{\mathrm{pub}},\quad \boldsymbol{r}'_{\mathrm{pub}},\quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + \widetilde{\mathbf{E}}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes \widetilde{\mathbf{E}}_{\mathsf{id},\mathfrak{g}\mathfrak{g}}^{\mathsf{T}} + \widetilde{\mathbf{E}}_{\mathsf{id},\mathfrak{g}\mathfrak{o}}^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$

where for all $\mathsf{id}\in\mathsf{ID}$ and $\mathfrak{e}\in\{\mathfrak{o},\mathfrak{g}\mathfrak{g},\mathfrak{g}\mathfrak{o}\}$,

$$\widetilde{\mathbf{E}}_{\mathsf{id},\mathfrak{e}} = \begin{cases} \mathbf{E}_{\mathsf{id},\mathfrak{e}}, & \text{if } \sigma_{\mathrm{pre}} \geq \frac{\sigma_{\mathrm{pre},0}}{2^{\kappa+6}\sqrt{\kappa}} \text{ so } \sigma'_{\mathrm{pre}} = \sigma_{\mathrm{pre}}; \\ \mathbf{E}_{\mathsf{id},\mathfrak{e}} + \mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{e}}, & \text{otherwise;} \end{cases}$$

with $\mathbf{E}_{\mathsf{id},\mathfrak{o}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}},\leq\sigma_{\mathrm{pre}}\sqrt{\kappa}}^{J_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{o}}}$, $\mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{g}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}},\leq\sigma_{\mathrm{pre}}\sqrt{\kappa}}^{J_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{g}}}$, $\mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{o}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}},\leq\sigma_{\mathrm{pre}}\sqrt{\kappa}}^{KJ_{\mathsf{id}}\times I_{\mathsf{id},\mathfrak{g}}}$.

In the first case, the two hybrids are statistically close by Lemma 2.1, and in the second case, by Lemma 2.2, as $\sigma'_{\mathrm{pre}} = \sigma_{\mathrm{pre},0} \geq 2^{\kappa+6}\sigma_{\mathrm{pre}}\sqrt{\kappa}$. Therefore, $\mathsf{H}_4 \approx_{\mathrm{s}} \mathsf{H}_5$.

- In H_6, we no longer truncate $\mathbf{E}_{\mathsf{id},\mathfrak{e}}$'s, i.e.,

$$r_{\mathrm{pub}},\quad \boldsymbol{r}'_{\mathrm{pub}},\quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},\quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}'\in\mathsf{ID}}^{\mathsf{id}\neq\mathsf{id}'},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{o}} + \mathbf{E}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}} + (\mathbf{E}''_{\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$
$$\left\{\mathbf{U}_{\mathsf{id},\mathfrak{g}}^{\mathsf{T}}\mathbf{V}_{\mathsf{id},\mathfrak{g}} + \widetilde{\mathbf{g}}^{\mathsf{T}} \otimes \mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{g}}^{\mathsf{T}} + \mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{o}}^{\mathsf{T}} + (\mathbf{E}''_{\mathsf{id},\mathfrak{g}})^{\mathsf{T}}\right\}_{\mathsf{id}\in\mathsf{ID}},$$

where $\mathbf{E}_{\mathsf{id},\mathfrak{o}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{J_{\mathsf{id}} \times I_{\mathsf{id},\mathfrak{o}}}$, $\mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{g}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{J_{\mathsf{id}} \times I_{\mathsf{id},\mathfrak{g}}}$, $\mathbf{E}_{\mathsf{id},\mathfrak{g}\mathfrak{o}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}}}^{KJ_{\mathsf{id}} \times I_{\mathsf{id},\mathfrak{g}}}$, and $\mathbf{E}''_{\mathsf{id},\mathfrak{o}}$ (resp. $\mathbf{E}''_{\mathsf{id},\mathfrak{g}}$) is either $\mathbf{0}$ or $\mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{o}}$ (resp. $\widetilde{\mathbf{g}} \otimes \mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{g}} + \mathbf{E}'_{\mathsf{id},\mathsf{id},\mathfrak{g}\mathfrak{o}}$) for all $\mathsf{id} \in \mathsf{ID}$.

By Lemma 2.1, we have $\mathsf{H}_5 \approx_s \mathsf{H}_6$.

- H_7 is the second distribution in $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}'}$, i.e.,

$$r_{\mathrm{pub}}, \quad \boldsymbol{r}'_{\mathrm{pub}}, \quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}' \in \mathsf{ID}}^{\mathsf{id} \neq \mathsf{id}'}, \quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}})^{\mathsf{T}}\right\}_{\mathsf{id},\mathsf{id}' \in \mathsf{ID}}^{\mathsf{id} \neq \mathsf{id}'},$$
$$\left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id},\mathfrak{o}})^{\mathsf{T}}\right\}_{\mathsf{id} \in \mathsf{ID}}, \quad \left\{(\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id},\mathfrak{g}})^{\mathsf{T}}\right\}_{\mathsf{id} \in \mathsf{ID}},$$

where

$$\boldsymbol{\Delta}'_{1,\mathfrak{s}} = \begin{pmatrix} \boldsymbol{\Delta}'_{*,*,\mathfrak{s}} & \boldsymbol{\Delta}'_{\mathsf{id},*,\mathfrak{s}} & \cdots \\ \boldsymbol{\Delta}'_{*,\mathsf{id}',\mathfrak{s}} & \boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{s}} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}_{\mathsf{id}',\mathsf{id} \in \mathsf{ID}} \qquad \text{for all } \mathfrak{s} \in \{\mathfrak{o},\mathfrak{g}\}$$

with $\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{o}} \xleftarrow{\$} \mathbb{Z}_q^{J_{\mathsf{id}'} \times I_{\mathsf{id},\mathfrak{o}}}$, $\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id}',\mathfrak{g}} \xleftarrow{\$} \mathbb{Z}_q^{KJ_{\mathsf{id}'} \times I_{\mathsf{id},\mathfrak{g}}}$ for all $\mathsf{id}, \mathsf{id}' \in \mathsf{ID}$.

$\mathsf{H}_6 \approx \mathsf{H}_7$ follows from $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}}$, with $\boldsymbol{\Delta}'_{\mathsf{id},\mathsf{id},\mathfrak{s}}$ being either $\boldsymbol{\Delta}_{\mathsf{id},\mathfrak{s}}$ or $(\boldsymbol{\Delta}_{\mathsf{id},\mathfrak{s}} - \mathbf{E}''_{\mathsf{id},\mathsf{id},\mathfrak{s}})$.

By hybrid argument, $\mathsf{H}_0 \approx \mathsf{H}_7$, which is exactly $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}'}$. □

7.4.5 Scheme with Structured Noises

To support structured noises, we modify Construction 7.1, fitting it into the shape of evasive learning with structured errors. We explain a subtlety in the adaptation. In the basic scheme, a key $\mathbf{k}$ for $\mathbf{0}$ makes $\mathbf{d}^{\mathsf{T}}\mathbf{B}\mathbf{k}$ small (not pseudorandom). Therefore, in the presence of such a key, no ciphertext can be provably secure from evasive LWE. This is not a problem for Construction 7.1, because every (plaintext) vector has non-pseudorandom noisy inner product with $\mathbf{0}$. Similar properties of keys are not allowed in a scheme with structured noises, as security should hold for a ciphertext of random vector for $\mathfrak{g}$, given a key for $\mathbf{v}_{\mathfrak{o}} = \mathbf{0}$ and random $\mathbf{V}_{\mathfrak{g}}$. Preventing noise structure mix-and-match is akin to having two identities, for which we employ a mechanism similar to yet simpler than that of Construction 7.2.

Ingredients of Construction 7.3. We rely on Lemma 2.3 for correctness. For security, we need LWE (Assumption 2.2) and evasive learning with structured errors (Assumption 7.1).

Construction 7.3 (evIPFE-w/sn). Our scheme works as follows.

- $\mathsf{Setup}(q, 1^K, 1^Z)$ takes as input q, K, Z. It sets $\mathcal{I}_{q,K,Z} = \{1\}$, and *deterministically* picks n and $m \geq m_0(n(K+1), q)$ and $\sigma_{-1} \geq \sigma_0(n, m)$ appropriate for Lemma 2.3, as well as $\sigma_{\mathrm{post}}, \kappa$. The algorithm samples and sets

$$(\widetilde{\mathbf{B}}, \tau) \xleftarrow{\$} \mathsf{TrapGen}(1^{n(K+1)}, 1^m, q), \qquad \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times 2(Z+n)\lceil \log_2 q \rceil},$$
$$\mathsf{impk} = (1^n, 1^m, q, 1^K, 1^Z, \sigma_{-1}, \sigma_{\mathrm{post}}, 1^\kappa, \mathbf{A}, \mathbf{B}), \qquad \mathsf{imsk} = (\mathsf{impk}, \tau).$$

  where $\mathbf{B} = (\mathbf{B}_{-1}, \ldots, \mathbf{B}_{K-1})$ for $\widetilde{\mathbf{B}} = (\mathbf{B}_{-1}^\intercal, \ldots, \mathbf{B}_{K-1}^\intercal)^\intercal$. It outputs $(\mathsf{impk}, \mathsf{imsk})$.

- $\mathsf{KeyGen}(\mathsf{imsk}, \mathsf{id}, \mathbf{v}_\mathfrak{o}, \mathbf{V}_\mathfrak{g})$ imsk, $\mathsf{id} = 1$, $\mathbf{v}_\mathfrak{o} \in \mathbb{Z}_q^Z$, and $\mathbf{V}_\mathfrak{g} \in \mathbb{Z}_q^{Z \times K}$. It samples and runs

$$\boldsymbol{\psi}_\mathfrak{o} \xleftarrow{\$} \mathbb{Z}_q^n, \qquad \boldsymbol{\Psi}_\mathfrak{g} \xleftarrow{\$} \mathbb{Z}_q^{n \times K}, \qquad \mathbf{K} \xleftarrow{\$} \mathsf{SampD}\left(\widetilde{\mathbf{B}}, \tau, \widetilde{\mathbf{P}}, \sigma_{-1}\right),$$

  where $\widetilde{\mathbf{P}} = (\mathbf{P}_{-1}^\intercal, \ldots, \mathbf{P}_{K-1}^\intercal)^\intercal$ for

$$\mathbf{P} = (\mathbf{P}_{-1}, \ldots, \mathbf{P}_{K-1}) = \mathbf{A}\mathbf{G}^{-1}\begin{pmatrix} \mathbf{v}_\mathfrak{o} & \\ \boldsymbol{\psi}_\mathfrak{o} & \\ & \mathbf{V}_\mathfrak{g} \\ & \boldsymbol{\Psi}_\mathfrak{g} \end{pmatrix}.$$

  The algorithm outputs $\mathsf{isk} = (\boldsymbol{\psi}_\mathfrak{o}, \boldsymbol{\Psi}_\mathfrak{g}, \mathbf{K})$.

- $\mathsf{Enc}(\mathsf{impk}, \mathsf{id}, \mathfrak{s}, \mathbf{u}^\intercal)$ takes as input impk, $\mathsf{id} = 1$, $\mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}$, and $\mathbf{u} \in \mathbb{Z}_q^Z$. It samples and sets

$$\boldsymbol{\varphi} \xleftarrow{\$} \mathbb{Z}_q^n, \qquad \mathbf{d} \xleftarrow{\$} \mathbb{Z}_q^n, \qquad \mathbf{e}_0 \xleftarrow{\$} \mathcal{D}^{2(Z+n)\lceil \log_2 q \rceil}_{\mathbb{Z}, \sigma_{\mathrm{post}}, \leq \sigma_{\mathrm{post}}\sqrt{\kappa}},$$
$$\mathbf{e}_1 \xleftarrow{\$} \mathcal{D}^{m(K+1)}_{\mathbb{Z}, \sigma_{\mathrm{post}}, \leq \sigma_{\mathrm{post}}\sqrt{\kappa}}, \qquad \mathbf{e}_3 \xleftarrow{\$} \mathcal{D}^{m}_{\mathbb{Z}, \sigma_{\mathrm{post}}, \leq \sigma_{\mathrm{post}}\sqrt{\kappa}},$$
$$(\mathbf{u}')^\intercal \leftarrow \begin{cases} (\mathbf{u}^\intercal, \mathbf{0}_{1\times n}, \mathbf{0}_{1\times Z}, \boldsymbol{\varphi}^\intercal), & \text{if } \mathfrak{s} = \mathfrak{o}; \\ (\mathbf{0}_{1\times Z}, \boldsymbol{\varphi}^\intercal\mathbf{G}, \mathbf{u}^\intercal\mathbf{G}, \mathbf{0}_{1\times n}), & \text{if } \mathfrak{s} = \mathfrak{g}. \end{cases}$$

  The algorithm outputs $\mathsf{ict} = (\mathbf{d}^\intercal\mathbf{B} + (\mathbf{0}_{1\times m}, \widetilde{\mathbf{g}}^\intercal \otimes \mathbf{e}_3^\intercal) + \mathbf{e}_1^\intercal,\ \mathbf{d}^\intercal\mathbf{A} + (\mathbf{u}')^\intercal\mathbf{G} + \mathbf{e}_0^\intercal)$.

- $\mathsf{Dec}(\mathsf{impk}, \mathsf{id}, \mathbf{v}_{\mathfrak{o}}, \mathbf{V}_{\mathfrak{g}}, \mathsf{isk}, \mathfrak{s}, \mathsf{ict})$ computes and outputs

$$\mathsf{ict} \cdot \begin{pmatrix} -\mathbf{I}_{K+1} \otimes \mathbf{K} \\ \mathbf{G}^{-1}\begin{pmatrix} \mathbf{v}_{\mathfrak{o}} & \\ \boldsymbol{\psi}_{\mathfrak{o}} & \\ & \mathbf{V}_{\mathfrak{g}} \\ & \boldsymbol{\Psi}_{\mathfrak{g}} \end{pmatrix} \end{pmatrix} \cdot \begin{cases} (1, \mathbf{0}_{1\times K})^{\intercal}, & \text{if } \mathfrak{s} = \mathfrak{o}; \\ (\mathbf{0}_{K\times 1}, \mathbf{I}_K)^{\intercal}, & \text{if } \mathfrak{s} = \mathfrak{g}. \end{cases}$$

Correctness. It is readily verified, similarly to Construction 7.1, that the scheme is B-correct for

$$B = \kappa^{1/2}\sigma_{\mathsf{post}}(m^{3/2}\sigma_{-1} + 2(Z+n)\lceil \log_2 q \rceil).$$

Theorem 7.9 (¶). *Construction 7.3 is* restricted-σ_{post}-secure *(Definition 7.3) under evasive learning with structured errors (Assumption 7.1) and* $\mathsf{LWE}_{n,m(K+1)+2(Z+n)\lceil \log_2 q\rceil,q,\sigma_{\mathsf{help}}}$ *(Assumption 2.2) for some* $\sigma_{\mathsf{help}} \leq \frac{\sigma_{\mathsf{post}}}{(2^{\kappa+6}\sqrt{\kappa})^2 \cdot 2(Z+n)\lceil \log_2 q\rceil}$*, where* q, K, Z *are those picked by* $\mathcal{S}$ *and* $n, m, \sigma_{\mathsf{post}}, \kappa$ *are those picked by* Setup.

Proof Sketch (Theorem 7.9). The proof done by combining the techniques demonstrated earlier in this section. Roughly speaking, the desired can be cast as a postcondition of evLWsE. In the precondition of evLWsE, first use IPFE simulation as in the proof of Theorem 7.2 so that we only consider the inner products, both for matching and non-matching noise structures. Then, use the proof of Claim 7.8 to argue that non-matching inner products, due to the presence of $\boldsymbol{\psi}^{\intercal}\boldsymbol{\varphi}_{\mathfrak{o}}$ or $\boldsymbol{\psi}^{\intercal}\boldsymbol{\Phi}_{\mathfrak{g}}$, are pseudorandom. Lastly, use the premise that the matching inner products are pseudorandom to conclude the proof. □

Applying Construction 7.2 to Construction 7.3, we obtain the following:

Corollary 7.10. *Under* $\mathsf{LWE}_{n,\mathrm{poly}(\lambda),q,\sigma_{\mathsf{help}}}$ *(Assumption 2.2) and evasive learning with structured errors (Assumption 7.1), there exists an IB-evIPFE-w/sn (Definition 7.1) with identity space* $[q]$ *and correctness bound*

$$B = \kappa^{1/2}\sigma_{\mathsf{post}}(m^{3/2}\sigma_{-1} + 2(Z+3n)\lceil \log_2 q \rceil)$$

that is restricted-σ_{post}-secure *(Definition 7.3), where* $\sigma_{\mathsf{help}} \leq \frac{\sigma_{\mathsf{post}}}{(2^{\kappa+6}\sqrt{\kappa})^2 \cdot 2(Z+3n)\lceil \log_2 q\rceil}$.

7.5 Noisy Linear Garbling

We consider a notion of noisy linear garbling with authenticity (i.e., conditional disclosure of secrets)[83] reusable for multiple inputs. Our formulation characterizes a *promise* primitive. Given a function $f : \mathbb{Z}^L \to \{0, 1, \perp\}$, the garbling consists of labels decomposable and affine in the input $\mathbf{x}$, a garbled table, a secret, and some auxiliary information. For correctness, if $f(\mathbf{x}) = 1$, the secret can be approximately recovered from the other information; for security, if $f(\mathbf{x}) = 0$, the secret remains hidden; when $f(\mathbf{x}) = \perp$, we require neither correctness nor security. The *promise* feature of the definition is used to exclude $\mathbf{x}$ leading to out-of-bound wire values in arithmetic computations, and $\mathbf{x} \notin \{0, 1\}^\star$ in Boolean computations.

Definition 7.4 (noisy linear garbling). Let $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{param}}\}_{\lambda\in\mathbb{N},\mathsf{param}\in\mathsf{Params}_\lambda}$ be a sequence of function families, where $\mathsf{Params} = \{\mathsf{Params}_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of function family description sets and each $f \in \mathcal{F}_{\lambda,\mathsf{param}}$ is a function $\mathbb{Z}^L \to \{0, 1, \perp\}$ (with potentially different L for each f). A *noisy linear garbling scheme* for $\mathcal{F}$ consists of three efficient algorithms.

- $\mathsf{Setup}(1^\lambda, \mathsf{param})$ takes the function family description $\mathsf{param} \in \mathsf{Params}_\lambda$ as input. It outputs some public parameter $\mathsf{pp} = (q, \dots)$, where $q \in \mathbb{N}_{\geq 2}$ is the modulus.[84]

- $\mathsf{GenF}(1^\lambda, \mathsf{pp}, f)$ takes as input pp and $f \in \mathcal{F}_{\lambda,\mathsf{param}} : \mathbb{Z}^L \to \{0, 1, \perp\}$. It outputs a deterministic dimension 1^{n_f} of garbling randomness, a secret vector $\mathbf{w}_{\mathrm{out}}$, label functions $\{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}$, a garbled table $\mathbf{T}$, and some reusable information $\mathbf{R}$. Here, $\mathbf{w}_{\mathrm{out}}$, $\mathbf{W}$'s, and $\mathbf{T}$ are n_f-row matrices over $\mathbb{Z}$, and their shapes, including n_f, are fully determined by pp, f.

- $\mathsf{Eval}(1^\lambda, \mathsf{pp}, f, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\intercal\}_{\ell\in[L]}, \mathbf{t}^\intercal)$ takes as input pp, f, $\mathbf{R}$, input $\mathbf{x}$ (to f), one set of

[83] The terminology is changed from "secret sharing" in introduction and technical overview, because the policy function might not be monotone.

[84] In the conference version [HLL24a], the definition requires q be deterministic. While that version still works (by smuggling q into param and checking its suitability in Setup), it was an unintended error — it is easier and more consistent to let Setup sample q instead. This is currently necessary since we need prime q and it is unknown [TCH12] how to generate large primes in deterministic polynomial time.

randomized labels, and the randomized garbled table $\mathbf{t}$. If $f(\mathbf{x}) = 1$, it is supposed to output an approximation of the randomized secret w_{out}. Here, $w_{\text{out}}, \{\mathbf{w}_\ell\}_{\ell\in[L]}$, $\mathbf{t}$ are over $\mathbb{Z}_q$.

Let B_{in} and B_{out} be functions mapping (λ, param) to natural numbers. The scheme is $(B_{\text{in}}, B_{\text{out}})$-*correct* if for all $\lambda \in \mathbb{N}$, $\text{param} \in \mathsf{Params}_\lambda$, $f \in \mathcal{F}_{\lambda,\text{param}} : \mathbb{Z}^L \to \{0, 1, \bot\}$, $\mathbf{x} \in \mathbb{Z}^L$, $e_{\text{out}}, \mathbf{e}_1, \ldots, \mathbf{e}_L, \mathbf{e}_t \in [-B_{\text{in}}(\lambda, \text{param}), B_{\text{in}}(\lambda, \text{param})]^\star$ (of suitable dimensions)[85] such that $f(\mathbf{x}) = 1$, it holds that

$$\Pr\left[\begin{array}{rl} & \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \text{param}) \\ \left(\begin{smallmatrix} 1^{n_f}, \mathbf{w}_{\text{out}}, \\ \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \\ \mathbf{T}, \mathbf{R} \end{smallmatrix}\right) & \xleftarrow{\$} \mathsf{GenF}(1^\lambda, \mathsf{pp}, f) \\ & \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{n_f} \end{array} : \begin{array}{r} \mathsf{Eval}\left(\begin{smallmatrix} 1^\lambda, \mathsf{pp}, f, \mathbf{R}, \mathbf{x}, \\ \{\mathbf{s}^\intercal(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_\ell^\intercal\}_{\ell\in[L]}, \\ \mathbf{s}^\intercal\mathbf{T} + \mathbf{e}_t^\intercal \end{smallmatrix}\right) \\ - (\mathbf{s}^\intercal\mathbf{w}_{\text{out}} + e_{\text{out}}) \\ \in [-B_{\text{out}}(\lambda, \text{param}), B_{\text{out}}(\lambda, \text{param})] \end{array}\right] = 1.$$

Definition 7.5 (short garbling). A noisy linear garbling scheme (Definition 7.4) is B_{short}-*short* (for some function B_{short} mapping (λ, param) to natural numbers) if for all $\lambda \in \mathbb{N}$, $\text{param} \in \mathsf{Params}_\lambda$, $f \in \mathcal{F}_{\lambda,\text{param}} : \mathbb{Z}^L \to \{0, 1, \bot\}$, $\mathbf{x} \in \mathbb{Z}^L$ such that $f(\mathbf{x}) = 0$,

$$\Pr\left[\begin{array}{rl} & \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \text{param}) \\ \left(\begin{smallmatrix} 1^{n_f}, \mathbf{w}_{\text{out}}, \\ \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \\ \mathbf{T}, \mathbf{R} \end{smallmatrix}\right) & \xleftarrow{\$} \mathsf{GenF}(1^\lambda, \mathsf{pp}, f) \end{array} : \begin{array}{r} \|\mathbf{w}_{\text{out}}^\intercal\|, \|\mathbf{T}^\intercal\| \le B_{\text{short}}(\lambda, \text{param}), \\ \|\mathbf{W}_{\ell,0}^\intercal + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}^\intercal\| \le B_{\text{short}}(\lambda, \text{param}) \\ \text{for all } \ell \in [L] \end{array}\right] = 1.$$

Definition 7.6 (fixed randomness dimension). A noisy linear garbling scheme (Definition 7.4) has *fixed randomness dimension* if param is of the form $(q, 1^n, \ldots)$ and $n_f = n$ always holds.

Security. We require that the randomized (by noisy linear combination) secret, labels, and garbled table be pseudorandom if $f(\mathbf{x}) = 0$.

Definition 7.7 (noisy linear garbling security). Let $(\mathsf{Setup}, \mathsf{GenF}, \mathsf{Eval})$ be a noisy linear garbling scheme for $\mathcal{F}$ (Definition 7.4) and $\mathsf{GenNoise}$ an efficient algorithm with suitable input/output formats. The scheme is $\mathsf{GenNoise}$-*secure* if $\mathsf{Exp}^{\text{real}}_{\text{garble}} \approx \mathsf{Exp}^{\text{random}}_{\text{garble}}$, where $\mathsf{Exp}^{\text{real}}_{\text{garble}}(1^\lambda, \mathcal{A})$ and $\mathsf{Exp}^{\text{random}}_{\text{garble}}(1^\lambda, \mathcal{A})$ proceed as follows.

[85] The dimensions could depend on the randomness of Setup. Formally, they can be quantified over sufficiently large dimensions then truncated appropriately.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $\mathsf{param} \in \mathsf{Params}_\lambda$ from it. Sample r_{Setup}, run

$$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{param}; r_{\mathsf{Setup}}),$$

  and send r_{Setup} to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ chooses $f \in \mathcal{F}_{\lambda,\mathsf{param}} : \mathbb{Z}^L \to \{0, 1, \perp\}$ and $\mathbf{x} \in \mathbb{Z}^L$. Sample r_{GenF} and run

$$(1^{n_f}, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}) \leftarrow \mathsf{GenF}(1^\lambda, \mathsf{pp}, f; r_{\mathsf{GenF}}).$$

  Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^{n_f}$, run

$$(e_{\mathrm{out}}, \mathbf{e}_1, \ldots, \mathbf{e}_L, \mathbf{e}_{\mathrm{t}}) \xleftarrow{\$} \mathsf{GenNoise}(1^\lambda, \mathsf{param}, \mathsf{pp}, f, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}, \mathbf{x}),$$

  and compute

$$\begin{cases} w_{\mathrm{out}} \leftarrow \mathbf{s}^\intercal \mathbf{w}_{\mathrm{out}} + e_{\mathrm{out}}, & \mathbf{w}_\ell^\intercal \leftarrow \mathbf{s}^\intercal(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_\ell^\intercal, & \mathbf{t}^\intercal \leftarrow \mathbf{s}^\intercal \mathbf{T} + \mathbf{e}_{\mathrm{t}}^\intercal, & \text{in } \mathsf{Exp}^{\mathrm{real}}_{\mathrm{garble}}; \\ w_{\mathrm{out}} \xleftarrow{\$} \mathbb{Z}_q, & \mathbf{w}_\ell \xleftarrow{\$} \mathbb{Z}_q^\star \quad (\text{for all } \ell \in [L]), & \mathbf{t} \xleftarrow{\$} \mathbb{Z}_q^\star, & \text{in } \mathsf{Exp}^{\mathrm{random}}_{\mathrm{garble}}; \end{cases}$$

  where $\{\mathbf{w}_\ell\}_{\ell\in[L]}$ and $\mathbf{t}$ in $\mathsf{Exp}^{\mathrm{random}}_{\mathrm{garble}}$ are of the same dimensions as their counterparts in $\mathsf{Exp}^{\mathrm{real}}_{\mathrm{garble}}$. Send $(r_{\mathsf{GenF}}, w_{\mathrm{out}}, \{\mathbf{w}_\ell\}_{\ell\in[L]}, \mathbf{t})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs $\beta' \in \{0, 1\}$. The outcome of the experiment is β' if $f(\mathbf{x}) = 0$. Otherwise, the outcome is set to 0.

7.6 Noisy Linear Garbling for Bounded-Arithmetic Circuits

In this section, we construct a noisy linear garbling scheme for arithmetic circuits of bounded wire values. It is a variant of [AIK11,IW14] and Chapter 4. The garbling scheme serves as the central component in our CP-ABE for bounded-arithmetic circuits. (The notations have undergone some major changes since the conference version [HLL24a] — see Remark 7.4.)

Construction 7.4 (noisy linear garbling for bounded-arithmetic circuits). The function family of bounded-arithmetic circuits is

$$\mathsf{Params} = \left\{ (M, 1^d) \;\middle|\; M \in \mathbb{N}_+ \text{ and } d \in \mathbb{N} \right\},$$

$$\mathcal{F}_{M,1^d} = \left\{ f_{M,C} \;\middle|\; C \text{ is an arithmetic circuit of depth no more than } d \right\},$$

$$f_{M,C}(\mathbf{x}) = \begin{cases} \perp, & \text{if some wire value} \notin [-M, M] \text{ when evaluating } C(\mathbf{x}); \\ 1, & \text{if } C(\mathbf{x}) = 0 \text{ and all wire values} \in [-M, M]; \\ 0, & \text{if } C(\mathbf{x}) \neq 0 \text{ and all wire values} \in [-M, M]; \end{cases}$$

$$\text{for } C : \mathbb{Z}^L \to \mathbb{Z} \text{ and } \mathbf{x} \in \mathbb{Z}^L.$$

Since M is known from param, the function $f_{M,C}$ is simply represented by C. The scheme works as follows.

- $\mathsf{Setup}(M, 1^d)$ picks suitable q, N and outputs $\mathsf{pp} = (q, M, 1^d, 1^N)$. The constraints of q, N are specified in Theorems 7.11, 7.12, and 7.13.
- $\mathsf{GenF}(\mathsf{pp}, C)$ generates the garbling for $C : \mathbb{Z}^L \to \mathbb{Z}$ in the following steps.
  - It parses C into L input gates $1, \ldots, L$ and $(|C| - L)$ arithmetic gates $L+1, \ldots, |C|$. They are sorted in topological order so that gate $|C|$ is the output gate of C and the inputs to every non-input gate i are connected to gates $\mathsf{gin}[i,1], \mathsf{gin}[i,2] < i$.
  - For each gate i, the algorithm samples its shrunken label function

    $$\mathbf{W}_{i,0} \xleftarrow{\$} \{0,1\}^{N \times N}, \qquad \mathbf{W}_{i,\times} \xleftarrow{\$} \{0,1\}^{N \times N}.$$

    We write $\mathbf{W}_{i,x} = \mathbf{W}_{i,0} + x\mathbf{W}_{i,\times}$, where $x \in [-M, M]$ is a potential output value of gate i, for the corresponding shrunken label (before randomization).
  - For each non-input gate i, the algorithm samples $\widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],0}, \widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],\times}$, $\widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],0}, \widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],\times}$, which contribute to the expanded label function of each input to gate i. Here, the superscript indicates the purpose of the part (due to being an input to gate i), and the subscript indicates the matrix containing that part, so

    $$\widetilde{\mathbf{W}}_{i',0} = \left(\ldots, \widetilde{\mathbf{W}}^{(i,\gamma)}_{i',0}, \ldots \right)_{\mathsf{gin}[i,\gamma]=i'}, \qquad \widetilde{\mathbf{W}}_{i',\times} = \left(\ldots, \widetilde{\mathbf{W}}^{(i,\gamma)}_{i',\times}, \ldots \right)_{\mathsf{gin}[i,\gamma]=i'}.$$

Similarly, we write $\widetilde{\mathbf{W}}_{i',x} = \widetilde{\mathbf{W}}_{i',0} + x\widetilde{\mathbf{W}}_{i',\times}$ for the expanded label (before randomization) corresponding to a potential output $x \in [-M, M]$ of gate i'. Suppose the input values to gate i are x_1, x_2, the goal is to compute the shrunken label $\mathbf{W}_{i,x}$ from the expanded labels $\widetilde{\mathbf{W}}_{\mathsf{gin}[i,1],x_1}, \widetilde{\mathbf{W}}_{\mathsf{gin}[i,2],x_2}$. Following [AIK11] (adapted for authenticity only), first sample $\mathbf{U}_i \xleftarrow{\$} \{0,1\}^{N\times N}$.

* If gate i is addition, set

$$\widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],0} \leftarrow \mathbf{U}_i, \qquad \widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],\times} \leftarrow \mathbf{W}_{i,\times},$$
$$\widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],0} \leftarrow \mathbf{W}_{i,0} - \mathbf{U}_i, \qquad \widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],\times} \leftarrow \mathbf{W}_{i,\times},$$

  which are subject to $\widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],x_1} + \widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],x_2} = \mathbf{W}_{i,x_1+x_2}$.

* If gate i is multiplication, set

$$\widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],0} \leftarrow \mathbf{U}_i, \qquad \widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],\times} \leftarrow \mathbf{W}_{i,\times},$$
$$\widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],0} \leftarrow \mathbf{W}_{i,0}, \qquad \widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],\times} \leftarrow -\mathbf{U}_i,$$

  which are subject to $x_2\widetilde{\mathbf{W}}^{(i,1)}_{\mathsf{gin}[i,1],x_1} + \widetilde{\mathbf{W}}^{(i,2)}_{\mathsf{gin}[i,2],x_2} = \mathbf{W}_{i,x_1x_2}$.

– For the output gate $|C|$, sample the expanded label function

$$\widetilde{\mathbf{W}}_{|C|,0} \xleftarrow{\$} \{0,1\}^{N\times 1}, \qquad \widetilde{\mathbf{W}}_{|C|,\times} \xleftarrow{\$} \{0,1\}^{N\times 1}.$$

  Here, $\widetilde{\mathbf{W}}_{|C|,0}, \widetilde{\mathbf{W}}_{|C|,\times}$ are vectors (single-column matrices; still in upper case for consistency with the other components).

– The algorithm computes the garbled table, which enables conversion to the expanded label $\widetilde{\mathbf{W}}_{i',x}$ from the shrunken label $\mathbf{W}_{i',x}$ for each gate i' with output value x, akin to the key-shrinking gadget in [AIK11]. For each tuple (i', i, γ) with $\mathsf{gin}[i,\gamma] = i'$ (i.e., gate i' is the γ^{th} input to gate i), sample and set

$$\mathbf{R}^{(i,\gamma)}_{i'} \xleftarrow{\$} \{0,1\}^{N\times N},$$
$$\mathbf{T}^{(i,\gamma)}_{i',0} \leftarrow \mathbf{W}_{i',0}\mathbf{R}^{(i,\gamma)}_{i'} + \widetilde{\mathbf{W}}^{(i,\gamma)}_{i',0}, \qquad \mathbf{T}^{(i,\gamma)}_{i',\times} \leftarrow \mathbf{W}_{i',\times}\mathbf{R}^{(i,\gamma)}_{i'} + \widetilde{\mathbf{W}}^{(i,\gamma)}_{i',\times}.$$

Similarly define $\mathbf{T}^{(i,\gamma)}_{i',x}$, then $\mathbf{T}^{(i,\gamma)}_{i',x} = \mathbf{W}_{i',x}\mathbf{R}^{(i,\gamma)}_{i'} + \widetilde{\mathbf{W}}^{(i,\gamma)}_{i',x}$. For the output gate, sample $\mathbf{R}_{|C|} \xleftarrow{\$} \{0,1\}^{N\times 1}$ and set

$$\mathbf{T}_{|C|,0} \leftarrow \mathbf{W}_{|C|,0}\mathbf{R}_{|C|} + \widetilde{\mathbf{W}}_{|C|,0}, \qquad \mathbf{T}_{|C|,\times} \leftarrow \mathbf{W}_{|C|,\times}\mathbf{R}_{|C|} + \widetilde{\mathbf{W}}_{|C|,\times}.$$

Here, $\mathbf{R}_{|C|}, \mathbf{T}_{|C|,0}, \mathbf{T}_{|C|,\times}$ are vectors (single-column matrices; still in upper case for consistency with the other components).

All the samplings are done in a straight-forward way.[86] The algorithm outputs

$$1^{n_C} = 1^N, \qquad \mathbf{w}_{\mathrm{out}} = \widetilde{\mathbf{W}}_{|C|,0}, \qquad \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]},$$
$$\mathbf{T} = \left(\ldots, \mathbf{T}^{(i,\gamma)}_{i',0}, \mathbf{T}^{(i,\gamma)}_{i',\times}, \ldots\right)_{\mathsf{gin}[i,\gamma]=i'}, \qquad \mathbf{R} = \left(\ldots, \mathbf{R}^{(i,\gamma)}_{i'}, \ldots\right)_{\mathsf{gin}[i,\gamma]=i'}.$$

- $\mathsf{Eval}(\mathsf{pp}, C, \mathbf{R}, \mathbf{x}, \{\mathbf{w}^{\mathsf{T}}_\ell\}_{\ell\in[L]}, \mathbf{t}^{\mathsf{T}})$ parses $C, \mathbf{R}$ as in GenF and

$$\mathbf{t}^{\mathsf{T}} = \left(\ldots, (\mathbf{t}^{(i,\gamma)}_{i',0})^{\mathsf{T}}, (\mathbf{t}^{(i,\gamma)}_{i',\times})^{\mathsf{T}}, \ldots\right)_{\mathsf{gin}[i,\gamma]=i'}.$$

  It evaluates $C(\mathbf{x})$ for the wire values $\{x_i\}_{i\in[|C|]}$ of each gate i, namely,

$$x_i \leftarrow \begin{cases} \mathbf{x}[\ell], & \text{if gate } i \text{ is an input gate } (i = \ell \in [L]); \\ x_{\mathsf{gin}[i,1]} + x_{\mathsf{gin}[i,2]}, & \text{if gate } i \text{ is addition;} \\ x_{\mathsf{gin}[i,1]} x_{\mathsf{gin}[i,2]}, & \text{if gate } i \text{ is multiplication.} \end{cases}$$

  The algorithm has the (shrunken) input labels $\{\mathbf{w}^{\mathsf{T}}_\ell\}_{\ell\in[L]}$ as input. It computes the label of each non-input gate in increasing order by its index i.

  - The algorithm decrypts for the expanded label using

$$(\widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1]})^{\mathsf{T}} \leftarrow (\mathbf{t}^{(i,1)}_{\mathsf{gin}[i,1],x_{\mathsf{gin}[i,1]}})^{\mathsf{T}} - \mathbf{w}^{\mathsf{T}}_{\mathsf{gin}[i,1]} \mathbf{R}^{(i,1)}_{\mathsf{gin}[i,1]},$$
$$(\widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2]})^{\mathsf{T}} \leftarrow (\mathbf{t}^{(i,2)}_{\mathsf{gin}[i,2],x_{\mathsf{gin}[i,2]}})^{\mathsf{T}} - \mathbf{w}^{\mathsf{T}}_{\mathsf{gin}[i,2]} \mathbf{R}^{(i,2)}_{\mathsf{gin}[i,2]},$$

    where $\mathbf{t}^{(i,1)}_{\mathsf{gin}[i,1],x_{\mathsf{gin}[i,1]}} = \mathbf{t}^{(i,1)}_{\mathsf{gin}[i,1],0} + x_{\mathsf{gin}[i,1]} \mathbf{t}^{(i,1)}_{\mathsf{gin}[i,1],\times}$ (similarly for $\mathbf{t}^{(i,2)}_{\mathsf{gin}[i,2],x_{\mathsf{gin}[i,2]}}$) can be computed from $\mathbf{t}$ (input to Eval) and x's.

  - The algorithm recovers the shrunken label as

$$\mathbf{w}_i \leftarrow \begin{cases} \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1]} + \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2]}, & \text{if gate } i \text{ is addition;} \\ x_{\mathsf{gin}[i,2]} \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1]} + \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2]}, & \text{if gate } i \text{ is multiplication.} \end{cases}$$

[86] Precisely speaking, r_{GenF} conditioned on the sampled matrices must be efficiently sampleable given those matrices (with negligible statistical error), e.g., when the matrices are just the bits read sequentially from r_{GenF}. This ensures that no sampled matrix has a known trapdoor, and is important because r_{GenF} is incorporated into the sampler's randomness in a security proof.

Lastly, the algorithm computes and outputs the secret

$$w_{\text{out}} \leftarrow \mathbf{t}_{|C|,x_{|C|}}^{\intercal} - \mathbf{w}_{|C|}^{\intercal}\mathbf{R}_{|C|}.$$

Here, $\mathbf{t}_{|C|,x_{|C|}}$ is a scalar (one-dimensional vector; still boldfaced for consistency with the other components).

Remark 7.4 (changes since the conference version [HLL24a]). We list major changes for those who have read the conference version of Construction 7.4.

- In Params, the wire value bound M is now encoded in binary (instead of unary) for stronger functionality since it need not be $\text{poly}(\lambda)$-bounded.
- The dimension of secret is N (instead of n, m) since it is for flipped LWE secret. It is now encoded in unary in pp to emphasize $\text{poly}(\lambda)$-boundedness.
- Gates are named simply by their indices (instead of g subscript index).
- The notations i, i' are more consistent. When both appear in a context, gate i' is always an input to gate i, and i' never appears as component superscript. The notations in the proof are also improved for consistency.
- The expanded label function of the output gate is now $\widetilde{\mathbf{W}}$ (instead of $\widetilde{\mathbf{w}}$) to reduce ambiguity.

7.6.1 Correctness and Shortness

Theorem 7.11 (¶). *Construction 7.4 is* $(B_{\text{in}}, B_{\text{out}})$-correct *(Definition 7.4) if*

$$(6NM^2)^{d+1}B_{\text{in}}(M,d) \le B_{\text{out}}(M,d).$$

Proof (Theorem 7.11). Let $C : \mathbb{Z}^L \to \mathbb{Z}$ be an arithmetic circuit of depth at most d and $\mathbf{x} \in \mathbb{Z}^L$ an input with $C(\mathbf{x}) = 0$ such that all wire values in $C(\mathbf{x})$ are bounded by M. Let $\mathbf{s} \in \mathbb{Z}_q^N$ be a secret and $e_{\text{out}}, \mathbf{e}_1, \ldots, \mathbf{e}_L, \mathbf{e}_{\text{t}}$ any B_{in}-bounded errors. We extend the notation of $\mathbf{e}$'s by $\mathbf{e}_i = \mathbf{w}_i - (\mathbf{s}^{\intercal}\mathbf{W}_{i,x_i})^{\intercal}$ for all $L < i \le |C|$, where x_i's are the wire values. The extension symbolically agrees with $\mathbf{e}_1, \ldots, \mathbf{e}_L$.

We show by induction that during evaluation, $\|\mathbf{e}_i\| \le (6NM^2)^{d_i} B_{\text{in}}$ for every gate i, where d_i is the depth of gate i (the input gates are of depth 0). The base case (for $\mathbf{e}_1, \dots, \mathbf{e}_L$) is by assumption. For the inductive case, let $L < i \le |C|$. Assuming the induction hypothesis for all $i' < i$, then it applies to $i' = \text{gin}[i, \gamma] < i$ for both $\gamma \in \{1, 2\}$, and together with $d_{i'} \le d_i - 1$, we have

$$\|\mathbf{e}_{i'}\| \le (6NM^2)^{d_{i'}} B_{\text{in}} \le (6NM^2)^{d_i - 1} B_{\text{in}}.$$

By the definition of $\mathbf{t}$ and how Eval proceeds,

$$\begin{aligned}
\left(\widetilde{\mathbf{w}}_{i'}^{(i,\gamma)}\right)^{\mathsf{T}} &= \left(\mathbf{s}^{\mathsf{T}} \mathbf{T}_{i',0}^{(i,\gamma)} + (\mathbf{e}_{\text{t},i',0}^{(i,\gamma)})^{\mathsf{T}}\right) + x_{i'}\left(\mathbf{s}^{\mathsf{T}} \mathbf{T}_{i',\times}^{(i,\gamma)} + (\mathbf{e}_{\text{t},i',\times}^{(i,\gamma)})^{\mathsf{T}}\right) - (\mathbf{s}^{\mathsf{T}} \mathbf{W}_{i',x_{i'}} + \mathbf{e}_{i'}^{\mathsf{T}}) \mathbf{R}_{i'}^{(i,\gamma)} \\
&= \mathbf{s}^{\mathsf{T}} \widetilde{\mathbf{W}}_{i',x_{i'}}^{(i,\gamma)} + \left(\mathbf{e}_{\text{t},i',0}^{(i,\gamma)} + x_{i'} \mathbf{e}_{\text{t},i',\times}^{(i,\gamma)}\right)^{\mathsf{T}} - \mathbf{e}_{i'}^{\mathsf{T}} \mathbf{R}_{i'}^{(i,\gamma)} = \mathbf{s}^{\mathsf{T}} \widetilde{\mathbf{W}}_{i',x_{i'}}^{(i,\gamma)} + \left(\widetilde{\mathbf{e}}_{i'}^{(i,\gamma)}\right)^{\mathsf{T}},
\end{aligned}$$

where superscripts/subscripts follow the usual meaning of taking the relevant parts. It follows that the errors on the expanded labels are

$$\begin{aligned}
\left\|\widetilde{\mathbf{e}}_{i'}^{(i,\gamma)}\right\| &\le \left\|\mathbf{e}_{\text{t},i',0}^{(i,\gamma)}\right\| + |x_{i'}| \cdot \left\|\mathbf{e}_{\text{t},i',\times}^{(i,\gamma)}\right\| + \left\|(\mathbf{R}_{i'}^{(i,\gamma)})^{\mathsf{T}}\right\| \cdot \|\mathbf{e}_{i'}\| \\
&\le B_{\text{in}} + M \cdot B_{\text{in}} + N \cdot (6NM^2)^{d_i - 1} B_{\text{in}} \\
&\le (N + M + 1)(6NM^2)^{d_i - 1} B_{\text{in}}.
\end{aligned}$$

We proceed to a case analysis.

- If gate i is addition,

$$\begin{aligned}
\mathbf{e}_i^{\mathsf{T}} &= \overbrace{\left(\mathbf{s}^{\mathsf{T}} \widetilde{\mathbf{W}}_{\text{gin}[i,1],x_{\text{gin}[i,1]}}^{(i,1)} + (\widetilde{\mathbf{e}}_{\text{gin}[i,1]}^{(i,1)})^{\mathsf{T}}\right) + \left(\mathbf{s}^{\mathsf{T}} \widetilde{\mathbf{W}}_{\text{gin}[i,2],x_{\text{gin}[i,2]}}^{(i,2)} + (\widetilde{\mathbf{e}}_{\text{gin}[i,2]}^{(i,2)})^{\mathsf{T}}\right)}^{\mathbf{w}_i = \widetilde{\mathbf{w}}_{\text{gin}[i,1]}^{(i,1)} + \widetilde{\mathbf{w}}_{\text{gin}[i,2]}^{(i,2)}} - \mathbf{s}^{\mathsf{T}} \mathbf{W}_{i,x_i} \\
&= \left(\mathbf{e}_{\text{gin}[i,1]}^{(i,1)} + \mathbf{e}_{\text{gin}[i,2]}^{(i,2)}\right)^{\mathsf{T}},
\end{aligned}$$

  so $\|\mathbf{e}_i\| \le 2(N + M + 1)(6NM^2)^{d_i - 1} B_{\text{in}} \le (6NM^2)^{d_i} B_{\text{in}}$.

- If gate i is multiplication,

$$\begin{aligned}
\mathbf{e}_i^{\mathsf{T}} &= \overbrace{x_{\text{gin}[i,2]}\left(\mathbf{s}^{\mathsf{T}} \widetilde{\mathbf{W}}_{\text{gin}[i,1],x_{\text{gin}[i,1]}}^{(i,1)} + (\widetilde{\mathbf{e}}_{\text{gin}[i,1]}^{(i,1)})^{\mathsf{T}}\right) + \left(\mathbf{s}^{\mathsf{T}} \widetilde{\mathbf{W}}_{\text{gin}[i,2],x_{\text{gin}[i,2]}}^{(i,2)} + (\widetilde{\mathbf{e}}_{\text{gin}[i,2]}^{(i,2)})^{\mathsf{T}}\right)}^{\mathbf{w}_i = x_{\text{gin}[i,2]} \widetilde{\mathbf{w}}_{\text{gin}[i,1]}^{(i,1)} + \widetilde{\mathbf{w}}_{\text{gin}[i,2]}^{(i,2)}} - \mathbf{s}^{\mathsf{T}} \mathbf{W}_{i,x_i} \\
&= \left(x_{\text{gin}[i,2]} \mathbf{e}_{\text{gin}[i,1]}^{(i,1)} + \mathbf{e}_{\text{gin}[i,2]}^{(i,2)}\right)^{\mathsf{T}},
\end{aligned}$$

  so $\|\mathbf{e}_i\| \le (M + 1)(N + M + 1)(6NM^2)^{d_i - 1} B_{\text{in}} \le (6NM^2)^{d_i} B_{\text{in}}$.

This completes the induction.

Applying the error bound to the output gate, we have

$$\|\mathbf{e}_{|C|}\| \le (6NM^2)^{d_{|C|}} B_{\text{in}} \le (6NM^2)^d B_{\text{in}}.$$

Note that $x_{|C|} = C(\mathbf{x}) = 0$ by assumption, and we have

$$\begin{aligned}
&w_{\text{out}} - (\mathbf{s}^\intercal \mathbf{w}_{\text{out}} + e_{\text{out}}) \\
&= \left(\mathbf{t}_{|C|,0}^\intercal - \mathbf{w}_{|C|}^\intercal \mathbf{R}_{|C|}\right) - (\mathbf{s}^\intercal \mathbf{w}_{\text{out}} + e_{\text{out}}) \\
&= \left(\mathbf{s}^\intercal \mathbf{T}_{|C|,0} + \mathbf{e}_{\text{t},|C|,0}^\intercal\right) - \left(\mathbf{s}^\intercal \mathbf{W}_{|C|,0} + \mathbf{e}_{|C|}^\intercal\right)\mathbf{R}_{|C|} - \mathbf{s}^\intercal \mathbf{w}_{\text{out}} - e_{\text{out}} \\
&= \mathbf{e}_{\text{t},|C|,0}^\intercal - \mathbf{e}_{|C|}^\intercal \mathbf{R}_{|C|} - e_{\text{out}} + \mathbf{s}^\intercal \underbrace{\left(\mathbf{W}_{|C|,0}\mathbf{R}_{|C|} + \widetilde{\mathbf{W}}_{|C|,0} - \mathbf{W}_{|C|,0}\mathbf{R}_{|C|} - \widetilde{\mathbf{W}}_{|C|,0}\right)}_{\mathbf{0}}.
\end{aligned}$$

Therefore, the output error is bounded by

$$\begin{aligned}
\|\mathbf{e}_{\text{t},|C|,0}\| + \left\|\mathbf{R}_{|C|}^\intercal\right\| \cdot \|\mathbf{e}_{|C|}\| + |e_{\text{out}}| &\le B_{\text{in}} + N \cdot (6NM^2)^d B_{\text{in}} + B_{\text{in}} \\
&\le (6NM^2)^{d+1} B_{\text{in}} \le B_{\text{out}}.
\end{aligned}$$

□

Theorem 7.12 (¶). *Construction 7.4 is* B_{short}-short *(Definition 7.5) if*

$$B_{\text{short}}(M, d) \ge (N + M + 2)N.$$

Proof (Theorem 7.12). By definition, $\mathbf{w}_{\text{out}}^\intercal \in \{0,1\}^{1\times N}$, so

$$\|\mathbf{w}_{\text{out}}^\intercal\| \le N \le (N + M + 2)N \le B_{\text{short}}.$$

For all $\ell \in [L]$, when $|\mathbf{x}[\ell]| \le M$,

$$\|\mathbf{W}_{\ell,\mathbf{x}[\ell]}^\intercal\| \le \|\mathbf{W}_{\ell,0}^\intercal\| + M\|\mathbf{W}_{\ell,\times}^\intercal\| \le (M+1)N \le (N + M + 2)N \le B_{\text{short}},$$

because $\mathbf{W}$'s are in $\{0,1\}^{N\times N}$. For $\mathbf{T}$, observe that each $\widetilde{\mathbf{W}}$ is a signed sum of at most two matrices in $\{0,1\}^{N\times N}$, so each $\widetilde{\mathbf{W}}^\intercal$ has norm bound $2N$. Therefore,

$$\|\mathbf{T}^\intercal\| \le \overbrace{M}^{\mathbf{R}^\intercal} \cdot \overbrace{N}^{\mathbf{W}^\intercal} + \overbrace{2N}^{\widetilde{\mathbf{W}}^\intercal} = (M+2)N \le (N + M + 2)N \le B_{\text{short}}.$$

□

7.6.2 Security with Gaussian Noise

Theorem 7.13 (¶). *Suppose*

$$\mathsf{GenGaussian}(M, 1^d, (q, M, 1^d, 1^N), C, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}, \mathbf{x})$$

samples (truncated) Gaussian noises with appropriate width σ of suitable shape

$$(e_{\mathrm{out}}, \mathbf{e}_1, \ldots, \mathbf{e}_L, \mathbf{e}_{\mathrm{t}}) \xleftarrow{\$} \mathcal{D}^{\star}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\kappa}},$$

then Construction 7.4 is GenGaussian-secure *(Definition 7.7) if q is always a prime and* $\mathsf{FlipLWE}_{N,\mathrm{poly}(\lambda),q,\sigma'}$ *(Assumption 2.3) holds for some $\sigma' \le \frac{\sigma}{2^{\kappa+6}(2N+M+2)\sqrt{\kappa}}$.*[87]

It is known (see [AIK11,IW14] and Chapter 4) that the information-theoretic version of arithmetic garbling satisfies the following simple simulation property — the labels are uniformly random conditioned on evaluation correctness. Construction 7.4 is similar in this regard. The garbling is a noisy linear randomization of the public matrices. Once LWE is applied:

- the expanded labels of fan-ins are uniformly random conditioned on the correct evaluation into the shrunken labels (of fan-outs),
- the garbled table is uniformly random conditioned on the correct recovery of the expanded labels from the shrunken labels (of the same gate), and
- the expanded labels of the output gate are uniformly random conditioned on the approximate recovery of $\mathbf{s}^{\mathsf{T}}(\widetilde{\mathbf{W}}_{|C|,0} + C(\mathbf{x}) \cdot \widetilde{\mathbf{W}}_{|C|,\times})$.

Combining these points, the input labels and the garbled table are jointly random, conditioned on the approximate recovery of $\mathbf{s}^{\mathsf{T}}(\widetilde{\mathbf{W}}_{|C|,0} + C(\mathbf{x}) \cdot \widetilde{\mathbf{W}}_{|C|,\times})$. When this latter value itself is random, the overall distribution of input labels and garbled table is simply random. Recall that security is considered only when $C(\mathbf{x}) \neq 0$. Since q is a prime, the evaluation result will be independent of the secret approximating $\mathbf{s}^{\mathsf{T}}\widetilde{\mathbf{W}}_{|C|,0}$ once LWE is applied. Therefore, the revealed components (secret, input labels, garbled table) are jointly pseudorandom. The proof below formalizes this idea.

[87] The constant has been changed from the conference version [HLL24a]. Both versions are correct. The new version fits the proof better.

Proof (Theorem 7.13). We show $\mathsf{Exp}^{\mathrm{real}}_{\mathrm{garble}} \approx \mathsf{Exp}^{\mathrm{random}}_{\mathrm{garble}}$ with the following hybrids.

- H_0 is $\mathsf{Exp}^{\mathrm{real}}_{\mathrm{garble}}$. The adversary sees $r_{\mathsf{Setup}}, r_{\mathsf{GenF}}$ and

$$w_{\mathrm{out}} = \mathbf{s}^\intercal \widetilde{\mathbf{W}}_{|C|,0} + e_{\mathrm{out}}, \qquad \mathbf{w}_\ell^\intercal = \mathbf{s}^\intercal(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_\ell^\intercal,$$
$$(\mathbf{t}_{i',0}^{(i,\gamma)})^\intercal = \mathbf{s}^\intercal \mathbf{T}_{i',0}^{(i,\gamma)} + (\mathbf{e}_{i',0}^{(i,\gamma)})^\intercal, \qquad (\mathbf{t}_{i',\times}^{(i,\gamma)})^\intercal = \mathbf{s}^\intercal \mathbf{T}_{i',\times}^{(i,\gamma)} + (\mathbf{e}_{i',\times}^{(i,\gamma)})^\intercal,$$

  where the indices for $\mathbf{t}$ are constrained by $\mathsf{gin}[i,\gamma] = i'$ (henceforth same and omitted).

- In H_1, we rewrite the garbling using distributivity and associativity. Recall that each block in $\mathbf{T}$ is a signed sum of $\mathbf{W}$'s and $\mathbf{U}$'s. Instead of computing $\mathbf{T}$, multiplying $\mathbf{s}$, then adding the noise, we compute $\mathbf{s}$ multiplied by $\mathbf{W}$'s and $\mathbf{U}$'s, sum them with appropriate signs, then add the noise. Namely, for all $i \in [|C|]$, compute

$$\mathbf{w}_{i,0}^\intercal = \mathbf{s}^\intercal \mathbf{W}_{i,0}, \qquad \mathbf{w}_{i,\times}^\intercal = \mathbf{s}^\intercal \mathbf{W}_{i,\times}, \qquad (\text{if } i > L) \quad \mathbf{u}_i^\intercal = \mathbf{s}^\intercal \mathbf{U}_i,$$

  and $\widetilde{w}_{\mathrm{out},0} = \mathbf{s}^\intercal \widetilde{\mathbf{W}}_{|C|,0}$ and $\widetilde{w}_{\mathrm{out},\times} = \mathbf{s}^\intercal \widetilde{\mathbf{W}}_{|C|,\times}$. Those components are called "initial $\mathbf{w}$'s and $\mathbf{u}$'s", which are not given directly to the adversary. Next, compute the randomized expanded label functions as follows (compare with GenF of Construction 7.4).

  - If gate i is addition, set

$$\widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],0} = \mathbf{u}_i, \qquad \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],\times} = \mathbf{w}_{i,\times},$$
$$\widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],0} = \mathbf{w}_{i,0} - \mathbf{u}_i, \qquad \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],\times} = \mathbf{w}_{i,\times}.$$

  - If gate i is multiplication, set

$$\widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],0} = \mathbf{u}_i, \qquad \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],\times} = \mathbf{w}_{i,\times},$$
$$\widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],0} = \mathbf{w}_{i,0}, \qquad \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],\times} = -\mathbf{u}_i.$$

  Then, compute the $\mathbf{s}$-linear parts (denoted by $\widetilde{\mathbf{t}}$) of $\mathbf{t}$ as

$$(\widetilde{\mathbf{t}}_{i',0}^{(i,\gamma)})^\intercal = \mathbf{w}_{i',0}^\intercal \mathbf{R}_{i'}^{(i,\gamma)} + (\widetilde{\mathbf{w}}_{i',0}^{(i,\gamma)})^\intercal, \qquad (\widetilde{\mathbf{t}}_{i',\times}^{(i,\gamma)})^\intercal = \mathbf{w}_{i',\times}^\intercal \mathbf{R}_{i'}^{(i,\gamma)} + (\widetilde{\mathbf{w}}_{i',\times}^{(i,\gamma)})^\intercal.$$

The randomized secret, input labels, and garble table are

$$w_{\text{out}} = \widetilde{w}_{\text{out},0} + e_{\text{out}}, \qquad \mathbf{w}_\ell = \mathbf{w}_{\ell,0} + \mathbf{x}[\ell]\mathbf{w}_{\ell,\times} + \mathbf{e}_\ell,$$
$$\mathbf{t}^{(i,\gamma)}_{i',0} = \widetilde{\mathbf{t}}^{(i,\gamma)}_{i',0} + \mathbf{e}^{(i,\gamma)}_{\text{t},i',0}, \qquad \mathbf{t}^{(i,\gamma)}_{i',\times} = \widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times} + \mathbf{e}^{(i,\gamma)}_{\text{t},i',\times}.$$

Clearly, $\mathsf{H}_0 \equiv \mathsf{H}_1$.

- In H_2, additional small noises are attached to the initial $\mathbf{w}$'s and $\mathbf{u}$'s as follows. Sample $\overline{\mathbf{e}}_{i,0}, \overline{\mathbf{e}}_{i,\times}, \overline{\mathbf{e}}_{\text{U},i}$'s of appropriate dimensions and $\overline{e}_{\text{out},0}, \overline{e}_{\text{out},\times}$, each entry from $\mathcal{D}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\kappa}}$, and set

$$\mathbf{w}^{\mathsf{T}}_{i,0} = \mathbf{s}^{\mathsf{T}}\mathbf{W}_{i,0} + \overline{\mathbf{e}}^{\mathsf{T}}_{i,0}, \qquad \mathbf{w}^{\mathsf{T}}_{i,\times} = \mathbf{s}^{\mathsf{T}}\mathbf{W}_{i,\times} + \overline{\mathbf{e}}^{\mathsf{T}}_{i,\times}, \qquad (i > L) \quad \mathbf{u}^{\mathsf{T}}_i = \mathbf{s}^{\mathsf{T}}\mathbf{U}_i + \overline{\mathbf{e}}^{\mathsf{T}}_{\text{U},i},$$
$$\widetilde{w}_{\text{out},0} = \mathbf{s}^{\mathsf{T}}\widetilde{\mathbf{W}}_{|C|,0} + \overline{e}_{\text{out},0}, \quad \widetilde{w}_{\text{out},\times} = \mathbf{s}^{\mathsf{T}}\widetilde{\mathbf{W}}_{|C|,\times} + \overline{e}_{\text{out},\times}.$$

  Recall that the initial $\mathbf{w}$'s and $\mathbf{u}$'s are not directly given to the adversary. In the revealed components (secret, input labels, garbled table), the magnitude of errors contributed by the additional small noises is bounded by

$$\sigma'\sqrt{\kappa} \cdot \max\{M+1, N+2\} \le (N+M+2)\sigma'\sqrt{\kappa} \le 2^{-\kappa-6}\sigma.$$

  Therefore, they are flooded (Lemma 2.2) by the randomizing noises ($\mathbf{e}$'s and e's, without bars), i.e., $\mathsf{H}_1 \approx_{\text{s}} \mathsf{H}_2$.

- In H_3, the small noises $\overline{\mathbf{e}}_{i,0}, \overline{\mathbf{e}}_{i,\times}, \overline{\mathbf{e}}_{\text{U},i}$'s and $\overline{e}_{\text{out},0}, \overline{e}_{\text{out},\times}$ are no longer truncated, i.e., the entries are from $\mathcal{D}_{\mathbb{Z},\sigma'}$. We have $\mathsf{H}_2 \approx_{\text{s}} \mathsf{H}_3$ by Lemma 2.1.

- In H_4, the initial $\mathbf{w}$'s and $\mathbf{u}$'s are replaced by random values, i.e.,

$$\mathbf{w}_{i,0} \xleftarrow{\$} \mathbb{Z}_q^N, \quad \mathbf{w}_{i,\times} \xleftarrow{\$} \mathbb{Z}_q^N, \quad (i > L)\ \mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_q^N, \quad \widetilde{w}_{\text{out},0} \xleftarrow{\$} \mathbb{Z}_q, \quad \widetilde{w}_{\text{out},\times} \xleftarrow{\$} \mathbb{Z}_q.$$

  $\mathsf{H}_3 \approx \mathsf{H}_4$ follows from $\mathsf{FlipLWE}_{N,N(L+|C|)+2,q,\sigma'}$.

  For this step, we rely on the "straight-forwardness", i.e., r_{GenF}, which must be produced by the reduction algorithm for the underlying adversary, is efficiently sampleable conditioned on and given $\mathbf{W}_{i,0}, \mathbf{W}_{i,\times}, \mathbf{U}_i$'s and $\widetilde{\mathbf{W}}_{|C|,0}, \widetilde{\mathbf{W}}_{|C|,\times}$, which are LWE public matrices received by the reduction algorithm.

- In H_5, instead of sampling $\mathbf{w}_{i,0}$'s and $\mathbf{u}_i$'s, we sample and set

$$\begin{aligned} & \mathbf{w}_{i,x_i} \xleftarrow{\$} \mathbb{Z}_q^N, && \mathbf{w}_{i,0} = \mathbf{w}_{i,x_i} - x_i \mathbf{w}_{i,\times}, \\ (i > L) \qquad & \overline{\mathbf{u}}_i \xleftarrow{\$} \mathbb{Z}_q^N, && \mathbf{u}_i = \overline{\mathbf{u}}_i - x_{\mathsf{gin}[i,1]} \mathbf{w}_{i,\times}, \end{aligned}$$

where x_i's are the wire values in the evaluation of $C(\mathbf{x})$. This is simply a change of variable, so $\mathsf{H}_4 \equiv \mathsf{H}_5$.

Approximations of $\mathbf{w}_{i,x_i}$'s would be computed during an honest evaluation — we call $\mathbf{w}_{i,x_i}$'s the "active" shrunken labels. Note that the input labels given to the adversary are just the active ones, i.e., $\mathbf{w}_\ell = \mathbf{w}_{i,0} + \mathbf{x}[\ell]\mathbf{w}_{i,\times} + \mathbf{e}_\ell = \mathbf{w}_{i,x_i} + \mathbf{e}_\ell$. For the "active" expanded labels, we use a case analysis. If gate i is addition,

$$\begin{aligned} \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],x_{\mathsf{gin}[i,1]}} &= \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],0} + x_{\mathsf{gin}[i,1]} \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],\times} = \mathbf{u}_i + x_{\mathsf{gin}[i,1]} \mathbf{w}_{i,\times} = \overline{\mathbf{u}}_i, \\ \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],x_{\mathsf{gin}[i,2]}} &= \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],0} + x_{\mathsf{gin}[i,2]} \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],\times} \\ &= (\mathbf{w}_{i,0} - \mathbf{u}_i) + x_{\mathsf{gin}[i,2]} \mathbf{w}_{i,\times} \\ &= \mathbf{w}_{i,0} + (x_{\mathsf{gin}[i,1]} + x_{\mathsf{gin}[i,2]}) \mathbf{w}_{i,\times} - (\mathbf{u}_i + x_{\mathsf{gin}[i,1]} \mathbf{w}_{i,\times}) \\ (x_i = x_{\mathsf{gin}[i,1]} + x_{\mathsf{gin}[i,2]}) \quad &= \mathbf{w}_{i,x_i} - \overline{\mathbf{u}}_i. \end{aligned}$$

If gate i is multiplication,

$$\begin{aligned} \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],x_{\mathsf{gin}[i,1]}} &= \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],0} + x_{\mathsf{gin}[i,1]} \widetilde{\mathbf{w}}^{(i,1)}_{\mathsf{gin}[i,1],\times} = \mathbf{u}_i + x_{\mathsf{gin}[i,1]} \mathbf{w}_{i,\times} = \overline{\mathbf{u}}_i, \\ \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],x_{\mathsf{gin}[i,2]}} &= \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],0} + x_{\mathsf{gin}[i,2]} \widetilde{\mathbf{w}}^{(i,2)}_{\mathsf{gin}[i,2],\times} \\ &= \mathbf{w}_{i,0} - x_{\mathsf{gin}[i,2]} \mathbf{u}_i \\ &= \mathbf{w}_{i,0} + x_{\mathsf{gin}[i,1]} x_{\mathsf{gin}[i,2]} \mathbf{w}_{i,\times} - x_{\mathsf{gin}[i,2]} (\mathbf{u}_i + x_{\mathsf{gin}[i,1]} \mathbf{w}_{i,\times}) \\ (x_i = x_{\mathsf{gin}[i,1]} x_{\mathsf{gin}[i,2]}) \quad &= \mathbf{w}_{i,x_i} - x_{\mathsf{gin}[i,2]} \overline{\mathbf{u}}_i. \end{aligned}$$

The point is to see that the active expanded labels are random subject to correct evaluation into active shrunken labels, and that they are independent of values with subscript "×". Moreover, $\widetilde{\mathbf{t}}$ components with subscript "0" now become

$$\begin{aligned} (\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',0})^\intercal &= \mathbf{w}^\intercal_{i',0} \mathbf{R}^{(i,\gamma)}_{i'} + (\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',0})^\intercal \\ &= (\mathbf{w}^\intercal_{i',x_{i'}} - x_{i'} \mathbf{w}^\intercal_{i',\times}) \mathbf{R}^{(i,\gamma)}_{i'} + \big((\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',x_{i'}})^\intercal - x_{i'} (\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',\times})^\intercal\big) \\ &= \mathbf{w}^\intercal_{i',x_{i'}} \mathbf{R}^{(i,\gamma)}_{i'} + (\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',x_{i'}})^\intercal - x_{i'} (\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times})^\intercal, \end{aligned}$$

i.e., the active expanded label ($\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',x_{i'}}$) padded by the active shrunken label ($\mathbf{w}_{i',x_{i'}}$ stretched by $\mathbf{R}$), shifted by some known amount ($x_{i'}\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times}$).

- In H_6, additional small noises are attached to $\widetilde{\mathbf{t}}$ components with subscript "$\times$" as follows. Sample $\mathbf{w}_{i,x_i}, \mathbf{w}_{i,\times}, \overline{\mathbf{u}}_i$'s at random, compute $\widetilde{\mathbf{w}}^{(i,\gamma)}_{x',x_{i'}}$'s as explained in H_5, and compute $\widetilde{\mathbf{w}}^{(i,\gamma)}_{x',\times}$ as usual. Sample $\overline{\mathbf{e}}^{(i,\gamma)}_{\mathrm{t},i',\times}$'s with entries independently from $\mathcal{D}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\kappa}}$ and set

$$(\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times})^\intercal = \mathbf{w}^\intercal_{i',\times}\mathbf{R}^{(i,\gamma)}_{i'} + (\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',\times})^\intercal + (\overline{\mathbf{e}}^{(i,\gamma)}_{\mathrm{t},i',\times})^\intercal.$$

  Compute $\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',0}$'s as explained in H_5, and lastly the revealed components (secret, input labels, garbled table). The magnitude of inserted noises in the revealed components (in fact, just the randomized garbled table $\mathbf{t}$) is bounded by

$$\sigma'\sqrt{\kappa}\cdot\max\{|x_i|\} \le (N+M+2)\sigma'\sqrt{\kappa} \le 2^{-\kappa-6}\sigma,$$

  so $\mathsf{H}_5 \approx_{\mathrm{s}} \mathsf{H}_6$ by Lemma 2.2.

- In H^0_7, the small noises in $\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times}$'s are no longer truncated, i.e., entries of $\overline{\mathbf{e}}^{(i,\gamma)}_{\mathrm{t},i',\times}$ follow $\mathcal{D}_{\mathbb{Z},\sigma'}$. We have $\mathsf{H}_6 \approx_{\mathrm{s}} \mathsf{H}^0_7$ by Lemma 2.1.

- In $\mathsf{H}^{\mathfrak{i}'}_7$ (for $\mathfrak{i}'$ from 1 up to $|C|$), the components $\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times}$ with $i' \le \mathfrak{i}'$ are replaced by random. Comparing $\mathsf{H}^{\mathfrak{i}'-1}_7$ and $\mathsf{H}^{\mathfrak{i}'}_7$, the only change is

$$\begin{array}{rlll} (\widetilde{\mathbf{t}}^{(i,\gamma)}_{\mathfrak{i}',\times})^\intercal & \text{from} & \mathbf{w}^\intercal_{\mathfrak{i}',\times}\mathbf{R}^{(i,\gamma)}_{\mathfrak{i}'} + (\widetilde{\mathbf{w}}^{(i,\gamma)}_{\mathfrak{i}',\times})^\intercal + (\overline{\mathbf{e}}^{(i,\gamma)}_{\mathrm{t},\mathfrak{i}',\times})^\intercal & \text{in } \mathsf{H}^{\mathfrak{i}'-1}_7 \\ & \text{to} & \text{random} & \text{in } \mathsf{H}^{\mathfrak{i}'}_7, \text{ where } \mathsf{gin}[i,\gamma] = \mathfrak{i}'. \end{array}$$

  The indistinguishability follows from flipped LWE with secret $\mathbf{w}_{\mathfrak{i}',\times}$ and public matrix $(\ldots, \mathbf{R}^{(i,\gamma)}_{\mathfrak{i}'}, \ldots)_{\mathsf{gin}[i,\gamma]=\mathfrak{i}'}$. To verify that the reduction works, we inspect where $\mathbf{w}_{\mathfrak{i}',\times}$ might appear in the revealed components.

  - The secret is $w_{\mathrm{out}} = \widetilde{w}_{\mathrm{out},0} + e_{\mathrm{out}}$, independent of $\mathbf{w}_{\mathfrak{i}',\times}$.
  - The input labels are $\mathbf{w}_\ell = \mathbf{w}_{\ell,x_\ell} + \mathbf{e}_\ell$, independent of $\mathbf{w}_{\mathfrak{i}',\times}$.
  - The garbled table is $\widetilde{\mathbf{t}}$ plus randomizing noises (without bars, independent of $\mathbf{w}_{\mathfrak{i}',\times}$). The $\widetilde{\mathbf{t}}$ components with subscript "$\times$" require a case analysis.

* For $i' < \mathfrak{i}'$, the $\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times}$'s are already replaced by random, hence independent of $\mathbf{w}_{\mathfrak{i}',\times}$.
* For $i' \geq \mathfrak{i}'$, we have

$$(\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times})^\intercal = \mathbf{w}^\intercal_{i',\times}\mathbf{R}^{(i,\gamma)}_{i'} + (\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',\times})^\intercal + (\overline{\mathbf{e}}^{(i,\gamma)}_{\mathsf{t},i',\times})^\intercal.$$

  The first and third terms are either received together as input to the reduction algorithm (for $i' = \mathfrak{i}'$) or independent of $\mathbf{w}_{\mathfrak{i}',\times}$ (for $i' > \mathfrak{i}'$). The second term cannot contain $\mathbf{w}_{\mathfrak{i}',\times}$ since by construction (see H_1 and how $\overline{\mathbf{u}}$'s are set in H_5), it is computed from $\mathbf{w}_{i,\times}, \overline{\mathbf{u}}_i$ with subscript $i > i' \geq \mathfrak{i}'$ (topological sorting implies $i > i'$) as well as the wiring of C and the wire values of $C(\mathbf{x})$.

The $\widetilde{\mathbf{t}}$ components with subscript "0", as explained in H_5, are computed from $\mathbf{w}_{i,x_i}$'s and $\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',x_{i'}}$'s (independent of $\mathbf{w}_{\mathfrak{i}',\times}$) and $\widetilde{\mathbf{t}}$ components with subscript "$\times$" (analyzed above, either received or sampled by reduction).

Therefore, $\mathsf{H}^{\mathfrak{i}'-1}_7 \approx \mathsf{H}^{\mathfrak{i}'}_7$ follows from $\mathsf{FlipLWE}_{N,m',q,\sigma'}$ (the minimum required m' varies by how gate $\mathfrak{i}'$ is connected in C; bounded by some fixed $\mathrm{poly}(N, |C|)$). This step also relies on the "straight-forwardness".

- In $\mathsf{H}^{|C|+1}_8$, we replace $\widetilde{\mathbf{t}}_{|C|,0}$ (actually a scalar) by random. Comparing $\mathsf{H}^{|C|}_7$ and $\mathsf{H}^{|C|+1}_8$, the only change is

$$\begin{array}{rlll} \widetilde{\mathbf{t}}^\intercal_{|C|,0} & \text{from} & \mathbf{w}^\intercal_{|C|,x_{|C|}}\mathbf{R}_{|C|} + \widetilde{\mathbf{w}}^\intercal_{|C|,x_{|C|}} - x_{|C|}\widetilde{\mathbf{t}}^\intercal_{|C|,\times} & \text{in } \mathsf{H}^{|C|}_7 \\ & \text{to} & \text{random} & \text{in } \mathsf{H}^{|C|+1}_8. \end{array}$$

  Note that the active expanded label is $\widetilde{\mathbf{w}}_{|C|,x_{|C|}} = \widetilde{w}_{|C|,0} + x_{|C|}\widetilde{w}_{|C|,\times}$. Since q is a prime, $x_{|C|} = C(\mathbf{x}) \neq 0$ (constraint for security), and $\widetilde{w}_{|C|,\times}$ is not used elsewhere in the revealed components (secret, input labels, garbled table except $\mathbf{t}_{|C|,0}$), it follows that $x_{|C|}\widetilde{w}_{|C|,\times}$ is a one-time pad for $\widetilde{\mathbf{t}}_{|C|,0}$, so $\mathsf{H}^{|C|}_7 \equiv \mathsf{H}^{|C|+1}_8$.

- In $\mathsf{H}^{\mathfrak{i}}_8$ (for $\mathfrak{i}$ from $|C|$ down to $(L+1)$), the components $\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',0}$ with $i \geq \mathfrak{i}$ (note that the indices being compared are $i, \mathfrak{i}$, not $i', \mathfrak{i}'$ as in $\mathsf{H}^{\mathfrak{i}'}_7$'s) are replaced by random.

Comparing $\mathsf{H}_8^{\mathsf{i}+1}$ and $\mathsf{H}_8^{\mathsf{i}}$, the only change is

$$(\widetilde{\mathbf{t}}^{(\mathsf{i},\gamma)}_{\mathsf{gin}[\mathsf{i},\gamma],0})^\intercal \quad \text{from} \quad \mathbf{w}^\intercal_{\mathsf{gin}[\mathsf{i},\gamma],x_{\mathsf{gin}[\mathsf{i},\gamma]}} \mathbf{R}^{(\mathsf{i},\gamma)}_{\mathsf{gin}[\mathsf{i},\gamma]} + (\widetilde{\mathbf{w}}^{(\mathsf{i},\gamma)}_{\mathsf{gin}[\mathsf{i},\gamma],x_{\mathsf{gin}[\mathsf{i},\gamma]}})^\intercal - x_{\mathsf{gin}[\mathsf{i},\gamma]} (\widetilde{\mathbf{t}}^{(\mathsf{i},\gamma)}_{\mathsf{gin}[\mathsf{i},\gamma],\times})^\intercal \quad \text{in } \mathsf{H}_8^{\mathsf{i}+1}$$

$$\text{to} \quad \text{random} \quad \text{in } \mathsf{H}_8^{\mathsf{i}}, \text{ where } \gamma \in \{1,2\}.$$

Observe that

$$(\widetilde{\mathbf{w}}^{(\mathsf{i},1)}_{\mathsf{gin}[\mathsf{i},1],x_{\mathsf{gin}[\mathsf{i},1]}}, \widetilde{\mathbf{w}}^{(\mathsf{i},2)}_{\mathsf{gin}[\mathsf{i},2],x_{\mathsf{gin}[\mathsf{i},2]}}) = \begin{cases} (\overline{\mathbf{u}}_{\mathsf{i}}, \mathbf{w}_{\mathsf{i},x_{\mathsf{i}}} - \overline{\mathbf{u}}_{\mathsf{i}}), & \text{if gate } i \text{ is addition;} \\ (\overline{\mathbf{u}}_{\mathsf{i}}, \mathbf{w}_{\mathsf{i},x_{\mathsf{i}}} - x_{\mathsf{gin}[\mathsf{i},2]}\overline{\mathbf{u}}_{\mathsf{i}}), & \text{if gate } i \text{ is multiplication;} \end{cases}$$

so $\overline{\mathbf{u}}_{\mathsf{i}}, \mathbf{w}_{\mathsf{i},x_{\mathsf{i}}}$ can serve as one-time pads if they do not appear elsewhere in the revealed components. We inspect their appearances.

- The secret is $w_{\text{out}} = \widetilde{w}_{\text{out},0} + e_{\text{out}}$, no appearance.
- The input labels are $\mathbf{w}_\ell = \mathbf{w}_{\ell,x_\ell} + \mathbf{e}_\ell$, no appearance (note $\mathsf{i} > L \geq \ell$).
- The garbled table is $\widetilde{\mathbf{t}}$ plus randomizing noises (without bars, no $\overline{\mathbf{u}}_{\mathsf{i}}, \mathbf{w}_{\mathsf{i},x_{\mathsf{i}}}$). The $\widetilde{\mathbf{t}}$ components with subscript "$\times$" are already replaced by random, so no appearance. The $\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',0}$ components require a case analysis.
  - For $i > \mathsf{i}$, they are already replaced by random, so no appearance.
  - For $i \leq \mathsf{i}$, we have

    $$(\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',0})^\intercal = \mathbf{w}^\intercal_{i',x_{i'}} \mathbf{R}^{(i,\gamma)}_{i'} + (\widetilde{\mathbf{w}}^{(i,\gamma)}_{i',x_{i'}})^\intercal - x_{i'} (\widetilde{\mathbf{t}}^{(i,\gamma)}_{i',\times})^\intercal.$$

    The first term has no appearance of $\overline{\mathbf{u}}_{\mathsf{i}}, \mathbf{w}_{\mathsf{i},x_{\mathsf{i}}}$, since $i' = \mathsf{gin}[i,\gamma] < i \leq \mathsf{i}$ by topological sorting. Neither does the third term. The second term either (if $i = \mathsf{i}$) is among the ones being one-time-padded, or (if $i < \mathsf{i}$) cannot contain $\overline{\mathbf{u}}_{\mathsf{i}}, \mathbf{w}_{\mathsf{i},x_{\mathsf{i}}}$, because by construction (see H_5), they are computed from only $\overline{\mathbf{u}}_i, \mathbf{w}_{i,x_i}$ (note $i < \mathsf{i}$), the wiring of C, and the wire values of $C(\mathbf{x})$.

Therefore, $\mathsf{H}_8^{\mathsf{i}+1} \equiv \mathsf{H}_8^{\mathsf{i}}$.

- In H_9, all the revealed components (secret, input labels, garbled table) are replaced by random. We inspect them in H_8^{L+1}.

- The secret is $w_{\mathrm{out}} = \widetilde{w}_{\mathrm{out},0} + e_{\mathrm{out}}$ and $\widetilde{w}_{\mathrm{out},0}$ serves as a one-time pad.
- The input labels are $\mathbf{w}_\ell = \mathbf{w}_{\ell,x_\ell} + \mathbf{e}_\ell$ and $\mathbf{w}_{\ell,x_\ell}$'s serve as one-time pads.
- The garbled table is $\widetilde{\mathbf{t}}$ plus randomizing noises. The $\widetilde{\mathbf{t}}$ components with subscript "$\times$" are independent and random thanks to $\mathsf{H}_7^{i'}$'s, and those with subscript "0", thanks to H_8^{i}'s — the output component is handled in $\mathsf{H}_8^{|C|+1}$ and the others must belong to some $\widetilde{\mathbf{t}}_{i',0}^{(i,\gamma)}$ (for some $L < i \le |C|$) hence indeed handled in H_8^{i}.

Therefore, $\mathsf{H}_8^{L+1} \equiv \mathsf{H}_9$.

Clearly, H_9 is just $\mathsf{Exp}^{\mathrm{random}}_{\mathsf{garble}}$. By hybrid argument, $\mathsf{Exp}^{\mathrm{real}}_{\mathsf{garble}} \equiv \mathsf{H}_0 \approx \mathsf{H}_9 \equiv \mathsf{Exp}^{\mathrm{random}}_{\mathsf{garble}}$. □

7.7 Ciphertext-Policy ABE from Short Noisy Linear Garbling

In this section, we show how to generically construct a CP-ABE scheme from a noisy linear garbling scheme and an identity-based evasive IPFE scheme.

Ingredients of Construction 7.5. We rely on

- a noisy linear garbling scheme NLG supporting $\mathcal{F} = \{\mathcal{F}_{\mathsf{param}'}\}_{\mathsf{param}' \in \mathsf{Params}'}$ that is $(B_{\mathrm{in}}, B_{\mathrm{out}})$-correct, B_{short}-short, and $\mathsf{GenNoise}$-secure such that the output of $\mathsf{GenNoise}$ is in $[-B_{\mathsf{NLG}}, B_{\mathsf{NLG}}]^\star$ (Definitions 7.4, 7.5, and 7.7), and
- an identity-based evasive IPFE scheme IPFE for $\mathcal{I} \supseteq [0, L]$ that is B_{IPFE}-correct and restricted-σ_{IPFE}-secure (Definitions 7.1 and 7.3).

Construction 7.5 (CP-ABE). Define (see Definition 2.3)

$$\mathsf{Params} = \left\{ \mathsf{param} = (\mathsf{param}', 1^L) \;\middle|\; \mathsf{param}' \in \mathsf{Params}', L \in \mathbb{N} \right\}, \qquad F_{\mathsf{param}',1^L} = \mathbb{Z}^L,$$
$$X_{\mathsf{param}',1^L} = \left\{ f : \mathbb{Z}^L \to \{0,1,\perp\} \;\middle|\; f \in \mathcal{F}_{\mathsf{param}'} \right\}, \qquad P_{\mathsf{param}',1^L}(\mathbf{x}, f) = f(\mathbf{x}).$$

Our CP-ABE for P works as follows.

- $\mathsf{Setup}(\mathsf{param}', 1^L)$ runs

$$\mathsf{pp} = (q, \dots) \xleftarrow{\$} \mathsf{NLG.Setup}(\mathsf{param}'),$$

deterministically picks a suitable n_{r} based on q, and sets $K = 0$ and $Z = 2n_{\mathrm{r}}$. It then sets up the IPFE scheme by

$$(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(q, 1^K, 1^Z).$$

The algorithm outputs $\mathsf{mpk} = (\mathsf{pp}, 1^{n_{\mathrm{r}}}, 1^K, 1^Z, \mathsf{impk})$ and $\mathsf{msk} = (\mathsf{mpk}, \mathsf{imsk})$.

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^{n_{\mathrm{r}}}$ and sets

$$\mathbf{v}_0 \leftarrow \begin{pmatrix} \mathbf{r} \\ 1 \\ \mathbf{0}_{n_{\mathrm{r}}-1} \end{pmatrix}, \qquad \mathbf{v}_\ell \leftarrow \begin{pmatrix} \mathbf{r} \\ \\ \mathbf{x}[\ell]\mathbf{r} \\ \\ \end{pmatrix} \quad \text{for all } \ell \in [L].$$

It generates IPFE secret keys

$$\begin{aligned} \mathsf{isk}_0 &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, \overbrace{0}^{\text{id}}, \mathbf{v}_0, \overbrace{\perp}^{\text{structured noises, unused}}), \\ \mathsf{isk}_\ell &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, \ \ell \ , \mathbf{v}_\ell, \ \perp \ ) \qquad \text{for all } \ell \in [L], \end{aligned}$$

and outputs $\mathsf{sk} = (\mathbf{r}, \mathsf{isk}_0, \mathsf{isk}_1, \ldots, \mathsf{isk}_L)$.

- $\mathsf{Enc}(\mathsf{mpk}, f, \mu)$ generates a noisy linear garbling of f by

$$(1^{n_f}, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell \in [L]}, \mathbf{T}, \mathbf{R}) \xleftarrow{\$} \mathsf{NLG.GenF}(\mathsf{pp}, f).$$

It samples a random secret matrix $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n_f \times n_{\mathrm{r}}}$ and a sufficiently large message encoding scalar $s_{\mathrm{msg}} \xleftarrow{\$} \mathbb{Z}_q \setminus [-2 \cdot 2^{-\kappa} q, 2 \cdot 2^{-\kappa} q]$. The algorithm computes

$$\begin{aligned} \mathbf{u}_{\mathrm{msg}}^\intercal &\leftarrow (\mathbf{w}_{\mathrm{out}}^\intercal \mathbf{S}, \mu s_{\mathrm{msg}}, \mathbf{0}_{1 \times (n_{\mathrm{r}}-1)}), & \mathbf{U}_{\mathrm{t}}^\intercal &\leftarrow (\mathbf{T}^\intercal \mathbf{S}, \mathbf{0}_{\star \times n_{\mathrm{r}}}), \\ \mathbf{U}_\ell^\intercal &\leftarrow (\mathbf{W}_{\ell,0}^\intercal \mathbf{S}, \mathbf{W}_{\ell,\times}^\intercal \mathbf{S}) \qquad \text{for all } \ell \in [L]. \end{aligned}$$

Lastly, it generates IPFE ciphertexts

$$\begin{aligned} \mathsf{ict}_{\mathrm{msg}} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 0, \mathfrak{d}, \mathbf{u}_{\mathrm{msg}}^\intercal), \qquad \mathsf{ict}_{\mathrm{t}} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 0, \mathfrak{d}, \mathbf{U}_{\mathrm{t}}^\intercal), \\ \mathsf{ict}_\ell &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, \ell, \mathfrak{d}, \mathbf{U}_\ell^\intercal) \qquad \text{for all } \ell \in [L], \end{aligned}$$

and outputs $\mathsf{ct} = (\mathbf{R}, \mathsf{ict}_{\mathrm{msg}}, \mathsf{ict}_{\mathrm{t}}, \mathsf{ict}_1, \ldots, \mathsf{ict}_L)$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}, \mathsf{sk}, f, \mathsf{ct})$ outputs $\perp$ if $f(\mathbf{x}) \neq 1$. Otherwise, it parses sk, ct as in KeyGen, Enc, and recomputes $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_L$ as in KeyGen. The algorithm decrypts all the IPFE ciphertexts for the rerandomized garbling instance, i.e.,

$$\begin{aligned} w_{\mathrm{msg}} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 0, \mathbf{v}_0, \perp, \mathsf{isk}_0, \mathfrak{o}, \mathsf{ict}_{\mathrm{msg}}), \\ \mathbf{t} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 0, \mathbf{v}_0, \perp, \mathsf{isk}_0, \mathfrak{o}, \mathsf{ict}_{\mathrm{t}}), \\ \mathbf{w}_\ell &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, \ell, \mathbf{v}_\ell, \perp, \mathsf{isk}_\ell, \mathfrak{o}, \mathsf{ict}_\ell) \qquad \text{for all } \ell \in [L]. \end{aligned}$$

  It evaluates the garbling by

$$w_{\mathrm{out}} \leftarrow \mathsf{NLG.Eval}(\mathsf{pp}, f, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\intercal\}_{\ell \in [L]}, \mathbf{t}^\intercal)$$

  and outputs the decryption result

$$\mu' \leftarrow \begin{cases} 0, & \text{if } w_{\mathrm{msg}} - w_{\mathrm{out}} \in [-2^{-\kappa}q, 2^{-\kappa}q]; \\ 1, & \text{otherwise.} \end{cases}$$

7.7.1 Correctness

Theorem 7.14 (¶). *Construction 7.5 is* correct *(Definition 2.3) if*

- *the modulus q output by* $\mathsf{NLG.Setup}(\mathsf{param}')$ *is always a prime and always satisfies* $B_{\mathrm{out}}(\mathsf{param}') \leq 2^{-\kappa}q$*, and*
- *it holds that* $B_{\mathsf{IPFE}}(q, 0, 2n_{\mathrm{r}}) \leq B_{\mathrm{in}}(\mathsf{param}')$.

Proof (Theorem 7.14). The correctness of Construction 7.5 follows directly from the correctness of the underlying IPFE and garbling schemes. Since q is a prime, by the B_{IPFE}-correctness of the evasive IPFE scheme,

$$\begin{aligned} w_{\mathrm{msg}} &= \mathbf{u}_{\mathrm{msg}}^\intercal \mathbf{v}_0 + e_{\mathrm{out}} = \mathbf{w}_{\mathrm{out}}^\intercal \mathbf{S}\mathbf{r} + \mu s_{\mathrm{msg}} + e_{\mathrm{out}} = (\mathbf{S}\mathbf{r})^\intercal \mathbf{w}_{\mathrm{out}} + e_{\mathrm{out}} + \mu s_{\mathrm{msg}}, \\ \mathbf{t}^\intercal &= \mathbf{v}_0^\intercal \mathbf{U}_{\mathrm{t}} + \mathbf{e}_{\mathrm{t}}^\intercal = (\mathbf{S}\mathbf{r})^\intercal \mathbf{T} + \mathbf{e}_{\mathrm{t}}^\intercal, \\ \mathbf{w}_\ell^\intercal &= \mathbf{v}_\ell^\intercal \mathbf{U}_\ell + \mathbf{e}_\ell^\intercal = (\mathbf{S}\mathbf{r})^\intercal (\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_\ell^\intercal, \end{aligned}$$

where $\mathbf{S}\mathbf{r} \in \mathbb{Z}_q^{n_f}$ randomizes the garbling and the errors (from IPFE decryption) are bounded by $B_{\mathsf{IPFE}}(q, 0, 2n_{\mathrm{r}}) \leq B_{\mathrm{in}}(\mathsf{param}')$. By the $(B_{\mathrm{in}}, B_{\mathrm{out}})$-correctness of garbling,

$$w_{\mathrm{out}} - ((\mathbf{S}\mathbf{r})^\intercal \mathbf{w}_{\mathrm{out}} + e_{\mathrm{out}}) \in [-B_{\mathrm{out}}(\mathsf{param}'), B_{\mathrm{out}}(\mathsf{param}')] \subseteq [-2^{-\kappa}q, 2^{-\kappa}q].$$

If $\mu = 0$, it always holds that $w_{\mathrm{msg}} - w_{\mathrm{out}} \in [-2^{-\kappa}q, 2^{-\kappa}q]$, which never holds when $\mu = 1$ by our choice of s_{msg}. We conclude that Construction 7.5 is correct. □

7.7.2 Security

Theorem 7.15 (¶). *Construction 7.5 is* very selectively secure *(Definition 2.3) if*[88]

- $\mathsf{LWE}_{n_{\mathrm{r}},\mathrm{poly}(\lambda),q,\sigma_{\mathrm{r}}}$ *(Assumption 2.2) holds for some* $\sigma_{\mathrm{r}} \leq \frac{2^{-\kappa-6}\sigma_{\mathsf{IPFE}}}{\sqrt{\kappa}\cdot B_{\mathsf{short}}(\mathsf{param}')}$*, and*
- *it holds that* $2^{\kappa+6}B_{\mathsf{NLG}}(\mathsf{param}') \leq \sigma_{\mathsf{IPFE}}$.

Proof (Theorem 7.15). Let $\mathcal{A}$ be an efficient adversary. We assume that it always chooses challenges such that $f(\mathbf{x}_j) = 0$ for all $j \in [J]$ — otherwise, we alter $\mathcal{A}$ so that when the condition fails, it resets J to zero and aborts when it receives back the ABE components (incomplete per its expectation), as its output is irrelevant to $\mathsf{Exp}^{\beta}_{\mathrm{ABE}}$ in that case. Consider the following evasive IPFE sampler $\mathcal{S}^{\beta} = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}^{\beta}_{\mathrm{U}})$ per Definition 7.2.

- $\mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}})$ parses $r_{\mathrm{pub}} = (r_{\mathcal{A}}, r_{\mathsf{NLG.Setup}}, r_{\mathrm{r}}, r_{\mathsf{NLG.GenF}})$. It runs $\mathcal{A}(r_{\mathcal{A}})$ to obtain

$$\mathsf{param} = (\mathsf{param}', 1^L), \qquad \{\mathbf{x}_j\}_{j\in[J]} \quad (\mathbf{x}_j \in \mathbb{Z}^L), \qquad f : \mathbb{Z}^L \to \{0, 1, \bot\}.$$

  The algorithm next sets up the noisy linear garbling by

$$\mathsf{pp} = (q, \dots) \leftarrow \mathsf{NLG.Setup}(\mathsf{param}'; r_{\mathsf{NLG.Setup}}),$$

  and deterministically picks n_{r}, K, Z as in Setup of Construction 7.5. It then uses r_{r} to sample $\mathbf{r}_1, \dots, \mathbf{r}_J \in \mathbb{Z}_q^{n_{\mathrm{r}}}$ uniformly at random in a straight-forward way.[89] The algorithm sets for all $j \in [J]$,

$$\mathbf{v}_{j,0} = \begin{pmatrix} \mathbf{r}_j \\ 1 \\ \mathbf{0}_{n_{\mathrm{r}}-1} \end{pmatrix}, \qquad \mathbf{v}_{j,\ell} = \begin{pmatrix} \mathbf{r}_j \\ \\ \mathbf{x}_j[\ell]\mathbf{r}_j \end{pmatrix} \quad (\text{for all } \ell \in [L]).$$

[88] The constants have been changed from the conference version [HLL24a]. Both versions are correct. The new version fits the proof better.

[89] Precisely speaking, r_{r} must be efficiently sampleable conditioned on and given $\mathbf{r}_1, \dots, \mathbf{r}_J$.

It runs

$$(1^{n_f}, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}) \leftarrow \mathsf{NLG.GenF}(\mathsf{pp}, f; r_{\mathsf{NLG.GenF}}).$$

Suppose $\mathbf{W}_{\ell,0} \in \mathbb{Z}_q^{n_f \times m_\ell}$ and $\mathbf{T} \in \mathbb{Z}_q^{n_f \times m_{\mathrm{t}}}$, the algorithm outputs

$$q, 1^K, 1^Z, \quad \mathsf{ID} = [0, L], \quad 1^{J_{\mathsf{id}}} = 1^J, \quad 1^{I_{0,\mathfrak{o}}} = 1^{m_{\mathrm{t}}+1}, \quad 1^{I_{\ell,\mathfrak{o}}} = 1^{m_\ell},$$
$$1^{I_{\mathsf{id},\mathfrak{g}}} = 1^0, \quad \sigma_{\mathrm{pre}} = \sigma_{\mathsf{IPFE}}, \quad \mathbf{V}_{\mathsf{id},\mathfrak{o}} = (\mathbf{v}_{1,\mathsf{id}}, \ldots, \mathbf{v}_{J,\mathsf{id}}), \quad \mathbf{V}_{\mathsf{id},\mathfrak{g}} = \bot,$$

where the indices are $\mathsf{id} \in \mathsf{ID}$ and $\ell \in [L]$.

- $\mathcal{S}_{\mathrm{U}}^{\beta}(r_{\mathrm{pub}}; r_{\mathrm{priv}})$ first runs the deterministic subprocedure $\mathcal{S}_{\mathrm{A0}}^{\beta}(r_{\mathrm{pub}})$, which does the following. It first reruns $\mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}})$ to obtain $\mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}$. The algorithm then rearranges them into matrices $\mathbf{A}_{\mathrm{msg}}, \{\mathbf{A}_{\mathrm{t},i}\}_{i\in[m_{\mathrm{t}}]}, \{\mathbf{A}_{\ell,i}\}_{\ell\in[L],i\in[m_\ell]}$ so that for all $\widetilde{\mathbf{s}}_1, \ldots, \widetilde{\mathbf{s}}_{n_{\mathrm{r}}} \in \mathbb{Z}_q^{n_f}$ and $s_{\mathrm{msg}} \in \mathbb{Z}_q$,

$$\mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathrm{msg}} = \mathbf{u}_{\mathrm{msg}}^{\mathsf{T}} = (\mathbf{w}_{\mathrm{out}}^{\mathsf{T}}\mathbf{S}, \beta s_{\mathrm{msg}}, \mathbf{0}_{1\times(n_{\mathrm{r}}-1)}),$$
$$(\mathbf{A}_{\mathrm{t},1}^{\mathsf{T}}\mathbf{d}_0, \ldots, \mathbf{A}_{\mathrm{t},m_{\mathrm{t}}}^{\mathsf{T}}\mathbf{d}_0)^{\mathsf{T}} = \mathbf{U}_{\mathrm{t}}^{\mathsf{T}} = (\mathbf{T}^{\mathsf{T}}\mathbf{S}, \mathbf{0}_{m_{\mathrm{t}}\times n_{\mathrm{r}}}),$$
$$(\mathbf{A}_{\ell,1}^{\mathsf{T}}\mathbf{d}_0, \ldots, \mathbf{A}_{\ell,m_\ell}^{\mathsf{T}}\mathbf{d}_0)^{\mathsf{T}} = \mathbf{U}_{\ell}^{\mathsf{T}} = (\mathbf{W}_{\ell,0}^{\mathsf{T}}\mathbf{S}, \mathbf{W}_{\ell,\times}^{\mathsf{T}}\mathbf{S}) \qquad (\text{for all } \ell \in [L]),$$

where $\mathbf{d}_0^{\mathsf{T}} = (s_{\mathrm{msg}}, \widetilde{\mathbf{s}}_1^{\mathsf{T}}, \ldots, \widetilde{\mathbf{s}}_{n_{\mathrm{r}}}^{\mathsf{T}})$ and $\mathbf{S} = (\widetilde{\mathbf{s}}_1, \ldots, \widetilde{\mathbf{s}}_{n_{\mathrm{r}}})$. This is possible since every entry on the right-hand side is linear in $\mathbf{d}_0$. The algorithm $\mathcal{S}_{\mathrm{A0}}$ outputs

$$1^{n'} = 1^{n_f n_{\mathrm{r}}+1}, \qquad \mathbf{A}_{0,\mathfrak{o},0,i} = \mathbf{A}_{\mathrm{t},i} \qquad (\text{for all } i \in [m_{\mathrm{t}}]),$$
$$\mathbf{A}_{0,\mathfrak{o},0,m_{\mathrm{t}}+1} = \mathbf{A}_{\mathrm{msg}}, \qquad \mathbf{A}_{\ell,\mathfrak{o},0,i} = \mathbf{A}_{\ell,i} \qquad (\text{for all } \ell \in [L],\ i \in [m_\ell]).$$

Completing $\mathcal{S}_{\mathrm{A0}}$, the algorithm $\mathcal{S}_{\mathrm{U}}$ samples $\mathbf{d}_0 \xleftarrow{\$} \mathbb{Z}_q^{n'}$ using r_{priv} and outputs

$$\mathbf{U}_{\mathsf{id},\mathfrak{o}}^{\mathsf{T}} = \begin{pmatrix} \mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathsf{id},\mathfrak{o},0,1} \\ \vdots \\ \mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathsf{id},\mathfrak{o},0,I_{\mathsf{id},\mathfrak{o}}} \end{pmatrix} \qquad \text{for all } \mathsf{id} \in \mathsf{ID}.$$

By definition, $\mathcal{S}^{\beta}$ is a restricted sampler (Definition 7.3). We have the following:

Claim 7.16 (¶). $\mathcal{S}^{\beta}$ has pseudorandom noisy inner products, i.e., $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}^{\beta}}$ (Definition 7.2) holds for both $\beta \in \{0, 1\}$.

The very selective security of Construction 7.5 follows from Claim 7.16. Consider the following hybrids.

- H_0^β is $\mathsf{Exp}_{\mathrm{ABE}}^\beta$ for Construction 7.5. Note that s_{msg} in the challenge ciphertext is uniformly random over $\mathbb{Z}_q \setminus [-2 \cdot 2^{-\kappa} q, 2 \cdot 2^{-\kappa} q]$.
- In H_1^β, the scalar s_{msg} is changed to be uniformly random over $\mathbb{Z}_q$. Clearly, $\mathsf{H}_0^\beta \approx_{\mathrm{s}} \mathsf{H}_1^\beta$. Moreover, H_1^β corresponds to the left distribution of $\mathsf{IPFEsec}_{\mathrm{post}}^{\mathcal{S}^\beta}$.
- In H_2^β, the vectors inside IPFE ciphertexts (components of the ABE challenge ciphertext) are replaced by random, which corresponds to the right distribution of $\mathsf{IPFEsec}_{\mathrm{post}}^{\mathcal{S}^\beta}$. By Claim 7.16 and the restricted-σ_{IPFE}-security of IPFE, we have $\mathsf{H}_1^\beta \approx \mathsf{H}_2^\beta$.

Lastly, $\mathsf{H}_2^0 \equiv \mathsf{H}_2^1$. By hybrid argument, we conclude $\mathsf{Exp}_{\mathrm{ABE}}^0 \approx \mathsf{Exp}_{\mathrm{ABE}}^1$, completing the proof. □

Proof (Claim 7.16). Fix $\beta \in \{0,1\}$ and consider the following hybrids.

- H_0 is the left distribution of $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}^\beta}$. Recall that the public randomness is $r_{\mathrm{pub}} = (r_{\mathcal{A}}, r_{\mathsf{NLG.Setup}}, r_{\mathrm{r}}, r_{\mathsf{NLG.GenF}})$. Adopting the notations of the ABE decryption algorithm, the noisy inner products are

$$\begin{aligned} w_{\mathrm{msg},j} &= \mathbf{u}_{\mathrm{msg}}^\intercal \mathbf{v}_{j,0} + e_{\mathrm{out},j} = (\mathbf{S}\mathbf{r}_j)^\intercal \mathbf{w}_{\mathrm{out}} + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \\ \mathbf{t}_j^\intercal &= \mathbf{v}_{j,0}^\intercal \mathbf{U}_{\mathrm{t}} + \mathbf{e}_{\mathrm{t},j}^\intercal = (\mathbf{S}\mathbf{r}_j)^\intercal \mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal, \\ \mathbf{w}_{j,\ell}^\intercal &= \mathbf{v}_{j,\ell}^\intercal \mathbf{U}_\ell + \mathbf{e}_{j,\ell}^\intercal = (\mathbf{S}\mathbf{r}_j)^\intercal (\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^\intercal \qquad (\text{for all } \ell \in [L]), \end{aligned}$$

  where $j \in [J]$ and the entries of $\mathbf{e}$'s are independent $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{IPFE}}}$.
- In H_1, additional small noises are attached to $\{\mathbf{S}\mathbf{r}_j\}_{j\in[J]}$. For all $j \in [J]$, sample $\mathbf{e}_{\mathrm{r},j} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{r}},\le\sigma_{\mathrm{r}}\sqrt{\kappa}}^{n_f}$ and set the inner products to

$$\begin{aligned} w_{\mathrm{msg},j} &= (\mathbf{S}\mathbf{r}_j + \mathbf{e}_{\mathrm{r},j})^\intercal \mathbf{w}_{\mathrm{out}} + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \qquad \mathbf{t}_j^\intercal = (\mathbf{S}\mathbf{r}_j + \mathbf{e}_{\mathrm{r},j})^\intercal \mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal, \\ \mathbf{w}_{j,\ell}^\intercal &= (\mathbf{S}\mathbf{r}_j + \mathbf{e}_{\mathrm{r},j})^\intercal (\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^\intercal \qquad (\text{for all } \ell \in [L]). \end{aligned}$$

  The errors introduced into the inner products are bounded by

$$B_{\mathrm{short}}(\mathsf{param}') \cdot \sigma_{\mathrm{r}}\sqrt{\kappa} \le 2^{-\kappa-6}\sigma_{\mathsf{IPFE}},$$

so they are flooded (Lemma 2.2) by $\{e_{\mathrm{out},j}, \mathbf{e}_{\mathrm{t},j}, \{\mathbf{e}_{j,\ell}\}_{\ell\in[L]}\}_{j\in[J]}$, and $\mathsf{H}_0 \approx_{\mathrm{s}} \mathsf{H}_1$.

- In H_2, the $\mathbf{e}_{\mathrm{r},j}$'s are no longer truncated, i.e., $\mathbf{e}_{\mathrm{r},j} \xleftarrow{\$} \mathcal{D}^{n_f}_{\mathbb{Z},\sigma_{\mathrm{r}}}$. We have $\mathsf{H}_1 \approx_{\mathrm{s}} \mathsf{H}_2$ by Lemma 2.1.

- In H_3, the garblings use uniform randomness. For all $j \in [J]$, sample $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_q^{n_f}$ and set the inner products to

$$w_{\mathrm{msg},j} = \mathbf{s}_j^{\mathsf{T}}\mathbf{w}_{\mathrm{out}} + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \qquad \mathbf{t}_j^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^{\mathsf{T}},$$
$$\mathbf{w}_{j,\ell}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^{\mathsf{T}} \qquad (\text{for all } \ell \in [L]).$$

  By n_f hybrids of applying $\mathsf{LWE}_{n_{\mathrm{r}},J,q,\sigma_{\mathrm{r}}}$ over the rows of $\mathbf{S}$ with public matrix $(\mathbf{r}_1, \ldots, \mathbf{r}_J)$, we have

$$\mathbf{S}(\mathbf{r}_1, \ldots, \mathbf{r}_J) + (\mathbf{e}_{\mathrm{r},1}, \ldots, \mathbf{e}_{\mathrm{r},J}) \approx (\mathbf{s}_1, \ldots, \mathbf{s}_J),$$

  which implies $\mathsf{H}_2 \approx \mathsf{H}_3$. In this step, we rely on the "straight-forwardness" of sampling $\mathbf{r}_j$'s from r_{r} in $\mathcal{S}^\beta$.

- In H_4, additional noises generated by GenNoise are attached to the garblings. For all $j \in [J]$, run

$$(\overline{e}_{\mathrm{out},j}, \{\overline{\mathbf{e}}_{j,\ell}\}_{\ell\in[L]}, \overline{\mathbf{e}}_{\mathrm{t},j}) \xleftarrow{\$} \mathsf{GenNoise}(\mathsf{param}', \mathsf{pp}, f, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}\mathbf{T}, \mathbf{R}, \mathbf{x}_j),$$

  and set the inner products to

$$w_{\mathrm{msg},j} = \mathbf{s}_j^{\mathsf{T}}\mathbf{w}_{\mathrm{out}} + \overline{e}_{\mathrm{out},j} + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \qquad \mathbf{t}_j^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}\mathbf{T} + \overline{\mathbf{e}}_{\mathrm{t},j}^{\mathsf{T}} + \mathbf{e}_{\mathrm{t},j}^{\mathsf{T}},$$
$$\mathbf{w}_{j,\ell}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \overline{\mathbf{e}}_{j,\ell}^{\mathsf{T}} + \mathbf{e}_{j,\ell}^{\mathsf{T}} \qquad (\text{for all } \ell \in [L]).$$

  The $\overline{e}, \overline{\mathbf{e}}$'s introduced into the inner products are bounded by $B_{\mathsf{NLG}} \le 2^{-\kappa-6}\sigma_{\mathsf{IPFE}}$, so they are flooded (Lemma 2.2) by $e, \mathbf{e}$'s. We have $\mathsf{H}_3 \approx_{\mathrm{s}} \mathsf{H}_4$.

- In H_5, all the inner products are replaced by random, i.e., the right distribution of $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{pre}}$. Note that the inner products in H_4 are well-randomized garblings (by $\mathbf{s}, \overline{e}, \overline{\mathbf{e}}$'s) plus independent noises ($e, \mathbf{e}$'s). Therefore, by J hybrids of applying GenNoise-security of NLG, we conclude $\mathsf{H}_4 \approx \mathsf{H}_5$.

By hybrid argument, $\mathsf{H}_0 \approx \mathsf{H}_5$, i.e., $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{pre}}$ holds. □

7.7.3 Unbounded Attribute and Summary

Unbounded Attribute. Construction 7.5 can be modified for attributes of unbounded length. Before presenting the scheme, we shall clarify the predicate family of such a scheme, in particular, what $P(\mathbf{x}, f)$ is when $\mathbf{x}$ is shorter than an input to f. There are multiple approaches in the literature. The following is the discussion in Section 8.5.3. For monotone functions, the missing attribute bits can be assumed to be zero. For general functions: the specification is silent in many works, which should be interpreted as $P(\mathbf{x}, f) = \perp$ (neither correct nor secure); it might require exact length matching, or only reject $\mathbf{x}$ shorter than input to f (prefix-matching), or attach index sets to $\mathbf{x}, f$ and require equal-match or subset-match.

The "silent" specification is straight-forward to achieve. We explain how to achieve prefix-matching. For simplicity, we require that $f(\mathbf{x}) \neq \perp$ for all $\mathbf{x} \in \{0,1\}^L$. The predicate family is (see Definition 2.3)

$$\mathsf{Params} = \left\{ \mathsf{param} = \mathsf{param}' \,\middle|\, \mathsf{param}' \in \mathsf{Params}' \right\},$$

$$F_{\mathsf{param}'} = \mathbb{Z}^{<2^\lambda}, \qquad X_{\mathsf{param}'} = \left\{ f : \mathbb{Z}^L \to \{0,1,\perp\} \,\middle|\, f \in \mathcal{F}_{\mathsf{param}'} \right\},$$

$$P_{\mathsf{param}'}(\mathbf{x}, f) = \begin{cases} f(\mathbf{x}'), & \text{if the input length of } f \text{ is } L \text{ and } \mathbf{x}' \text{ is the } L\text{-prefix of } \mathbf{x}; \\ 0, & \text{otherwise.} \end{cases}$$

Construction 7.5 with L (textually) removed is already quite close to an unbounded ABE — it becomes one with the following modifications similar to Section 8.5.3.

- The identity space satisfies $\mathcal{I} \supseteq \{0, 1, \ldots, \lambda, \lambda+1, \ldots, \lambda + 2^\lambda - 1\}$.
- When generating a key for $\mathbf{x}$, instead generate a key for $(L_{\mathrm{x}}, \mathbf{x})$, where L_{x} is the λ-bit encoding of $\mathbf{x}$.
- When generating a ciphertext for f, instead generate a ciphertext for

$$f'(L_{\mathrm{x}}, \mathbf{x}') = \begin{cases} 0, & \text{if } L_{\mathrm{x}} \text{ is less than the input length of } f; \\ f(\mathbf{x}'), & \text{otherwise.} \end{cases}$$

- The security reduces to that of Construction 7.5 by appropriately appending zeros to every key attribute.

Summary. By instantiating our CP-ABE with the garbling scheme of Construction 7.4 and the IPFE scheme of Construction 7.2, we obtain CP-ABE for bounded-arithmetic circuits.

Corollary 7.17. *Under LWE (Assumption 2.2) and evasive LWE (Assumption 7.2), there exist CP-ABE schemes (bounded/unbounded in attribute length) for bounded-arithmetic circuits with* $|\mathsf{mpk}| = \mathrm{poly}(\lambda, \log M, d)$ *and*

$$|\mathsf{sk}_{\mathbf{x}}| = (|\mathbf{x}| + 1)\,\mathrm{poly}(\lambda, \log M, d), \qquad |\mathsf{ct}_C| = (|C| + 1)\,\mathrm{poly}(\lambda, \log M, d),$$

where M *is the maximum wire value and* d *is the maximum depth.*

7.8 ABE for DFA

We define DFA in a minimal format convenient for garbling. The notable difference from Definition 4.16 is that Definition 7.8 uses *rejection states* instead of accepting states, because it is more convenient for noisy linear garbling (Definition 7.4) to authorize recovery of message when the computation yields zero.

Definition 7.8 (DFA). A *deterministic finite automaton* Γ over the binary alphabet is a tuple $(1^{\mathfrak{Q}}, \mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \boldsymbol{\xi})$, where $\mathfrak{Q} \in \mathbb{N}$ is its *number of states*, $\mathbf{\Gamma}_0, \mathbf{\Gamma}_1 \in \{0,1\}^{\mathfrak{Q}\times\mathfrak{Q}}$ are its *transition matrices*, $\boldsymbol{\xi} \in \{0,1\}^{\mathfrak{Q}}$ is its *rejection state vector*, and each column of $\mathbf{\Gamma}_0, \mathbf{\Gamma}_1$ contains exactly one 1. The DFA *accepts* an input $\mathbf{x} \in \{0,1\}^L$ if and only if

$$\boldsymbol{\xi}^{\intercal} \cdot \prod_{\ell=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell]} \cdot \boldsymbol{\iota}_1 = 0.$$

Implicitly, for Γ, the set of states is $[\mathfrak{Q}]$, the initial state is 1, and the set of accept states is $\{\, \mathfrak{q} \in [\mathfrak{Q}] \mid \boldsymbol{\xi}[\mathfrak{q}] = 0 \,\}$. After reading $x \in \{0,1\}$, the machine transitions from $\mathfrak{q}$ to $\mathfrak{q}'$ if and only if $\mathbf{\Gamma}_x[\mathfrak{q}', \mathfrak{q}] = 1$, which is equivalent to $\mathbf{\Gamma}_x \boldsymbol{\iota}_{\mathfrak{q}} = \boldsymbol{\iota}_{\mathfrak{q}'}$ (recall that $\boldsymbol{\iota}_{\mathfrak{q}} \in \{0,1\}^{\mathfrak{Q}}$ is the $\mathfrak{q}^{\text{th}}$ standard basis vector). The following lemma, readily verified, will be handy:

Lemma 7.18. *For all DFA* $(1^{\mathfrak{Q}}, \mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \boldsymbol{\xi})$ *and input* $\mathbf{x} \in \{0,1\}^L$*, it holds that*

$$\left\| \prod_{\ell=1}^{L} \mathbf{\Gamma}_{\mathbf{x}[\ell]}^{\intercal} \right\| \le 1, \qquad \boldsymbol{\xi}^{\intercal} \cdot \prod_{\ell=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell]} \cdot \boldsymbol{\iota}_1 \in \{0,1\}.$$

7.8.1 Noisy Linear Garbling for DFA

DFA is defined for inputs of arbitrary length, yet our notion of garbling is only for functions with fixed-length input. Following the paradigm in Section 4.6, we consider the garbling scheme for each possible input length. The existing (see [Wat12] or Construction 4.8) linear secret sharing scheme for DFA can be cast into a noisy linear garbling (Definition 7.4).

Construction 7.6 (DFA garbling, adapted from Construction 4.8). Let[90]

$$\begin{aligned}
\mathsf{Params} &= \left\{ (q, \overline{L}) \,\middle|\, q, \overline{L} \in \mathbb{N},\ q \geq 2 \right\}, \\
\mathcal{F}_{q,\overline{L}} &= \left\{ f_{\Gamma,L} : \mathbb{Z}^L \to \{0,1,\bot\} \,\middle|\, \Gamma \text{ is a DFA and } L \in [0,\overline{L}] \right\}, \\
f_{\Gamma,L}(\mathbf{x}) &= \begin{cases} 1, & \text{if } \mathbf{x} \in \{0,1\}^L \text{ and } \Gamma \text{ accepts } \mathbf{x}; \\ 0, & \text{if } \mathbf{x} \in \{0,1\}^L \text{ and } \Gamma \text{ rejects } \mathbf{x}; \\ \bot, & \text{if } \mathbf{x} \notin \{0,1\}^L. \end{cases}
\end{aligned}$$

The function $f_{\Gamma,L}$ is represented by $(\Gamma, 1^L)$. The noisy linear garbling scheme works as follows.

- $\mathsf{Setup}(\mathsf{param})$ outputs $\mathsf{pp} = \mathsf{param} = (q, \overline{L})$.
- $\mathsf{GenF}(\mathsf{pp}, f_{\Gamma,L})$ parses $\Gamma = (1^{\mathfrak{Q}}, \boldsymbol{\Gamma}_0, \boldsymbol{\Gamma}_1, \boldsymbol{\xi})$, sets

$$n_{\Gamma,L} = (L+1)\mathfrak{Q} + 1, \qquad \mathbf{w}_{\mathsf{out}}^\intercal = (-\boldsymbol{\iota}_1^\intercal, \mathbf{0}_{1\times L\mathfrak{Q}}, 0), \qquad \mathbf{R} = \bot,$$

$$\mathbf{W}_{\ell,0} = \begin{pmatrix} \mathbf{0}_{(\ell-1)\mathfrak{Q}\times\mathfrak{Q}} \\ -\mathbf{I}_{\mathfrak{Q}} \\ \boldsymbol{\Gamma}_0 \\ \mathbf{0}_{(L-\ell)\mathfrak{Q}\times\mathfrak{Q}} \\ \mathbf{0}_{1\times\mathfrak{Q}} \end{pmatrix}, \qquad \mathbf{W}_{\ell,\times} = \begin{pmatrix} \mathbf{0}_{(\ell-1)\mathfrak{Q}\times\mathfrak{Q}} \\ \mathbf{0}_{\mathfrak{Q}\times\mathfrak{Q}} \\ \boldsymbol{\Gamma}_1 - \boldsymbol{\Gamma}_0 \\ \mathbf{0}_{(L-\ell)\mathfrak{Q}\times\mathfrak{Q}} \\ \mathbf{0}_{1\times\mathfrak{Q}} \end{pmatrix}, \qquad \mathbf{T} = \begin{pmatrix} \mathbf{0}_{L\mathfrak{Q}\times\mathfrak{Q}} \\ -\mathbf{I}_{\mathfrak{Q}} \\ \boldsymbol{\xi}^\intercal \end{pmatrix},$$

  and outputs $(1^{n_{\Gamma,L}}, \mathbf{w}_{\mathsf{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R})$. Here, $\boldsymbol{\iota}_1 \in \{0,1\}^{\mathfrak{Q}}$.

[90] Here, the DFA input length upper bound $\overline{L}$ is needed due to perfect correctness requirement in the presence of noise accumulation. For our garbling and ABE constructions, we can simply set $\overline{L} = 2^\lambda$ to support arbitrary polynomial-size computations.

- $\mathsf{Eval}(\mathsf{pp}, f_{\Gamma,L}, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\intercal\}_{\ell\in[L]}, \mathbf{t}^\intercal)$ computes and outputs

$$\mathbf{t}^\intercal \cdot \prod_{\ell'=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 + \sum_{\ell=L}^{1} \left(\mathbf{w}_\ell^\intercal \cdot \prod_{\ell'=\ell-1}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \right).$$

Theorem 7.19 (¶). *Construction 7.6 is* $(B_{\mathrm{in}}, B_{\mathrm{out}})$-correct *(Definition 7.4) if*

$$B_{\mathrm{out}} \geq (\overline{L} + 2) B_{\mathrm{in}}.$$

It is 2-short *(Definition 7.5) and* GenNoise-secure *(Definition 7.7) for all* GenNoise.

Proof (Theorem 7.19). The proof is basically that for Construction 4.4 plus handling the noise. It suffices to consider $\mathbf{x} \in \{0,1\}^L$. Note that (the second by Lemma 7.18)

$$\begin{aligned}
\mathbf{w}_{\mathrm{out}}^\intercal = (-\boldsymbol{\iota}_1^\intercal, \mathbf{0}) &\implies \|\mathbf{w}_{\mathrm{out}}^\intercal\| \leq 1 \qquad \leq 2, \\
\mathbf{W}_{\ell,0}^\intercal + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}^\intercal = (\mathbf{0}, -\mathbf{I}_{\mathfrak{Q}}, \mathbf{\Gamma}_{\mathbf{x}[\ell]}^\intercal, \mathbf{0}) &\implies \|\mathbf{W}_{\ell,0}^\intercal + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}^\intercal\| \leq 1 + 1 \leq 2, \\
\mathbf{T}^\intercal = (\mathbf{0}, -\mathbf{I}_{\mathfrak{Q}}, \boldsymbol{\xi}) &\implies \|\mathbf{T}^\intercal\| \leq 1 + 1 \leq 2,
\end{aligned}$$

so the scheme is 2-short.

Let the garbling randomness/noise be

$$\mathbf{s} = (\mathbf{s}_0^\intercal, \mathbf{s}_1^\intercal, \ldots, \mathbf{s}_L^\intercal, s_{L+1})^\intercal, \qquad \mathbf{e} = (e_0, \mathbf{e}_1^\intercal, \ldots, \mathbf{e}_L^\intercal, \mathbf{e}_{L+1}^\intercal)^\intercal,$$

where each $\mathbf{s}_\ell, \mathbf{e}_\ell$ is $\mathfrak{Q}$-dimensional, then

$$w_{\mathrm{out}} = -\mathbf{s}_0^\intercal \boldsymbol{\iota}_1 + e_0, \qquad \mathbf{w}_\ell^\intercal = \mathbf{s}_\ell^\intercal \mathbf{\Gamma}_{\mathbf{x}[\ell]} - \mathbf{s}_{\ell-1}^\intercal + \mathbf{e}_\ell^\intercal, \qquad \mathbf{t}^\intercal = s_{L+1}\boldsymbol{\xi}^\intercal - \mathbf{s}_L^\intercal + \mathbf{e}_{L+1}^\intercal.$$

Since evaluation is linear in $\mathbf{w}$'s and $\mathbf{t}$, we first consider the non-noisy part,

$$\begin{aligned}
&(\mathbf{t} - \mathbf{e}_{L+1})^\intercal \cdot \prod_{\ell'=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 + \sum_{\ell=L}^{1} \left((\mathbf{w}_\ell - \mathbf{e}_\ell)^\intercal \cdot \prod_{\ell'=\ell-1}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \right) \\
&= (s_{L+1}\boldsymbol{\xi}^\intercal - \mathbf{s}_L^\intercal) \cdot \prod_{\ell'=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 + \sum_{\ell=L}^{1} \left((\mathbf{s}_\ell^\intercal \mathbf{\Gamma}_{\mathbf{x}[\ell]} - \mathbf{s}_{\ell-1}^\intercal) \cdot \prod_{\ell'=\ell-1}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \right) \\
&= s_{L+1}\boldsymbol{\xi}^\intercal \cdot \prod_{\ell'=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 - \mathbf{s}_L^\intercal \cdot \prod_{\ell'=L}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \\
&\quad \text{(telescoping)} \quad + \sum_{\ell=L}^{1} \left(\mathbf{s}_\ell^\intercal \cdot \prod_{\ell'=\ell}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 - \mathbf{s}_{\ell-1}^\intercal \cdot \prod_{\ell'=\ell-1}^{1} \mathbf{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \right)
\end{aligned}$$

$$= s_{L+1}\boldsymbol{\xi}^\mathsf{T} \cdot \prod_{\ell'=L}^{1} \boldsymbol{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \; - \mathbf{s}_0^\mathsf{T}\boldsymbol{\iota}_1 = (w_{\mathsf{out}} - e_0) + \begin{cases} 0, & \text{if } \Gamma \text{ accepts } \mathbf{x}; \\ s_{L+1}, & \text{otherwise.} \end{cases}$$

Writing

$$e_{\mathsf{Eval}} = \mathbf{e}_{L+1}^\mathsf{T} \cdot \prod_{\ell'=L}^{1} \boldsymbol{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \; + \sum_{\ell=L}^{1} \left(\mathbf{e}_\ell^\mathsf{T} \cdot \prod_{\ell'=\ell-1}^{1} \boldsymbol{\Gamma}_{\mathbf{x}[\ell']} \cdot \boldsymbol{\iota}_1 \right) - e_0,$$

we have

$$\mathsf{Eval}(\mathsf{pp}, f_{\Gamma,L}, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\mathsf{T}\}_{\ell\in[L]}, \mathbf{t}^\mathsf{T}) = w_{\mathsf{out}} + e_{\mathsf{Eval}} + \begin{cases} 0, & \text{if } \Gamma \text{ accepts } \mathbf{x}; \\ s_{L+1}, & \text{otherwise.} \end{cases}$$

Transposing e_{Eval} and applying Lemma 7.18, we find

$$|e_{\mathsf{Eval}}| \le \|\boldsymbol{\iota}_1^\mathsf{T}\| \cdot \underbrace{\prod_{\ell'=1}^{L} \|\boldsymbol{\Gamma}_{\mathbf{x}[\ell']}^\mathsf{T}\|}_{\le 1} \cdot \|\mathbf{e}_{L+1}\| \; + \sum_{\ell=L}^{1} \left(\|\boldsymbol{\iota}_1^\mathsf{T}\| \cdot \underbrace{\prod_{\ell'=1}^{\ell-1} \|\boldsymbol{\Gamma}_{\mathbf{x}[\ell']}^\mathsf{T}\|}_{\le 1} \cdot \|\mathbf{e}_\ell\| \right) + |e_0|$$

$$\le (L+2)B_{\mathsf{in}} \le (\overline{L}+2)B_{\mathsf{in}} \le B_{\mathsf{out}}.$$

This proves $(B_{\mathsf{in}}, B_{\mathsf{out}})$-correctness.

Perfect security without noise is Theorem 4.18, which implies security with any noise. To recap, observe that with param, pp, $f_{\Gamma,L}$, output of GenF, and $\mathbf{x} \in \{0,1\}^L$ fixed, the values $(\mathbf{w}_1, \ldots, \mathbf{w}_L, s_{L+1})$ are jointly uniformly random — thanks to the subtraction of $\mathbf{s}_\ell$ in each $\mathbf{w}_\ell$. When $\mathbf{x}$ is rejected by Γ,

$$w_{\mathsf{out}} = \underbrace{\mathsf{Eval}(\mathsf{pp}, f_{\Gamma,L}, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\mathsf{T}\}_{\ell\in[L]}, \mathbf{t}^\mathsf{T}) - e_{\mathsf{Eval}}}_{\text{independent of } s_{L+1}} - s_{L+1},$$

so $(w_{\mathsf{out}}, \mathbf{w}_1, \ldots, \mathbf{w}_L)$ is again jointly uniformly random. □

7.8.2 Construction of KP-ABE for DFA

We present our KP-ABE for DFA, utilizing the telescoping-sum structure of the DFA garbling. The idea is similar to Section 4.6.5 — an ABE key contains roughly $\mathfrak{Q}$ IPFE keys and an ABE ciphertext, L IPFE ciphertexts, so that decryption yields $\Theta(L\mathfrak{Q})$ inner products corresponding to a garbling generated using pseudorandomness. The proof, yet, is much simpler than that in Section 4.6 (as well as those of all previous pairing-based ABE for DFA), thanks to the easy-to-use security of evasive IPFE (Definition 7.2).

Ingredients of Construction 7.7. We rely on

- Construction 7.6, a specific noisy linear garbling NLG for DFA, and
- an identity-based evasive IPFE scheme IPFE for $\mathcal{I} \supseteq \{0,1,2\}$ that is B_{IPFE}-correct and restricted-σ_{IPFE}-secure (Definitions 7.1 and 7.3).

Construction 7.7 (KP-ABE for DFA). Let (see Definition 2.3)

$$\mathsf{Params} = \left\{ \overline{L} \,\middle|\, \overline{L} \in \mathbb{N} \right\}, \qquad F_{\overline{L}} = \left\{ \text{DFA } \Gamma \right\}, \qquad X_{\overline{L}} = \left\{ \mathbf{x} \in \{0,1\}^{L} \,\middle|\, L \le \overline{L} \right\},$$

$$P_{\overline{L}}(\Gamma, \mathbf{x}) = \begin{cases} 1, & \text{if } \Gamma \text{ accepts } \mathbf{x}; \\ 0, & \text{if } \Gamma \text{ rejects } \mathbf{x}. \end{cases}$$

Our KP-ABE for DFA works as follows.

- $\mathsf{Setup}(\overline{L})$ picks suitable q, n in a straight-forward way.[91] The algorithm sets $\mathsf{pp} = (q, \overline{L})$ and $K = 0, Z = 3n$. It runs $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(q, 1^K, 1^Z)$ and outputs

$$\mathsf{mpk} = (\mathsf{pp}, 1^n, 1^K, 1^Z, \mathsf{impk}), \qquad \mathsf{msk} = (\mathsf{mpk}, \mathsf{imsk}).$$

- $\mathsf{KeyGen}(\mathsf{msk}, \Gamma)$ parses $\Gamma = (1^{\Omega}, \mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \boldsymbol{\xi})$. It samples and sets

$$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_q^{n \times \Omega}, \qquad \mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^{n},$$

$$\mathbf{v}_{\mathrm{msg}} = \begin{pmatrix} -\mathbf{R}\boldsymbol{\iota}_1 \\ \mathbf{0}_n \\ 1 \\ \mathbf{0}_{n-1} \end{pmatrix}, \qquad \mathbf{V}_{\mathrm{t}} = \begin{pmatrix} -\mathbf{R} \\ \mathbf{r}\boldsymbol{\xi}^{\intercal} \\ \mathbf{0}_{n \times \Omega} \end{pmatrix}, \qquad \mathbf{V}_{\mathrm{in}} = \begin{pmatrix} \mathbf{R}\mathbf{\Gamma}_0 \\ \mathbf{R}\mathbf{\Gamma}_1 \\ -\mathbf{R} \end{pmatrix},$$

  where $\boldsymbol{\iota}_1 = (1, 0, \ldots, 0)^{\intercal} \in \{0,1\}^{\Omega}$ is the first standard basis vector. The algorithm generates IPFE secret keys

$$\mathsf{isk}_{\mathrm{msg}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 0, \mathbf{v}_{\mathrm{msg}}, \bot),$$

$$\mathsf{isk}_{\mathrm{t}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 1, \mathbf{V}_{\mathrm{t}}, \bot),$$

$$\mathsf{isk}_{\mathrm{in}} \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 2, \mathbf{V}_{\mathrm{in}}, \bot).$$

[91] Precisely speaking, the randomness used to sample q, n must be separate from that to run IPFE.Setup and efficiently sampleable conditioned on and given q, n.

It outputs $\mathsf{sk} = (\mathbf{R}, \mathbf{r}, \mathsf{isk}_{\mathrm{msg}}, \mathsf{isk}_{\mathrm{t}}, \mathsf{isk}_{\mathrm{in}})$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ samples

$$\widetilde{\mathbf{s}}_0, \widetilde{\mathbf{s}}_1, \ldots, \widetilde{\mathbf{s}}_L, \widetilde{\mathbf{s}}_{L+1} \xleftarrow{\$} \mathbb{Z}_q^n, \qquad s_{\mathrm{msg}} \xleftarrow{\$} \mathbb{Z}_q \setminus [-2 \cdot 2^{-\kappa} q, 2 \cdot 2^{-\kappa} q].$$

It sets

$$\begin{aligned}
\mathbf{u}_{\mathrm{msg}}^\intercal &= (&& \widetilde{\mathbf{s}}_0^\intercal, && \mathbf{0}_{1\times n}, && \mu s_{\mathrm{msg}}, \mathbf{0}_{1\times(n-1)} &&), \\
\mathbf{u}_{\mathrm{t}}^\intercal &= (&& \widetilde{\mathbf{s}}_L^\intercal, && \widetilde{\mathbf{s}}_{L+1}^\intercal, && \mathbf{0}_{1\times n} &&), \\
\mathbf{u}_\ell^\intercal &= (&& (1-\mathbf{x}[\ell])\widetilde{\mathbf{s}}_\ell^\intercal, && \mathbf{x}[\ell]\widetilde{\mathbf{s}}_\ell^\intercal, && \widetilde{\mathbf{s}}_{\ell-1}^\intercal &&) \qquad \text{for all } \ell \in [L],
\end{aligned}$$

generates IPFE ciphertexts

$$\begin{aligned}
\mathsf{ict}_{\mathrm{msg}} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 0, \mathfrak{v}, \mathbf{u}_{\mathrm{msg}}^\intercal), \\
\mathsf{ict}_{\mathrm{t}} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 1, \mathfrak{v}, \mathbf{u}_{\mathrm{t}}^\intercal), \\
\mathsf{ict}_\ell &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 2, \mathfrak{v}, \mathbf{u}_\ell^\intercal) \qquad \text{for all } \ell \in [L],
\end{aligned}$$

and outputs $\mathsf{ct} = (\mathsf{ict}_{\mathrm{msg}}, \mathsf{ict}_{\mathrm{t}}, \mathsf{ict}_1, \ldots, \mathsf{ict}_L)$.

- $\mathsf{Dec}(\mathsf{mpk}, \Gamma, \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ outputs $\perp$ and terminates if Γ rejects $\mathbf{x}$. Otherwise, it parses sk, ct as in $\mathsf{KeyGen}, \mathsf{Enc}$. Suppose $\mathbf{x} \in \{0,1\}^L$, the algorithm recomputes $\mathbf{v}_{\mathrm{msg}}, \mathbf{V}_{\mathrm{t}}$, $\mathbf{V}_{\mathrm{in}}$ as in KeyGen and performs IPFE decryptions

$$\begin{aligned}
w_{\mathrm{msg}} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 0, \mathbf{v}_{\mathrm{msg}}, \perp, \mathsf{isk}_{\mathrm{msg}}, \mathfrak{v}, \mathsf{ict}_{\mathrm{msg}}), \\
\mathbf{t}^\intercal &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 1, \mathbf{V}_{\mathrm{t}}, \perp, \mathsf{isk}_{\mathrm{t}}, \mathfrak{v}, \mathsf{ict}_{\mathrm{t}}), \\
\mathbf{w}_\ell^\intercal &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 2, \mathbf{V}_{\mathrm{in}}, \perp, \mathsf{isk}_{\mathrm{in}}, \mathfrak{v}, \mathsf{ict}_\ell) \qquad \text{for all } \ell \in [L].
\end{aligned}$$

It evaluates the DFA garbling

$$w_{\mathrm{out}} \leftarrow \mathsf{NLG.Eval}(\mathsf{pp}, (\Gamma, 1^L), \perp, \mathbf{x}, \{\mathbf{w}_\ell^\intercal\}_{\ell\in[L]}, \mathbf{t}^\intercal)$$

and outputs

$$\mu' \leftarrow \begin{cases} 0, & \text{if } w_{\mathrm{msg}} - w_{\mathrm{out}} \in [-2^{-\kappa} q, 2^{-\kappa} q]; \\ 1, & \text{otherwise.} \end{cases}$$

Theorem 7.20 (¶). *Construction 7.7 is* correct *(Definition 2.3) if*

$$(\overline{L}+2)B_{\mathsf{IPFE}}(q,0,3n) \le 2^{-\kappa}q.$$

Proof (Theorem 7.20). The IPFE decryption results in Dec are

$$\begin{aligned}
w_{\mathrm{msg}} &= -\widetilde{\mathbf{s}}_0^\intercal \mathbf{R}\boldsymbol{\iota}_1 + \mu s_{\mathrm{msg}} + e_{\mathrm{out}} = \widetilde{\mathbf{s}}_0^\intercal \mathbf{R}\cdot(-\boldsymbol{\iota}_1) + \mu s_{\mathrm{msg}} + e_{\mathrm{out}},\\
\mathbf{t}^\intercal &= \mathbf{u}_{\mathrm{t}}^\intercal \mathbf{V}_{\mathrm{t}} + \mathbf{e}_{\mathrm{t}}^\intercal = -\widetilde{\mathbf{s}}_L^\intercal \mathbf{R} + \widetilde{\mathbf{s}}_{L+1}^\intercal \mathbf{r}\cdot\boldsymbol{\xi}^\intercal + \mathbf{e}_{\mathrm{t}}^\intercal,\\
\mathbf{w}_\ell^\intercal &= \mathbf{u}_\ell^\intercal \mathbf{V}_{\mathrm{in}} + \mathbf{e}_\ell^\intercal = (1-\mathbf{x}[\ell])\widetilde{\mathbf{s}}_\ell^\intercal \mathbf{R}\boldsymbol{\Gamma}_0 + \mathbf{x}[\ell]\widetilde{\mathbf{s}}_\ell^\intercal \mathbf{R}\boldsymbol{\Gamma}_1 - \widetilde{\mathbf{s}}_{\ell-1}^\intercal \mathbf{R} + \mathbf{e}_\ell^\intercal\\
&= \widetilde{\mathbf{s}}_\ell^\intercal \mathbf{R}\cdot\boldsymbol{\Gamma}_{\mathbf{x}[\ell]} - \widetilde{\mathbf{s}}_{\ell-1}^\intercal \mathbf{R} + \mathbf{e}_\ell^\intercal \qquad \text{for all } \ell\in[L],
\end{aligned}$$

where the errors $e_{\mathrm{out}}, \mathbf{e}_{\mathrm{t}}, \{\mathbf{e}_\ell\}_{\ell\in[L]}$ are B_{IPFE}-bounded by the correctness of IPFE. They form a DFA garbling with randomness

$$\mathbf{s} = (\widetilde{\mathbf{s}}_0^\intercal \mathbf{R}, \widetilde{\mathbf{s}}_1^\intercal \mathbf{R}, \dots, \widetilde{\mathbf{s}}_L^\intercal \mathbf{R}, \widetilde{\mathbf{s}}_{L+1}^\intercal \mathbf{r})^\intercal.$$

Therefore, by the $(B_{\mathsf{IPFE}}, (\overline{L}+2)B_{\mathsf{IPFE}})$-correctness of NLG (Theorem 7.19), we have

$$\begin{aligned}
w_{\mathrm{out}} - (\widetilde{\mathbf{s}}_0^\intercal \mathbf{R}\cdot(-\boldsymbol{\iota}_1) + e_{\mathrm{out}}) &\in [-(\overline{L}+2)B_{\mathsf{IPFE}}(q,0,3n), (\overline{L}+2)B_{\mathsf{IPFE}}(q,0,3n)]\\
&\subseteq [-2^{-\kappa}q, 2^{\kappa}q].
\end{aligned}$$

If $\mu = 0$, it always holds that $w_{\mathrm{msg}} - w_{\mathrm{out}} \in [-2^{-\kappa}q, 2^{-\kappa}q]$, which never holds when $\mu = 1$ by our choice of s_{msg}. We conclude that Construction 7.7 is correct. □

7.8.3 Security of KP-ABE for DFA

Theorem 7.21 (¶). *If* $\mathsf{LWE}_{n,\mathrm{poly}(\lambda),q,\sigma_{\mathrm{r}}}$ *(Assumption 2.2) holds for some* $\sigma_{\mathrm{r}} \le \frac{2^{-\kappa-6}\sigma_{\mathsf{IPFE}}}{2\sqrt{\kappa}}$,[92] *then Construction 7.7 is* very selectively secure *(Definition 2.3).*

Proof (Theorem 7.21). The proof is similar to that of Theorem 7.15. Let $\mathcal{A}$ be an efficient adversary and assume that it always chooses challenges such that Γ_j rejects $\mathbf{x}$ for all $j\in[J]$. Consider the following evasive IPFE sampler $\mathcal{S}^\beta = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}_{\mathrm{U}}^\beta)$ per Definition 7.2.

- $\mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}})$ parses $r_{\mathrm{pub}} = (r_{\mathcal{A}}, r_{\mathrm{qn}}, r_{\mathrm{r}})$. It runs $\mathcal{A}(r_{\mathcal{A}})$ to obtain

$$\mathsf{param} = \overline{L}, \qquad \{\Gamma_j\}_{j\in[J]} \quad (\Gamma_j = (1^{\mathfrak{Q}_j}, \boldsymbol{\Gamma}_{j,0}, \boldsymbol{\Gamma}_{j,1}, \xi_j)), \qquad \mathbf{x}\in\{0,1\}^{\le\overline{L}}.$$

[92] This condition is the same as in Theorem 7.15, instantiated with $B_{\mathsf{short}} = 2$ and $B_{\mathsf{NLG}} = 0$.

The algorithm uses r_{qn} to sample q, n as in Setup of Construction 7.7. It uses r_{r} to sample $\mathbf{R}_j \in \mathbb{Z}_q^{n \times \mathfrak{Q}_j}$ and $\mathbf{r}_j \in \mathbb{Z}_q^n$ uniformly at random in a straight-forward way[93] for all $j \in [J]$. It computes

$$\mathbf{v}_{\mathrm{msg},j} = \begin{pmatrix} -\mathbf{R}_j \boldsymbol{\iota}_1 \\ \mathbf{0}_n \\ 1 \\ \mathbf{0}_{n-1} \end{pmatrix}, \qquad \mathbf{V}_{\mathrm{t},j} = \begin{pmatrix} -\mathbf{R}_j \\ \mathbf{r}_j \boldsymbol{\xi}_j^\intercal \\ \mathbf{0}_{n \times \mathfrak{Q}_j} \end{pmatrix}, \qquad \mathbf{V}_{\mathrm{in},j} = \begin{pmatrix} \mathbf{R}_j \boldsymbol{\Gamma}_{j,0} \\ \mathbf{R}_j \boldsymbol{\Gamma}_{j,1} \\ -\mathbf{R}_j \end{pmatrix},$$

and sets

$$K = 0, \quad Z = 3n, \quad \mathsf{ID} = \{0,1,2\}, \quad J_0 = J, \quad J_1 = J_2 = \sum_{j \in [J]} \mathfrak{Q}_j,$$

$$I_{0,\mathfrak{o}} = I_{1,\mathfrak{o}} = 1, \quad I_{2,\mathfrak{o}} = L, \quad I_{0,\mathfrak{g}} = I_{1,\mathfrak{g}} = I_{2,\mathfrak{g}} = 0, \quad \sigma_{\mathrm{pre}} = \sigma_{\mathsf{IPFE}},$$

$$\mathbf{V}_{0,\mathfrak{o}} = (\mathbf{v}_{\mathrm{msg},1}, \ldots, \mathbf{v}_{\mathrm{msg},J}), \quad \mathbf{V}_{1,\mathfrak{o}} = (\mathbf{v}_{\mathrm{t},1}, \ldots, \mathbf{v}_{\mathrm{t},J}),$$

$$\mathbf{V}_{2,\mathfrak{o}} = (\mathbf{v}_{\mathrm{in},1}, \ldots, \mathbf{v}_{\mathrm{in},J}), \quad \mathbf{V}_{0,\mathfrak{g}} = \mathbf{V}_{1,\mathfrak{g}} = \mathbf{V}_{2,\mathfrak{g}} = \bot.$$

Lastly, the algorithm outputs the required components.

- $\mathcal{S}_{\mathrm{U}}^{\beta}(r_{\mathrm{pub}}; r_{\mathrm{priv}})$ first runs the deterministic subprocedure $\mathcal{S}_{\mathrm{A0}}^{\beta}(r_{\mathrm{pub}})$, which does the following. It first reruns $\mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}})$ to obtain $q, n, \mathbf{x}, L$. The algorithm finds matrices $\mathbf{A}_{\mathrm{msg}}, \mathbf{A}_{\mathrm{t}}, \{\mathbf{A}_\ell\}_{\ell \in [L]}$ such that for all $\widetilde{\mathbf{s}}_0, \ldots, \widetilde{\mathbf{s}}_{L+1} \in \mathbb{Z}_q^n$ and $s_{\mathrm{msg}} \in \mathbb{Z}_q$,

$$\mathbf{d}_0^\intercal \mathbf{A}_{\mathrm{msg}} = \mathbf{u}_{\mathrm{msg}}^\intercal = (\widetilde{\mathbf{s}}_0^\intercal, \mathbf{0}_{1\times n}, \beta s_{\mathrm{msg}}, \mathbf{0}_{1\times(n-1)}), \qquad \mathbf{d}_0^\intercal \mathbf{A}_{\mathrm{t}} = \mathbf{u}_{\mathrm{t}}^\intercal = (\widetilde{\mathbf{s}}_L^\intercal, \widetilde{\mathbf{s}}_{L+1}^\intercal, \mathbf{0}_{1\times n}),$$
$$\mathbf{d}_0^\intercal \mathbf{A}_\ell = \mathbf{u}_\ell^\intercal = ((1-\mathbf{x}[\ell])\widetilde{\mathbf{s}}_\ell^\intercal, \mathbf{x}[\ell]\widetilde{\mathbf{s}}_\ell^\intercal, \widetilde{\mathbf{s}}_{\ell-1}^\intercal) \qquad (\text{for all } \ell \in [L]),$$

where $\mathbf{d}_0^\intercal = (s_{\mathrm{msg}}, \widetilde{\mathbf{s}}_0^\intercal, \ldots, \widetilde{\mathbf{s}}_{L+1}^\intercal)$. The algorithm $\mathcal{S}_{\mathrm{A0}}$ outputs

$$1^{n'} = 1^{n(L+2)+1}, \quad \mathbf{A}_{0,\mathfrak{o},0,1} = \mathbf{A}_{\mathrm{msg}}, \quad \mathbf{A}_{1,\mathfrak{o},0,1} = \mathbf{A}_{\mathrm{t}}, \quad \mathbf{A}_{2,\mathfrak{o},0,\ell} = \mathbf{A}_\ell \ (\text{for all } \ell \in [L]).$$

Completing $\mathcal{S}_{\mathrm{A0}}$, the algorithm $\mathcal{S}_{\mathrm{U}}$ samples $\mathbf{d}_0 \xleftarrow{\$} \mathbb{Z}_q^{n'}$ using r_{priv} and outputs

$$\mathbf{U}_{\mathsf{id},\mathfrak{o}}^\intercal = \begin{pmatrix} \mathbf{d}_0^\intercal \mathbf{A}_{\mathsf{id},\mathfrak{o},0,1} \\ \vdots \\ \mathbf{d}_0^\intercal \mathbf{A}_{\mathsf{id},\mathfrak{o},0,I_{\mathsf{id},\mathfrak{o}}} \end{pmatrix} \qquad \text{for all } \mathsf{id} \in \mathsf{ID}.$$

[93] Precisely speaking, r_{r} must be efficiently sampleable conditioned on and given $\mathbf{R}_1, \mathbf{r}_1, \ldots, \mathbf{R}_J, \mathbf{r}_J$.

By definition, $\mathcal{S}^\beta$ is a restricted sampler (Definition 7.3). We have the following:

Claim 7.22 (¶). *$\mathcal{S}^\beta$ has pseudorandom noisy inner products, i.e., $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{pre}}$ (Definition 7.2) holds for both $\beta \in \{0,1\}$.*

The very selective security of Construction 7.7 follows from Claim 7.22. Consider the following hybrids.

- H^β_0 is $\mathsf{Exp}^\beta_{\mathrm{ABE}}$ for Construction 7.7. Note that s_{msg} in the challenge ciphertext is uniformly random over $\mathbb{Z}_q \setminus [-2 \cdot 2^{-\kappa} q, 2 \cdot 2^{-\kappa} q]$.
- In H^β_1, the scalar s_{msg} is changed to be uniformly random over $\mathbb{Z}_q$. Clearly, $\mathsf{H}^\beta_0 \approx_{\mathrm{s}} \mathsf{H}^\beta_1$. Moreover, H^β_1 corresponds to the left distribution of $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{post}}$.
- In H^β_2, the vectors inside IPFE ciphertexts (components of the ABE challenge ciphertext) are replaced by random, which corresponds to the right distribution of $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{post}}$. By Claim 7.22 and the restricted-σ_{IPFE}-security of IPFE, we have $\mathsf{H}^\beta_1 \approx \mathsf{H}^\beta_2$.

Lastly, $\mathsf{H}^0_2 \equiv \mathsf{H}^1_2$. By hybrid argument, we conclude $\mathsf{Exp}^0_{\mathrm{ABE}} \approx \mathsf{Exp}^1_{\mathrm{ABE}}$, completing the proof. □

Proof (Claim 7.22). Fix $\beta \in \{0,1\}$ and consider the following hybrids.

- H_0 is the left distribution of $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{pre}}$. Recall that the public randomness is $r_{\mathrm{pub}} = (r_{\mathcal{A}}, r_{\mathrm{qn}}, r_{\mathrm{r}})$. Adopting the notations of the ABE decryption algorithm, the noisy inner products are

$$\begin{aligned} w_{\mathrm{msg},j} &= \mathbf{u}^\top_{\mathrm{msg}} \mathbf{v}_{\mathrm{msg},j} + e_{\mathrm{out},j} = \widetilde{\mathbf{s}}_0^\top \mathbf{R}_j \cdot (-\boldsymbol{\iota}_1) + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \\ \mathbf{t}^\top_j &= \mathbf{u}^\top_{\mathrm{t}} \mathbf{V}_{\mathrm{t},j} + \mathbf{e}^\top_{\mathrm{t},j} = -\widetilde{\mathbf{s}}_L^\top \mathbf{R}_j + \widetilde{\mathbf{s}}_{L+1}^\top \mathbf{r}_j \cdot \boldsymbol{\xi}_j^\top + \mathbf{e}^\top_{\mathrm{t},j}, \\ \mathbf{w}^\top_{j,\ell} &= \mathbf{u}^\top_\ell \mathbf{V}_\ell + \mathbf{e}^\top_{j,\ell} = \widetilde{\mathbf{s}}_\ell^\top \mathbf{R}_j \cdot \boldsymbol{\Gamma}_{j,\mathbf{x}[\ell]} - \widetilde{\mathbf{s}}_{\ell-1}^\top \mathbf{R}_j + \mathbf{e}^\top_{j,\ell} \qquad (\text{for all } \ell \in [L]), \end{aligned}$$

  where $j \in [J]$ and the entries of $e, \mathbf{e}$'s are independent $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{IPFE}}}$.
- In H_1, additional small noises are attached to $\widetilde{\mathbf{s}}^\top\mathbf{R}, \widetilde{\mathbf{s}}^\top\mathbf{r}$'s. For all $j \in [J]$, set the

inner products to

$$\begin{aligned} w_{\mathrm{msg},j} &= (\widetilde{\mathbf{s}}_0^\intercal \mathbf{R}_j + \mathbf{e}_{\mathrm{r},j,0}^\intercal)(-\boldsymbol{\iota}_1) + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \\ \mathbf{t}_j^\intercal &= -(\widetilde{\mathbf{s}}_L^\intercal \mathbf{R}_j + \mathbf{e}_{\mathrm{r},j,L}^\intercal) + (\widetilde{\mathbf{s}}_{L+1}^\intercal \mathbf{r}_j + e_{\mathrm{r},j,L+1})\boldsymbol{\xi}_j^\intercal + \mathbf{e}_{\mathrm{t},j}^\intercal, \\ \mathbf{w}_{j,\ell}^\intercal &= (\widetilde{\mathbf{s}}_\ell^\intercal \mathbf{R}_j + \mathbf{e}_{\mathrm{r},j,\ell}^\intercal)\boldsymbol{\Gamma}_{j,\mathbf{x}[\ell]} - (\widetilde{\mathbf{s}}_{\ell-1}^\intercal \mathbf{R}_j + \mathbf{e}_{\mathrm{r},j,\ell-1}^\intercal) + \mathbf{e}_{j,\ell}^\intercal \qquad (\text{for all } \ell \in [L]), \end{aligned}$$

where $\mathbf{e}_{\mathrm{r},j,\ell} \xleftarrow{\$} \mathcal{D}^{\mathfrak{Q}_j}_{\mathbb{Z},\sigma_\mathrm{r},\le\sigma_\mathrm{r}\sqrt{\kappa}}$ for all $\ell \in [0,L]$ and $e_{\mathrm{r},j,L+1} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_\mathrm{r},\le\sigma_\mathrm{r}\sqrt{\kappa}}$. The errors introduced into the inner products are bounded by

$$2\sigma_\mathrm{r}\sqrt{\kappa} \le 2^{-\kappa-6}\sigma_{\mathsf{IPFE}},$$

so they are flooded (Lemma 2.2) by $\{e_{\mathrm{out},j}, \mathbf{e}_{\mathrm{t},j}, \{\mathbf{e}_{j,\ell}\}_{\ell\in[L]}\}_{j\in[J]}$, and $\mathsf{H}_0 \approx_\mathrm{s} \mathsf{H}_1$.

- In H_2, the $\mathbf{e}_\mathrm{r}$'s are no longer truncated, i.e., for all $j \in [J]$, sample $\mathbf{e}_{\mathrm{r},j,\ell} \xleftarrow{\$} \mathcal{D}^{\mathfrak{Q}_j}_{\mathbb{Z},\sigma_\mathrm{r}}$ for all $\ell \in [0,L]$ and $e_{\mathrm{r},j,L+1} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma_\mathrm{r}}$. We have $\mathsf{H}_1 \approx_\mathrm{s} \mathsf{H}_2$ by Lemma 2.1.

- In H_3, the garblings use uniform randomness. For all $j \in [J]$, sample $\mathbf{s}_{j,\ell} \xleftarrow{\$} \mathbb{Z}_q^n$ for all $\ell \in [0,L]$ and $s_{j,L+1} \xleftarrow{\$} \mathbb{Z}_q$, and set the inner products to

$$\begin{aligned} w_{\mathrm{msg},j} &= \mathbf{s}_{j,0}^\intercal \cdot (-\boldsymbol{\iota}_1) + \beta s_{\mathrm{msg}} + e_{\mathrm{out},j}, \\ \mathbf{t}_j^\intercal &= -\mathbf{s}_{j,L}^\intercal + s_{j,L+1}\boldsymbol{\xi}_j^\intercal + \mathbf{e}_{\mathrm{t},j}^\intercal, \\ \mathbf{w}_{j,\ell}^\intercal &= \mathbf{s}_{j,\ell}^\intercal \boldsymbol{\Gamma}_{j,\mathbf{x}[\ell]} - \mathbf{s}_{j,\ell-1}^\intercal + \mathbf{e}_{j,\ell}^\intercal \qquad (\text{for all } \ell \in [L]), \end{aligned}$$

  By $(L+1)$ hybrids of applying $\mathsf{LWE}_{n,J_2,q,\sigma_\mathrm{r}}$ with public matrix $(\mathbf{R}_1,\ldots,\mathbf{R}_J)$ and secrets $\widetilde{\mathbf{s}}_0,\ldots,\widetilde{\mathbf{s}}_L$, then $\mathsf{LWE}_{n,J,q,\sigma_\mathrm{r}}$ with public matrix $(\mathbf{r}_1,\ldots,\mathbf{r}_J)$ and secret $\widetilde{\mathbf{s}}_{L+1}$, we have

$$\left\{ \{\widetilde{\mathbf{s}}_\ell^\intercal \mathbf{R}_j + \mathbf{e}_{\mathrm{r},j,\ell}^\intercal\}_{\ell\in[0,L]}, \widetilde{\mathbf{s}}_{L+1}^\intercal \mathbf{r}_j + e_{\mathrm{r},j,L+1} \right\}_{j\in[J]} \approx \left\{ \{\mathbf{s}_{j,\ell}^\intercal\}_{\ell\in[0,L]}, s_{j,L+1} \right\}_{j\in[J]},$$

  which implies $\mathsf{H}_2 \approx \mathsf{H}_3$. In this step, we rely on the "straight-forwardness" of sampling $\mathbf{R}, \mathbf{r}$'s from r_r in $\mathcal{S}^\beta$.

- In H_4, all the inner products are replaced by random, i.e., the right distribution of $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{pre}}$. Note that the inner products in H_3 are well-randomized garblings (by $\mathbf{s}$'s) plus independent noises ($e, \mathbf{e}$'s). Therefore, by J hybrids of applying the security of NLG, we conclude $\mathsf{H}_3 \equiv \mathsf{H}_4$.

By hybrid argument, $\mathsf{H}_0 \approx \mathsf{H}_4$, i.e., $\mathsf{IPFEsec}^{\mathcal{S}^\beta}_{\mathrm{pre}}$ holds. □

7.8.4 CP-ABE for DFA and Summary

Our CP-ABE for DFA is similar to KP-ABE for DFA and general CP-ABE and features a mixture of how components are set in both schemes.

Ingredients of Construction 7.8. We rely on

- Construction 7.6, a specific noisy linear garbling NLG for DFA, and
- an identity-based evasive IPFE scheme IPFE for $\mathcal{I} \supseteq \{0,1,2\}$ that is B_{IPFE}-correct and restricted-σ_{IPFE}-secure (Definitions 7.1 and 7.3).

Construction 7.8 (CP-ABE for DFA). Let (see Definition 2.3)

$$\mathsf{Params} = \left\{ \overline{L} \,\middle|\, \overline{L} \in \mathbb{N} \right\}, \qquad F_{\overline{L}} = \left\{ \mathbf{x} \in \{0,1\}^L \,\middle|\, L \leq \overline{L} \right\}, \qquad X_{\overline{L}} = \left\{ \text{DFA } \Gamma \right\},$$

$$P_{\overline{L}}(\mathbf{x}, \Gamma) = \begin{cases} 1, & \text{if } \Gamma \text{ accepts } \mathbf{x}; \\ 0, & \text{if } \Gamma \text{ rejects } \mathbf{x}. \end{cases}$$

Our CP-ABE for DFA works as follows.

- $\mathsf{Setup}(\overline{L})$ picks suitable q, n in a straight-forward way. It sets $\mathsf{pp} = (q, \overline{L}), K = 0$, $Z = 3n$. The algorithm runs $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(q, 1^K, 1^Z)$ and outputs

$$\mathsf{mpk} = (\mathsf{pp}, 1^n, 1^K, 1^Z, \mathsf{impk}), \qquad \mathsf{msk} = (\mathsf{mpk}, \mathsf{imsk}).$$

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$, given $\mathbf{x} \in \{0,1\}^L$, samples and sets

$$\mathbf{r}_0, \ldots, \mathbf{r}_L \xleftarrow{\$} \mathbb{Z}_q^n, \qquad r_{L+1} \xleftarrow{\$} \mathbb{Z}_q,$$

$$\mathbf{v}_{\mathsf{msg}} = \begin{pmatrix} \mathbf{r}_0 \\ 1 \\ \mathbf{0}_{2n-1} \end{pmatrix}, \qquad \mathbf{v}_{\mathsf{t}} = \begin{pmatrix} \mathbf{r}_L \\ r_{L+1} \\ \mathbf{0}_{2n-1} \end{pmatrix}, \qquad \mathbf{v}_\ell = \begin{pmatrix} (1 - \mathbf{x}[\ell])\mathbf{r}_\ell \\ \mathbf{x}[\ell]\mathbf{r}_\ell \\ \mathbf{r}_{\ell-1} \end{pmatrix} \quad \text{for all } \ell \in [L].$$

  It generates IPFE secret keys

$$\begin{aligned} \mathsf{isk}_{\mathsf{msg}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 0, \mathbf{v}_{\mathsf{msg}}, \bot), \\ \mathsf{isk}_{\mathsf{t}} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 1, \mathbf{v}_{\mathsf{t}}, \bot), \\ \mathsf{isk}_\ell &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 2, \mathbf{v}_\ell, \bot) \qquad \text{for all } \ell \in [L]. \end{aligned}$$

  The algorithm outputs $\mathsf{sk} = (\mathbf{r}_0, \ldots, \mathbf{r}_L, r_{L+1}, \mathsf{isk}_{\mathsf{msg}}, \mathsf{isk}_{\mathsf{t}}, \mathsf{isk}_1, \ldots, \mathsf{isk}_L)$.

- $\mathsf{Enc}(\mathsf{mpk}, \Gamma, \mu)$ parses $\Gamma = (1^{\mathfrak{Q}}, \mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \boldsymbol{\xi})$. It samples

$$\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{\mathfrak{Q} \times n}, \qquad s_{-1} \xleftarrow{\$} \mathbb{Z}_q, \qquad s_{\mathrm{msg}} \xleftarrow{\$} \mathbb{Z}_q \setminus [-2 \cdot 2^{-\kappa} q, 2 \cdot 2^{-\kappa} q],$$

  and sets

$$\mathbf{u}_{\mathrm{msg}}^{\intercal} = (-\boldsymbol{\iota}_1^{\intercal} \mathbf{S}, \mu s_{\mathrm{msg}}, \mathbf{0}_{1 \times (2n-1)}), \quad \mathbf{U}_{\mathsf{t}}^{\intercal} = (-\mathbf{S}, s_{-1} \boldsymbol{\xi}, \mathbf{0}_{\mathfrak{Q} \times (2n-1)}), \quad \mathbf{U}_{\mathrm{in}}^{\intercal} = (\mathbf{\Gamma}_0^{\intercal} \mathbf{S}, \mathbf{\Gamma}_1^{\intercal} \mathbf{S}, -\mathbf{S}).$$

  The algorithm generates IPFE ciphertexts

$$\begin{aligned} \mathsf{ict}_{\mathrm{msg}} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 0, \mathfrak{o}, \mathbf{u}_{\mathrm{msg}}^{\intercal}), \\ \mathsf{ict}_{\mathsf{t}} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 1, \mathfrak{o}, \mathbf{U}_{\mathsf{t}}^{\intercal}), \\ \mathsf{ict}_{\mathrm{in}} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 2, \mathfrak{o}, \mathbf{U}_{\mathrm{in}}^{\intercal}), \end{aligned}$$

  and outputs $\mathsf{ct} = (\mathsf{ict}_{\mathrm{msg}}, \mathsf{ict}_{\mathsf{t}}, \mathsf{ict}_{\mathrm{in}})$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}, \mathsf{sk}, \Gamma, \mathsf{ct})$ outputs $\perp$ and terminates if Γ rejects $\mathbf{x}$. Otherwise, it parses sk, ct as in $\mathsf{KeyGen}, \mathsf{Enc}$. Suppose $\mathbf{x} \in \{0,1\}^L$, the algorithm recomputes $\mathbf{v}_{\mathrm{msg}}, \mathbf{v}_{\mathsf{t}}$, $\mathbf{v}_1, \ldots, \mathbf{v}_L$ as in KeyGen and performs IPFE decryptions

$$\begin{aligned} w_{\mathrm{msg}} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 0, \mathbf{v}_{\mathrm{msg}}, \perp, \mathsf{isk}_{\mathrm{msg}}, \mathfrak{o}, \mathsf{ict}_{\mathrm{msg}}), \\ \mathbf{t} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 1, \mathbf{v}_{\mathsf{t}}, \perp, \mathsf{isk}_{\mathsf{t}}, \mathfrak{o}, \mathsf{ict}_{\mathsf{t}}), \\ \mathbf{w}_\ell &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 2, \mathbf{v}_\ell, \perp, \mathsf{isk}_\ell, \mathfrak{o}, \mathsf{ict}_{\mathrm{in}}) \qquad \text{for all } \ell \in [L]. \end{aligned}$$

  It evaluates the DFA garbling

$$w_{\mathrm{out}} \leftarrow \mathsf{NLG.Eval}(\mathsf{pp}, (\Gamma, 1^L), \perp, \mathbf{x}, \{\mathbf{w}_\ell^{\intercal}\}_{\ell \in [L]}, \mathbf{t}^{\intercal})$$

  and outputs

$$\mu' \leftarrow \begin{cases} 0, & \text{if } w_{\mathrm{msg}} - w_{\mathrm{out}} \in [-2^{-\kappa} q, 2^{-\kappa} q]; \\ 1, & \text{otherwise.} \end{cases}$$

Theorem 7.23. *Construction 7.8 is* correct *(Definition 2.3) if* $(\overline{L} + 2) B_{\mathsf{IPFE}}(q, 0, 3n) \leq 2^{-\kappa} q$.

Theorem 7.24. *If* $\mathsf{LWE}_{n, \mathrm{poly}(\lambda), q, \sigma_{\mathrm{r}}}$ *(Assumption 2.2) holds for some* $\sigma_{\mathrm{r}} \leq \frac{2^{-\kappa-6} \sigma_{\mathsf{IPFE}}}{2\sqrt{\kappa}}$*, then Construction 7.8 is* very selectively secure *(Definition 2.3).*

Summary. By instantiating our ABE schemes with the IPFE scheme of Construction 7.2, we obtain KP- and CP-ABE for DFA.

Corollary 7.25. *Under LWE (Assumption 2.2) and evasive LWE (Assumption 7.2), there exists KP-ABE for DFA with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \qquad |\mathsf{sk}_\Gamma| = (|\Gamma| + 1)\,\mathrm{poly}(\lambda), \qquad |\mathsf{ct}_{\mathbf{x}}| = (|\mathbf{x}| + 1)\,\mathrm{poly}(\lambda),$$

and CP-ABE for DFA with

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \qquad |\mathsf{sk}_{\mathbf{x}}| = (|\mathbf{x}| + 1)\,\mathrm{poly}(\lambda), \qquad |\mathsf{ct}_\Gamma| = (|\Gamma| + 1)\,\mathrm{poly}(\lambda),$$

where $|\Gamma|$ *is the number of states of* Γ.

We remark that, starting from the linear secret sharing scheme for L (Construction 4.4), we also obtain ABE for L via the same construction. However, ABE for NFA or NL cannot be obtained so. The schemes for DFA and L crucially rely on the fact that, when noises are added to the LSSS shares, the output noise grows *linearly* with the computation size. On the other hand, the secret sharing schemes for NFA and NL, once cast as noisy linear garbling, incur exponential noise growth during reconstruction. The latter fact makes the resultant ABE bounded.

7.9 CP-ABE with Succinct Ciphertexts

In this section, we show that with the help of evasive IPFE with structured noises, we can construct CP-ABE from general noisy linear garbling scheme (not necessarily short). When instantiated with the noisy linear garbling implicit in [BGG+14], this yields a CP-ABE scheme with ciphertext size dependent only on circuit depth instead of circuit size.

7.9.1 Succinct Noisy Linear Garbling for Circuits

The work of [BGG+14] presented a KP-ABE scheme with succinct keys for circuits based on lattices. At the core of their construction are a pair of deterministic homomorphic

evaluation algorithms $\mathsf{EvalC}, \mathsf{EvalCX}$, which, for circuit $C : \{0,1\}^L \to \{0,1\}$, input $x \in \{0,1\}^L$, and public matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m(L+1)}$, satisfies the relation

$$\mathsf{EvalC}(\mathbf{A}, C) = \mathbf{A}_C \in \mathbb{Z}_q^{n \times m},$$
$$\mathsf{EvalCX}(\mathbf{s}^\intercal(\mathbf{A} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}^\intercal, f, \mathbf{x}) = \mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_C^\intercal,$$

where $\|\mathbf{e}_C\|$, relative to $\|\mathbf{e}\|$, grows exponentially with the depth of C. (In [BGG+14], it also works for bounded-arithmetic circuits, but we only present the Boolean version.) It naturally yields a noisy linear garbling scheme.

Construction 7.9 (garbling adapted from [BGG+14]). Let

$$\mathsf{Params} = \left\{ 1^d \;\middle|\; d \in \mathbb{N} \right\},$$
$$\mathcal{F}_{1^d} = \left\{ f_C \;\middle|\; C \text{ is a Boolean circuit of depth no more than } d \right\},$$
$$f_C(\mathbf{x}) = \begin{cases} \bot, & \text{if } \mathbf{x} \notin \{0,1\}^L; \\ \neg C(\mathbf{x}), & \text{if } \mathbf{x} \in \{0,1\}^L; \end{cases} \qquad \text{for } C : \{0,1\}^L \to \{0,1\} \text{ and } \mathbf{x} \in \mathbb{Z}^L.$$

The noisy linear garbling scheme works as follows.

- $\mathsf{Setup}(1^d)$ picks suitable q, n, m (with $m \geq n\lceil \log_2 q \rceil$) and outputs $\mathsf{pp} = (q, 1^n, 1^m)$.
- $\mathsf{GenF}(\mathsf{pp}, C)$ samples $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_L \xleftarrow{\$} \mathbb{Z}_q^{n \times m(L+1)}$ and $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$. It computes

  $$\mathbf{A}_C \leftarrow \mathsf{EvalC}((\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_L), C).$$

  The algorithm outputs

  $$1^{n_C} = 1^n, \quad \mathbf{w}_{\text{out}} = \mathbf{A}_C\mathbf{G}^{-1}(\mathbf{a}), \quad \mathbf{W}_{\ell,0} = \mathbf{A}_\ell, \qquad \mathbf{W}_{\ell,\times} = -\mathbf{G}, \quad (\text{for all } \ell \in [L])$$
  $$\mathbf{T} = \mathbf{A}_0 - \mathbf{G}, \qquad \mathbf{R} = \mathbf{a}.$$

  All the samplings are done in a straight-forward way.
- $\mathsf{Eval}(\mathsf{pp}, C, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\intercal\}_{\ell \in [L]}, \mathbf{t}^\intercal)$ computes and outputs

  $$\mathsf{EvalCX}((\mathbf{t}^\intercal, \mathbf{w}_1^\intercal, \ldots, \mathbf{w}_L^\intercal), C, \mathbf{x}) \cdot \mathbf{G}^{-1}(\mathbf{a}).$$

Lemma 7.26 ([BGG+14]). *Construction 7.9 is* correct *(Definition 7.4) if*

$$(m+2)^{d+1}B_{\mathrm{in}}(d) \le B_{\mathrm{out}}(d).$$

It has fixed randomness dimension *(Definition 7.6). Suppose*

$$\mathsf{GenGaussian}'(1^d, (q,n,m), C, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}, \mathbf{x})$$

samples (truncated) Gaussian noises with appropriate width σ of suitable shape

$$(e_{\mathrm{out}}, \mathbf{e}_1, \dots, \mathbf{e}_L, \mathbf{e}_{\mathrm{t}}) \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\kappa}}^{1+(L+1)m},$$

then Construction 7.9 is GenGaussian′-secure *(Definition 7.7) if* $\mathsf{LWE}_{n,\mathrm{poly}(\lambda),q,\sigma'}$ *(Assumption 2.3) holds for some* $\sigma' \le \frac{\sigma}{2^{\kappa+6}(m+2)^{d+1}\sqrt{\kappa}}$.

7.9.2 CP-ABE from General Noisy Linear Garbling

We present a construction of CP-ABE from noisy linear garbling with fixed randomness dimension that is not necessarily short. It has a lot in common with Construction 7.5. Both compute rerandomized garbling using IPFE. In Construction 7.5, security relies on flooding to introduce noises to argue that the garblings use good pseudorandomness. Here, we use structured noises, together with some algebraic tweaks ("noisy secret sharing", see Section 7.2), to attach the same noise to the garbling pseudorandomness.

Ingredients of Construction 7.10. We rely on

- a noisy linear garbling scheme NLG supporting $\mathcal{F} = \{\mathcal{F}_{\mathsf{param}'}\}_{\mathsf{param}'\in\mathsf{Params}'}$ that is $(B_{\mathrm{in}}, B_{\mathrm{out}})$-correct, has fixed randomness dimension, and is GenNoise-secure such that the output of GenNoise is in $[-B_{\mathsf{NLG}}, B_{\mathsf{NLG}}]^\star$ (Definitions 7.4, 7.6, and 7.7), and
- an identity-based evasive IPFE with structured noises scheme IPFE for identity space $\mathcal{I} \supseteq [-1, L]$ that is B_{IPFE}-correct and restricted-σ_{IPFE}-secure (Definitions 7.1 and 7.3).

Construction 7.10 (CP-ABE from general garbling). Let (see Definition 2.3)

$$\mathsf{Params} = \left\{ \mathsf{param} = (\mathsf{param}', 1^L) \,\middle|\, \mathsf{param}' \in \mathsf{Params}', L \in \mathbb{N} \right\}, \quad F_{\mathsf{param}',1^L} = \{0,1\}^L,$$
$$X_{\mathsf{param}',1^L} = \left\{ f : \mathbb{Z}^L \to \{0,1,\bot\} \,\middle|\, f \in \mathcal{F}_{\mathsf{param}'} \right\}, \qquad P_{\mathsf{param}',1^L}(\mathbf{x}, f) = f(\mathbf{x}).$$

(Note that $\mathbf{x}$ is confined to Boolean.) Our CP-ABE scheme works as follows.

- $\mathsf{Setup}(\mathsf{param}', 1^L)$ runs

$$\mathsf{pp} = (q, 1^n, \dots) \xleftarrow{\$} \mathsf{NLG.Setup}(\mathsf{param}').$$

  The algorithm sets $K = \lceil \log_2 q \rceil$ to be the dimension of $\widetilde{\mathbf{g}}$ (for structured noises), and $Z = 2(n + n^2K)$. It runs $(\mathsf{impk}, \mathsf{imsk}) \xleftarrow{\$} \mathsf{IPFE.Setup}(q, 1^K, 1^Z)$ and outputs

$$\mathsf{mpk} = (\mathsf{pp}, 1^K, 1^Z, \mathsf{impk}), \qquad \mathsf{msk} = (\mathsf{mpk}, \mathsf{imsk}).$$

- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\mathbf{Q} \xleftarrow{\$} \mathbb{Z}_q^{n \times nK}$. Recall that $\mathsf{col}(\mathbf{Q})$ is a column of dimension n^2K concatenating the columns of $\mathbf{Q}$. The algorithm sets

$$\mathbf{V}_{\mathrm{Q}} = \begin{pmatrix} \mathbf{Q} \\ \mathbf{0}_{(Z-n)\times nK} \end{pmatrix}, \quad \mathbf{v}_0 = \begin{pmatrix} \mathbf{r} \\ \mathsf{col}(\mathbf{Q}) \\ 1 \\ \mathbf{0}_{n-1+n^2K} \end{pmatrix}, \quad \mathbf{v}_\ell = \begin{pmatrix} (1-\mathbf{x}[\ell])\mathbf{r} \\ (1-\mathbf{x}[\ell])\mathsf{col}(\mathbf{Q}) \\ \mathbf{x}[\ell]\mathbf{r} \\ \mathbf{x}[\ell]\mathsf{col}(\mathbf{Q}) \end{pmatrix} \text{ for all } \ell \in [L].$$

  It generates IPFE secret keys

$$\begin{aligned} \mathsf{isk}_{-1} &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, -1, \mathbf{0}, \mathbf{V}_{\mathrm{Q}}), \\ \mathsf{isk}_0 &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, 0, \mathbf{v}_0, \mathbf{0}), \\ \mathsf{isk}_\ell &\xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{imsk}, \ell, \mathbf{v}_\ell, \mathbf{0}) \qquad \text{for all } \ell \in [L], \end{aligned}$$

  and outputs $\mathsf{sk} = (\mathbf{r}, \mathbf{Q}, \mathsf{isk}_{-1}, \mathsf{isk}_0, \mathsf{isk}_1, \dots, \mathsf{isk}_L)$.

- $\mathsf{Enc}(\mathsf{mpk}, f, \mu)$ runs

$$(1^n, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}) \xleftarrow{\$} \mathsf{NLG.GenF}(\mathsf{pp}, f).$$

  It samples $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$, $\mathbf{s}_{\mathrm{Q}} \xleftarrow{\$} \mathbb{Z}_q^n$, $s_{\mathrm{msg}} \xleftarrow{\$} \mathbb{Z}_q \setminus [-2\cdot 2^{-\kappa}q, 2\cdot 2^{-\kappa}q]$, and sets

$$\begin{aligned} \mathbf{u}_{\mathrm{Q}}^\intercal &= \left(\mathbf{s}_{\mathrm{Q}}^\intercal, \mathbf{0}_{1\times(Z-n)}\right), \\ \mathbf{u}_{\mathrm{msg}}^\intercal &= \left(\mathbf{w}_{\mathrm{out}}^\intercal \mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}) \otimes \mathbf{s}_{\mathrm{Q}})^\intercal, \mu s_{\mathrm{msg}}, \mathbf{0}_{1\times(n-1+n^2K)}\right), \\ \mathbf{U}_{\mathrm{t}}^\intercal &= \left(\mathbf{T}^\intercal \mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{T}) \otimes \mathbf{s}_{\mathrm{Q}})^\intercal, \mathbf{0}_{\star\times(n+n^2K)}\right), \end{aligned}$$

$$\mathbf{U}_\ell^\intercal = \big(\mathbf{W}_{\ell,0}^\intercal \mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0}) \otimes \mathbf{s}_\mathrm{Q})^\intercal, (\mathbf{W}_{\ell,0}^\intercal + \mathbf{W}_{\ell,\times}^\intercal)\mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{W}_{\ell,\times}) \otimes \mathbf{s}_\mathrm{Q})^\intercal\big),$$

where $\ell \in [L]$. The algorithm generates IPFE ciphertexts

$$\begin{aligned}
\mathsf{ict}_\mathrm{Q} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, -1, \mathfrak{g}, \mathbf{u}_\mathrm{Q}^\intercal), \\
\mathsf{ict}_\mathrm{msg} &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 0, \mathfrak{o}, \mathbf{u}_\mathrm{msg}^\intercal), \qquad \mathsf{ict}_\mathrm{t} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, 0, \mathfrak{o}, \mathbf{U}_\mathrm{t}^\intercal), \\
\mathsf{ict}_\ell &\xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{impk}, \ell, \mathfrak{o}, \mathbf{U}_\ell^\intercal) \qquad \text{for all } \ell \in [L],
\end{aligned}$$

and outputs $\mathsf{ct} = (\mathbf{R}, \mathsf{ict}_\mathrm{Q}, \mathsf{ict}_\mathrm{msg}, \mathsf{ict}_\mathrm{t}, \mathsf{ict}_1, \ldots, \mathsf{ict}_L)$.

- $\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}, \mathsf{sk}, f, \mathsf{ct})$ outputs $\perp$ and terminates if $f(\mathbf{x}) \neq 1$. Otherwise, it parses sk, ct as in KeyGen, Enc, recomputes $\mathbf{V}_\mathrm{Q}, \mathbf{v}_0, \{\mathbf{v}_\ell\}_{\ell\in[L]}$, and performs IPFE decryptions for the shares

$$\begin{aligned}
\mathsf{sh}_\mathrm{Q}^\intercal &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, -1, \mathbf{0}, \mathbf{V}_\mathrm{Q}, \mathsf{isk}_{-1}, \mathfrak{g}, \mathsf{ict}_\mathrm{Q}), \\
\mathsf{sh}_\mathrm{msg} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 0, \mathbf{v}_0, \mathbf{0}, \mathsf{isk}_0, \mathfrak{o}, \mathsf{ict}_\mathrm{msg}), \\
\mathsf{sh}_\mathrm{t} &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, 0, \mathbf{v}_0, \mathbf{0}, \mathsf{isk}_0, \mathfrak{o}, \mathsf{ict}_\mathrm{t}), \\
\mathsf{sh}_\ell &\leftarrow \mathsf{IPFE.Dec}(\mathsf{impk}, \ell, \mathbf{v}_\ell, \mathbf{0}, \mathsf{isk}_\ell, \mathfrak{o}, \mathsf{ict}_\ell) \qquad \text{for all } \ell \in [L].
\end{aligned}$$

The algorithm reconstructs the garbling from the shares as

$$\begin{aligned}
w_\mathrm{msg} &\leftarrow \mathsf{sh}_\mathrm{msg} + \mathsf{sh}_\mathrm{Q}^\intercal \widetilde{\mathbf{G}}^{-1}(\mathbf{w}_\mathrm{out}), \qquad & \mathbf{t}^\intercal &\leftarrow \mathsf{sh}_\mathrm{t}^\intercal + \mathsf{sh}_\mathrm{Q}^\intercal \widetilde{\mathbf{G}}^{-1}(\mathbf{T}), \\
\mathbf{w}_\ell^\intercal &\leftarrow \mathsf{sh}_\ell^\intercal + \mathsf{sh}_\mathrm{Q}^\intercal \widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) \qquad & \text{for all } \ell &\in [L].
\end{aligned}$$

It evaluates the garbling by

$$w_\mathrm{out} \leftarrow \mathsf{NLG.Eval}(\mathsf{pp}, f, \mathbf{R}, \mathbf{x}, \{\mathbf{w}_\ell^\intercal\}_{\ell\in[L]}, \mathbf{t}^\intercal)$$

and outputs

$$\mu' \leftarrow \begin{cases} 0, & \text{if } w_\mathrm{msg} - w_\mathrm{out} \in [-2^{-\kappa}q, 2^\kappa q]; \\ 1, & \text{otherwise.} \end{cases}$$

Theorem 7.27 (¶). *Construction 7.5 is* correct *(Definition 2.3) if*

- *the modulus q output by* $\mathsf{NLG.Setup}(\mathsf{param}')$ *is always a prime and always satisfies* $B_\mathrm{out}(\mathsf{param}') \leq 2^{-\kappa}q$*, and*

- *it holds that* $(nK+1)B_{\mathsf{IPFE}}(q,K,2(n+n^2K)) \le B_{\mathrm{in}}(\mathsf{param}')$.

Proof (Theorem 7.27). By the correctness of IPFE and how we set the vectors, we have

$$\mathsf{sh}_{\mathrm{Q}}^{\intercal} = \mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q} + \widetilde{\mathbf{g}}^{\intercal} \otimes \mathbf{e}_{\mathrm{sr}}^{\intercal} + \mathbf{e}_{\mathrm{Q}}^{\intercal} = \mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q} + \mathbf{e}_{\mathrm{sr}}^{\intercal}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q}}^{\intercal}$$

for some B_{IPFE}-bounded errors $\mathbf{e}_{\mathrm{sr}}, \mathbf{e}_{\mathrm{Q}}$. For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times\star}$,

$$\mathsf{sh}_{\mathrm{Q}}^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{A}) = (\mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q} + \mathbf{e}_{\mathrm{sr}}^{\intercal}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q}}^{\intercal})\widetilde{\mathbf{G}}^{-1}(\mathbf{A}) = \underbrace{\mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q}\widetilde{\mathbf{G}}^{-1}(\mathbf{A})}_{\text{one-time pad}} + \underbrace{\mathbf{e}_{\mathrm{sr}}^{\intercal}\mathbf{A}}_{\text{“attached” error}} + \underbrace{\mathbf{e}_{\mathrm{Q}}^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{A})}_{\text{small error}}.$$

For the table shares, we have

$$\begin{aligned}\mathsf{sh}_{\mathrm{t}} = \mathbf{U}_{\mathrm{t}}^{\intercal}\mathbf{v}_0 + \mathbf{e}_{\mathrm{t}} &= \mathbf{T}^{\intercal}\mathbf{S}\mathbf{r} - ((\widetilde{\mathbf{G}}^{-1}(\mathbf{T}))^{\intercal} \otimes \mathbf{s}_{\mathrm{Q}}^{\intercal})\mathsf{col}(\mathbf{Q}) + \mathbf{e}_{\mathrm{t}} \\ &= \mathbf{T}^{\intercal}\mathbf{S}\mathbf{r} - \mathsf{col}(\mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q}\widetilde{\mathbf{G}}^{-1}(\mathbf{T})) + \mathbf{e}_{\mathrm{t}} = \left((\mathbf{S}\mathbf{r})^{\intercal}\mathbf{T} + \mathbf{e}_{\mathrm{t}}^{\intercal} - \mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q}\widetilde{\mathbf{G}}^{-1}(\mathbf{T})\right)^{\intercal},\end{aligned}$$

where the third equality follows from $\mathsf{col}(\mathbf{ABC}) = (\mathbf{C}^{\intercal} \otimes \mathbf{A})\mathsf{col}(\mathbf{B})$ and the last equality holds because $\mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q}\widetilde{\mathbf{G}}^{-1}(\mathbf{T})$ is a row vector. The reconstructed garbled table is hence

$$\mathbf{t}^{\intercal} = \underbrace{(\mathbf{S}\mathbf{r})^{\intercal}\mathbf{T} + \mathbf{e}_{\mathrm{t}}^{\intercal} - \mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q}\widetilde{\mathbf{G}}^{-1}(\mathbf{T})}_{\mathsf{sh}_{\mathrm{t}}^{\intercal}} + \underbrace{\mathbf{s}_{\mathrm{Q}}^{\intercal}\mathbf{Q}\widetilde{\mathbf{G}}^{-1}(\mathbf{T}) + \mathbf{e}_{\mathrm{sr}}^{\intercal}\mathbf{T} + \mathbf{e}_{\mathrm{Q}}^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{T})}_{\mathsf{sh}_{\mathrm{Q}}^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{T})} = (\mathbf{S}\mathbf{r} + \mathbf{e}_{\mathrm{sr}})^{\intercal}\mathbf{T} + \widetilde{\mathbf{e}}_{\mathrm{t}}^{\intercal},$$

where $\widetilde{\mathbf{e}}_{\mathrm{t}} = \mathbf{e}_{\mathrm{t}} + (\widetilde{\mathbf{G}}^{-1}(\mathbf{T}))^{\intercal}\mathbf{e}_{\mathrm{Q}}$ is $(nK+1)B_{\mathsf{IPFE}}$-bounded. Similarly,

$$\begin{aligned} w_{\mathrm{msg}} &= (\mathbf{S}\mathbf{r} + \mathbf{e}_{\mathrm{sr}})^{\intercal}\mathbf{w}_{\mathrm{out}} + \mu s_{\mathrm{msg}} + \widetilde{e}_{\mathrm{out}}, \\ \mathbf{w}_{\ell}^{\intercal} &= (\mathbf{S}\mathbf{r} + \mathbf{e}_{\mathrm{sr}})^{\intercal}(\mathbf{W}_{\ell,0} + \mathbf{x}[\ell]\mathbf{W}_{\ell,\times}) + \widetilde{\mathbf{e}}_{\ell}^{\intercal} \qquad \text{for all } \ell \in [L],\end{aligned}$$

with $\widetilde{e}_{\mathrm{out}}, \widetilde{\mathbf{e}}_{\ell}$'s bounded by $(nK+1)B_{\mathsf{IPFE}}$. The garbling uses secret $\mathbf{S}\mathbf{r} + \mathbf{e}_{\mathrm{sr}} \in \mathbb{Z}_q^n$ and its noises ($\widetilde{\mathbf{e}}$'s) are bounded by $(nK+1)B_{\mathsf{IPFE}}(q,K,Z) \le B_{\mathrm{in}}(\mathsf{param}')$. Therefore, by the correctness of NLG,

$$w_{\mathrm{out}} - ((\mathbf{S}\mathbf{r} + \mathbf{e}_{\mathrm{sr}})^{\intercal}\mathbf{w}_{\mathrm{out}} + e'_{\mathrm{out}}) \in [-B_{\mathrm{out}}(\mathsf{param}'), B_{\mathrm{out}}(\mathsf{param}')] \subseteq [-2^{-\kappa}q, 2^{-\kappa}q].$$

If $\mu = 0$, it always holds that $w_{\mathrm{msg}} - w_{\mathrm{out}} \in [-2^{-\kappa}q, 2^{-\kappa}q]$, which never holds when $\mu = 1$ by our choice of s_{msg}. We conclude that Construction 7.10 is correct. □

7.9.3 Security and Summary

Theorem 7.28 (¶). *Construction 7.10 is* very selectively secure *(Definition 2.3) if*

- $\mathsf{LWE}_{n,\mathrm{poly}(\lambda),q,\sigma}$ *(Assumption 2.2) holds for some* $\sigma \leq \frac{2^{-\kappa-6}\sigma_{\mathsf{IPFE}}}{nK\sqrt{\kappa}}$*, and*
- *it holds that* $2^{\kappa+6}B_{\mathsf{NLG}}(\mathsf{param}') \leq \sigma_{\mathsf{IPFE}}$.

Proof (Theorem 7.28). Let $\mathcal{A}$ be an efficient adversary and assume that it always chooses challenges such that $f(\mathbf{x}_j) = 0$ for all $j \in [J]$. Consider the following evasive IPFE sampler $\mathcal{S}^\beta = (\mathcal{S}_{\mathrm{V}}, \mathcal{S}^\beta_{\mathrm{U}})$ per Definition 7.2.

- $\mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}})$ parses $r_{\mathrm{pub}} = (r_{\mathcal{A}}, r_{\mathsf{NLG.Setup}}, r_{\mathrm{rq}}, r_{\mathsf{NLG.GenF}})$. It runs $\mathcal{A}(r_{\mathcal{A}})$ to obtain

$$\mathsf{param} = (\mathsf{param}', 1^L), \qquad \{\mathbf{x}_j\}_{j\in[J]} \quad (\mathbf{x}_j \in \{0,1\}^L), \qquad f : \mathbb{Z}^L \to \{0,1,\perp\}.$$

  The algorithm next sets up the noisy linear garbling by

$$\mathsf{pp} = (q, 1^n, \dots) \leftarrow \mathsf{NLG.Setup}(\mathsf{param}'; r_{\mathsf{NLG.Setup}}),$$

  and computes K, Z as in Setup of Construction 7.10. It then uses r_{rq} to sample $\mathbf{r}_1, \dots, \mathbf{r}_J \in \mathbb{Z}_q^n$, $\mathbf{Q}_1, \dots, \mathbf{Q}_J \in \mathbb{Z}_q^{n\times nK}$ uniformly at random in a straight-forward way.[94] The algorithm sets for all $j \in [J]$,

$$\mathbf{V}_{\mathrm{Q},j} = \begin{pmatrix} \mathbf{Q}_j \\ \mathbf{0}_{(Z-n)\times nK} \end{pmatrix}, \quad \mathbf{v}_{j,0} = \begin{pmatrix} \mathbf{r}_j \\ \mathsf{col}(\mathbf{Q}_j) \\ 1 \\ \mathbf{0}_{n-1+n^2K} \end{pmatrix}, \quad \mathbf{v}_{j,\ell} = \begin{pmatrix} (1-\mathbf{x}_j[\ell])\mathbf{r}_j \\ (1-\mathbf{x}_j[\ell])\mathsf{col}(\mathbf{Q}_j) \\ \mathbf{x}_j[\ell]\mathbf{r}_j \\ \mathbf{x}_j[\ell]\mathsf{col}(\mathbf{Q}_j) \end{pmatrix} \text{ for all } \ell \in [L].$$

  It runs

$$(1^n, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}, \mathbf{R}) \leftarrow \mathsf{NLG.GenF}(\mathsf{pp}, f; r_{\mathsf{NLG.GenF}}).$$

  Suppose $\mathbf{W}_{\ell,0} \in \mathbb{Z}_q^{n\times m_\ell}$ and $\mathbf{T} \in \mathbb{Z}_q^{n\times m_{\mathrm{t}}}$, the algorithm outputs

$$\begin{aligned} &q, 1^K, 1^Z, && \mathsf{ID} = [-1, L], && 1^{J_{\mathsf{id}}} = 1^J, \\ &1^{I_{-1,\mathfrak{o}}} = 1^0, && 1^{I_{0,\mathfrak{o}}} = 1^{m_{\mathrm{t}}+1}, && 1^{I_{\ell,\mathfrak{o}}} = 1^{m_\ell}, \\ &1^{I_{-1,\mathfrak{g}}} = 1^1, && 1^{I_{0,\mathfrak{g}}} = 1^{I_{\ell,\mathfrak{g}}} = 1^0, && \sigma_{\mathrm{pre}} = \sigma_{\mathsf{IPFE}}, \end{aligned}$$

[94] Precisely speaking, r_{rq} must be efficiently sampleable conditioned on and given $\mathbf{r}_1, \dots, \mathbf{r}_J$ and $\mathbf{Q}_1, \dots, \mathbf{Q}_J$.

$$\mathbf{v}_{-1,\mathfrak{g}} = (\mathbf{v}_{\mathrm{Q},1}, \ldots, \mathbf{v}_{\mathrm{Q},J}), \qquad \mathbf{v}_{-1,\mathfrak{o}} = \mathbf{0},$$
$$\mathbf{V}_{0,\mathfrak{o}} = (\mathbf{v}_{0,0}, \ldots, \mathbf{v}_{J,0}), \qquad \mathbf{V}_{0,\mathfrak{g}} = \mathbf{0},$$
$$\mathbf{V}_{\ell,\mathfrak{o}} = (\mathbf{v}_{1,\ell}, \ldots, \mathbf{v}_{J,\ell}), \qquad \mathbf{V}_{\ell,\mathfrak{g}} = \mathbf{0},$$

where the indices are $\mathsf{id} \in \mathsf{ID}$ and $\ell \in [L]$.

- $\mathcal{S}_{\mathrm{U}}^{\beta}(r_{\mathrm{pub}}; r_{\mathrm{priv}})$ first runs the deterministic subprocedure $\mathcal{S}_{\mathrm{A0}}^{\beta}(r_{\mathrm{pub}})$, which does the following. It first reruns $\mathcal{S}_{\mathrm{V}}(r_{\mathrm{pub}})$ to obtain $\mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}, \mathbf{T}$. The algorithm then rearranges them into $\mathbf{A}_{\mathrm{Q}}, \mathbf{A}_{\mathrm{msg}}, \{\mathbf{A}_{\mathrm{t},i}\}_{i\in[m_{\mathrm{t}}]}, \{\mathbf{A}_{\ell,i}\}_{\ell\in[L],i\in[m_\ell]}$ such that for all $\mathbf{s}_{\mathrm{Q}}, \widetilde{\mathbf{s}}_1, \ldots, \widetilde{\mathbf{s}}_n \in \mathbb{Z}_q^n$ and $s_{\mathrm{msg}} \in \mathbb{Z}_q$,

$$\begin{aligned}
\mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathrm{Q}} &= \mathbf{u}_{\mathrm{Q}}^{\mathsf{T}} = \big(\mathbf{s}_{\mathrm{Q}}^{\mathsf{T}}, \mathbf{0}_{1\times(Z-n)}\big),\\
\mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathrm{msg}} &= \mathbf{u}_{\mathrm{msg}}^{\mathsf{T}} = \big(\mathbf{w}_{\mathrm{out}}^{\mathsf{T}}\mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}) \otimes \mathbf{s}_{\mathrm{Q}})^{\mathsf{T}}, \beta s_{\mathrm{msg}}, \mathbf{0}_{1\times(n-1+n^2K)}\big),\\
(\mathbf{A}_{\mathrm{t},1}^{\mathsf{T}}\mathbf{d}_0, \ldots, \mathbf{A}_{\mathrm{t},m_{\mathrm{t}}}^{\mathsf{T}}\mathbf{d}_0)^{\mathsf{T}} &= \mathbf{U}_{\mathrm{t}}^{\mathsf{T}} = \big(\mathbf{T}^{\mathsf{T}}\mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{T}) \otimes \mathbf{s}_{\mathrm{Q}})^{\mathsf{T}}, \mathbf{0}_{m_{\mathrm{t}}\times(n+n^2K)}\big),\\
(\mathbf{A}_{\ell,1}^{\mathsf{T}}\mathbf{d}_0, \ldots, \mathbf{A}_{\ell,m_\ell}^{\mathsf{T}}\mathbf{d}_0)^{\mathsf{T}} &= \mathbf{U}_{\ell}^{\mathsf{T}}\\
&= \big(\mathbf{W}_{\ell,0}^{\mathsf{T}}\mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0}) \otimes \mathbf{s}_{\mathrm{Q}})^{\mathsf{T}}, (\mathbf{W}_{\ell,0}^{\mathsf{T}} + \mathbf{W}_{\ell,\times}^{\mathsf{T}})\mathbf{S}, -(\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{W}_{\ell,\times}) \otimes \mathbf{s}_{\mathrm{Q}})^{\mathsf{T}}\big),
\end{aligned}$$

where $\ell \in [L]$ and $\mathbf{d}_0^{\mathsf{T}} = (s_{\mathrm{msg}}, \mathbf{s}_{\mathrm{Q}}^{\mathsf{T}}, \widetilde{\mathbf{s}}_1^{\mathsf{T}}, \ldots, \widetilde{\mathbf{s}}_n^{\mathsf{T}})$ and $\mathbf{S} = (\widetilde{\mathbf{s}}_1, \ldots, \widetilde{\mathbf{s}}_n)$. This is possible since every entry on the right-hand side is linear in $\mathbf{d}_0$. The algorithm $\mathcal{S}_{\mathrm{A0}}$ outputs

$$\begin{aligned}
1^{n'} &= 1^{n(n+1)+1}, & \mathbf{A}_{-1,\mathfrak{g},0,1} &= \mathbf{A}_{\mathrm{Q}},\\
\mathbf{A}_{0,\mathfrak{o},0,m_{\mathrm{t}}+1} &= \mathbf{A}_{\mathrm{msg}}, & \mathbf{A}_{0,\mathfrak{o},0,i} &= \mathbf{A}_{\mathrm{t},i} && \text{(for all } i \in [m_{\mathrm{t}}]),\\
& & \mathbf{A}_{\ell,\mathfrak{o},0,i} &= \mathbf{A}_{\ell,i} && \text{(for all } \ell \in [L],\ i \in [m_\ell]).
\end{aligned}$$

Completing $\mathcal{S}_{\mathrm{A0}}$, the algorithm $\mathcal{S}_{\mathrm{U}}$ samples $\mathbf{d}_0 \xleftarrow{\$} \mathbb{Z}_q^{n'}$ using r_{priv} and outputs

$$\mathbf{U}_{\mathsf{id},\mathfrak{s}}^{\mathsf{T}} = \begin{pmatrix} \mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathsf{id},\mathfrak{s},0,1} \\ \vdots \\ \mathbf{d}_0^{\mathsf{T}}\mathbf{A}_{\mathsf{id},\mathfrak{s},0,I_{\mathsf{id},\mathfrak{s}}} \end{pmatrix} \qquad \text{for all } \mathsf{id} \in \mathsf{ID}, \mathfrak{s} \in \{\mathfrak{o}, \mathfrak{g}\}.$$

By definition, $\mathcal{S}^{\beta}$ is a restricted sampler (Definition 7.3). We have the following:

Claim 7.29 (¶). $\mathcal{S}^{\beta}$ has pseudorandom structurally noisy inner products, i.e., $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}^{\beta}}$ *(Definition 7.2) holds for both* $\beta \in \{0, 1\}$.

The very selective security of Construction 7.10 follows from Claim 7.29. Consider the following hybrids.

- H_0^β is $\mathsf{Exp}_{\mathrm{ABE}}^\beta$ for Construction 7.10. Note that s_{msg} in the challenge ciphertext is uniformly random over $\mathbb{Z}_q \setminus [-2 \cdot 2^{-\kappa} q, 2 \cdot 2^{-\kappa} q]$.
- In H_1^β, the scalar s_{msg} is changed to be uniformly random over $\mathbb{Z}_q$. Clearly, $\mathsf{H}_0^\beta \approx_{\mathrm{s}} \mathsf{H}_1^\beta$. Moreover, H_1^β corresponds to the left distribution of $\mathsf{IPFEsec}_{\mathrm{post}}^{\mathcal{S}^\beta}$.
- In H_2^β, the vectors inside IPFE ciphertexts (components of the ABE challenge ciphertext) are replaced by random, which corresponds to the right distribution of $\mathsf{IPFEsec}_{\mathrm{post}}^{\mathcal{S}^\beta}$. By Claim 7.29 and the restricted-σ_{IPFE}-security of IPFE, we have $\mathsf{H}_1^\beta \approx \mathsf{H}_2^\beta$.

Lastly, $\mathsf{H}_2^0 \equiv \mathsf{H}_2^1$. By hybrid argument, we conclude $\mathsf{Exp}_{\mathrm{ABE}}^0 \approx \mathsf{Exp}_{\mathrm{ABE}}^1$, completing the proof. □

Proof (Claim 7.29). Fix $\beta \in \{0,1\}$ and consider the following hybrids.

- H_0 is the left distribution of $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}^\beta}$. Recall that the public randomness is $r_{\mathrm{pub}} = (r_{\mathcal{A}}, r_{\mathsf{NLG.Setup}}, r_{\mathrm{rq}}, r_{\mathsf{NLG.GenF}})$. Adopting the notations of the ABE decryption algorithm, the noisy inner products are

$$\begin{aligned}
\mathsf{sh}_{\mathrm{Q},j}^\intercal &= \mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j + \mathbf{e}_{\mathrm{sr},j}^\intercal \widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q},j}^\intercal, \\
\mathsf{sh}_{\mathrm{msg},j} &= (\mathbf{S}\mathbf{r}_j)^\intercal \mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - \mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j \widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}), \\
\mathsf{sh}_{\mathrm{t},j}^\intercal &= (\mathbf{S}\mathbf{r}_j)^\intercal \mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal - \mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j \widetilde{\mathbf{G}}^{-1}(\mathbf{T}), \\
\mathsf{sh}_{j,\ell}^\intercal &= (\mathbf{S}\mathbf{r}_j)^\intercal (\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^\intercal - \mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j \widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) \quad (\ell \in [L]),
\end{aligned}$$

  where $j \in [J]$ and the entries of $\mathbf{e}$'s are independent $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{IPFE}}}$.

- In H_1, additional small noises are attached. For all $j \in [J]$, the inner products become

$$\begin{aligned}
\mathsf{sh}_{\mathrm{Q},j}^\intercal &= \mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j}^\intercal + \widetilde{\mathbf{e}}_{\mathrm{sr},j}^\intercal \widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{sr},j}^\intercal \widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q},j}^\intercal, \\
\mathsf{sh}_{\mathrm{msg},j} &= (\mathbf{S}\mathbf{r}_j)^\intercal \mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - (\mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j}^\intercal)\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}), \\
\mathsf{sh}_{\mathrm{t},j}^\intercal &= (\mathbf{S}\mathbf{r}_j)^\intercal \mathbf{T} + \mathbf{e}_{\mathrm{t},j}^\intercal - (\mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j}^\intercal)\widetilde{\mathbf{G}}^{-1}(\mathbf{T}), \\
\mathsf{sh}_{j,\ell}^\intercal &= (\mathbf{S}\mathbf{r}_j)^\intercal (\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^\intercal - (\mathbf{s}_{\mathrm{Q}}^\intercal \mathbf{Q}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j}^\intercal)\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}),
\end{aligned}$$

where $\ell \in [L]$ and the entries of $\widetilde{\mathbf{e}}$ are independent $\mathcal{D}_{\mathbb{Z},\sigma,\le\sigma\sqrt{\kappa}}$. In $\mathsf{sh}_{\mathrm{Q},j}$'s, the $\widetilde{\mathbf{e}}_{\mathrm{Q},j}$'s and $\widetilde{\mathbf{e}}_{\mathrm{sr},j}$'s are flooded by $\mathbf{e}_{\mathrm{Q},j}$'s and $\mathbf{e}_{\mathrm{sr},j}$'s, respectively. In the other inner products, each $\widetilde{\mathbf{e}}_{\mathrm{Q},j}^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\cdots)$ is flooded by the e or $\mathbf{e}$ in that term. Since

$$\begin{aligned}
&\text{(for } \mathsf{sh}_{\mathrm{Q},j}\text{'s)} && \sigma\sqrt{\kappa} \le \frac{2^{-\kappa-6}\sigma_{\mathsf{IPFE}}}{nK} && \le 2^{-\kappa-6}\sigma_{\mathsf{IPFE}},\\
&\text{(for the rest)} && nK\cdot\sigma\sqrt{\kappa} \le nK\cdot\frac{2^{-\kappa-6}\sigma_{\mathsf{IPFE}}}{nK} && = 2^{-\kappa-6}\sigma_{\mathsf{IPFE}},
\end{aligned}$$

Lemma 2.2 applies and $\mathsf{H}_0 \approx_{\mathrm{s}} \mathsf{H}_1$.

- In H_2, the $\widetilde{\mathbf{e}}$'s are no longer truncated, i.e., their entries follow $\mathcal{D}_{\mathbb{Z},\sigma}$. We have $\mathsf{H}_1 \approx_{\mathrm{s}} \mathsf{H}_2$ by Lemma 2.1.

- In H_3, the LWE samples $(\mathbf{s}_{\mathrm{Q}}^{\mathsf{T}}\mathbf{Q}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j}^{\mathsf{T}})$ are replaced by random. For all $j \in [J]$, sample $\mathbf{q}_j \xleftarrow{\$} \mathbb{Z}_q^n$ and the inner products are

$$\begin{aligned}
\mathsf{sh}_{\mathrm{Q},j}^{\mathsf{T}} &= \mathbf{q}_j^{\mathsf{T}} + \widetilde{\mathbf{e}}_{\mathrm{sr},j}^{\mathsf{T}}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{sr},j}^{\mathsf{T}}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q},j}^{\mathsf{T}},\\
\mathsf{sh}_{\mathrm{msg},j} &= (\mathbf{S}\mathbf{r}_j)^{\mathsf{T}}\mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - \mathbf{q}_j^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}),\\
\mathsf{sh}_{\mathrm{t},j}^{\mathsf{T}} &= (\mathbf{S}\mathbf{r}_j)^{\mathsf{T}}\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^{\mathsf{T}} - \mathbf{q}_j^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\mathbf{T}),\\
\mathsf{sh}_{j,\ell}^{\mathsf{T}} &= (\mathbf{S}\mathbf{r}_j)^{\mathsf{T}}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^{\mathsf{T}} - \mathbf{q}_j^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) \quad (\ell \in [L]).
\end{aligned}$$

By $\mathsf{LWE}_{n,nJ,q,\sigma}$, we have $\mathsf{H}_2 \approx \mathsf{H}_3$. In this step, we rely on $\mathbf{Q}$'s sampling being "straight-forward" (for reduction to sample r_{rq} conditioned on and given $\mathbf{Q}$'s).

- In H_4, we perform a change of variable. For all $j \in [J]$, sample $\widetilde{\mathbf{q}}_j \xleftarrow{\$} \mathbb{Z}_q^n$ and set $\mathbf{q}_j^{\mathsf{T}} = \widetilde{\mathbf{q}}_j^{\mathsf{T}} - \widetilde{\mathbf{e}}_{\mathrm{Q},j}^{\mathsf{T}}\widetilde{\mathbf{G}}$, then the inner products are

$$\begin{aligned}
\mathsf{sh}_{\mathrm{Q},j}^{\mathsf{T}} &= \widetilde{\mathbf{q}}_j^{\mathsf{T}} + \mathbf{e}_{\mathrm{sr},j}^{\mathsf{T}}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q},j}^{\mathsf{T}},\\
\mathsf{sh}_{\mathrm{msg},j} &= (\mathbf{S}\mathbf{r}_j)^{\mathsf{T}}\mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - (\widetilde{\mathbf{q}}_j^{\mathsf{T}} - \widetilde{\mathbf{e}}_{\mathrm{Q},j}^{\mathsf{T}}\widetilde{\mathbf{G}})\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}})\\
&= (\mathbf{S}\mathbf{r}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j})^{\mathsf{T}}\mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - \widetilde{\mathbf{q}}_j^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}),\\
\mathsf{sh}_{\mathrm{t},j}^{\mathsf{T}} &= (\mathbf{S}\mathbf{r}_j)^{\mathsf{T}}\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^{\mathsf{T}} - (\widetilde{\mathbf{q}}_j^{\mathsf{T}} - \widetilde{\mathbf{e}}_{\mathrm{Q},j}^{\mathsf{T}}\widetilde{\mathbf{G}})\widetilde{\mathbf{G}}^{-1}(\mathbf{T})\\
&= (\mathbf{S}\mathbf{r}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j})^{\mathsf{T}}\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^{\mathsf{T}} - \widetilde{\mathbf{q}}_j^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\mathbf{T}),\\
\mathsf{sh}_{j,\ell}^{\mathsf{T}} &= (\mathbf{S}\mathbf{r}_j)^{\mathsf{T}}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^{\mathsf{T}} - (\widetilde{\mathbf{q}}_j^{\mathsf{T}} - \widetilde{\mathbf{e}}_{\mathrm{Q},j}^{\mathsf{T}}\widetilde{\mathbf{G}})\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times})\\
&= (\mathbf{S}\mathbf{r}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j})^{\mathsf{T}}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^{\mathsf{T}} - \widetilde{\mathbf{q}}_j^{\mathsf{T}}\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}),
\end{aligned}$$

where $\ell \in [L]$. Clearly, $\mathsf{H}_3 \equiv \mathsf{H}_4$.

- In H_5, the LWE samples $(\mathbf{S}\mathbf{r}_j + \widetilde{\mathbf{e}}_{\mathrm{Q},j})$ are replaced by random. For all $j \in [J]$, sample $\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_q^n$ and the inner products are

$$\begin{aligned}
\mathsf{sh}_{\mathrm{Q},j}^{\intercal} &= \widetilde{\mathbf{q}}_j^{\intercal} + \mathbf{e}_{\mathrm{sr},j}^{\intercal}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q},j}^{\intercal},\\
\mathsf{sh}_{\mathrm{msg},j} &= \mathbf{s}_j^{\intercal}\mathbf{w}_{\mathrm{out}} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - \widetilde{\mathbf{q}}_j^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}),\\
\mathsf{sh}_{\mathrm{t},j}^{\intercal} &= \mathbf{s}_j^{\intercal}\mathbf{T} + \mathbf{e}_{\mathrm{t},j}^{\intercal} - \widetilde{\mathbf{q}}_j^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{T}),\\
\mathsf{sh}_{j,\ell}^{\intercal} &= \mathbf{s}_j^{\intercal}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \mathbf{e}_{j,\ell}^{\intercal} - \widetilde{\mathbf{q}}_j^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) \qquad (\ell \in [L]).
\end{aligned}$$

  By n hybrids of $\mathsf{LWE}_{n,J,q,\sigma}$ over the rows of $\mathbf{S}$ with public matrix $(\mathbf{r}_1, \ldots, \mathbf{r}_J)$, we have $\mathsf{H}_4 \approx \mathsf{H}_5$.

- In H_6, additional noises generated by GenNoise are attached to the garblings. For all $j \in [J]$, run

$$(\overline{e}_{\mathrm{out},j}, \{\overline{\mathbf{e}}_{j,\ell}\}_{\ell\in[L]}, \overline{\mathbf{e}}_{\mathrm{t},j}) \xleftarrow{\$} \mathsf{GenNoise}(\mathsf{param}', \mathsf{pp}, f, \mathbf{w}_{\mathrm{out}}, \{\mathbf{W}_{\ell,0}, \mathbf{W}_{\ell,\times}\}_{\ell\in[L]}\mathbf{T}, \mathbf{R}, \mathbf{x}_j),$$

  and set the inner products to

$$\begin{aligned}
\mathsf{sh}_{\mathrm{Q},j}^{\intercal} &= \widetilde{\mathbf{q}}_j^{\intercal} + \mathbf{e}_{\mathrm{sr},j}^{\intercal}\widetilde{\mathbf{G}} + \mathbf{e}_{\mathrm{Q},j}^{\intercal},\\
\mathsf{sh}_{\mathrm{msg},j} &= \mathbf{s}_j^{\intercal}\mathbf{w}_{\mathrm{out}} + \overline{e}_{\mathrm{out},j} + e_{\mathrm{out},j} + \beta s_{\mathrm{msg}} - \widetilde{\mathbf{q}}_j^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{w}_{\mathrm{out}}),\\
\mathsf{sh}_{\mathrm{t},j}^{\intercal} &= \mathbf{s}_j^{\intercal}\mathbf{T} + \overline{\mathbf{e}}_{\mathrm{t},j}^{\intercal} + \mathbf{e}_{\mathrm{t},j}^{\intercal} - \widetilde{\mathbf{q}}_j^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{T}),\\
\mathsf{sh}_{j,\ell}^{\intercal} &= \mathbf{s}_j^{\intercal}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) + \overline{\mathbf{e}}_{j,\ell}^{\intercal} + \mathbf{e}_{j,\ell}^{\intercal} - \widetilde{\mathbf{q}}_j^{\intercal}\widetilde{\mathbf{G}}^{-1}(\mathbf{W}_{\ell,0} + \mathbf{x}_j[\ell]\mathbf{W}_{\ell,\times}) \quad (\ell \in [L]).
\end{aligned}$$

  The $\overline{e}, \overline{\mathbf{e}}$'s introduced into the inner products are bounded by $B_{\mathsf{NLG}} \le 2^{-\kappa-6}\sigma_{\mathsf{IPFE}}$, so they are flooded (Lemma 2.2) by $e, \mathbf{e}$'s. We have $\mathsf{H}_5 \approx_{\mathrm{s}} \mathsf{H}_6$.

- In H_7, all the inner products are replaced by random, i.e., the right distribution of $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}^\beta}$. Note that in H_6, non-$\mathsf{sh}_{\mathrm{Q},j}$'s are well-randomized garblings (by $\mathbf{s}, \overline{e}, \overline{\mathbf{e}}$'s) plus independent noises and values derived from $\mathsf{sh}_{\mathrm{Q},j}$. Therefore, by J hybrids of applying GenNoise-security of NLG, the non-$\mathsf{sh}_{\mathrm{Q},j}$'s can be replaced by random, after which $\widetilde{\mathbf{q}}_j$'s ensure $\mathsf{sh}_{\mathrm{Q},j}$'s are independent random. Therefore, $\mathsf{H}_6 \approx \mathsf{H}_7$.

By hybrid argument, $\mathsf{H}_0 \approx \mathsf{H}_7$, i.e., $\mathsf{IPFEsec}_{\mathrm{pre}}^{\mathcal{S}^\beta}$ holds. □

Summary. Combining Corollary 7.10, Construction 7.9, and Construction 7.10, we obtain a CP-ABE scheme for Boolean circuits with succinct ciphertexts.

Corollary 7.30. *Under LWE (Assumption 2.2) and evasive learning with structured errors (Assumption 7.1), there exists a CP-ABE scheme for Boolean circuits with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda, d), \qquad |\mathsf{sk}_{\mathbf{x}}| = (L+1)\,\mathrm{poly}(\lambda, d), \qquad |\mathsf{ct}_C| = (L+1)\,\mathrm{poly}(\lambda, d),$$

where $L = |\mathbf{x}|$ is the attribute length and d is the maximum depth.

7.10 Further Discussions

Here are further comments around the research of this chapter.

Evasive LWE as Unstucker/Pathway. I have not mastered lattice techniques to heart and often feel stuck when trying to prove the security of lattice-based constructions based on standard lattice techniques. In general, the community has been stuck on lattice-based ABE for a while. Take CP-ABE for example, the work of [BV22] was published in 2022, first preprinted in 2020, although its scheme has been circulating since 2015 [BV20; Section 1.1] — that is one year after [BGG+14] then seven years without an attack nor a security proof.

In between, there were LWE-based schemes for limited class of functions [Tsa19] (with adaptive security). The needle started to move following [AY20,AWY20] (see Chapter 6 on this line), which in addition relied on pairing. The formulation of evasive LWE [Wee22,Tsa22] greatly improved the momentum, and there have been a flurry of new results — see the many works cited in this chapter and Chapter 8, as well as the two chapters themselves.

Given the "generic-model" view of evasive LWE, I regard it as a pathway to unstuck progress, by providing *some* security proofs for constructions that are previously stuck due to a lack of security proof. As excited as I am to see the new ideas and results, it is also important to research methods of bringing those ideas into schemes provable under falsifiable assumptions. There has been some effort in this direction, including [CW23,CW24,Wee24,AMR25a,Wee25].

Part II

More

Chapter 8

ABE for Circuits of Unbounded Depth from Lattices

This chapter is an adaptation of [HLL23b,HLL23a,HLL25].[95]

In this chapter, we push the frontier of ABE in the realm of (or at least close to) implementable cryptography. The structure and hardness of integer lattices has been a rare gem fueling many powerful cryptography primitives, notably fully homomorphic encryption (FHE) supporting all circuits, without bounds on size or depth. It can also be used to construct ABE supporting polynomial-size circuits, but the known constructions require setting a polynomial upper bound on the maximum circuit depth upon initializing the scheme. Despite the structural connections between lattice-based FHE and ABE schemes, removing the depth bound has remained elusive for almost fifteen years. We construct the first ABE supporting circuits of unbounded depth (and size) from lattices.

8.1 Introduction

Over the past few decades, there has been remarkable progress in developing a diverse array of homomorphic primitives. They enable computations involving secrets in a non-interactive and reusable manner, all while ensuring data privacy and/or integrity. This advancement has realized long-sought-after primitives such as fully homomorphic encryption (FHE) [RAD78,Gen09], attribute-based encryption (ABE) [SW05,GPSW06], functional encryption (FE) [SW05,GPSW06,O'N10,BSW11], and indistinguishability

[95] Yao-Ching Hsieh, Huijia Lin, Ji Luo:
Attribute-Based Encryption for Circuits of Unbounded Depth from Lattices,
© 2023 IEEE, DOI 10.1109/FOCS57990.2023.00031;
Attribute-Based Encryption for Circuits of Unbounded Depth from Lattices: Garbled Circuits of Optimal Size, Laconic Functional Evaluation, and More,
© 2025 SIAM, DOI 10.1137/24M1629316.

obfuscation ($i\mathcal{O}$) [DH76,BGI+01]. Additionally, a plethora of intriguing concepts have emerged, including laconic function evaluation (LFE) [QWW18], reusable garbling [GKP+13b], homomorphic commitments and signatures [BF11,GVW15b], constrained pseudorandom functions [BW13], among others.

A central research objective is to develop homomorphic primitives supporting all *unrestricted* polynomial-size computations, with no predetermined bound on the parameters such as description size, input length, and depth. Notable successes in this pursuit include the development of FHE from circular security assumptions [Gen09, BV11], as well as FE and $i\mathcal{O}$ from well-studied assumptions [JLS21,JLS22]. These constructions accommodate unrestricted polynomial-size circuits. Since $i\mathcal{O}$ serves as a powerful tool for achieving other cryptographic goals, the latter results imply the feasibility of FHE [CLTV15] and ABE (see [GGH+13a] and Chapter 9) for unrestricted polynomial-size computations, and potentially other homomorphic primitives.

However, beyond these two successes, progress has stagnated, particularly concerning ABE and related primitives. ABE provides fined-grained access control for encrypted data, allowing data owners to encrypt data tied to public attributes x. Users hold partial decryption keys linked to various access policies f, ensuring that a ciphertext can only be decrypted by keys with matching policy, i.e., $f(x) = 1$. The state-of-the-art ABE construction based on the hardness of learning with errors (LWE) by Boneh *et al.* [BGG+14], following the initial work by Gorbunov, Vaikuntanathan, and Wee [GVW13], only supports circuits up to *predetermined* (polynomially bounded) depth, akin to leveled homomorphic encryption. This depth dependency presents itself in two ways. Functionally, it requires fixing a bound d on the maximum computation depth when generating the master public key, limiting subsequent computations to depths below d. In terms of efficiency, the scheme's components — master public keys, secret keys, and ciphertexts — grow in length polynomial in d. Such dependency has been inherited by other homomorphic primitives that employ techniques developed for ABE, including predicate encryption [GVW15a] (PE, enhanced from ABE by further hiding the attribute in the non-decrypting case), homomorphic signatures [GVW15b], and constrained pseudorandom functions [BV15]. For ABE, Chapter 6 and the work

of [CW23] reduced the size of secret keys to be independent of d, and the works of [GKW16,BV16,CW24] made the schemes unbounded in input length, but they did not support unbounded depth, unlike the bootstrapping of FHE.

Even in the simpler scenario with just one secret key published, there is no known solution to overcome the depth limitation. Hence, we only have 1-key ABE for bounded-depth circuits, and similarly for LFE [QWW18], 1-key FE [SS10,GVW12,GKP+13b], and reusable garbling [GKP+13b].

In summary, without using $i\mathcal{O}$, we are limited to *leveled* versions of ABE and related homomorphic primitives. However, these primitives are intuitively closer to homomorphic encryption in terms of capabilities and techniques than to $i\mathcal{O}$. Gentry [Gen09] introduced the bootstrapping technique based on circular security assumptions to remove depth dependency in FHE. However, after nearly *fifteen* years, there is no equivalent of bootstrapping for ABE, even when considering circular security assumptions and other lattice-based assumptions. In this chapter, we aim to fill the gap and give direct lattice-based constructions of unbounded depth ABE, PE, LFE, 1-key FE, and reusable garbling. Such direct constructions without relying on $i\mathcal{O}$ yield simpler, more efficient, and post-quantum secure schemes. (There is no known candidate of post-quantum $i\mathcal{O}$.)

8.1.1 Our Results

We obtain the first lattice-based constructions of ABE and several related primitives supporting circuits of unbounded depth (and size). Our results come in two versions. In the single-key setting, we achieve 1-key ABE, LFE, 1-key FE, and compact reusable garbling schemes, based on a circular security assumption. In the multi-key setting, we achieve full-fledged ABE (secure against unbounded collusion) and PE by leveraging a new *evasive circular* LWE assumption, which is a variant of the recently proposed *evasive* LWE assumptions [Wee22,Tsa22]. Notably, our LFE, 1-key FE, and reusable garbling schemes enjoy almost optimal succinctness (up to polynomial factors in the security parameter). Their input encoding/ciphertexts scale in length linear in the input length, while function encoding/secret keys are of constant size (for Boolean-

output circuits), independent of circuit size or depth. In fact, we obtain the first garbled circuits where neither online nor offline size depends on the circuit size, without using $i\mathcal{O}$, irrespective of the reusability property. Our ABE and PE schemes have the same level of succinctness. Compared to prior constructions, our schemes eliminate the multiplicative overheads that grow polynomially with the maximum depth of the computations.

Only attribute-based LFE (AB-LFE) and full-fledged ABE are included in this dissertation. See Section 8.6 for discussion of the other results.

Depth-Unbounded AB-LFE from Circular LWE. Circular security assumptions postulate that LWE samples are pseudorandom even when they are used to encrypt the underlying secret vector. We rely on the following specific circular LWE assumption (Assumption 8.1):

$$\text{with} \qquad \mathbf{A}_{\mathsf{fhe}} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^{\mathsf{T}}\overline{\mathbf{A}}_{\mathsf{fhe}} \end{pmatrix}, \ \mathbf{S} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - \mathsf{bits}(\mathbf{r}^{\mathsf{T}}, -1) \otimes \mathbf{G},$$

$$\text{it holds that} \qquad \mathbf{A}_{\mathsf{fhe}}, \ \mathbf{S}, \ \overline{\mathbf{A}}', \ \mathbf{r}^{\mathsf{T}}\overline{\mathbf{A}}' \approx \$, \ \$, \ \$, \ \$,$$

where $\mathbf{r}$ consists of small Gaussian entries, $\mathbf{G}$ is the gadget matrix, $\overline{\mathbf{A}}_{\mathsf{fhe}} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\overline{\mathbf{A}}' \xleftarrow{\$} \mathbb{Z}_q^{n \times m'}$, $\mathbf{R} \xleftarrow{\$} \{0,1\}^{m \times m''}$ for some appropriate m, m'' and any polynomial m', and independent Gaussian noises are added to the terms with wavy underlines. Here, $\mathbf{A}_{\mathsf{fhe}}$ is a public key with the corresponding secret key $\mathbf{s} = (\mathbf{r}^{\mathsf{T}}, -1)^{\mathsf{T}}$ satisfying $\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{fhe}} = \underset{\sim}{\mathbf{0}}$, and $\mathbf{S}$ is a circular encryption of (the bits of) $\mathbf{s}$ under the secret key $\mathbf{s}$, both generated honestly using the Gentry–Sahai–Waters FHE scheme [GSW13]. The assumption states that this circular encryption together with the other LWE samples are jointly pseudorandom. It is almost identical to the circular security assumption needed for bootstrapping the [GSW13] FHE scheme, except here the secret key is small Gaussian instead of uniformly random. In the literature, circular security assumptions with small secrets have already been used, e.g., for bootstrapping the Brakerski–Gentry–Vaikuntanathan FHE scheme [BGV12].

Assuming the above circular LWE assumption, we construct depth-unbounded AB-LFE.

LACONIC FUNCTION EVALUATION. Introduced by Quach, Wee, and Wichs [QWW18], LFE enables one party, Alice, to compress a large circuit C into a short digest. Using this digest, Bob can encrypt any input $\mathbf{x}$ in a way that allows Alice to recover $C(\mathbf{x})$ without learning anything else about $\mathbf{x}$. In LFE, the digest size and the encryption time (consequently, the ciphertext) are small — much smaller than the circuit size. In AB-LFE, Bob encrypts message μ with public attribute $\mathbf{x}$, and Alice can recover μ if and only if $C(\mathbf{x}) = 0$. It is a useful primitive in secure computation, implying Bob-optimized 2-party function evaluation and MPC protocols with laconic online computation. The prior construction of LFE [QWW18] uses techniques developed in the constructions of ABE by [BGG+14] and 1-key FE of [GKP+13b], and inherits their depth dependency. The common reference string, circuit digests, and ciphertexts all grow in length polynomially in a bound on the computation depth (specified when generating the common reference string). Assuming the circular LWE assumption, we remove such dependency.

Construction 8.3 (AB-LFE). *Under the circular LWE assumption, there exists a very selectively secure*[96] *AB-LFE scheme for circuits of unbounded depth and bounded input/message lengths L, L' with*

$$|\mathsf{crs}| = \mathrm{O}(L), \qquad |\mathsf{digest}_C| = \mathrm{O}(L'), \qquad T_{\mathsf{Enc}} = \mathrm{O}(L + L'), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(L + L').$$

In the above statement and rest of this introduction, $\mathrm{O}(\cdot)$ hides $\mathrm{poly}(\lambda)$ factors.

Full-Fledged Depth-Unbounded ABE from Evasive Circular LWE. We also construct full-fledged depth-unbounded ABE assuming a stronger assumption, called the *evasive circular LWE assumption*, which incorporates circularity into the evasive LWE assumption of [Wee22] (more on that later). Our ABE is a direct construction.

Construction 8.4 (depth-unbounded ABE). *Under the evasive circular LWE assumption and the circular LWE assumption, there exists a very selectively secure ABE scheme for*

[96] In very selective security (including that for ABE later in the introduction), all adversarial choices (notably, functions and attributes) must be made independently of any information (notably, public parameters or master public key) about the scheme instance.

circuits of unbounded depth and bounded input length L with

$$|\mathsf{mpk}| = \mathrm{O}(L), \qquad |\mathsf{sk}_C| = \mathrm{O}(1), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(L).$$

The secret key size of our ABE scheme is constant, while the master public key and the ciphertexts are compact, of size linear in the maximum attribute length. We can further remove the predetermined bound L on attribute length by applying the generic transformation of [GKW16] to obtain an ABE scheme for, truly, all polynomial-size computations, at the price of increasing the secret key size to be linear in the input length of the function encoded.

Corollary 8.17 (attribute-unbounded depth-unbounded ABE). *Under the evasive circular LWE assumption and the circular LWE assumption, there also exists a very selectively secure ABE scheme for circuits of unbounded depth and input length with*

$$|\mathsf{mpk}| = \mathrm{O}(1), \qquad |\mathsf{sk}_C| = \mathrm{O}(L), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(|\mathbf{x}|),$$

where L is the input length of C in sk_C (not necessarily the same as $|\mathbf{x}|$).

The Evasive Circular LWE Assumption. We explain the evasive circular LWE assumption at a high level. The evasive LWE assumption [Wee22,Tsa22] asserts that LWE samples $(\mathbf{s}^\intercal\mathbf{B} + \mathbf{e}_{\mathrm{B}}^\intercal)$ remain secure even in the presence of low-norm trapdoors $\mathbf{B}^{-1}(\mathbf{P})$ mapping $\mathbf{B}$ to another (not necessarily random) matrix $\mathbf{P}$, provided that $(\mathbf{r}^\intercal\mathbf{B} + \mathbf{e}_{\mathrm{B}}^\intercal, \mathbf{r}^\intercal\mathbf{P} + \mathbf{e}_{\mathrm{P}}^\intercal)$ (with fresh random noises $\mathbf{e}_{\mathrm{P}}$) are jointly pseudorandom. A simple formal version is the following. Fix an efficiently sampleable joint distribution $\mathcal{S}$ of matrices $\overline{\mathbf{A}}', \mathbf{P}$ and auxiliary information $\mathsf{aux} \in \{0,1\}^\star$, the evasive LWE assumption postulates that

$$\begin{aligned} \text{if} \quad & ① : \mathbf{B},\ \overline{\mathbf{A}}',\ \mathbf{P},\ \mathbf{r}^\intercal\mathbf{B},\ \mathbf{r}^\intercal\overline{\mathbf{A}}',\ \mathbf{r}^\intercal\mathbf{P},\ \mathsf{aux} \approx ② : \mathbf{B},\ \overline{\mathbf{A}}',\ \mathbf{P},\ \$,\ \$,\ \$,\ \mathsf{aux}, \\ \text{then} \quad & ③ : \mathbf{B},\ \overline{\mathbf{A}}',\ \mathbf{P},\ \mathbf{r}^\intercal\mathbf{B},\ \mathbf{r}^\intercal\overline{\mathbf{A}}',\ \mathbf{K}\ ,\ \mathsf{aux} \approx ④ : \mathbf{B},\ \overline{\mathbf{A}}',\ \mathbf{P},\ \$,\ \$,\ \mathbf{K},\ \mathsf{aux}. \end{aligned}$$

Here, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ for $m = \Theta(n \log q)$, and $\mathbf{K} \xleftarrow{\$} \mathbf{B}^{-1}(\mathbf{P})$ is a low-norm matrix satisfying $\mathbf{BK} = \mathbf{P}$.[97] The public-coin version of this assumption requires that aux contain the random coins used for sampling from $\mathcal{S}$.

[97] $\mathbf{K}$ can be efficiently sampled using a trapdoor [MP12] of $\mathbf{B}$.

Our multi-key depth-unbounded ABE schemes rely on a stronger variant of the evasive LWE assumption, where both the precondition and the postcondition additionally include a public key $\mathbf{A}_{\mathsf{fhe}}$ of [GSW13] and a circular *encoding* of the secret key. The circular encoding consists of two parts — a circular encryption $\mathbf{S}$ of the secret key under itself, and the attribute encoding [BGG+14] of $\mathbf{S}$ using the same secret and a matrix $\mathbf{A}_{\mathsf{circ}}$. More formally, fix an efficiently sampleable joint distribution of $\mathbf{A}_{\mathsf{circ}}, \overline{\mathbf{A}}', \mathbf{P}, \mathsf{aux}$, the evasive circular LWE assumption (Assumption 8.2) stipulates that

$$\begin{aligned} \text{if} \quad & ①, \mathbf{A}_{\mathsf{fhe}}, \mathbf{S}, \mathbf{A}_{\mathsf{circ}}, \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G}) \approx ②, \$, \$, \mathbf{A}_{\mathsf{circ}}, \$, \\ \text{then} \quad & ③, \mathbf{A}_{\mathsf{fhe}}, \mathbf{S}, \mathbf{A}_{\mathsf{circ}}, \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G}) \approx ④, \$, \$, \mathbf{A}_{\mathsf{circ}}, \$. \end{aligned}$$

Recall that $\mathbf{s} = (\mathbf{r}^\intercal, -1)^\intercal$ is essentially the same as $\mathbf{r}$. Note also that $\mathbf{A}_{\mathsf{circ}}, \overline{\mathbf{A}}', \mathbf{P}, \mathsf{aux}$ are sampled independent of the public key $\mathbf{A}_{\mathsf{fhe}}$ and the circular ciphertext $\mathbf{S}$, which are honestly generated.

The works of [Wee22,VWW22] argue that for evasive LWE, if the precondition holds, then no attacks are known against the postcondition. In particular, the family of zeroizing attacks (e.g., [CHL+15,CVW18,HJL21,JLLS23]) that have successfully ruled out many post-quantum obfuscation candidates and several recently proposed LWE with leakage assumptions (e.g., [GP21,WW21,DQV+21]) do not work. These attacks crucially rely on collecting equations of LWE secrets over the integers from correlated LWE samples (provided by or derived from the assumption or construction being analyzed). For evasive LWE, this strategy fails, since the precondition ensures that all LWE samples that can be obtained are jointly pseudorandom, and hence one cannot collect any useful equation over the integers. Our circular variant adds an FHE public key and a circular encoding to the precondition and the postcondition, while maintaining the same justification.

Lastly, we remark that it is possible to construct contrived auxiliary information with respect to which the (circular or non-circular) evasive LWE assumption becomes false (e.g., aux contains an obfuscation [VWW22]). However, no such counterexamples are known in the public-coin case, when aux contains the randomness used for sampling the matrices. Our KP-ABE schemes only rely on the public-coin version of the evasive circular LWE assumption.

Barriers to Adaptivity. Our ABE is very selectively secure — the functions and the challenge attribute must be chosen prior to seeing the master public key. This is weaker than selective security often considered in the literature, where the functions, but not the challenge attribute, can be chosen dependent on the master public key, the challenge ciphertext, and all previously queried keys. The security proof of the ABE scheme in this chapter, as well as those of [Wee22] and Chapter 7, relies on evasive LWE to deal with the trapdoor preimages in all the keys. Since evasive LWE is non-interactive, the functions in the keys cannot be chosen adaptively.

We note that proving adaptive security for lattice-based ABE for circuits [GVW13, BGG+14] has been a long-standing open problem. In addition, the recent work of [BM24] shows evidence that black-box reductions to LWE (or any bounded-round assumption) are implausible, and the recent work of [AMR25b] presents an adaptive attack against the ABE scheme of [BGG+14] when the LWE parameters do not grow fast enough with the attribute length.

A straight-forward idea to enable proof of (non-very) selective security is to consider an interactive version of evasive LWE, as suggested in [Wee22]. However, there is a simple counterexample to even the most modestly interactive formulation. Consider a version of evasive LWE where $\mathbf{P}$ (determined by functions in ABE keys) could depend on $\mathbf{B}$ (in ABE master public key). The high-level idea is that when $\mathbf{P}$ is correlated with $\mathbf{B}$, the preimage $\mathbf{K}$ could be more useful than one-time multiplication with LWE samples, enabling attacks to the postcondition not modeled by the precondition. In our counterexample, it is used for *two*-time multiplication. The sampler could set

$$\mathbf{P} = \mathbf{Q}\mathbf{M}\mathbf{Q}^{-1}\mathbf{B}, \qquad \text{with } \mathbf{M} = \begin{pmatrix} & \mathbf{I}_{n/2} \\ \mathbf{I}_{n/2} & \end{pmatrix},$$

where $\mathbf{Q}$ is a random $n \times n$ invertible matrix. Writing

$$\mathbf{s}^\intercal\mathbf{Q} = (\mathbf{s}_1^\intercal, \mathbf{s}_2^\intercal), \qquad \mathbf{Q}^{-1}\mathbf{B} = (\mathbf{B}_1^\intercal, \mathbf{B}_2^\intercal)^\intercal,$$

then

$$\mathbf{s}^\intercal\mathbf{B} + \mathbf{e}_{\mathrm{B}}^\intercal = \boxed{\mathbf{s}_1^\intercal\mathbf{B}_1 + \mathbf{e}_{\mathrm{B}}^\intercal} + \mathbf{s}_2^\intercal\mathbf{B}_2, \qquad \mathbf{s}^\intercal\mathbf{P} + \mathbf{e}_{\mathrm{P}}^\intercal = \mathbf{s}_2^\intercal\mathbf{B}_1 + \boxed{\mathbf{s}_1^\intercal\mathbf{B}_2 + \mathbf{e}_{\mathrm{P}}^\intercal},$$

so the precondition holds under LWE. However, since $\mathbf{M}^2 = \mathbf{I}$,

$$\mathbf{BK}^2 = \mathbf{PK} = \mathbf{QMQ}^{-1}\mathbf{BK} = \mathbf{QMQ}^{-1}\mathbf{P} = \mathbf{QMQ}^{-1}\mathbf{QMQ}^{-1}\mathbf{B} = \mathbf{B},$$

and we have

$$(\mathbf{s}^\intercal\mathbf{B} + \mathbf{e}_{\mathrm{B}}^\intercal)(\mathbf{K}^2 - \mathbf{I}) = \mathbf{s}^\intercal(\mathbf{BK}^2 - \mathbf{B}) + \mathbf{e}_{\mathrm{B}}^\intercal(\mathbf{K}^2 - \mathbf{I}) = \overbrace{\mathbf{e}_{\mathrm{B}}^\intercal(\mathbf{K}^2 - \mathbf{I})}^{\text{small}}.$$

We conjecture that $\mathbf{K}^2 - \mathbf{I} \neq \mathbf{0}$ with noticeable (in fact, quite high) probability, then for the postcondition, $\mathbf{c}_{\mathrm{B}}^\intercal(\mathbf{K}^2 - \mathbf{I})$ is small if $\mathbf{c}_{\mathrm{B}}$ consists of LWE samples, and large if $\mathbf{c}_{\mathrm{B}}$ is random, i.e., the postcondition fails.

Note that the counterexample is not known to apply to ABE adversaries (it is not clear how to efficiently find functions leading to such $\mathbf{P}$), similarly to the case of private-coin evasive LWE and witness encryption [VWW22]. However, the simplicity of our counterexample renders proving selective security based on an even stronger assumption less meaningful, even if it could be done. In contrast, the (usual, non-interactive) version of evasive circular LWE used in this chapter is not known to suffer from any counterexample. We leave the question of obtaining selective security (or ambitiously, even adaptive security) for future work.

Our Techniques in a Nutshell. Our key technical contribution is a new bootstrapping method for the ABE schemes of [BGG+14].

To briefly recap [BGG+14], starting from an input encoding $(\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}^\intercal)$, for any circuit C, the homomorphic evaluation procedure produces an output encoding $(\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_+^\intercal)$, where the noise is exponential in d. A key authorizing decryption when $C(\mathbf{x}) = 0$ enables transforming $(\mathbf{s}^\intercal\mathbf{A}_C + \mathbf{e}_+^\intercal)$ to approximately $\mathbf{s}^\intercal\mathbf{z}$ for some $\mathbf{z}$ chosen during set-up, which is used as a one-time pad to hide the message. To make decryption work, the modulus q (chosen when the scheme is set up) must be larger than the maximum output noise, upper-bounding the maximum depth of circuits that can be handled. Since q, hence the maximum d, is determined during set-up before the circuit is known, the scheme of [BGG+14] only supports evaluating circuits with *a priori* bounded depth.

To support circuits of unbounded depth, our bootstrapping technique allows transforming an attribute encoding $(\mathbf{s}^\intercal(\mathbf{A} - v\mathbf{G}) + \mathbf{e}_+^\intercal)$ of a bit v with large noises $\mathbf{e}_+$

into another encoding $(\mathbf{s}^\intercal(\mathbf{A}' - v\mathbf{G}) + (\mathbf{e}')^\intercal)$ of the same bit with smaller noises $\mathbf{e}'$ of some fixed magnitude. Moreover, the new matrix $\mathbf{A}'$ can be derived efficiently from $\mathbf{A}$ and other public matrices, independent of $\mathbf{s}$, v, and the noises. Now, using our bootstrapping technique, when the noise becomes too large, we can simply reduce the noise to obtain a *refreshed* encoding $(\mathbf{s}^\intercal(\mathbf{A}' - v\mathbf{G}) + (\mathbf{e}')^\intercal)$, and perform further homomorphic evaluation on it. Since bootstrapping can be applied for an unbounded number of times, we can handle circuits of unbounded depth.

Our ABE bootstrapping is inspired by the FHE bootstrapping [Gen09], but differs significantly. The idea of FHE bootstrapping is publishing a circular encryption $\mathbf{S}$ of the secret key $\mathbf{s}$, and whenever a ciphertext $\mathbf{C}$ becomes too noisy, one can homomorphically evaluate the decryption function $\mathsf{Dec}(\cdot, \mathbf{C})$ with the ciphertext hardcoded inside, over the circular ciphertext $\mathbf{S}$ to obtain a new, less noisy, ciphertext $\mathbf{C}'$ of the same plaintext. Unfortunately, throughout the past decade, it remained unknown how to adapt FHE bootstrapping to the context of ABE.

Our ABE bootstrapping proceeds roughly in two steps. The first step adapts the rounding (or modulus reduction) technique used in the FHE scheme of [BGV12] to the ABE setting. Rounding an attribute encoding $(\mathbf{s}^\intercal(\mathbf{A} - v\mathbf{G}) + \mathbf{e}_+^\intercal)$ naïvely would cause the modulus to decrease, leading again to depth-bounded evaluation. Instead, our new technique first rounds the encoding, which brings down the modulus, and then restores the modulus. But after restoration, the result is not a well-formed attribute encoding, instead, it can be regarded as a "*noiseless* ciphertext" of the form $(\mathsf{RndPad}(\mathbf{s}) - v\mathbf{s}^\intercal\mathbf{G})$, where $v\mathbf{s}^\intercal\mathbf{G}$ is the payload and $\mathsf{RndPad}(\mathbf{s})$ is a "blinding factor" that depends only on $\mathbf{s}$ and public matrices. We complement round-then-restore with another procedure generating an encoding of the form $(\mathbf{s}^\intercal\mathbf{A}' - \mathsf{RndPad}(\mathbf{s}) + (\mathbf{e}')^\intercal)$ with small noises, whose sum with the "noiseless ciphertext" is a well-formed attribute encoding $(\mathbf{s}^\intercal(\mathbf{A}' - v\mathbf{G}) + (\mathbf{e}')^\intercal)$ for the same bit v with small noises. The second step crucially relies on an attribute encoding $(\mathbf{s}^\intercal(\mathbf{A}_{\mathrm{circ}} - \mathbf{S} \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{circ}}^\intercal)$ of the circular ciphertext. More details can be found in the technical overview (Section 8.1.3).

Lastly, we mention that it is not clear how to combine our bootstrapping techniques with the lattice trapdoor techniques employed in prior ABE constructions [GVW13,

BGG+14]. As a result, our security proof does not rely on trapdoor simulation, more similar to recent works (see [QWW18,Wee22] and Chapter 6). This contributed to our full-fledged ABE schemes relying on the evasive circular LWE assumption. We leave the question of removing the new assumption as an interesting future direction.

8.1.2 Related Works

Table 8.1 summarizes the state of KP-ABE for circuits when the underlying work was initially written. (We discuss concurrent and follow-up works later.)

ABE Constructions Using $i\mathcal{O}$ and Related Primitives. Prior to the work underlying this chapter, depth-unbounded ABE and (1-key or multi-key) ABE with depth-independent succinctness were only known using the strong tools of $i\mathcal{O}$ or $i\mathcal{O}$-related primitives. The work of [GKP+13a] builds ABE for Turing machines from extractable witness encryption and SNARK. Both the existence of extractable witness encryption and that of SNARK are knowledge assumptions in nature, and the only candidate of extractable witness encryption relies on differing-input obfuscation [ABG+13], which is even stronger than $i\mathcal{O}$. It is well known that $i\mathcal{O}$ itself implies FE for unbounded-depth circuits [GGH+13a], which implies ABE for unbounded-depth circuits. However, the direct construction yields an FE with non-succinct secret keys, of size polynomial in the circuit size. In Chapter 9, we use such FE with non-succinct keys to construct FE and ABE for random-access machines (RAM) with succinct components. All the components of that scheme, the master public key, secret keys, and ciphertexts, are of constant size, whereas the master public key and ciphertexts of the ABE schemes in this chapter still scales with input length.

ABE Constructions Without $i\mathcal{O}$. As mentioned earlier, the lattice-based ABE schemes of [GVW13,BGG+14] support circuits of *a priori* bounded depth and input length, and the scheme of [BGG+14] has components of size polynomial in the maximum depth. Several follow-up works improve their construction on the fronts considered in this chapter. Chapter 6 improves the secret key size from $\mathrm{poly}(\lambda, d)$ to $\mathrm{poly}(\lambda)$, but unfortunately still suffer the constraint of being depth-bounded and have the

Table 8.1. Comparison among select KP-ABE schemes for circuits.

reference	**depth-unbounded**	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{sk}_C\rvert$	$\lvert\mathsf{ct}_{\mathbf{x}}\rvert$	**assumptions**
[GGH+13a]	✓	$\mathrm{poly}(L)$	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{poly}(L)$	$i\mathcal{O}$
[GKP+13a]	✓	$\mathrm{O}(1)$	$\mathrm{O}(1)$	$\mathrm{poly}(L)$	exWE & SNARK
[AS16]	✓	$\mathrm{O}(1)$	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{poly}(L)$	$i\mathcal{O}$
[AJS17]	✓	$\mathrm{O}(1)$	$\mathrm{O}(\lvert C\rvert)$	$\mathrm{O}(L)$	$i\mathcal{O}$ (subexp)
[AM18]	✓	$\mathrm{O}(1)$	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{O}(L)$	FE
[KNTY19]	✓	$\mathrm{O}(1)$	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{poly}(L)$	FE
[GWZ22]	✓	$\mathrm{poly}(L)$	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{O}(L)$	$i\mathcal{O}$
[ACFQ22]	✓	$\mathrm{O}(1)$	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{poly}(L)$	FE & DE-PIR
Chapter 9	✓	$\mathrm{O}(1)$	$\mathrm{O}(1)$	$\mathrm{O}(1)$	FE
[GGH+13b]		$L\,\mathrm{poly}(d)$	$\lvert C\rvert\,\mathrm{poly}(d)$	$L\,\mathrm{poly}(d)$	MMaps
[GVW13]		$L\,\mathrm{poly}(d)$	$\lvert C\rvert\,\mathrm{poly}(d)$	$L\,\mathrm{poly}(d)$	LWE
[BGG+14]		$L\,\mathrm{poly}(d)$	$\mathrm{poly}(d)$	$L\,\mathrm{poly}(d)$	LWE
[BV16]		$\mathrm{O}(1)$	$\mathrm{O}(L) + \mathrm{poly}(d)$	$\lvert\mathbf{x}\rvert\,\mathrm{poly}(d)$	LWE
this, 1-key	✓	$\mathrm{O}(L)$	$\mathrm{O}(1)$	$\mathrm{O}(L)$	csLWE
this	✓	$\mathrm{O}(L)$	$\mathrm{O}(1)$	$\mathrm{O}(L)$	evcsLWE
this	✓	$\mathrm{O}(1)$	$\mathrm{O}(L)$	$\mathrm{O}(\lvert\mathbf{x}\rvert)$	evcsLWE

$L, d, |C|$ are the input length, the depth, and the size of C, and $|\mathbf{x}|$ is the length of $\mathbf{x}$ in attribute-unbounded schemes. For schemes supporting Turing machines or random-access machines, the shown efficiency is that when the scheme is used for circuits. In component sizes, $\mathrm{poly}(\lambda)$ factors are ignored. For assumptions: exWE is extractable witness encryption; subexp means subexponential security; FE is for circuits; DE-PIR is doubly efficient private information retrieval; MMaps is multilinear maps; csLWE is circular small-secret LWE; evcsLWE is evasive circular small-secret LWE; only the heaviest assumptions are listed.

master public key and ciphertexts of size polynomially dependent on d. In addition, that scheme is designed in the generic pairing group model, thus not post-quantum secure.

Unbounded Attributes. Brakerski and Vaikuntanathan [BV16] presented an ABE scheme supporting unbounded attributes, based on and modified from the scheme of [BGG+14]. Goyal, Koppula, and Waters [GKW16] presented a generic transformation converting any attribute-bounded ABE into an attribute-unbounded one. (Both also make security stronger.) As mentioned above, the transformation of [GKW16] can also be applied to our depth-unbounded ABE scheme to further remove the predetermined bound on attribute length, yielding a scheme that handles truly all polynomial-size computations.

Concurrent and Follow-Up Works. The concurrent work of [CW23] achieves ABE for circuits (of bounded depth) with $\mathrm{poly}(\lambda)$-size secret keys and $(L+1)\,\mathrm{poly}(\lambda, d)$-size master public key and ciphertexts, based solely on LWE (same efficiency as Chapter 6, with pairing removed). The technical core of this chapter, unbounded homomorphic evaluation for attribute encodings, has been used in many follow-up works. Oftentimes, if a lattice-based construction uses the homomorphic evaluation of [BGG+14], then swapping it out by ours yields depth-unbounded and more efficient schemes. The only case of follow-up work where it is explicitly stated that the encoding of [BGG+14] cannot be swapped out for ours is [ARS24]. Otherwise, it *is* the case for ABE for RAM [DHM+24] and ABE for circuits [Wee24] with secret key size $\mathrm{poly}(\lambda, d)$ and ciphertext size sublinear in L (attribute length). The work of [AKY24a] tweaks our encoding and evaluation for rerandomization, achieving CP-ABE for circuits of unbounded depth and ABE for Turing machines. The work of [AKY24b] further generalized [AKY24a] to achieve, not just ABE, but a new kind of partially hiding functional encryption, "pseudorandom PHFE", whose security holds when the outputs are pseudorandom, with optimal component sizes. The pseudorandom PHFE can be bootstrapped to KP-ABE, KP-PE, CP-ABE for circuits of unbounded depth with optimal component sizes, which in turn implies ABE for TM with improved efficiency. Surprisingly, in [AKY24b], no circularity assumption is needed. However, the variants

of evasive LWE used in [AKY24a,AKY24b] allow private coins (and such feature is indeed used in the proof). This renders our result (public-coin, circular) and theirs (private-coin, non-circular) incomparable. See Section 8.2.3 for a discussion of variants of evasive LWE.

8.1.3 Technical Overview

In this section, we present the core techniques of our unbounded homomorphic evaluation before exemplifying its usage with attribute-based laconic function evaluation (AB-LFE).

[BGG+14] Bounded Homomorphism. Our starting point is the attribute encoding and its homomorphic evaluation due to [BGG+14]. Let $\mathbf{G}$ be the gadget matrix. Given public matrix $\mathbf{A}$ and LWE secret $\mathbf{s}$, the encoding of $\mathbf{x}$ (bit-string) is $\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})$, where the wavy underline indicates the presence of noise. Given $\mathbf{A}$ and a circuit C with Boolean output, one can compute a low-norm matrix $\mathbf{H}_C$, and given $\mathbf{A}, C, \mathbf{x}$, a low-norm matrix $\mathbf{H}_{C,\mathbf{x}}$, satisfying

$$(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}\mathbf{H}_C - C(\mathbf{x})\mathbf{G} \quad \Longrightarrow \quad \mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{s}^\intercal(\mathbf{A}\mathbf{H}_C - C(\mathbf{x})\mathbf{G}).$$

The procedure can be regarded as evaluation on a matrix-valued circuit $\mathbf{x} \mapsto C(\mathbf{x})\mathbf{G}$, and it can be extended to such circuits having arbitrary matrix output (not just multiples of $\mathbf{G}$).

The norms of $\mathbf{H}_C$ and $\mathbf{H}_{C,\mathbf{x}}$ grow exponentially with the depth of C, which translates to the noise growth. Once the modulus q is fixed, it puts a polynomial bound on the depth — $\mathrm{O}(\log q)$ at the very most. After some *a priori* fixed polynomial depth, the noise will grow beyond tolerance. Beyond correctness, the security proof of [BGG+14] also relies on $\mathbf{H}_C, \mathbf{H}_{C,\mathbf{x}}$ being low-norm. Clearly, the crux of the matter is controlling noise growth during homomorphic evaluation.

Inspirations from FHE. In fully homomorphic encryption literature, there are two major ways of dealing with noise, *rounding* and *bootstrapping*.

Let M (factor of q) be a rounding resolution, C a low-depth subcircuit, $\mathbf{x}$ an input, and $\mathbf{A}_C = \mathbf{A}\mathbf{H}_C$ the public matrix for C. As a first attempt, after evaluation of C with

noise about to overflow, we might try

$$\left\lfloor \frac{(\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}^\intercal_{\text{large}}) \bmod q}{M} \right\rceil = (\mathbf{s}^\intercal \mathbf{A}_{C,\text{small}} - C(\mathbf{x})\mathbf{s}^\intercal \mathbf{G}_{\text{small}} + \mathbf{e}^\intercal_{\text{small}}) \bmod \frac{q}{M}. \quad (8.1)$$

The secret $\mathbf{s}$ is now assumed to be low-norm so that it can be "factored" out of rounding. Here, $\mathbf{e}_{\text{small}}$ contains both the rounded $\mathbf{e}_{\text{large}}$ and the rounding error. Rounding reduces the absolute magnitude of noise, but it also shrinks the modulus, and the modulus-to-noise ratio remains large. Therefore, rounding does not enable unbounded homomorphic evaluation *by itself*.

Bootstrapping reduces the noise without diminishing the modulus. Let $\mathsf{hct}(\mathbf{x})$ represent an FHE ciphertext of $\mathbf{x}$. Suppose HEval is the FHE evaluation procedure such that

$$\mathsf{HEval}_C(\mathsf{hct}(\mathbf{x})) = \mathsf{hct}(C(\mathbf{x})) \qquad \text{for any circuit } C : \mathbf{x} \mapsto C(\mathbf{x}),$$

where the noise magnitude of the output $\mathsf{hct}(C(\mathbf{x}))$ depends on that of the input $\mathsf{hct}(\mathbf{x})$ and the depth of C. To bootstrap, we publish $\mathsf{hct}(\mathsf{hsk})$, an FHE ciphertext of the FHE secret key hsk (encrypted under itself). Given $\mathsf{hct}_{\text{large}} = \mathsf{hct}(x)$, which, though still decryptable to x, might contain large noise, we let $C_u(v)$ be a circuit with u hardwired that performs FHE decryption on u (ciphertext) using input v (key), and run

$$\mathsf{hct}_{\text{small}} = \mathsf{HEval}_{C_{\mathsf{hct}_{\text{large}}}}(\mathsf{hct}(\mathsf{hsk})) = \mathsf{hct}(C_{\mathsf{hct}_{\text{large}}}(\mathsf{hsk})) = \mathsf{hct}(x).$$

The output $\mathsf{hct}_{\text{small}}$ is again a ciphertext of x, but its noise magnitude only depends on that of $\mathsf{hct}(\mathsf{hsk})$ and the depth of FHE decryption circuit, not that of $\mathsf{hct}_{\text{large}}$. This procedure restores the noise to a fixed amount and helps achieving (non-leveled) FHE. However, it is not clear how bootstrapping can be applied to attribute encoding [BGG+14] (or more generally, ABE).

Difficulties of Bootstrapping ABE. A naïve idea to reduce the noise in $\mathbf{c}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G})$ by bootstrapping is to homomorphically evaluate, on an attribute encoding of $\mathbf{s}$, a circuit $C_u(v)$ with $u = \mathbf{c}$ hardwired that outputs the value (i.e., $C(\mathbf{x})$)

encoded using the input $v = \mathbf{s}$ so that[98]

$$\text{(wishful thinking)} \quad \mathbf{s}^\intercal(\mathbf{A} - \mathbf{s} \otimes \mathbf{G})\mathbf{H}_{C_\mathbf{c},\mathbf{s}} = \mathbf{s}^\intercal(\mathbf{A}\mathbf{H}_{C_\mathbf{c}} - C_\mathbf{c}(\mathbf{s})\mathbf{G}) = \mathbf{s}^\intercal(\mathbf{A}\mathbf{H}_{C_\mathbf{c}} - C(\mathbf{x})\mathbf{G}),$$

where the output noise only depends on that in $\mathbf{s}^\intercal(\mathbf{A} - \mathbf{s} \otimes \mathbf{G})$ and the depth of $C_\mathbf{c}$, but not that in $\mathbf{c}$. There are two issues with this approach.

One is that the circuit $C_\mathbf{c}$ is ciphertext-dependent, thus unknown at key generation time, making it difficult, if possible at all, to generate the key corresponding to the correct circuit. (Concretely, the key depends on $\mathbf{H}$ of the circuit, supposedly independent of the ciphertext.) This is also highlighted by the difference between the security of FHE and ABE — in FHE, decryption works regardless of the homomorphic computation applied to the ciphertext, whereas in ABE, since the key is bound to a specific circuit, the homomorphic evaluation must be *authenticated* and decryption must implicitly verify that the correct computation is performed.

The other difficulty is that in [BGG+14] homomorphism, it is necessary to know the value being encoded, so evaluating a circuit on $\mathbf{s}$ requires knowing $\mathbf{s}$, which we cannot afford as revealing it would destroy security.

Our Unbounded Homomorphic Evaluation. We achieve unbounded homomorphism for [BGG+14] attribute encoding by combining rounding and (circular-FHE-style) bootstrapping.

Noise Removal. We make rounding *noiseless*. Recall that $\mathbf{e}_{\text{small}}$ in Equation (8.1) contains both the rounded $\mathbf{e}_{\text{large}}$ and the rounding error. To remove the former, we round when $\|\mathbf{e}_{\text{large}}\|$ is much less than M. To get rid of the latter, we draw inspiration from the learning with rounding (LWR) assumption — instead of taking $\mathbf{s}$ out from rounding, we keep it inside:

$$\left\lfloor \frac{(\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{\text{large}}^\intercal) \bmod q}{M} \right\rceil = \left(\left\lfloor \frac{(\mathbf{s}^\intercal\mathbf{A}_C + \mathbf{e}_{\text{large}}^\intercal) \bmod q}{M} \right\rceil - C(\mathbf{x})\mathbf{s}^\intercal\mathbf{G}_{\text{small}} \right) \bmod \frac{q}{M}$$

$$\text{(with high probability)} \quad = \left(\left\lfloor \frac{\mathbf{s}^\intercal\mathbf{A}_C \bmod q}{M} \right\rceil - C(\mathbf{x})\mathbf{s}^\intercal\mathbf{G}_{\text{small}} \right) \bmod \frac{q}{M}.$$

[98] Precisely speaking, the attribute encoding (and FHE ciphertexts of $\mathbf{s}$ later) is bit-by-bit for $\mathbf{s}$ (and $\mathbf{S}$ later), and needs to include a helper encoding of 1. We let go of those details for a simplified exposition in this overview.

For the first equality to hold, we need M to be a power of two so that $M \mid \mathbf{G}$ (ignoring the small entries in $\mathbf{G}$ for now) and $C(\mathbf{x}), \mathbf{s}, \frac{\mathbf{G}}{M}$ are integral hence can be freely taken out of rounding. The second equality holds when $\mathbf{e}_{\text{large}}$ does not introduce carrying/borrowing. Intuitively, $\mathbf{s}^\intercal \mathbf{A}_C$ for $\mathbf{A}_C$ arising from homomorphic evaluation should just be random, hence is likely to be far away from the boundary of rounding jumps.

Rounding transfers the encoding to a smaller modulus $\frac{q}{M}$. We restore the large modulus q by multiplication:

$$M \left\lfloor \frac{(\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{\text{large}}^\intercal) \bmod q}{M} \right\rceil = \left(M \left\lfloor \frac{\mathbf{s}^\intercal \mathbf{A}_C \bmod q}{M} \right\rceil - C(\mathbf{x})\mathbf{s}^\intercal \cdot M\mathbf{G}_{\text{small}} \right) \bmod q.$$

We want to recover $\mathbf{G}$, but $M\mathbf{G}_{\text{small}}$ only contains the large powers of two in $\mathbf{G}$. We also glossed over the issue of entries in $\mathbf{G}$ less than M when rounding. The fix is to rearrange $\mathbf{G}$ by small and large portions $\mathbf{G}_{\text{L}}, \mathbf{G}_{\text{R}}$ and amplify $\mathbf{G}_{\text{L}}$ by M before rounding. Let $\mathbf{Q}$ be the permutation matrix such that $(\mathbf{G}_{\text{L}}, \mathbf{G}_{\text{R}})\mathbf{Q} = \mathbf{G}$, then the rounding procedure is

$$\left\lfloor \frac{\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G})\mathbf{G}^{-1}(M\mathbf{G}_{\text{L}}, \mathbf{G}_{\text{R}}) \bmod q}{M} \right\rceil \begin{pmatrix} \mathbf{I} & \\ & M\mathbf{I} \end{pmatrix} \mathbf{Q}.$$

The matrix multiplying $C(\mathbf{x})\mathbf{s}^\intercal$ is (note that the quantity being rounded is integral)

$$\left\lfloor \frac{\mathbf{G}\mathbf{G}^{-1}(M\mathbf{G}_{\text{L}}, \mathbf{G}_{\text{R}})}{M} \right\rceil \begin{pmatrix} \mathbf{I} & \\ & M\mathbf{I} \end{pmatrix} \mathbf{Q} = \left(\mathbf{G}_{\text{L}}, \tfrac{\mathbf{G}_{\text{R}}}{M}\right) \begin{pmatrix} \mathbf{I} & \\ & M\mathbf{I} \end{pmatrix} \mathbf{Q} = (\mathbf{G}_{\text{L}}, \mathbf{G}_{\text{R}})\mathbf{Q} = \mathbf{G},$$

and the other part is a *rounded pad* dependent on $\mathbf{A}_C$ and $\mathbf{s}$,

$$\mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s}) = \left\lfloor \frac{\mathbf{s}^\intercal \mathbf{A}_C \mathbf{G}^{-1}(M\mathbf{G}_{\text{L}}, \mathbf{G}_{\text{R}}) \bmod q}{M} \right\rceil \begin{pmatrix} \mathbf{I} & \\ & M\mathbf{I} \end{pmatrix} \mathbf{Q}.$$

We note that proving correctness for the small portion is quite different from that for the large portion and relies on $\mathbf{s}$ being low-norm (see Section 8.3.1).

BOOTSTRAPPING. After noise removal (rounding and modulus restoration), the encoding is no longer amenable to [BGG+14] homomorphism. If we can compute

$\mathbf{s}^\intercal \mathbf{A}'_C - \mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s})$ for some (other) public matrix $\mathbf{A}'_C$ related to C, we will be able to continue homomorphic evaluation using

$$\mathbf{s}^\intercal \mathbf{A}'_C - \mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s}) + (\mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s}) - C(\mathbf{x})\mathbf{s}^\intercal \mathbf{G}) = \mathbf{s}^\intercal(\mathbf{A}'_C - C(\mathbf{x})\mathbf{G}).$$

This naturally calls for homomorphic evaluation on $\mathbf{s}$. But again, we cannot perform [BGG+14] evaluation on $\mathbf{s}$ as it must not be known to the evaluator for security. Instead, we circularly encrypt $\mathbf{s}$, perform [BGG+14] evaluation on its ciphertext $\mathbf{S}$ under itself, and employ the dual-use technique of [BTVW17] for *automagic decryption*.

In the FHE scheme of [GSW13], the secret key is $\mathbf{s}^\intercal$, a ciphertext is a matrix $\mathbf{C}$, and decryption is noisy linear. Suppose $\mathbf{C}$ encrypts $\mathbf{x}^\intercal$, then $\mathbf{s}^\intercal \mathbf{C} = \mathbf{x}^\intercal + \text{noise}$.[99] The technique of [BTVW17] is to use the same $\mathbf{s}$ for attribute encoding and FHE secret key. We additionally publish

$$\text{(circular encryption)} \quad \mathbf{S} = \mathsf{hct}(\mathbf{s}), \qquad \text{(circular encoding)} \quad \mathbf{s}^\intercal(\mathbf{A}_{\mathrm{circ}} - \mathbf{S} \otimes \mathbf{G}).$$

Given $\mathbf{A}_C$, we evaluate $\mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}}$ on attribute $\mathbf{S}$, which yields the desired

$$\begin{aligned}
& \mathbf{s}^\intercal(\mathbf{A}_{\mathrm{circ}} - \mathbf{S} \otimes \mathbf{G})\mathbf{H}_{\mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}},\mathbf{S}} \qquad \mathbf{A}_{\mathrm{circ}}\mathbf{H}_{\mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}}} \uparrow \\
\text{([BGG+14])} \quad &= \mathbf{s}^\intercal \mathbf{A}_{\mathrm{circ}} \mathbf{H}_{\mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}}} - \mathbf{s}^\intercal \mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}}(\mathbf{S}) = \mathbf{s}^\intercal \mathbf{A}'_C - \mathbf{s}^\intercal \mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}}(\mathsf{hct}(\mathbf{s})) \\
\text{([GSW13])} \quad &= \mathbf{s}^\intercal \mathbf{A}'_C - \mathbf{s}^\intercal \mathsf{hct}(\mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s})) = \mathbf{s}^\intercal \mathbf{A}'_C - \mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s}).
\end{aligned}$$

Note that FHE decryption happens automagically when an FHE ciphertext is evaluated on the attribute. The noise in $\mathbf{s}^\intercal \mathbf{A}'_C - \mathsf{RndPad}_{\mathbf{A}_C}(\mathbf{s})$, thus that in $\mathbf{s}^\intercal(\mathbf{A}'_C - C(\mathbf{x})\mathbf{G})$, only grows with the depth of $\mathsf{HEval}_{\mathsf{RndPad}_{\mathbf{A}_C}}$, which is fixed and does not grow with that of C.

SUMMARY. By removing noise and bootstrapping after every gate, we keep the noise at a fixed level hence achieve unbounded homomorphic evaluation. Abstractly, the procedure gives rise to two efficient algorithms (corresponding to $\mathbf{H}_C, \mathbf{H}_{C,\mathbf{x}}$ of [BGG+14])

$$\mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C) = \mathbf{A}_C, \qquad \mathsf{UEvalCX}(\mathbf{A}_{\mathrm{attr}}, \mathbf{c}^\intercal_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{c}^\intercal_{\mathrm{attr}}, C, \mathbf{x}, \mathbf{S}) = \mathbf{c}^\intercal_C,$$

[99] The formulation in [GSW13] is different. For their bit encryption, we can regard $x\mathbf{s}^\intercal \mathbf{G}$ as the plaintext $\mathbf{x}^\intercal$. It extends to any vector $\mathbf{x}^\intercal$ and to ciphertexts homomorphically evaluated for vector-valued circuits. (See Section 8.2.4.)

satisfying (with high probability)

$$\mathsf{UEvalCX}(\mathbf{A}_{\mathrm{attr}}, \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{attr}} - \mathbf{x} \otimes \mathbf{G}), \mathbf{A}_{\mathrm{circ}}, \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{attr}} - \mathbf{S} \otimes \mathbf{G}), C, \mathbf{x}, \mathbf{S}) = \mathbf{s}^{\mathsf{T}}(\overset{\mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C)}{\widehat{\mathbf{A}_C}} - C(\mathbf{x})\mathbf{G})$$

if $\mathbf{S}$ is a circular [GSW13] encryption of $\mathbf{s}$.

One Minute Detail. We highlight a technicality here. Recall that noise removal relies on $\mathbf{s}^{\mathsf{T}}\mathbf{A}_C$ being far away from the boundary of rounding jumps. The intuition was $\mathbf{s}^{\mathsf{T}}\mathbf{A}_C$ is (pseudo-)random, but in reality, $\mathbf{s}^{\mathsf{T}}\mathbf{A}_C = \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{H}_C$ for some low-norm $\mathbf{H}_C$ dependent on $\mathbf{A}$. Although $\mathbf{s}^{\mathsf{T}}\mathbf{A}$ is marginally random, $\mathbf{H}_C$ has complicated dependency (described by C) on it, so the distribution of $\mathbf{s}^{\mathsf{T}}\mathbf{A}_C$ is not easy to work with. For our scheme, the last entry of $\mathbf{s}$ is always -1, in the worst scenario without considering the exact structure of $\mathbf{H}_C$, for every $\mathbf{z}$, there exists a function $H : \mathbf{A} \mapsto \mathbf{H}$ outputting a low-norm matrix that makes $\mathbf{s}^{\mathsf{T}}\mathbf{A}H(\mathbf{A}) = \mathbf{z}^{\mathsf{T}}$ happen with overwhelming probability.[100]

We take advantage of the fact that the low-norm $\mathbf{H}_C$ is efficiently computable. Our technique is to introduce (another, independent) noise $\mathbf{e}$. Under the LWE assumption, $(\mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}})\mathbf{H}_C$ is indistinguishable from $\boldsymbol{\delta}^{\mathsf{T}}\mathbf{H}_C$ (for $\boldsymbol{\delta}$ random and independent of $\mathbf{H}_C$). The latter can be shown to be far away from the boundary of rounding jumps. (See Lemma 8.6.)

Alternatively, the issue can be worked around by adding a random shift to the value before rounding. The shifts can be generated using a PRF key.

Laconic Function Evaluation. Putting things together, we present our AB-LFE construction using unbounded homomorphic evaluation. Its components are

$$\mathsf{crs} = (\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}), \qquad \mathsf{digest} = \mathbf{A}_C, \qquad \mathsf{ct}_{\mathbf{x}}(\mu) = (\mathbf{S}, \mathbf{c}_{\mathrm{attr}}, \mathbf{c}_{\mathrm{circ}}, \mathbf{z}, c_{\mathrm{msg}}),$$

where $\mathbf{S} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \mathbf{s}^{\mathsf{T}}\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}_{\mathrm{fhe}}^{\mathsf{T}} \end{pmatrix}\mathbf{R} - \mathbf{s} \otimes \mathbf{G}$ is the circular ciphertext, and

$$\mathbf{c}_{\mathrm{attr}}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{attr}} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{attr}}^{\mathsf{T}}, \qquad \mathbf{c}_{\mathrm{circ}}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{circ}} - \mathbf{S} \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{circ}}^{\mathsf{T}},$$
$$c_{\mathrm{msg}} = \mathbf{s}^{\mathsf{T}}\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}) + e_{\mathrm{msg}} + \mu \cdot \lfloor q/2 \rceil$$

[100] For most $\mathbf{z}$, this H is not known (nor believed) to be efficiently computable.

are the attribute/circular/message encodings. To decrypt when $C(\mathbf{x}) = 0$, run UEvalCX to obtain

$$\mathbf{c}_{C,\mathbf{x}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) = \mathbf{s}^\intercal\mathbf{A}_C,$$

which can be used to cancel $\mathbf{s}^\intercal\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z})$ in c_{msg} for message recovery.

The security proof when $C(\mathbf{x}) = 1$ involves two steps. First, simulate message encoding using

$$\mathbf{s}^\intercal\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}) + \mu \cdot \lfloor q/2 \rceil = \mu \cdot \lfloor q/2 \rceil + \underbrace{\mathbf{c}_{C,\mathbf{x}}^\intercal\mathbf{G}^{-1}(\mathbf{z})}_{\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G})\mathbf{G}^{-1}(\mathbf{z})} + \overbrace{C(\mathbf{x})}^{=1} \cdot \mathbf{s}^\intercal\mathbf{z}.$$

Then, invoke circular LWE on secret $\mathbf{s}$ to hide μ with the pseudorandom pad $\mathbf{s}^\intercal\mathbf{z}$.

Attribute-Based Encryption. Our ABE scheme has many components similar to our LFE scheme, with two major differences — ciphertexts are not specific to any circuit and security holds against multiple unauthorized keys. The first means c_{msg} cannot depend on $\mathbf{A}_C$. We use "one more level of indirection" of one-time pads. We sample $\mathbf{B}, \mathbf{z}$ during set-up and let

$$\mathbf{c}_{\mathrm{B}}^\intercal = \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}_{\mathrm{B}}^\intercal, \qquad c_{\mathrm{msg}} = \mathbf{s}^\intercal\mathbf{z} + e_{\mathrm{msg}} + \mu \cdot \lfloor q/2 \rceil.$$

For decryption, we let the secret key be a random vector $\mathbf{z}'$ together with a low-norm vector $\mathbf{k}$ (sampled using the trapdoor of $\mathbf{B}$) such that $\mathbf{B}\mathbf{k} = \mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}') + \mathbf{z}$. Think of $\mathbf{s}^\intercal\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}')$ as a one-time pad for $\mathbf{s}^\intercal\mathbf{z}$ (conditionally removable if $C(\mathbf{x}) = 0$), and the latter as a one-time pad for the message. When $C(\mathbf{x}) = 0$, the decryptor computes

$$c_{\mathrm{msg}} + \mathbf{c}_{\mathrm{B}}^\intercal\mathbf{k} - \mathbf{s}^\intercal\mathbf{A}_C \cdot \mathbf{G}^{-1}(\mathbf{z}') = \mu \cdot \lfloor q/2 \rceil.$$

To argue security based on the evasive circular LWE assumption, it suffices to prove the precondition, which asserts that all directly given LWE samples and all LWE samples computable from the trapdoor are jointly pseudorandom. The difference between the precondition proof and the LFE proof is that in the view, there are

$$\mathbf{c}_{\mathrm{B}}^\intercal\mathbf{k}_j : \qquad \mathbf{s}^\intercal\mathbf{A}_{C_j}\mathbf{G}^{-1}(\mathbf{z}_j') + \mathbf{s}^\intercal\mathbf{z} \qquad \text{for all queried keys for } C_j$$

due to the trapdoor. We simulate these components as

$$\underbrace{\mathbf{c}_{C_j,\mathbf{x}}^{\mathsf{T}}\mathbf{G}^{-1}(\mathbf{z}'_j)}_{\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{C_j}-C_j(\mathbf{x})\mathbf{G})\mathbf{G}^{-1}(\mathbf{z}'_j)} + \overbrace{C_j(\mathbf{x})}^{=1} \cdot \mathbf{s}^{\mathsf{T}}\mathbf{z}'_j + \mathbf{s}^{\mathsf{T}}\mathbf{z},$$

and invoke circular LWE to argue pseudorandomness.

8.2 Preliminaries

All circuits in this chapter are Boolean and use only conjunction, disjunction, and negation gates. For conceptual reasons, the domain/codomain of circuits might be written as any finite set, but pragmatically, the input/output are always encoded as bits. We write $A \xrightarrow{\$} B$ for distributions over B indexed by A, i.e., a randomized function mapping A to B.

Useful facts about intervals (intersected with the set of integers, see Chapter 2) are $|[a,b)| = \lceil b \rceil - \lceil a \rceil$ for all $a \le b$, and $2a - 1 < \lceil a \rceil - \lceil -a \rceil < 2a + 1$ for all a. For $z \in \mathbb{Z}_q$, the canonical bits(z) is the binary representation of its smallest non-negative representative (less than q), low to high, potentially with extra trailing (high) zeros. The expression $(z \bmod q)$ denotes the representative in $\left[-\frac{q}{2}, \frac{q}{2}\right)$. We also write $(z \bmod p)$ when p divides q. Remainder extends to matrices entry-wise.

Symbols. Table 8.2 explains select single-letter symbols used in this chapter.

8.2.1 Attribute-Based Laconic Function Evaluation

See Section 8.6 for further discussions on general LFE.[101]

Definition 8.1 (AB-LFE [QWW18]). Let $P = \{P_{\lambda,\mathsf{param}}\}_{\lambda\in\mathbb{N},\mathsf{param}\in\mathsf{Params}_\lambda}$ be a family of predicate sets, where each $P_{\lambda,\mathsf{param}}$ is a set of predicates $\{f : X_{\lambda,\mathsf{param}} \to \{0,1\}^{1\times L'}\}$, with $\mathsf{Params} = \{\mathsf{Params}_\lambda\}_{\lambda\in\mathbb{N}}$ being a sequence of parameter sets, $X_{\lambda,\mathsf{param}}$ a domain dependent only on λ and param, and L' the output length of f (different for each f).

[101] The definitions in this dissertation are specialized for AB-LFE and are textually quite different from the versions in [HLL23a,HLL25,HLL23b].

Table 8.2. Select single-letter symbols in this chapter.

symbol	meaning
$\lambda, \beta, \mathcal{A}, \mathcal{S}$	security parameter, challenge bit, adversary, sampler
J, j	key count, index
$f, x, \boldsymbol{\mu}$	function, attribute, messages [abstract]
$C, d, \mathbf{x}, L, \ell, \mu$	circuit, depth, input, length, index, message
c, r	commitment, opening/randomness
$\boldsymbol{\iota}, \mathbf{I}, \mathbf{g}, \mathbf{G}$	standard basis, identity matrix, gadget vector, gadget matrix
n, m	LWE dimension, sample count [shape of matrix with trapdoor]
m', q, σ	LWE sample count, modulus, Gaussian error width
ρ, B	hardness exponent, error bound
$\mathbf{A}, \overline{\mathbf{A}}, \underline{\mathbf{a}}^\intercal$	matrix without trapdoor, first n rows, last row
$\mathbf{B}, \tau$	matrix with trapdoor, trapdoor of matrix
$\mathbf{p}, \mathbf{P}, \mathbf{k}, \mathbf{K}$	image, many, Gaussian preimage, many
$\mathbf{r}, \mathbf{s}$	circular LWE secret without -1, with -1 [$\mathbf{s}^\intercal = (\mathbf{r}^\intercal, -1)$]
$e, \mathbf{e}, \mathbf{c}, \delta, \boldsymbol{\delta}, \boldsymbol{\Delta}$	error, many, LWE samples, random value, many, many
$\mathbf{R}, \mathbf{X}, \mathbf{C}, \mathbf{S}$	FHE randomness, ciphertexts of $\mathbf{x}$, of $C(\mathbf{x})$, of $\mathbf{s}$
$\mathbf{H}, \mathbf{h}, h$	homomorphic evaluation matrix, column, column function
M, p	rounding resolution (dividing q), generic factor of q
$\mathbf{G}_\mathrm{L}, \mathbf{G}_\mathrm{R}, \mathbf{Q}$	"left" part of $\mathbf{G}$, "right" part, permutation [$(\mathbf{G}_\mathrm{L}, \mathbf{G}_\mathrm{R})\mathbf{Q} = \mathbf{G}$]
$\mathbf{z}$	public vector creating one-time pad

To save space, "many" following X means "many X's", e.g., $\mathbf{P}$ is "many images".

An *attribute-based laconic function evaluation scheme* for P consists of four efficient algorithms.

- $\mathsf{GenCRS}(1^\lambda, \mathsf{param})$ takes some $\mathsf{param} \in \mathsf{Params}_\lambda$ as input and outputs a common reference string crs.
- $\mathsf{Compress}(1^\lambda, \mathsf{crs}, f)$ takes as input crs and some $f : X_{\lambda,\mathsf{param}} \to \{0,1\}^{1\times L'} \in P_{\lambda,\mathsf{param}}$. It outputs a potentially randomized digest, whose length only depends on L', λ, and param (not the description size of f). If the algorithm is randomized, it must be public-coin, i.e., digest is a pair whose first component is the random tape prefix read by that invocation of Compress.
- $\mathsf{Enc}(1^\lambda, \mathsf{crs}, L', \mathsf{digest}, x, \boldsymbol{\mu})$ takes as input crs, L', digest, some $x \in X_{\lambda,\mathsf{param}}$, and a multi-bit message $\boldsymbol{\mu} \in \{0,1\}^{L'}$. It outputs a ciphertext ct of $\boldsymbol{\mu}$ bound to f, x.
- $\mathsf{Dec}(1^\lambda, \mathsf{crs}, f, \mathsf{digest}, x, \mathsf{ct})$ is supposed to recover $\boldsymbol{\mu}[\ell]$ if $f(x)[1,\ell] = 1$.

Definition 8.2 (AB-LFE correctness). Given an AB-LFE (Definition 8.1), consider the following $\mathsf{Exp}^{\mathsf{adptv}}_{\text{AB-LFE}\checkmark}(1^\lambda, \mathcal{A})$.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $\mathsf{param} \in \mathsf{Params}_\lambda$ from it. Run

$$\mathsf{crs} \xleftarrow{\$} \mathsf{GenCRS}(1^\lambda, \mathsf{param})$$

  and send crs to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ chooses some $f : X_{\lambda,\mathsf{param}} \to \{0,1\}^{1\times L'} \in P_{\lambda,\mathsf{param}}$. Run

$$\mathsf{digest} \xleftarrow{\$} \mathsf{Compress}(1^\lambda, \mathsf{crs}, f)$$

  and send digest to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ chooses some $x \in X_{\lambda,\mathsf{param}}$ and $\boldsymbol{\mu} \in \{0,1\}^{L'}$. Run

$$\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{crs}, L', \mathsf{digest}, x, \boldsymbol{\mu}).$$

- **Test.** Run $\boldsymbol{\mu}' \xleftarrow{\$} \mathsf{Dec}(1^\lambda, \mathsf{crs}, f, \mathsf{digest}, x, \mathsf{ct})$. If $f(x)[1,\ell] = 1$ but $\boldsymbol{\mu}'[\ell] \neq \boldsymbol{\mu}[\ell]$ for some $\ell \in [L']$, the outcome of the experiment is 1. Otherwise, the outcome is 0.

$\mathsf{Exp}^{f\text{-sel}}_{\text{AB-LFE}\checkmark}$ is modified from $\mathsf{Exp}^{\text{adptv}}_{\text{AB-LFE}\checkmark}$ by requiring $\mathcal{A}$ to choose f during Setup together with param (before it receives crs).

The scheme is *(computationally)*[102] *adaptively correct* if $\Pr[\mathsf{Exp}^{\text{adptv}}_{\text{AB-LFE}\checkmark}(1^\lambda, \mathcal{A}) \to 1]$ is negligible for all efficient $\mathcal{A}$. For *statistical* correctness, $\mathcal{A}$ can be inefficient but its total output length $(|\mathsf{param}| + |f| + |x| + L')$ must be $\mathrm{poly}(\lambda)$-bounded. For *f-selective* correctness, the experiment is $\mathsf{Exp}^{f\text{-sel}}_{\text{AB-LFE}\checkmark}$.

Definition 8.3 (AB-LFE security). Given an AB-LFE scheme (Definition 8.1) and a simulator SimEnc, consider $\mathsf{Exp}^{\text{adptv,real}}_{\text{AB-LFE}}(1^\lambda, \mathcal{A})$ and $\mathsf{Exp}^{\text{adptv,sim}}_{\text{AB-LFE}}(1^\lambda, \mathcal{A})$.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $\mathsf{param} \in \mathsf{Params}_\lambda$ from it. Run

$$\mathsf{crs} \xleftarrow{\$} \mathsf{GenCRS}(1^\lambda, \mathsf{param})$$

  and send crs to $\mathcal{A}$.

- **Query.** $\mathcal{A}$ chooses some $f : X_{\lambda,\mathsf{param}} \to \{0,1\}^{1\times L'} \in P_{\lambda,\mathsf{param}}$. Run

$$\mathsf{digest} \xleftarrow{\$} \mathsf{Compress}(1^\lambda, \mathsf{crs}, f)$$

  and send digest to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ chooses some $x \in X_{\lambda,\mathsf{param}}$ and $\boldsymbol{\mu} \in \{0,1\}^{L'}$. Run

$$\mathsf{ct} \xleftarrow{\$} \begin{cases} \mathsf{Enc}(1^\lambda, \mathsf{crs}, L', \mathsf{digest}, x, \boldsymbol{\mu}), & \text{in } \mathsf{Exp}^{\text{adptv,real}}_{\text{AB-LFE}}; \\ \mathsf{SimEnc}(1^\lambda, \mathsf{crs}, f, \mathsf{digest}, x, (\boldsymbol{\mu}[\ell])_{f(x)[1,\ell]=1}), & \text{in } \mathsf{Exp}^{\text{adptv,sim}}_{\text{AB-LFE}}; \end{cases}$$

  and send ct to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$, which is the outcome of the experiment.

$\mathsf{Exp}^{x\text{-sel}}_{\text{AB-LFE}}$ (resp. $\mathsf{Exp}^{\text{very-sel}}_{\text{AB-LFE}}$) is modified from $\mathsf{Exp}^{\text{adptv}}_{\text{AB-LFE}}$ by requiring $\mathcal{A}$ to choose x (resp. f, x) during Setup together with param (before it receives crs).

The scheme is *adaptively (resp. x-selectively; very selectively) secure* if there exists an efficient simulator SimEnc such that $\mathsf{Exp}^{\text{adptv,real}}_{\text{AB-LFE}} \approx \mathsf{Exp}^{\text{adptv,sim}}_{\text{AB-LFE}}$ (resp. for $\mathsf{Exp}^{x\text{-sel}}_{\text{AB-LFE}}$; for $\mathsf{Exp}^{\text{very-sel}}_{\text{AB-LFE}}$).

[102] The strength of correctness is not the same as the nature of its proof. See Footnote 103.

8.2.2 Attribute-Based Encryption (Lesser Correctness)

We remind the readers of [HLL23b] that the meaning of φ outputting $\perp$ is different. In [HLL23b], it is used to characterize the gap in correctness and security for predicate encryption (PE). In this dissertation, it is for abandoning both correctness and security — used in Chapters 7 and 9.

We will define PE separately when we need it (see Definition 8.5). Below, we define the lesser versions of correctness for ABE, only for non-$\perp$ cases.

Definition 8.4 (ABE correctness). Given an ABE scheme (Definition 2.3 with T not important and P never $\perp$), consider the following $\mathsf{Exp}^{\mathsf{adptv+}}_{\mathrm{ABE}\checkmark}(1^\lambda, \mathcal{A})$.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive $\mathsf{param} \in \mathsf{Params}_\lambda$ from it. Set up the scheme by running $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathsf{param})$ and send $(\mathsf{mpk}, \mathsf{msk})$ to $\mathcal{A}$.
- **Query I.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$. In each round, $\mathcal{A}$ chooses $f_j \in F_{\lambda,\mathsf{param}}$. Run $\mathsf{sk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, f_j)$ and send sk_j to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ chooses $x \in X_{\lambda,\mathsf{param}}$ and $y \in Y_{\lambda,\mathsf{param}}$. Run $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, y)$ and send ct to $\mathcal{A}$.
- **Query II.** Same as Query I.
- **Test.** Run $z_j \xleftarrow{\$} \mathsf{Dec}^{\mathsf{mpk},f_j,\mathsf{sk}_j,x,\mathsf{ct}}(1^\lambda)$ for all j. If $P_{\lambda,\mathsf{param}}(f_j, x) = 1$ but $z_j \neq y$ for some j, then the outcome of the experiment is 1. Otherwise, the outcome is 0.

Variants of $\mathsf{Exp}^{\mathsf{adptv+}}_{\mathrm{ABE}\checkmark}$ are "xy-sel+", "f-sel+", "adptv", "xy-sel", "f-sel", "very-sel", where "+" (strong) means msk is given (default is non-strong) and "xy-sel" (resp. "f-sel"; "very-sel") means (x, y) (resp. all f_j's; (x, y) and all f_j's) must be chosen together with param (before mpk is generated).

The scheme is *computationally (resp. statistically)* which-*correct* (default is computational)[103] if $\Pr[\mathsf{Exp}^{\mathsf{which}}_{\mathrm{ABE}\checkmark}(1^\lambda, \mathcal{A}) \to 1]$ is negligible for all efficient $\mathcal{A}$ (resp. all, poten-

[103]The strength of correctness is not the same as the nature of its proof. Since there is no interaction in $\mathsf{Exp}^{\mathsf{very\text{-}sel}}_{\mathrm{ABE}\checkmark}$, computational very selective correctness against *non-uniform* adversaries is equivalent

tially inefficient, $\mathcal{A}$ whose total output length ($|\mathsf{param}| + \sum_j |f_j| + |x| + |y|$) is $\mathrm{poly}(\lambda)$-bounded). It is *perfectly correct*[104] if $\Pr[\mathsf{Exp}^{\text{very-sel}}_{\mathrm{ABE}\checkmark}(1^\lambda, \mathcal{A}) \to 1] = 0$ for all $\mathcal{A}$.

8.2.3 Lattices

For ergonomics, some lattice-related notations in this chapter are different from the usual convention in Chapter 2. Let $n, m \geq 1$ and $q \geq 2$ be integers such that $\log_2 q \leq \frac{m}{n+1} \in \mathbb{Z}$ (note that the denominator is now $(n+1)$, instead of n). Let

$$\mathbf{g} = \left(2^0, 2^1, \ldots, 2^{\frac{m}{n+1}-1}\right)^\intercal \qquad \text{and} \qquad \mathbf{G} = \mathbf{I}_{n+1} \otimes \mathbf{g}^\intercal = \begin{pmatrix} \overline{\mathbf{G}} \\ \boldsymbol{\iota}_{n+1}^\intercal \otimes \mathbf{g}^\intercal \end{pmatrix}$$

be the gadget vector and the gadget matrix (note that their shapes are changed), with $\overline{\mathbf{G}}$ being the first n rows of $\mathbf{G}$. The notations $\mathbf{G}^{-1}(\mathbf{p}), \mathbf{G}^{-1}(\mathbf{P})$ are for $(n+1)$-row $\mathbf{p}, \mathbf{P}$.

Circular LWE. We rely on the LWE assumption for small secrets with circular security.

Assumption 8.1 ((circular) small-secret LWE). Let $n, m, m' \leq \mathrm{poly}(\lambda)$, $q \leq 2^{\mathrm{poly}(\lambda)}$, $\sigma = \sigma(\lambda)$, $\sigma' = \sigma'(\lambda)$, and (with the argument λ omitted for those functions)

$$\begin{array}{llll}
\overline{\mathbf{A}}_{\mathrm{fhe}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, & \overline{\mathbf{A}}' \xleftarrow{\$} \mathbb{Z}_q^{n\times m'}, & \boxed{\mathbf{r} \xleftarrow{\$} \mathcal{D}^n_{\mathbb{Z},\sigma,\leq\sigma\sqrt{\lambda}}}, & \mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal, \\
\mathbf{e}_{\mathrm{fhe}} \xleftarrow{\$} \mathcal{D}^m_{\mathbb{Z},\sigma,\leq\sigma\sqrt{\lambda}}, & \mathbf{e}' \xleftarrow{\$} \mathcal{D}^{m'}_{\mathbb{Z},\sigma',\leq\sigma'\sqrt{\lambda}}, & \mathbf{R} \xleftarrow{\$} \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m}, & \\
\boldsymbol{\delta}_{\mathrm{fhe}} \xleftarrow{\$} \mathbb{Z}_q^m, & \boldsymbol{\delta}' \xleftarrow{\$} \mathbb{Z}_q^{m'}, & \boldsymbol{\Delta} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}. &
\end{array}$$

The *circular small-secret LWE assumption* $\mathsf{csLWE}_{n,m,m',q,\sigma,\sigma'}$ states that

$$\left\{\left(1^\lambda, \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}_{\mathrm{fhe}}^\intercal \end{pmatrix}, \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}_{\mathrm{fhe}}^\intercal \end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G}, \overline{\mathbf{A}}', \mathbf{r}^\intercal\overline{\mathbf{A}}' + (\mathbf{e}')^\intercal\right)\right\}_{\lambda\in\mathbb{N}}$$

$$\approx \left\{\left(1^\lambda, \underbrace{\begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \boldsymbol{\delta}_{\mathrm{fhe}}^\intercal \end{pmatrix}, \boldsymbol{\Delta}}_{\text{circular terms}}, \overline{\mathbf{A}}', (\boldsymbol{\delta}')^\intercal\right)\right\}_{\lambda\in\mathbb{N}}.$$

---

to statistical very selective correctness, and its proof might well rely on non-uniform computational hardness.

[104] This definition of perfect correctness is equivalent to the version per Definition 2.1.

For the *small-secret LWE assumption* $\mathsf{sLWE}_{n,m',q,\sigma,\sigma'}$, the circular terms are removed from both distributions.

Lemma 8.1. *The following hardness implications are true:*

- $\mathsf{csLWE}_{n,m,m',q,\sigma,\sigma'} \Longrightarrow \mathsf{sLWE}_{n,m',q,\sigma,\sigma'} \Longrightarrow \mathsf{LWE}_{n,m',q,\sigma'}$;
- $\mathsf{LWE}_{n,\lambda n+m',q,\sigma'} \Longrightarrow \mathsf{sLWE}_{n,m',q,\sigma',\sigma'}$ *[ACPS09];*
- $\mathsf{LWE}_{n,2\lambda+n\lceil\log_2 q\rceil,q,2^{-\lambda}\sigma'} \Longrightarrow \mathsf{LWE}_{n,m',q,\sigma'}$ *for all* $m' = \mathrm{poly}(\lambda)$;
- $\mathsf{sLWE}_{n,2\lambda+n\lceil\log_2 q\rceil,q,\sigma,2^{-\lambda}\sigma'} \Longrightarrow \mathsf{sLWE}_{n,m',q,\sigma,\sigma'}$ *for all* $m' = \mathrm{poly}(\lambda)$;
- $\mathsf{csLWE}_{n,m,2\lambda+n\lceil\log_2 q\rceil,q,\sigma,2^{-\lambda}\sigma'} \Longrightarrow \mathsf{csLWE}_{n,m,m',q,\sigma,\sigma'}$ *for all* $m' = \mathrm{poly}(\lambda)$.

In addition, if $m \geq 2\lambda + n\lceil\log_2 q\rceil$, *then* $\mathsf{csLWE}_{n,m,0,q,\sigma,0}$ *implies* $\mathsf{csLWE}_{n,m,m',q,\sigma,2^{\lambda}\sigma}$ *for all* $m' = \mathrm{poly}(\lambda)$.

Here, LWE is Assumption 2.2 (not directly used in this chapter), which is equivalent to Assumption 8.1 with the distribution of $\mathbf{r}$ *changed to* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n$ *and the circular terms removed (the two equivalent versions differ in truncation of noises).*

The first implication is by ignoring additional terms. The third to the last are by the leftover hash lemma and noise flooding. Due to the presence of circular encryption, the last two reductions in Lemma 8.1 cannot alter the distribution of $\mathbf{r}$.

PARAMETERS. We rely on csLWE with the following parameters:

- n is a fixed polynomial in λ and $m = 3(n+1)\lceil\log_2 q\rceil$;[105]
- m' varies over all possible polynomials in λ;
- $\log_2 q$ is an integer and fixed polynomial in n;
- σ, σ' are fixed functions of n such that $\frac{q}{\sigma\sqrt{n}}, \frac{q}{\sigma'\sqrt{n}} = \Omega(2^{n^\rho})$ for some constant $0 < \rho < \frac{1}{2}$.

[105] This choice of m enables the reduction of *non-circular* IND-CPA security of the [GSW13] FHE scheme (used in the circular terms) to $\mathsf{LWE}_{n,m,q,\sigma'}$, providing heuristic justification for our assumption. It also satisfies the constraint in Lemma 2.3 for trapdoor generation.

By Lemma 8.1, they reduce to a single $\mathsf{csLWE}_{n,m,2\lambda+n\lceil\log_2 q\rceil,q,\sigma,2^{-\lambda}\sigma'}$ assumption, which also implies all $\mathsf{sLWE}_{n,\mathrm{poly}(\lambda),q,\sigma,\sigma'}$ and $\mathsf{LWE}_{n,\mathrm{poly}(\lambda),q,\sigma'}$. In our derivation, we might use more specific choices of parameters.

We remark that we could solely rely on quasi-polynomial modulus-to-noise ratio. Clearly, the correctness error would not go below quasi-polynomial. There is a less obvious cost. Since our security proof relies on correctness, the provable advantage bound would also be quasi-polynomial, even if the assumptions have subexponential security. We choose to not go this route. In contrast, we obtain subexponential correctness and security from subexponential modulus-to-noise ratio if the assumptions are subexponentially secure, because our reductions are polynomial-time.

Evasive LWE. We formulate the variant of evasive LWE assumption needed for our KP-ABE. While it appears complicated due to incorporation of circular ciphertext and encoding, the assumption follows the same rationale as existing variants of evasive LWE. More discussion follows the definition.

Assumption 8.2 (evasive circular small-secret LWE). Let $\mathcal{S}(1^\lambda;\mathsf{aux})$ be an algorithm that, given randomness aux, outputs

$$\overline{\mathbf{A}}_{\mathrm{circ}} \in \mathbb{Z}_q^{n\times(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\quad \overline{\mathbf{A}}' \in \mathbb{Z}_q^{n\times m'},\quad \mathbf{P}\in\mathbb{Z}_q^{n\times J},\quad \sigma,\quad \sigma',\quad \sigma_{-1},\quad \sigma_{\mathrm{post}},\quad \sigma_{\mathrm{pre}},$$

where $m \ge m_0(n,q)$ and $\sigma_{-1}\ge\sigma_0(n,m)$ are constrained by Lemma 2.3 and $\sigma_{\mathrm{post}}\ge\sigma_{\mathrm{pre}}$. Suppose $\mathbf{e}_{\mathrm{B}}\in\mathbb{Z}^m$, $\mathbf{e}_{\mathrm{P}}\in\mathbb{Z}^J$, $\boldsymbol{\delta}_{\mathrm{B}}\xleftarrow{\$}\mathbb{Z}_q^m$, $\boldsymbol{\delta}_{\mathrm{P}}\xleftarrow{\$}\mathbb{Z}_q^J$, and

$$\begin{aligned}
&\overline{\mathbf{A}}_{\mathrm{fhe}}\xleftarrow{\$}\mathbb{Z}_q^{n\times m}, && (\mathbf{B},\tau)\xleftarrow{\$}\mathsf{TrapGen}(1^n,1^m,q), && \mathbf{K}\xleftarrow{\$}\mathsf{SampD}(\mathbf{B},\tau,\mathbf{P},\sigma_{-1}),\\
&\mathbf{e}_{\mathrm{fhe}}\xleftarrow{\$}\mathcal{D}^m_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}, && \mathbf{e}_{\mathrm{circ}}\xleftarrow{\$}\mathcal{D}^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}, && \mathbf{e}'\xleftarrow{\$}\mathcal{D}^{m'}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}},\\
&\boldsymbol{\delta}_{\mathrm{fhe}}\xleftarrow{\$}\mathbb{Z}_q^m, && \boldsymbol{\delta}_{\mathrm{circ}}\xleftarrow{\$}\mathbb{Z}_q^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m}, && \boldsymbol{\delta}'\xleftarrow{\$}\mathbb{Z}_q^{m'},\\
&\mathbf{r}\xleftarrow{\$}\mathcal{D}^n_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}, && \mathbf{R}\xleftarrow{\$}\{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m}, && \mathbf{S}=\begin{pmatrix}\overline{\mathbf{A}}_{\mathrm{fhe}}\\ \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}_{\mathrm{fhe}}^{\intercal}\end{pmatrix}\mathbf{R}-\mathsf{bits}(\mathbf{s})\otimes\mathbf{G}.\\
&\mathbf{s}\leftarrow(\mathbf{r}^{\intercal},-1)^{\intercal}, && \boldsymbol{\Delta}\xleftarrow{\$}\mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}, &&
\end{aligned}$$

In the precondition, the entries of $\mathbf{e}_{\mathrm{B}},\mathbf{e}_{\mathrm{P}}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{pre}},\le\sigma_{\mathrm{pre}}\sqrt{\lambda}}$,

and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{pre}}$ states that

$$\left\{\begin{pmatrix} 1^\lambda,\ \mathsf{aux},\ \overline{\mathbf{A}}_{\mathrm{fhe}},\ \mathbf{B},\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}^\intercal_{\mathrm{fhe}},\ \mathbf{S},\\ \mathbf{r}^\intercal(\overline{\mathbf{A}}_{\mathrm{circ}}-(1,\mathsf{bits}(\mathbf{S}))\otimes\overline{\mathbf{G}})+\mathbf{e}^\intercal_{\mathrm{circ}},\\ \mathbf{r}^\intercal\overline{\mathbf{A}}'+(\mathbf{e}')^\intercal,\ \mathbf{r}^\intercal\mathbf{B}+\mathbf{e}^\intercal_{\mathrm{B}},\ \mathbf{r}^\intercal\mathbf{P}+\mathbf{e}^\intercal_{\mathrm{P}}\end{pmatrix}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda,\ \mathsf{aux},\ \overline{\mathbf{A}}_{\mathrm{fhe}},\ \mathbf{B},\ \boldsymbol{\delta}^\intercal_{\mathrm{fhe}},\ \boldsymbol{\Delta},\\ \boldsymbol{\delta}^\intercal_{\mathrm{circ}},\\ (\boldsymbol{\delta}')^\intercal,\ \boldsymbol{\delta}^\intercal_{\mathrm{B}},\ \boldsymbol{\delta}^\intercal_{\mathrm{P}}\end{pmatrix}\right\}_{\lambda\in\mathbb{N}}.$$

In the postcondition, the entries of $\mathbf{e}_{\mathrm{B}}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathrm{post}},\le\sigma_{\mathrm{post}}\sqrt{\lambda}}$, and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{post}}$ states that

$$\left\{\begin{pmatrix} 1^\lambda,\ \mathsf{aux},\ \overline{\mathbf{A}}_{\mathrm{fhe}},\ \mathbf{B},\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}^\intercal_{\mathrm{fhe}},\ \mathbf{S},\\ \mathbf{r}^\intercal(\overline{\mathbf{A}}_{\mathrm{circ}}-(1,\mathsf{bits}(\mathbf{S}))\otimes\overline{\mathbf{G}})+\mathbf{e}^\intercal_{\mathrm{circ}},\\ \mathbf{r}^\intercal\overline{\mathbf{A}}'+(\mathbf{e}')^\intercal,\ \mathbf{r}^\intercal\mathbf{B}+\mathbf{e}^\intercal_{\mathrm{B}},\quad \mathbf{K}\end{pmatrix}\right\}_{\lambda\in\mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda,\ \mathsf{aux},\ \overline{\mathbf{A}}_{\mathrm{fhe}},\ \mathbf{B},\ \boldsymbol{\delta}^\intercal_{\mathrm{fhe}},\ \boldsymbol{\Delta},\\ \boldsymbol{\delta}^\intercal_{\mathrm{circ}},\\ (\boldsymbol{\delta}')^\intercal,\ \boldsymbol{\delta}^\intercal_{\mathrm{B}},\ \mathbf{K}\end{pmatrix}\right\}_{\lambda\in\mathbb{N}}.$$

The *evasive circular small-secret LWE assumption* postulates that $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{pre}}$ implies $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{post}}$ for all efficient sampler $\mathcal{S}$.

Remarks. We briefly compare known variants of evasive LWE assumptions. Assumption 8.2 and those in [Wee22,WWW22] and Chapter 7 are only for public-coin samplers, enforced by providing the sampler randomness to the distinguisher. Those in [Tsa22, VWW22,ARYY23] consider private-coin samplers, where the auxiliary information passed to the distinguisher is an additional output of the sampler. Assuming evasive LWE only for public-coin samplers avoids obfuscation-based counterexamples, where $\mathbf{P}$ is sampled with a trapdoor and the auxiliary information contains an obfuscated program with the trapdoor hardwired. From a provable security perspective, the desideratum for public-coin variants (achieved by all such ones) is that the assumption should not suffer from any known counterexamples and should suffice for the application.

The exact formulations vary — whether the matrix $\mathbf{B}$ (with trapdoor) is public or not, whether the secret $\mathbf{S}$ is sampled by $\mathcal{S}$ (only for private-coin versions) or prescribed, specification of advantage relation[106] between precondition and postcondition, number of matrices with trapdoors, how additional LWE samples can be

[106]There are three levels of specifications. The most coarse is "polynomial indistinguishability implies polynomial indistinguishability", which is the approach taken here and in Chapter 7. The intermediate version [Wee22,WWW22,ARYY23] requires unspecified polynomial relation between the two adversaries

formed (i.e., $\mathbf{A}$), noise structure, circularity, etc. They are formulated to be natural (in the sense that they deal with preimages of general $\mathbf{A}, \mathbf{P}$, decoupled from the specific distributions of $\mathbf{A}, \mathbf{P}$ used in the application) and sufficient for the intended application. The common underlying intuition is that the trapdoor preimages are only useful in breaking pseudorandomness by its natural usage, i.e., multiplying the LWE samples under the matrix with trapdoor.

As suggested in [Wee22], the noise magnitude in the postcondition can be made larger than that in the precondition for a more conservative assumption. We can implement it by requiring evasive LWE to hold only for samplers with a certain gap between σ_{post} and σ_{pre}. We additionally allow a worse Gaussian preimage $\mathbf{K}$. Otherwise, our assumption is stronger than the version in [Wee22], and is incomparable to those in [WWW22] and Chapter 7.

We note that in our formulation, in contrast to $\mathbf{e}_{\text{B}}, \mathbf{e}_{\text{P}}$, the distributions of $\mathbf{r}, \mathbf{e}_{\text{fhe}}, \mathbf{e}_{\text{circ}}, \mathbf{e}'$ are invariant across $\mathsf{evcsLWE}_{\text{pre}}$ and $\mathsf{evcsLWE}_{\text{post}}$. Intuitively, $\mathbf{e}_{\text{B}}, \mathbf{e}_{\text{P}}$ being smaller in the precondition compensates for the fact that in the postcondition, the noise attached to $\mathbf{r}^{\intercal}\mathbf{BK}$ is not an independent $\mathbf{e}_{\text{P}}$ but correlated with $\mathbf{e}_{\text{B}}$. However, this does not apply to $\mathbf{r}, \mathbf{e}_{\text{fhe}}, \mathbf{e}_{\text{circ}}, \mathbf{e}'$. In the presence of circular encryption, a distribution-varying $\mathbf{r}$ is difficult to interpret. Lastly, the formulation must consider different magnitudes for $\mathbf{e}', \mathbf{e}_{\text{P}}$ — for correctness, $\mathbf{e}_{\text{fhe}}, \mathbf{e}_{\text{circ}}, \mathbf{e}'$ (the last contains attribute encoding noises) must be small to be successfully refreshed; operationally, the (refreshed) homomorphic noises are larger than $\mathbf{e}_{\text{circ}}$; for security, $\mathbf{e}_{\text{P}}$ must be large to flood the homomorphic noises. We thus settle for the version above.

The sampler is allowed to choose $\overline{\mathbf{A}}_{\text{circ}}$. Since the public matrices for circuits depend on it, it must be known to the sampler, either (here) chosen by or (alternatively) given to the sampler. In both cases, the structure of $(\overline{\mathbf{A}}_{\text{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G})$ must be incorporated into the assumption, because the sampler cannot choose $\mathbf{S}$. We settle for one possible version here.

---

in the two conditions. The finest level stipulates exact polynomial relation between the two adversaries, which is needed if the assumption is to be applied super-constantly many times [Tsa22,VWW22] (pointed out in [AKY24b]).

For a more complete study of evasive LWE assumptions, we refer the reader to [BÜW24].

8.2.4 Homomorphic Encryption and Evaluation à la [GSW13]

We rely on the (leveled fully) homomorphic encryption due to [GSW13]. Since we use it as a building block and the security proof of our construction will be different, we only recall the format (without the distribution) of its components and the correctness property.

Lemma 8.2 ([GSW13]). *The leveled FHE scheme works as follows.*

- *The keys are*

$$\text{(public)}\quad \mathbf{A}_{\text{fhe}} = \begin{pmatrix} \overline{\mathbf{A}}_{\text{fhe}} \\ \mathbf{r}^{\mathsf{T}}\overline{\mathbf{A}}_{\text{fhe}} + \mathbf{e}_{\text{fhe}}^{\mathsf{T}} \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m}, \qquad \text{(secret)}\quad \mathbf{s}^{\mathsf{T}} = (\mathbf{r}^{\mathsf{T}}, -1) \in \mathbb{Z}^{n+1},$$

  where $\mathbf{r} \in \mathbb{Z}^n$, $\overline{\mathbf{A}}_{\text{fhe}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_{\text{fhe}} \in \mathbb{Z}^m$.

- *A ciphertext of* $x \in \{0,1\}$ *is*

$$\mathbf{X} = \mathbf{A}_{\text{fhe}}\mathbf{R} - x\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m},$$

  where $\mathbf{R} \in \mathbb{Z}^{m\times m}$ *is the encryption randomness. The decryption equation is*

$$\mathbf{s}^{\mathsf{T}}\mathbf{X} = -\mathbf{e}_{\text{fhe}}^{\mathsf{T}}\mathbf{R} - x\mathbf{s}^{\mathsf{T}}\mathbf{G} \in \mathbb{Z}_q^m,$$

  which can be used to extract x *via multiplication by* $\mathbf{G}^{-1}(\lfloor q/2 \rceil \boldsymbol{\iota}_{n+1})$.

- *There is an efficient algorithm*

$$\mathsf{MakeHEvalCkt}(1^n, 1^m, q, C) = \mathsf{HEval}_C$$

  that takes as input n, m, q *and a circuit* $C : \{0,1\}^L \to \{0,1\}$ *and outputs a circuit*

$$\mathsf{HEval}_C(\mathbf{X}_1, \ldots, \mathbf{X}_L) = \mathbf{C}$$

  taking L *ciphertexts as input and outputting a new ciphertext* $\mathbf{C}$.

- *The depth of* HEval_C *is* $d\,\mathrm{O}(\log m \log\log q)$,[107] *where* d *is the depth of* C.
- *Suppose* $\mathbf{X}_\ell = \mathbf{A}_{\mathrm{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ *for* $\ell \in [L]$ *with* $\mathbf{x} \in \{0,1\}^L$, *then*

$$\mathbf{C} = \mathbf{A}_{\mathrm{fhe}}\mathbf{R}_C - C(\mathbf{x})\mathbf{G},$$

where $\|\mathbf{R}_C^\intercal\| \le (m+2)^d \max_{\ell\in[L]} \|\mathbf{R}_\ell^\intercal\|$.

Additionally, in the circular version, ciphertexts of $\mathsf{bits}(\mathbf{s})$ *are published.*

It will be convenient for us to extend the homomorphic evaluation procedure for vector-valued functions similarly to [BTVW17].

Lemma 8.3 (homomorphic evaluation for vector-valued functions; ¶). *For the scheme in Lemma 8.2, there is an efficient algorithm*

$$\mathsf{MakeVEvalCkt}(n, m, q, C) = \mathsf{VEval}_C$$

that takes as input n, m, q *and a vector-valued circuit* $C : \{0,1\}^L \to \mathbb{Z}_q^{1\times m'}$ *and outputs a circuit*

$$\mathsf{VEval}_C(\mathbf{X}_1, \ldots, \mathbf{X}_L) = \mathbf{C}$$

taking L *ciphertexts as input and outputting a new ciphertext* $\mathbf{C}$ *of different format.*

- *The depth of* VEval_C *is* $\big(d\,\mathrm{O}(\log m \log\log q) + \mathrm{O}(\log^2 \log q)\big)$[108] *for* C *of depth* d.
- *Suppose* $\mathbf{X}_\ell = \mathbf{A}_{\mathrm{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ *for* $\ell \in [L]$ *with* $\mathbf{x} \in \{0,1\}^L$, *then*

$$\mathbf{C} = \mathbf{A}_{\mathrm{fhe}}\mathbf{R}_C - \begin{pmatrix} \mathbf{0}_{n\times m'} \\ C(\mathbf{x}) \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m'},$$

where $\|\mathbf{R}_C^\intercal\| \le (m+2)^d \lceil \log_2 q \rceil \max_{\ell\in[L]} \|\mathbf{R}_\ell^\intercal\|$. *The new decryption equation is*

$$\mathbf{s}^\intercal\mathbf{C} = -\mathbf{e}_{\mathrm{fhe}}^\intercal\mathbf{R}_C + C(\mathbf{x}) \in \mathbb{Z}_q^{1\times m'}.$$

[107] A better bound is $\mathrm{O}(\log m + \log\log q)$. Consequently, some parameters can be tighter. We choose to keep the bound used during the initial write-up to avoid an overhaul at this time.

[108] A better bound is $\big(d\,\mathrm{O}(\log m + \log\log q) + \mathrm{O}(\log\log q)\big)$. See Footnote 107.

Proof (Lemma 8.3). Recall that the codomain of C being $\mathbb{Z}_q^{1\times m'}$ is just conceptual — pragmatically, the output is computed bit by bit. Let $C_{u,v} : \{0,1\}^L \to \{0,1\}$ be the subcircuit of C computing the v^{th} bit (low to high) of the u^{th} scalar output, i.e., $C_{u,v}(\mathbf{x}) = \mathsf{bits}(C(\mathbf{x})[1,u])[1,v]$. The algorithm MakeVEvalCkt works as follows.

1. Given n, m, q, C, it splits C into $C_{u,v}$'s and runs

$$\mathsf{HEval}_{C_{u,v}} \leftarrow \mathsf{MakeHEvalCkt}(n, m, q, C_{u,v}) \qquad \text{for all } u \in [m'] \text{ and } v \in [0, \log_2 q).$$

2. It constructs and outputs the following circuit VEval_C.

   (a) Given $\mathbf{X}_1, \ldots, \mathbf{X}_L$, first use $\mathsf{HEval}_{C_{u,v}}$ to obtain $\mathbf{C}_{u,v}$ for all u, v in parallel.

   (b) Then compute, for all u in parallel,

$$\mathbf{C}_u = \sum_v \mathbf{C}_{u,v}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1}).$$

   (c) Lastly, output $\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_{m'})$.

To analyze the depth of VEval_C, let $d_{u,v}$ be the depth of $C_{u,v}$, then $d_{u,v} \le d$ since $C_{u,v}$ is a subcircuit of C, which is of depth d. Step 2a thus is of depth $d\,\mathrm{O}(\log m \log\log q)$ by Lemma 8.2. In Step 2b, multiplication by $\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1})$, which is a standard basis vector known to MakeHEvalCkt as a constant, can be implemented by wiring the correct entries (namely, the $(n \cdot \frac{m}{n+1} + v + 1)^{\text{st}}$ column) of $\mathbf{C}_{u,v}$, hence requires no additional depth. Summing up $\lceil \log_2 q \rceil$ values in $\mathbb{Z}_q$ takes $\mathrm{O}(\log^2 \log q)$[109] depth. Step 2c simply specifies the output, making no contribution to the depth. The bound for the total depth follows.

To see the new decryption equation and analyze $\|\mathbf{R}_C^\intercal\|$, by Lemma 8.2, we have $\mathbf{C}_{u,v} = \mathbf{A}_{\text{fhe}}\mathbf{R}_{C_{u,v}} - C_{u,v}(\mathbf{x})\mathbf{G}$, where

$$\|\mathbf{R}_{C_{u,v}}^\intercal\| \le (m+2)^{d_{u,v}} \max_{\ell\in[L]} \|\mathbf{R}_\ell^\intercal\| \le (m+2)^{d} \max_{\ell\in[L]} \|\mathbf{R}_\ell^\intercal\|.$$

[109] A better bound is $\mathrm{O}(\log\log q)$. See Footnote 107.

For each $u \in [m']$,

$$\mathbf{C}_u = \sum_v \mathbf{C}_{u,v}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1}) = \mathbf{A}_{\text{fhe}} \sum_v \mathbf{R}_{C_{u,v}}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1}) - \sum_v C_{u,v}(\mathbf{x})\mathbf{G}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1})$$
$$= \mathbf{A}_{\text{fhe}} \sum_v \mathbf{R}_{C_{u,v}}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1}) - \begin{pmatrix} \mathbf{0}_{n\times 1} \\ C(\mathbf{x})[1,u] \end{pmatrix},$$

and therefore,

$$\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_{m'}) = \mathbf{A}_{\text{fhe}}\mathbf{R}_C - \begin{pmatrix} \mathbf{0}_{n\times m'} \\ C(\mathbf{x}) \end{pmatrix},$$

$$\text{with} \qquad \mathbf{R}_C = \left(\sum_v \mathbf{R}_{C_{1,v}}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1}),\ \ \ldots,\ \ \sum_v \mathbf{R}_{C_{m',v}}\mathbf{G}^{-1}(2^v \boldsymbol{\iota}_{n+1}) \right).$$

Note that every row of $\mathbf{R}_C^\intercal$ is a sum of $\lceil \log_2 q \rceil$ rows among all the rows of $\mathbf{R}_{C_{u,v}}^\intercal$, from which the desired norm bound follows. □

8.2.5 Attribute Encoding and Homomorphic Evaluation

aux [BGG⁺14,BTVW17]

We use the attribute encoding and its homomorphic evaluation in [BGG⁺14], with the extension to matrix-valued circuits in [BTVW17]. We further extend the dual-use technique in [BTVW17] to circular encryption and vector-valued circuits.

Lemma 8.4 ([BGG⁺14,BTVW17]). *The attribute encoding and its homomorphic evaluation work as follows.*

- *For L-bit input, the public parameter is $\mathbf{A}_{\text{attr}} \in \mathbb{Z}_q^{(n+1)\times(L+1)m}$.*
- *The encoding of $\mathbf{x} \in \{0,1\}^L$ is*

  $$\mathbf{s}^\intercal(\mathbf{A}_{\text{attr}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\text{attr}}^\intercal,$$

  where $\mathbf{s}^\intercal = (\mathbf{r}^\intercal, -1)$ with $\mathbf{r} \in \mathbb{Z}^n$ and $\mathbf{e}_{\text{attr}} \in \mathbb{Z}^{(L+1)m}$.
- *There are efficient deterministic algorithms [BGG⁺14]*

  $$\mathsf{EvalC}(\mathbf{A}_{\text{attr}}, C) = \mathbf{H}_C \qquad \text{and} \qquad \mathsf{EvalCX}(\mathbf{A}_{\text{attr}}, C, \mathbf{x}) = \mathbf{H}_{C,\mathbf{x}}$$

  that take as input $\mathbf{A}_{\text{attr}}$, a circuit $C : \{0,1\}^L \to \{0,1\}$, and (for EvalCX) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m\times m}$.

  - *Suppose C is of depth d, then* $\|\mathbf{H}_C^\intercal\|, \|\mathbf{H}_{C,\mathbf{x}}^\intercal\| \le (m+2)^d$.
  - *They satisfy encoding homomorphism,*

$$(\mathbf{A}_{\text{attr}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\text{attr}}\mathbf{H}_C - C(\mathbf{x})\mathbf{G}.$$

- *There are efficient deterministic algorithms [BTVW17]*

$$\mathsf{MEvalC}(\mathbf{A}_{\text{attr}}, C) = \mathbf{H}_C \qquad \text{and} \qquad \mathsf{MEvalCX}(\mathbf{A}_{\text{attr}}, C, \mathbf{x}) = \mathbf{H}_{C,\mathbf{x}}$$

  that take as input $\mathbf{A}_{\text{attr}}$, a matrix-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{(n+1)\times m'}$, and (for MEvalCX*) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m\times m'}$.*

  - *Suppose C is of depth d, then* $\|\mathbf{H}_C^\intercal\|, \|\mathbf{H}_{C,\mathbf{x}}^\intercal\| \le (m+2)^d \lceil \log_2 q \rceil$.
  - *The matrix encoding homomorphism is*

$$(\mathbf{A}_{\text{attr}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\text{attr}}\mathbf{H}_C - C(\mathbf{x}).$$

Dual-Use Technique and Extension. In [BTVW17], the attribute encoded with secret $\mathbf{s}^\intercal$ is FHE ciphertexts under key $\mathbf{s}^\intercal$ (the same, "dual-use") and the circuit being MEvalCX'ed is some HEval_C. This leads to *automagic decryption*. Let C be a circuit with Boolean output, $\mathbf{x}$ an input, $\mathbf{X}$ a bunch of FHE ciphertexts of $\mathsf{bits}(\mathbf{x})$ under $\mathbf{s}^\intercal$, and $\mathbf{e}_{\text{attr}}, \mathbf{e}', \mathbf{e}''$ some unspecified noises, then

$$\begin{aligned}
& \left(\mathbf{s}^\intercal(\mathbf{A}_{\text{attr}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\text{attr}}^\intercal\right)\mathbf{H}_{\mathsf{HEval}_C,\mathbf{X}} \\
\text{(MEvalCX)} \quad & = \mathbf{s}^\intercal\mathbf{A}_{\text{attr}}\mathbf{H}_{\mathsf{HEval}_C} - \mathbf{s}^\intercal\mathsf{HEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \\
\text{(HEval decryption)} \quad & = \mathbf{s}^\intercal\mathbf{A}_{\text{attr}}\mathbf{H}_{\mathsf{HEval}_C} - \mathbf{s}^\intercal C(\mathbf{x})\mathbf{G} + (\mathbf{e}'')^\intercal \\
& = \mathbf{s}^\intercal(\mathbf{A}_{\text{attr}}\mathbf{H}_{\mathsf{HEval}_C} - C(\mathbf{x})\mathbf{G}) + (\mathbf{e}'')^\intercal.
\end{aligned}$$

To extend the dual-use technique to vector-valued circuits, let the codomain of C be $\mathbb{Z}_q^{1\times m'}$, then VEval_C is $\mathbb{Z}_q^{(n+1)\times m'}$-valued and

$$\begin{aligned}
& \left(\mathbf{s}^\intercal(\mathbf{A}_{\text{attr}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\text{attr}}^\intercal\right)\mathbf{H}_{\mathsf{VEval}_C,\mathbf{X}} \\
\text{(MEvalCX)} \quad & = \mathbf{s}^\intercal\mathbf{A}_{\text{attr}}\mathbf{H}_{\mathsf{VEval}_C} - \mathbf{s}^\intercal\mathsf{VEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \\
\text{(VEval decryption)} \quad & = \mathbf{s}^\intercal\mathbf{A}_{\text{attr}}\mathbf{H}_{\mathsf{VEval}_C} - C(\mathbf{x}) + (\mathbf{e}'')^\intercal.
\end{aligned}$$

Extension to circular encryption means setting $\mathbf{x} = \mathbf{s}$, for which we say $\mathbf{S}, \mathbf{A}_{\text{circ}}$ in place of $\mathbf{X}, \mathbf{A}_{\text{attr}}$.

8.3 Bootstrapping Homomorphic Evaluation

We first introduce some lemmas that will come handy later.

Lemma 8.5 (¶). *Fix q and $\mathbf{h} \in \mathbb{Z}^{m'}$. Let $g = \gcd(q, \mathbf{h}[1], \ldots, \mathbf{h}[m'])$ and $\mathbf{d} \xleftarrow{\$} \mathbb{Z}_q^{m'}$, then $\mathbf{d}^\intercal \mathbf{h}$ is uniformly random over*

$$g\mathbb{Z}_q = \{ \underbrace{g \cdot x}_{\text{multiplication over } \mathbb{Z}_q} \mid x \in \mathbb{Z}_q \} = \{ \underbrace{g \cdot (x \bmod (q/g))}_{\substack{\text{multiplication over } \mathbb{Z} \\ \text{then sent into } \mathbb{Z}_q}} \mid x \in \mathbb{Z}_{q/g} \}.$$

Proof (Lemma 8.5). Let i run through $[m']$. By Bézout's lemma, there exist $z_0, z_1, \ldots, z_{m'} \in \mathbb{Z}$ with

$$z_0 q + \sum_i z_i \mathbf{h}[i] = g.$$

Let $x \xleftarrow{\$} \mathbb{Z}_q$ and $\mathbf{d}'[i] = \mathbf{d}[i] - x z_i$, then $\mathbf{d}'$ is independent of x. Note that

$$\begin{aligned} \mathbf{d}^\intercal \mathbf{h} = gx + \mathbf{d}^\intercal \mathbf{h} - gx &= gx + \sum_i \mathbf{d}[i]\mathbf{h}[i] - x\left(z_0 q + \sum_i z_i \mathbf{h}[i]\right) \\ &= gx - \underbrace{qxz_0}_{=0} + \sum_i (\mathbf{d}[i] - xz_i)\mathbf{h}[i] = gx + (\mathbf{d}')^\intercal \mathbf{h}. \end{aligned}$$

Clearly, $(\mathbf{d}')^\intercal \mathbf{h} \in g\mathbb{Z}_q$. Since gx is uniformly random over $g\mathbb{Z}_q$ and $(\mathbf{d}')^\intercal \mathbf{h}$ is independent of x, we conclude that $\mathbf{d}^\intercal \mathbf{h} = gx + (\mathbf{d}')^\intercal \mathbf{h}$ follows the uniform distribution over $g\mathbb{Z}_q$. □

Lemma 8.6 (¶). *Let $B, B', n, m', q, \sigma, \sigma', p > 0$ and $h : \mathbb{Z}_q^{(n+1)\times m'} \xrightarrow{\$} \mathbb{Z}^{m'}$ be efficiently computable. Suppose $p \mid q$, $\Pr[\, \|(h(\mathbf{A}))^\intercal\| \le B' \,] = 1$, and $\mathsf{sLWE}_{n,m',q,\sigma,\sigma'}$ (Assumption 8.1) holds, then*

$$\Pr\left[\begin{array}{l} \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times m'} \\ \mathbf{r} \xleftarrow{\$} \mathcal{D}^n_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}} \\ \mathbf{h} \xleftarrow{\$} h(\mathbf{A}) \end{array} : \begin{array}{r} ((\mathbf{r}^\intercal, -1)\mathbf{A}\mathbf{h} \bmod p) \\ \in \left[-\frac{p}{2} + B, \frac{p}{2} - B\right) \end{array}\right] \ge 1 - \frac{2B + (2\sigma'\sqrt{\lambda} + 1)B'}{p} - \mathrm{negl}(\lambda).$$

Proof (Lemma 8.6). Let

$$\mathbf{A} = \begin{pmatrix} \overline{\mathbf{A}}' \\ \underline{\mathbf{a}}^\intercal \end{pmatrix}, \quad \mathbf{e} \xleftarrow{\$} \mathcal{D}^{m'}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}, \quad \boldsymbol{\delta} \xleftarrow{\$} \mathbb{Z}_q^{m'}, \quad \widetilde{B} = B + B'\sigma'\sqrt{\lambda}, \quad E_0 : |\mathbf{e}^\intercal \mathbf{h}| \le B'\sigma'\sqrt{\lambda},$$

$$E_1 : \qquad ((\mathbf{r}^\intercal, -1)\mathbf{A}\mathbf{h} \bmod p) \in \left[-\frac{p}{2} + B, \frac{p}{2} - B\right),$$

$$E_2: \quad ((\mathbf{r}^\mathsf{T}\overline{\mathbf{A}}' + \mathbf{e}^\mathsf{T} - \underline{\mathbf{a}}^\mathsf{T})\mathbf{h} \bmod p) \in \left[-\frac{p}{2} + \widetilde{B}, \frac{p}{2} - \widetilde{B}\right),$$
$$E_3: \quad ((\boldsymbol{\delta}^\mathsf{T} - \underline{\mathbf{a}}^\mathsf{T})\mathbf{h} \bmod p) \in \left[-\frac{p}{2} + \widetilde{B}, \frac{p}{2} - \widetilde{B}\right).$$

Note that $(\mathbf{r}^\mathsf{T}\overline{\mathbf{A}}' + \mathbf{e}^\mathsf{T} - \underline{\mathbf{a}}^\mathsf{T})\mathbf{h} = (\mathbf{r}^\mathsf{T}, -1)\mathbf{A}\mathbf{h} + \mathbf{e}^\mathsf{T}\mathbf{h}$, so $E_1 \supseteq E_0 \cap E_2$. Combining this with $\Pr[E_0] = 1$, we have

$$\Pr[E_1] \geq \Pr[E_0 \cap E_2] = \Pr[E_2].$$

It follows from $\mathsf{sLWE}_{n,m',q,\sigma,\sigma'}$ that $\Pr[E_2] \geq \Pr[E_3] - \mathrm{negl}(\lambda)$. (Roughly speaking, the reduction algorithm receives $\overline{\mathbf{A}}'$ and either $(\mathbf{r}^\mathsf{T}\overline{\mathbf{A}}' + \mathbf{e}^\mathsf{T})$ or random $\boldsymbol{\delta}^\mathsf{T}$, and performs the range testing after sampling $\underline{\mathbf{a}}, \mathbf{h}$.)

Moving to modulus p, write $g = \gcd(p, \mathbf{h}[1], \ldots, \mathbf{h}[m'])$ and let $\mathbf{d} \in \mathbb{Z}_p^{m'}$ be such that $\mathbf{d} = (\boldsymbol{\delta} - \underline{\mathbf{a}}) \bmod p$, then $\mathbf{d}$ is uniformly random and independent of $\mathbf{h}, g$. By Lemma 8.5, conditioned on g, the quantity $\mathbf{d}^\mathsf{T}\mathbf{h}$ is uniform over $g\mathbb{Z}_p$. Since $\|\mathbf{h}^\mathsf{T}\| \leq B'$, either $g = p$ (if $\mathbf{h} = \mathbf{0}$) or $g \leq \max|\mathbf{h}[i]| \leq B'$. We proceed with a case analysis.

- ($g = p$.) In this case, $\mathbf{d}^\mathsf{T}\mathbf{h} \bmod p = 0$, so $\Pr[\, E_3 \mid g = p \,] = 1 \geq 1 - \frac{B'+2\widetilde{B}}{p}$.
- ($g = g_0 \leq B'$.) In this case, $\frac{\mathbf{d}^\mathsf{T}\mathbf{h} \bmod p}{g_0}$ is uniformly random over $\left[-\frac{p}{2g_0}, \frac{p}{2g_0}\right)$, so

$$\begin{aligned}
\Pr[\, E_3 \mid g = g_0 \,] &= \Pr\left[u \xleftarrow{\$} \left[-\frac{p}{2g_0}, \frac{p}{2g_0}\right) \,:\, u \in \left[-\frac{p}{2g_0} + \frac{\widetilde{B}}{g_0}, \frac{p}{2g_0} - \frac{\widetilde{B}}{g_0}\right)\right] \\
&\geq \frac{\lceil (p - 2\widetilde{B})/2g_0 \rceil - \lceil -(p - 2\widetilde{B})/2g_0 \rceil}{p/g_0} > \frac{2(p - 2\widetilde{B})/2g_0 - 1}{p/g_0} \\
&\geq 1 - \frac{g_0 + 2\widetilde{B}}{p} \geq 1 - \frac{B' + 2\widetilde{B}}{p}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Pr[E_1] \geq \Pr[E_2] \geq \Pr[E_3] - \mathrm{negl}(\lambda) &= \mathbb{E}[\, \Pr[\, E_3 \mid g \,] \,] - \mathrm{negl}(\lambda) \\
&\geq 1 - \frac{B' + 2\widetilde{B}}{p} - \mathrm{negl}(\lambda) = 1 - \frac{2B + (2\sigma'\sqrt{\lambda} + 1)B'}{p} - \mathrm{negl}(\lambda). \qquad \square
\end{aligned}$$

Lemma 8.7 (¶). *For all $u, v \in \mathbb{Z}$ and $w \in \mathbb{Z}_+$,*

$$\left\lfloor \frac{u+v}{w} \right\rfloor = \left\lfloor \frac{u}{w} \right\rfloor \quad \text{if and only if} \quad (u+v) \bmod w = (u \bmod w) + v.$$

Proof (Lemma 8.7). Taking the difference between the two sides,

$$\left\lfloor \frac{u+v}{w} \right\rceil - \left\lfloor \frac{u}{w} \right\rceil = \frac{(u+v) - \big((u+v) \bmod w\big)}{w} - \frac{u - (u \bmod w)}{w}$$
$$= \frac{\big((u \bmod w) + v\big) - \big((u+v) \bmod w\big)}{w}.$$

The lemma follows from the equivalence of being zero for both sides. □

Notations and Parameters. Let

- q be the modulus,
- M the rounding resolution,
- B the bound of removable noises (also a loose bound on the secret),
- B' the bound of homomorphism matrices for one step of computation, and
- σ, σ' Gaussian widths.

For the purpose of this section, they must satisfy these relations:

- M is an exact power of two (so that M divides a large portion of $\mathbf{G}$) and $M \mid q$;
- $M^2 B/q$, $\sigma' B'/q$, B/M, $\sigma' B'/M$ are negligible;
- B' is $2^{\Omega(\log^6 \lambda)}$ and $\sigma\sqrt{\lambda} \le 2^{-\lambda} B$ and $\sigma'\sqrt{\lambda} \le 2^{-\lambda} B$.

For the applications except KP-ABE, we need another Gaussian width σ_{msg} and require

$$\sigma_{\mathrm{msg}}\sqrt{\lambda} + Bm + 1 < q/4, \qquad \sigma_{\mathrm{msg}} \ge 2^{\lambda+6}(Bm + \sigma'\sqrt{\lambda}).$$

For KP-ABE, we need $\sigma_{-1}, \sigma_{\mathrm{pre}}, \sigma_{\mathrm{post}}$ and require[110]

$$m \cdot \sigma_{\mathrm{post}}\sqrt{\lambda} \cdot \sigma_{-1}\sqrt{m} + \sigma'\sqrt{\lambda} + Bm + 1 < q/4,$$
$$\sigma_{\mathrm{pre}} \ge 2^{\lambda+6}(Bm + 2\sigma'\sqrt{\lambda}), \qquad \sigma_{\mathrm{post}} \ge \sigma_{\mathrm{pre}}.$$

[110] In the conference version [HLL23a], the inequality lower bounding σ_{pre} is different. Both are correct and the concrete values below are not changed. The current bound enables the transition $\mathsf{H}_2 \approx_s \mathsf{H}_3$ in the proof of Theorem 8.16 by invoking Lemma 2.2 once. The conference version requires two invocations.

We rely on $\mathsf{csLWE}_{n,m,2\lambda+n\lceil\log_2 q\rceil,q,\sigma,2^{-\lambda}\sigma'}$ (Assumption 8.1) for some $n \le \mathrm{poly}(\lambda)$ (dependent on ρ; see discussion after Lemma 8.1) and $m = 3(n+1)\lceil\log_2 q\rceil$. By Lemma 8.1, it implies $\mathsf{csLWE}_{n,m,m',q,\sigma,\sigma''}$, $\mathsf{sLWE}_{n,m',q,\sigma,\sigma''}$, $\mathsf{LWE}_{n,m',q,\sigma''}$ for all $m' \le \mathrm{poly}(\lambda)$ and $\sigma'' \ge \sigma'$.

We always write $\mathbf{s}^\intercal = (\mathbf{r}^\intercal, -1)$ for $\mathbf{r} \in \mathbb{Z}^n$. We define "left/right" gadgets

$$\mathbf{g}_{\mathrm{L}}^\intercal = (2^0, 2^1, 2^2, \ldots, M/2), \qquad \mathbf{G}_{\mathrm{L}} = \mathbf{I}_{n+1} \otimes \mathbf{g}_{\mathrm{L}}^\intercal,$$
$$\mathbf{g}_{\mathrm{R}}^\intercal = (M, 2M, 4M, \ldots, 2^{\frac{m}{n+1}-1}), \qquad \mathbf{G}_{\mathrm{R}} = \mathbf{I}_{n+1} \otimes \mathbf{g}_{\mathrm{R}}^\intercal,$$

then $\mathbf{g}^\intercal = (\mathbf{g}_{\mathrm{L}}^\intercal, \mathbf{g}_{\mathrm{R}}^\intercal)$ and $\mathbf{G} = (\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}})\mathbf{Q}$ for some efficiently computable permutation matrix $\mathbf{Q}$.

Concretely, for sufficiently large λ, we can set (exact numbers)

$$q = 2^{15\lambda}, \quad M = 2^{5\lambda}, \quad B = 2^{4\lambda}, \quad B' = 2^{\lambda}, \quad \sigma = 2^{\lambda}, \quad \sigma' = 2^{2\lambda},$$
$$\sigma_{\mathrm{msg}} = 2^{6\lambda}, \quad \sigma_{-1} = 2^{\lambda}, \quad \sigma_{\mathrm{pre}} = 2^{6\lambda}, \quad \sigma_{\mathrm{post}} = 2^{7\lambda},$$

and choose n appropriately. (The proofs in this section work with the general relations, not this particular version, and some are stated for conditions even weaker than the general ones set above.)

8.3.1 Noise Removal

In this subsection, we present the procedure $\mathsf{RemoveNoise}(\cdot)$. It takes as input an attribute encoding $(\mathbf{s}^\intercal(\mathbf{A} - x\mathbf{G}) + \mathbf{e}^\intercal) \in \mathbb{Z}_q^m$ with large noise $\mathbf{e}$. The deterministic algorithm, with overwhelming probability over the input, outputs a noiseless encoding $(\mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) - x\mathbf{s}^\intercal\mathbf{G}) \in \mathbb{Z}_q^m$, where the function $\mathsf{RndPad}_{\mathbf{A}}(\cdot)$ is fully described by $\mathbf{A}$ (not dependent on $x, \mathbf{e}$).

Construction 8.1 (noise removal). $\mathsf{RemoveNoise}(\mathbf{u}^\intercal \in \mathbb{Z}_q^{1\times m})$ computes

$$\mathbf{v}_{\mathrm{L}}^\intercal \leftarrow \mathbf{u}^\intercal\mathbf{G}^{-1}(M\mathbf{G}_{\mathrm{L}}), \quad \mathbf{v}_{\mathrm{R}}^\intercal \leftarrow \mathbf{u}^\intercal\mathbf{G}^{-1}(\mathbf{G}_{\mathrm{R}}), \quad \mathbf{w}^\intercal \leftarrow \left(\left\lfloor \frac{\mathbf{v}_{\mathrm{L}}^\intercal \bmod q}{M} \right\rceil, M\left\lfloor \frac{\mathbf{v}_{\mathrm{R}}^\intercal \bmod q}{M} \right\rceil\right)\mathbf{Q},$$

and outputs $\mathbf{w}^\intercal$ as an element of $\mathbb{Z}_q^m$. The function $\mathsf{RndPad}_{\mathbf{A}}(\mathbf{s})$ is $\mathsf{RemoveNoise}(\mathbf{s}^\intercal\mathbf{A})$.

Lemma 8.8. *A canonical Boolean circuit of* $\mathsf{RndPad}_{\mathbf{A}}(\cdot)$ *of depth* $\mathrm{O}(\log n \log\log q)$[111] *can be efficiently generated from* $\mathbf{A} \in \mathbb{Z}_q^{(n+1)\times m}$.

[111] A better bound is $\mathrm{O}(\log n + \log\log q)$. See Footnote 107.

RemoveNoise satisfies the following conditional correctness property.

Theorem 8.9 (¶). *Let* $\mathbf{\Gamma} = \mathbf{G}^{-1}(M\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}})$. *For all* $\mathbf{s}, \mathbf{A}$ *such that* $\|\mathbf{s}\| \le B$, *if both*

$$(\mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{\Gamma} \bmod q)^{\mathsf{T}} \in \left[-\frac{q}{2} + M^2B, \frac{q}{2} - M^2B\right)^m \text{ and } (\mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{\Gamma} \bmod M)^{\mathsf{T}} \in \left[-\frac{M}{2} + B, \frac{M}{2} - B\right)^m$$

hold, then for all $x \in \{0,1\}$ *and* $\mathbf{e}$ *such that* $\|\mathbf{e}\| \le B$,

$$\mathsf{RemoveNoise}(\mathbf{s}^{\mathsf{T}}(\mathbf{A} - x\mathbf{G}) + \mathbf{e}^{\mathsf{T}}) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) - x\mathbf{s}^{\mathsf{T}}\mathbf{G}.$$

Proof (Theorem 8.9). We simply expand RemoveNoise. Letting $\mathbf{u}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}(\mathbf{A} - x\mathbf{G}) + \mathbf{e}^{\mathsf{T}}$ and $\mathbf{v}^{\mathsf{T}} = \mathbf{u}^{\mathsf{T}}\mathbf{\Gamma}$, we can rewrite $\mathbf{v}$ as

$$\begin{aligned}(\mathbf{v}_{\mathrm{L}}^{\mathsf{T}}, \mathbf{v}_{\mathrm{R}}^{\mathsf{T}}) &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A} - x\mathbf{G}) + \mathbf{e}^{\mathsf{T}})\mathbf{G}^{-1}(M\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}}) \\ &= (\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} - Mx\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{L}} + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}},\ \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{R}} - x\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{R}} + \mathbf{e}_{\mathrm{R}}^{\mathsf{T}}),\end{aligned} \tag{8.2}$$

where $(\mathbf{A}_{\mathrm{L}}, \mathbf{A}_{\mathrm{R}}) = \mathbf{A}\mathbf{\Gamma}$ and $(\mathbf{e}_{\mathrm{L}}^{\mathsf{T}}, \mathbf{e}_{\mathrm{R}}^{\mathsf{T}}) = \mathbf{e}^{\mathsf{T}}\mathbf{\Gamma}$ are blocked correspondingly. Because M is an exact power of two and $M^2 \le q$, each column of $\mathbf{\Gamma} = \mathbf{G}^{-1}(M\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}})$ is a standard basis vector, which implies $\|\mathbf{\Gamma}^{\mathsf{T}}\| \le 1$. Since $(\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}})\mathbf{Q} = \mathbf{G}$, it suffices to prove both of

$$\begin{aligned}&\text{(left part)} && \left\lfloor \frac{\mathbf{v}_{\mathrm{L}}^{\mathsf{T}} \bmod q}{M} \right\rfloor = \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod q}{M} \right\rfloor - x\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{L}}, \\ &\text{(right part)} && M\left\lfloor \frac{\mathbf{v}_{\mathrm{R}}^{\mathsf{T}} \bmod q}{M} \right\rfloor \equiv M\left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{R}} \bmod q}{M} \right\rfloor - x\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{R}} \pmod q.\end{aligned}$$

For the left part, observe that

$$\|(-Mx\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{L}} + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}})^{\mathsf{T}}\| \le M \cdot \|\mathbf{s}\| \cdot \|\mathbf{G}_{\mathrm{L}}^{\mathsf{T}}\| + \|\mathbf{e}\| \cdot \|\mathbf{\Gamma}^{\mathsf{T}}\| \le M \cdot B \cdot M/2 + B \cdot 1 \le M^2B.$$

Combining it with the first premise, we obtain

$$(\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} - Mx\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{L}} + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}}) \bmod q = (\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod q) - Mx\mathbf{s}^{\mathsf{T}}\mathbf{G}_{\mathrm{L}} + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}}. \tag{8.3}$$

By the second premise and the fact that $\|\mathbf{e}_{\mathrm{L}}\| \le \|\mathbf{e}\| \cdot \|\mathbf{\Gamma}^{\mathsf{T}}\| \le B$,

$$\begin{aligned}((\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod q) + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}}) \bmod M &= ((\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod M) + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}}) \bmod M \\ &= (\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod M) + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}} = ((\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod q) \bmod M) + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}}.\end{aligned}$$

Applying Lemma 8.7 to it,

$$\left\lfloor \frac{(\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod q) + \mathbf{e}_{\mathrm{L}}^{\mathsf{T}}}{M} \right\rfloor = \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{L}} \bmod q}{M} \right\rfloor. \tag{8.4}$$

Now we have

$$\left\lfloor \frac{\mathbf{v}_{\mathrm{L}}^{\intercal} \bmod q}{M} \right\rceil \overset{(8.2)}{=\!=} \left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{L}} - Mx\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{L}} + \mathbf{e}_{\mathrm{L}}^{\intercal}) \bmod q}{M} \right\rceil \overset{(8.3)}{=\!=} \left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{L}} \bmod q) - Mx\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{L}} + \mathbf{e}_{\mathrm{L}}^{\intercal}}{M} \right\rceil$$
$$= \left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{L}} \bmod q) + \mathbf{e}_{\mathrm{L}}^{\intercal}}{M} \right\rceil - x\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{L}} \overset{(8.4)}{=\!=} \left\lfloor \frac{\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{L}} \bmod q}{M} \right\rceil - x\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{L}}.$$

This completes the proof for the left part.

For the right part, since $M \mid q$, for all $k, u \in \mathbb{Z}$, it holds that

$$M\left\lfloor \frac{u}{M} \right\rceil \equiv M\left\lfloor \frac{u}{M} \right\rceil + kq = M\left\lfloor \frac{u}{M} + k \cdot \frac{q}{M} \right\rceil \equiv M\left\lfloor \frac{u + kq}{M} \right\rceil \pmod q. \tag{8.5}$$

Therefore,

$$M\left\lfloor \frac{\mathbf{v}_{\mathrm{R}}^{\intercal} \bmod q}{M} \right\rceil \overset{(8.2)}{=\!=} M\left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{R}} - x\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{R}} + \mathbf{e}_{\mathrm{R}}^{\intercal}) \bmod q}{M} \right\rceil$$
$$\overset{(8.5)}{=\!=} M\left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{R}} \bmod q) - x\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{R}} + \mathbf{e}_{\mathrm{R}}^{\intercal}}{M} \right\rceil$$
$$(\text{by } M \mid \mathbf{G}_{\mathrm{R}}) \quad = M\left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{R}} \bmod q) + \mathbf{e}_{\mathrm{R}}^{\intercal}}{M} \right\rceil - x\mathbf{s}^{\intercal}\mathbf{G}_{\mathrm{R}} \pmod q.$$

By the second premise, the fact that $\|\mathbf{e}_{\mathrm{R}}\| \le \|\mathbf{e}\| \cdot \|\mathbf{\Gamma}^{\intercal}\| \le B$, and Lemma 8.7,

$$\left\lfloor \frac{(\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{R}} \bmod q) + \mathbf{e}_{\mathrm{R}}^{\intercal}}{M} \right\rceil = \left\lfloor \frac{\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{R}} \bmod q}{M} \right\rceil.$$

This completes the proof for the right part. □

8.3.2 Bootstrapping

In the previous subsection, we showed how to remove noise from an attribute encoding, yet the noiseless encoding is not amenable to further homomorphism. To support unbounded evaluation, we transform it back to an attribute encoding with smaller noise (magnitude independent of that of the old encoding). This is done with the help of a circular encoding of the secret.

Theorem 8.10 (bootstrapping; ¶). *It works as follows.*

- *The secret is* $\mathbf{s} = (\mathbf{r}^{\intercal}, -1)^{\intercal} \in \mathbb{Z}^{n+1}$ *with* $\mathsf{bits}(\mathbf{s}) \in \{0,1\}^{(n+1)\lceil \log_2 q \rceil}$.
- *The circular ciphertext is*

$$\mathbf{S} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}_{\mathrm{fhe}}^{\intercal} \end{pmatrix} (\mathbf{R}_1, \ldots, \mathbf{R}_{(n+1)\lceil \log_2 q \rceil}) - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log_2 q \rceil},$$

where $\overline{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{e}_{\mathsf{fhe}} \in \mathbb{Z}^m$, *and* $\mathbf{R}_\ell \in \{0,1\}^{m \times m}$ *for* $1 \le \ell \le (n+1)\lceil \log_2 q \rceil$.

Let $L_{\mathsf{S}} = m(n+1)^2 \lceil \log_2 q \rceil^2$ *be the bit-length of* $\mathbf{S}$ *so that* $\mathsf{bits}(\mathbf{S}) \in \{0,1\}^{1 \times L_{\mathsf{S}}}$.

- *The circular encoding is*

$$\mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal,$$

where $\mathbf{A}_{\mathsf{circ}} \in \mathbb{Z}_q^{(n+1) \times (L_{\mathsf{S}}+1)m}$ *and* $\mathbf{e}_{\mathsf{circ}} \in \mathbb{Z}^{(L_{\mathsf{S}}+1)m}$.

- *There are efficient deterministic algorithms*

$$\mathsf{EvalRndPad}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}) = \mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}} \quad \text{and} \quad \mathsf{EvalRndPadS}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}, \mathbf{S}) = \mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}}$$

that take as input $\mathbf{A}_{\mathsf{circ}}$, *a target matrix* $\mathbf{A} \in \mathbb{Z}_q^{(n+1) \times m}$, *and (for* EvalRndPadS*) some* $\mathbf{S}$, *and output some matrix in* $\mathbb{Z}^{(L_{\mathsf{S}}+1)m \times m}$. *It holds that*

$$\|(\mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}})^\intercal\|, \|(\mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}})^\intercal\| \le (m+2)^{\mathrm{O}(\log n \log m \log^2 \log q)} \le 2^{\mathrm{O}(\log^5 \lambda)}.$$

Moreover, when $\mathbf{S}$ *is indeed of the correct form,*

$$\mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) \mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}} = \mathbf{s}^\intercal \mathbf{A}_{\mathsf{circ}} \mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) + \mathbf{e}_{\mathsf{fhe}}^\intercal \mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}},$$

for $\mathsf{RndPad}_{\mathbf{A}}$ *defined in Construction 8.1 with*

$$\|\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}^\intercal\| \le (m+2)^{\mathrm{O}(\log n \log \log q)} \cdot \mathrm{O}(\log q) \le 2^{\mathrm{O}(\log^4 \lambda)}.$$

Before proceeding to the proof, we remark that the procedure, combined with RemoveNoise, indeed reduces the encoding noise. Suppose $\|\mathbf{e}_{\mathsf{fhe}}\|, \|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda} B$, then the output encoding is

$$\begin{aligned} &(\mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{A})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal) \mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}} \\ &= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{circ}} \mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) + \mathbf{e}_{\mathsf{fhe}}^\intercal \mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}} + \mathbf{e}_{\mathsf{circ}}^\intercal \mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}}, \end{aligned}$$

whose noise is bounded by

$$\|\mathbf{e}_{\mathsf{fhe}}\| \cdot \|\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}^\intercal\| + \|\mathbf{e}_{\mathsf{circ}}\| \cdot \|(\mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}})^\intercal\| \le B \cdot 2^{-\lambda + \mathrm{poly}(\log \lambda)}.$$

Adding it to the output of RemoveNoise, we restore the format of attribute encoding with the noise level reduced from close to B to much lower below it, which can be done after each step of homomorphic evaluation for unbounded homomorphism.

Proof (Theorem 8.10). We employ the dual-use technique from Section 8.2.5. By Lemmas 8.3 and 8.4, define

$$\begin{aligned}
\mathsf{VEval}_{\mathsf{RndPad}_{\mathbf{A}}} &= \mathsf{MakeVEvalCkt}(n, m, q, \mathsf{RndPad}_{\mathbf{A}}),\\
\mathsf{EvalRndPad}(\mathbf{A}_{\mathrm{circ}}, \mathbf{A}) &= \mathsf{MEvalC}(\mathbf{A}_{\mathrm{circ}}, \mathsf{VEval}_{\mathsf{RndPad}_{\mathbf{A}}}),\\
\mathsf{EvalRndPadS}(\mathbf{A}_{\mathrm{circ}}, \mathbf{A}, \mathbf{S}) &= \mathsf{MEvalCX}(\mathbf{A}_{\mathrm{circ}}, \mathsf{VEval}_{\mathsf{RndPad}_{\mathbf{A}}}, \mathsf{bits}(\mathbf{S})),
\end{aligned}$$

all of which can be efficiently computed. By Lemma 8.8, the depth of $\mathsf{RndPad}_{\mathbf{A}}$ is $\mathrm{O}(\log n \log\log q)$, so by Lemma 8.3, the circuit $\mathsf{VEval}_{\mathsf{RndPad}_{\mathbf{A}}}$ is of depth

$$\mathrm{O}(\log n \log\log q) \cdot \mathrm{O}(\log m \log\log q) + \mathrm{O}(\log^2 \log q) = \mathrm{O}(\log n \log m \log^2 \log q).$$

The desired norm bounds of $\mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}}$ and $\mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}}$ follow from Lemma 8.4.

When $\mathbf{S}$ is in the correct form,

$$\begin{aligned}
&\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G})\mathbf{H}_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}}\\
\text{(Lemma 8.4)} \quad &= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{circ}}\mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}} - \mathbf{s}^{\mathsf{T}}\mathsf{VEval}_{\mathsf{RndPad}_{\mathbf{A}}}(\mathbf{S})\\
\text{(Lemma 8.3)} \quad &= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathrm{circ}}\mathbf{H}_{\mathbf{A}}^{\mathsf{RndPad}} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) + \mathbf{e}_{\mathrm{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}.
\end{aligned}$$

Lastly, $\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}$ satisfies (again by Lemma 8.3)

$$\begin{aligned}
\|\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}^{\mathsf{T}}\| &\le (m+2)^{\mathrm{O}(\log n \log\log q)\lceil\log_2 q\rceil} \cdot \overbrace{\max_{1\le \ell \le (n+1)\lceil\log_2 q\rceil} \|\mathbf{R}_{\ell}^{\mathsf{T}}\|}^{\le m \le m+2}\\
&\le (m+2)^{\mathrm{O}(\log n \log\log q)} \cdot \mathrm{O}(\log q).
\end{aligned}$$

□

8.3.3 Unbounded Homomorphic Evaluation

In this subsection, we present homomorphic evaluation for circuits of unbounded depth. Unlike Lemma 8.4, our evaluation procedure is non-linear, and we cannot formulate them as producing low-norm linear transformation.

Construction 8.2 (unbounded homomorphic evaluation of attribute encoding). It works as follows.

- $\mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C)$ takes as input the public matrices

$$\begin{aligned}
&\mathbf{A}_{\mathrm{attr}} \in \mathbb{Z}_q^{(n+1)\times(L+1)m} \quad \text{(for attribute encoding)} \quad \text{and}\\
&\mathbf{A}_{\mathrm{circ}} \in \mathbb{Z}_q^{(n+1)\times(L_{\mathrm{S}}+1)m} \quad \text{(for circular encoding, with } L_{\mathrm{S}} = m(n+1)^2\lceil\log_2 q\rceil^2),
\end{aligned}$$

and a circuit $C : \{0,1\}^L \to \{0,1\}^{1\times L'}$ of arbitrary size and depth. It does the following.

1. Let C_ℓ (for $\ell \in [L]$) be the input gate of C corresponding to $\mathbf{x}[\ell]$, and $C_{L+1}, \ldots, C_{|C|}$ the other gates of C in topological order. Let

$$\mathbf{A}_{\mathrm{attr}} = (\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_L),$$

   where $\mathbf{A}_0$ corresponds to the constant 1 and $\mathbf{A}_\ell$ (for $\ell \in [L]$), the input $\mathbf{x}[\ell]$, in the attribute encoding.

2. For $i = L+1, \ldots, |C|$, let fan-ins of C_i be fan-outs of $C_{i'}, C_{i''}$ for some $i', i'' < i$ (for negation gates, ignore i''). Use Lemma 8.4 to compute[112]

$$\mathbf{H}'_{C_i} \leftarrow \mathsf{EvalC}((\mathbf{A}_0, \mathbf{A}_{i'}, \mathbf{A}_{i''}), \text{ gate } C_i \text{ as one-gate circuit}),$$
$$\mathbf{A}'_i \leftarrow (\mathbf{A}_0, \mathbf{A}_{i'}, \mathbf{A}_{i''})\mathbf{H}'_{C_i},$$

   then use Theorem 8.10 to compute

$$\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i} \leftarrow \mathsf{EvalRndPad}(\mathbf{A}_{\mathrm{circ}}, \mathbf{A}'_i), \qquad \mathbf{A}_i \leftarrow \mathbf{A}_{\mathrm{circ}}\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i}.$$

3. Let $i_1, \ldots, i_{L'}$ be the indices of the output gates. Set $\mathbf{A}_{C:\ell} \leftarrow \mathbf{A}_{i_\ell}$ for all $\ell \in [L']$ and output $\mathbf{A}_C = (\mathbf{A}_{C:1}, \ldots, \mathbf{A}_{C:L'})$.

- $\mathsf{UEvalCX}(\mathbf{A}_{\mathrm{attr}}, \mathbf{c}^\intercal_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{c}^\intercal_{\mathrm{circ}}, C, \mathbf{x}, \mathbf{S})$ takes as input all the input to UEvalC, attribute encoding $\mathbf{c}_{\mathrm{attr}} \in \mathbb{Z}_q^{(L+1)m}$, circular encoding $\mathbf{c}_{\mathrm{circ}} \in \mathbb{Z}_q^{(L_S+1)m}$, attribute $\mathbf{x} \in \{0,1\}^L$, and circular ciphertext $\mathbf{S} \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log_2 q\rceil}$. It does the following.

1. Parse C into $C_1, \ldots, C_L, C_{L+1}, \ldots, C_{|C|}$ and $\mathbf{A}_{\mathrm{attr}}$ into $(\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_L)$ as in UEvalC. Split $\mathbf{c}^\intercal_{\mathrm{attr}}$ into $(\mathbf{c}^\intercal_0, \mathbf{c}^\intercal_1, \ldots, \mathbf{c}^\intercal_L)$, where $\mathbf{c}_0$ is the encoding of the constant 1 and $\mathbf{c}_\ell$ (for $\ell \in [L]$), that of input $\mathbf{x}[\ell]$.

[112] The notations are different from those used in the technical overview. Here, primes are put on matrices before bootstrapping (an implementation detail) so that matrices defined for gates (the interface if the gate is an output) do not use primes (contrary to technical overview, where primes are attached to matrices appearing later in text).

2. For $i = L+1, \ldots, |C|$, let fan-ins of C_i be fan-outs of $C_{i'}, C_{i''}$ for some $i', i'' < i$ (for negation gates, ignore i''), and $\mathbf{x}[i'], \mathbf{x}[i'']$ the wire values out of $C_{i'}, C_{i''}$. Evaluate the gate and store the result into $\mathbf{x}[i]$ (extending $\mathbf{x}$). Use Lemma 8.4 to compute

$$\mathbf{H}'_{C_i,\mathbf{x}[i'],\mathbf{x}[i'']} \leftarrow \mathsf{EvalCX}((\mathbf{A}_0, \mathbf{A}_{i'}, \mathbf{A}_{i''}), \text{ gate } C_i \text{ as one-gate circuit}, (\mathbf{x}[i'], \mathbf{x}[i''])),$$

and set $(\mathbf{c}'_i)^\intercal \leftarrow (\mathbf{c}_0^\intercal, \mathbf{c}_{i'}^\intercal, \mathbf{c}_{i''}^\intercal)\mathbf{H}'_{C_i,\mathbf{x}[i'],\mathbf{x}[i'']}$. Next, run Construction 8.1,

$$(\mathbf{c}''_i)^\intercal \leftarrow \mathsf{RemoveNoise}((\mathbf{c}'_i)^\intercal).$$

Then, use Theorem 8.10 to compute

$$\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i,\mathbf{S}} \leftarrow \mathsf{EvalRndPad}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}'_i, \mathbf{S}),$$

and set $\mathbf{c}_i^\intercal \leftarrow (\mathbf{c}''_i)^\intercal + \mathbf{c}_{\mathsf{circ}}^\intercal \mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i,\mathbf{S}}$.

3. Let $i_1, \ldots, i_{L'}$ be the indices of the output gates. Set $\mathbf{c}_{C,\mathbf{x}:\ell} \leftarrow \mathbf{c}_{i_\ell}$ for all $\ell \in [L']$ and output $\mathbf{c}_{C,\mathbf{x}}^\intercal = (\mathbf{c}_{C,\mathbf{x}:1}^\intercal, \ldots, \mathbf{c}_{C,\mathbf{x}:L'}^\intercal)$.

We expect the homomorphism property — with overwhelming probability over the input, the output satisfies $\mathbf{c}_{C,\mathbf{x}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^\intercal$ for small $\mathbf{e}_{C,\mathbf{x}}$. We consider this in two steps. We first show a sufficient condition (barring "bad event") for correctness, and then prove that "bad event" happens with negligible probability.

Theorem 8.11 (¶). *Let λ be sufficiently large. For $C : \{0,1\}^L \to \{0,1\}^{1\times L'}$ and $\mathbf{x} \in \{0,1\}^L$, suppose*

$$\mathbf{R} \in \{0,1\}^{m \times m(n+1)\lceil \log_2 q \rceil}, \qquad \mathbf{S} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{s}^\intercal \overline{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G},$$

$$\mathbf{c}_{\mathsf{attr}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{attr}} + (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{attr}}^\intercal, \qquad \mathbf{c}_{\mathsf{circ}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} + (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal,$$

$\|\mathbf{s}\| \le B$, *and* $\|\mathbf{e}_{\mathsf{fhe}}\|, \|\mathbf{e}_{\mathsf{attr}}\|, \|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda} B$. *Define* $\mathbf{\Gamma} = \mathbf{G}^{-1}(M\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}})$ *as in Theorem 8.9 and let* $\mathbf{A}_i, \mathbf{A}'_i$ *be those in Construction 8.2. If both premises*

$$(\mathbf{s}^\intercal \mathbf{A}'_i \mathbf{\Gamma} \bmod q) \in \left[-\frac{q}{2} + M^2 B, \frac{q}{2} - M^2 B\right)^m \text{ and } (\mathbf{s}^\intercal \mathbf{A}'_i \mathbf{\Gamma} \bmod M) \in \left[-\frac{M}{2} + B, \frac{M}{2} - B\right)^m$$

hold for all $i \in [L+1, |C|]$*, then the output*

$$\mathbf{A}_C = \mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C)$$
$$\text{and} \quad \mathbf{c}^{\mathsf{T}}_{C,\mathbf{x}} = \mathsf{UEvalCX}(\mathbf{A}_{\mathrm{attr}}, \mathbf{c}^{\mathsf{T}}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{c}^{\mathsf{T}}_{\mathrm{circ}}, C, \mathbf{x}, \mathbf{S}) = \mathbf{s}^{\mathsf{T}}(\mathbf{A}_C - C(\mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{C,\mathbf{x}}$$

satisfy $\|\mathbf{e}_{C,\mathbf{x}}\| \le B$.

Proof (Theorem 8.11). Evaluate $C(\mathbf{x})$ and store the intermediate wire values into the extended $\mathbf{x}$ as done in $\mathsf{UEvalCX}$. It suffices to show the stronger statement that for all $i \in [0, |C|]$,

$$\|\mathbf{e}_i\| \le 2^{-\lambda/2} B \qquad \text{with} \quad \mathbf{c}^{\mathsf{T}}_i = \mathbf{s}^{\mathsf{T}}(\mathbf{A}_i - \mathbf{x}[i]\mathbf{G}) + \mathbf{e}^{\mathsf{T}}_i.$$

We prove it by induction on i. The basis case of $i \in [0, L]$ is by assumption. For the inductive case, suppose $i \in [L+1, |C|]$. By the definitions of $\mathsf{UEvalC}, \mathsf{UEvalCX}$ and Lemma 8.4,

$$\begin{aligned}
(\mathbf{c}'_i)^{\mathsf{T}} &= (\mathbf{c}^{\mathsf{T}}_0, \mathbf{c}^{\mathsf{T}}_{i'}, \mathbf{c}^{\mathsf{T}}_{i''}) \mathbf{H}'_{C_i, \mathbf{x}[i'], \mathbf{x}[i'']} \\
&= \left(\mathbf{s}^{\mathsf{T}}\left((\mathbf{A}_0, \mathbf{A}_{i'}, \mathbf{A}_{i''}) - (1, \mathbf{x}[i'], \mathbf{x}[i'']) \otimes \mathbf{G}\right) + (\mathbf{e}^{\mathsf{T}}_0, \mathbf{e}^{\mathsf{T}}_{i'}, \mathbf{e}^{\mathsf{T}}_{i''})\right) \mathbf{H}'_{C_i, \mathbf{x}[i'], \mathbf{x}[i'']} \\
&= \mathbf{s}^{\mathsf{T}}(\mathbf{A}'_i - \mathbf{x}[i]\mathbf{G}) + \underbrace{(\mathbf{e}^{\mathsf{T}}_0, \mathbf{e}^{\mathsf{T}}_{i'}, \mathbf{e}^{\mathsf{T}}_{i''}) \mathbf{H}'_{C_i, \mathbf{x}[i'], \mathbf{x}[i'']}}_{(\mathbf{e}'_i)^{\mathsf{T}}}.
\end{aligned}$$

Since a single gate C_i is of depth 1, by Lemma 8.4, it holds that $\|\mathbf{H}'_{C_i, \mathbf{x}[i'], \mathbf{x}[i'']}\| \le m + 2$, and

$$\|\mathbf{e}'_i\| \le \max\{\|\mathbf{e}_0\|, \|\mathbf{e}_{i'}\|, \|\mathbf{e}_{i''}\|\} \cdot \|\mathbf{H}^{\mathsf{T}}_{C_i, \mathbf{x}[i'], \mathbf{x}[i'']}\| \le 2^{-\lambda/2} B \cdot (m+2) < B,$$

where the second-last inequality invokes the induction hypothesis on $0, i', i'' < i$. Combining the premises with $\|\mathbf{e}'_i\|, \|\mathbf{s}\| \le B$, Theorem 8.9 yields

$$\begin{aligned}
(\mathbf{c}''_i)^{\mathsf{T}} = \mathsf{RemoveNoise}((\mathbf{c}'_i)^{\mathsf{T}}) &= \mathsf{RemoveNoise}(\mathbf{s}^{\mathsf{T}}(\mathbf{A}'_i - \mathbf{x}[i]\mathbf{G}) + (\mathbf{e}'_i)^{\mathsf{T}}) \\
&= \mathsf{RndPad}_{\mathbf{A}'_i}(\mathbf{s}) - \mathbf{x}[i]\mathbf{s}^{\mathsf{T}}\mathbf{G}.
\end{aligned}$$

By Theorem 8.10, we have

$$\begin{aligned}
\mathbf{c}^{\mathsf{T}}_{\mathrm{circ}} \mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i, \mathbf{S}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathrm{circ}}) \mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i, \mathbf{S}} \\
&= \mathbf{s}^{\mathsf{T}} \mathbf{A}_{\mathrm{circ}} \mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i} - \mathsf{RndPad}_{\mathbf{A}'_i}(\mathbf{s}) + \mathbf{e}^{\mathsf{T}}_{\mathrm{fhe}} \mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}'_i}} + \mathbf{e}^{\mathsf{T}}_{\mathrm{circ}} \mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i, \mathbf{S}}.
\end{aligned}$$

Because we set $\mathbf{A}_i = \mathbf{A}_{\mathsf{circ}}\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i}$, it follows that

$$\begin{aligned}\mathbf{e}_i^\intercal &= \mathbf{c}_i^\intercal - \mathbf{s}^\intercal(\mathbf{A}_i - \mathbf{x}[i]\mathbf{G}) = (\mathbf{c}_i'')^\intercal + \mathbf{c}_{\mathsf{circ}}^\intercal\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i,\mathbf{S}} + (-\mathbf{s}^\intercal\mathbf{A}_i + \mathbf{x}[i]\mathbf{s}^\intercal\mathbf{G}) \\ &= (\mathsf{RndPad}_{\mathbf{A}'_i}(\mathbf{s}) - \mathbf{x}[i]\mathbf{s}^\intercal\mathbf{G}) + \big(\mathbf{s}^\intercal\mathbf{A}_{\mathsf{circ}}\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i} - \mathsf{RndPad}_{\mathbf{A}'_i}(\mathbf{s}) \\ &\qquad + \mathbf{e}_{\mathsf{fhe}}^\intercal\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}'_i}} + \mathbf{e}_{\mathsf{circ}}^\intercal\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i,\mathbf{S}}\big) + (-\mathbf{s}^\intercal\mathbf{A}_i + \mathbf{x}[i]\mathbf{s}^\intercal\mathbf{G}) \\ &= \mathbf{e}_{\mathsf{fhe}}^\intercal\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}'_i}} + \mathbf{e}_{\mathsf{circ}}^\intercal\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i,\mathbf{S}}.\end{aligned}$$

Therefore,

$$\begin{aligned}\|\mathbf{e}_i\| &\le \|\mathbf{e}_{\mathsf{fhe}}\| \cdot \|\mathbf{R}^\intercal_{\mathsf{RndPad}_{\mathbf{A}'_i}}\| + \|\mathbf{e}_{\mathsf{circ}}\| \cdot \|(\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_i,\mathbf{S}})^\intercal\| \\ &\le 2^{-\lambda}B \cdot 2^{O(\log^4\lambda)} + 2^{-\lambda}B \cdot 2^{O(\log^5\lambda)} < 2^{-\lambda/2}B,\end{aligned}$$

where the second-last inequality relies on Theorem 8.10 and the assumption that $\|\mathbf{e}_{\mathsf{fhe}}\|, \|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda}B$. This completes the induction proof. □

Assuming small-secret LWE, the premises of Theorem 8.11 hold with overwhelming probability.

Theorem 8.12 (¶). *Let $\mathcal{A}$ be an efficient adversary choosing circuit $C : \{0,1\}^L \to \{0,1\}^{1\times L'}$ and suppose $\overline{L}$ is a polynomial upper bound of L that could ever be chosen by $\mathcal{A}$. Under $\mathsf{sLWE}_{n,(\overline{L}+L_S+2)m,q,\sigma,\sigma'}$ (Assumption 8.1) and our choice of parameters,*

$$\Pr\left[\begin{array}{ll} C \xleftarrow{\$} \mathcal{A}() & \text{there exists } i \in [L+1, |C|] \text{ such that} \\ \mathbf{A}_{\mathsf{attr}} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(L+1)m} & \\ \mathbf{A}_{\mathsf{circ}} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(L_S+1)m} & (\mathbf{s}^\intercal\mathbf{A}'_i\mathbf{\Gamma} \bmod q) \notin \left[-\frac{q}{2}+M^2B, \frac{q}{2}-M^2B\right)^m \\ \mathbf{r} \xleftarrow{\$} \mathcal{D}^n_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}} & \text{or } (\mathbf{s}^\intercal\mathbf{A}'_i\mathbf{\Gamma} \bmod M) \notin \left[-\frac{M}{2}+B, \frac{M}{2}-B\right)^m \end{array}\right]$$

is negligible, where the matrices $\mathbf{A}'_i$ are those in Construction 8.2.

Combined with Theorem 8.11 and the fact that $\|\mathbf{r}\| \le \sigma\sqrt{\lambda} \le 2^{-\lambda}B$ (by our choice of parameters), it follows as a corollary that

$$\Pr\left[\begin{array}{l}\text{for all } \mathbf{e}_{\mathsf{fhe}}, \mathbf{e}_{\mathsf{attr}}, \mathbf{e}_{\mathsf{circ}} \text{ such that } \|\mathbf{e}_{\mathsf{fhe}}\|, \|\mathbf{e}_{\mathsf{attr}}\|, \|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda}B, \\ \quad \text{all } \mathbf{R} \in \{0,1\}^{m\times m(n+1)\lceil\log_2 q\rceil}, \text{ and all } \mathbf{x} \in \{0,1\}^L, \\ \qquad\qquad \text{it holds that } \|\mathbf{e}_{C,\mathbf{x}}\| \le B.\end{array}\right]$$

is overwhelming, where the probability is taken over $C, \mathbf{A}_{\text{attr}}, \mathbf{A}_{\text{circ}}, \mathbf{r}$ *as in the previous one and the variables follow the relations in Theorem 8.11.*

We emphasize that the correctness is strong in the sense that it does *not* depend on the randomness of $\mathbf{e}$'s or $\mathbf{R}$, nor the choice of $\mathbf{x}$ — only reliant on $\mathbf{s}$ (i.e., $\mathbf{r}$) and $\mathbf{A}$'s following the right distribution. Moreover, in case of non-uniform assumptions, it holds for any C of polynomially bounded size (by letting the "worst" C be the advice of $\mathcal{A}$).

Proof (Theorem 8.12). Augment the probability space in the statement with

$$i^* \xleftarrow{\$} [L+1, |C|] \qquad \text{and} \qquad j^* \xleftarrow{\$} [m],$$

define events (with i, j running through $[L+1, |C|]$ and $[m]$)

$$\begin{aligned}
E_{i,j,1} &: \quad ((\mathbf{s}^\intercal \mathbf{A}'_i \boldsymbol{\Gamma})[1,j] \bmod q) \notin \left[-\frac{q}{2} + M^2 B, \frac{q}{2} - M^2 B\right), \\
E_{i,j,2} &: \quad ((\mathbf{s}^\intercal \mathbf{A}'_i \boldsymbol{\Gamma})[1,j] \bmod M) \notin \left[-\frac{M}{2} + B, \frac{M}{2} - B\right), \\
E &: \quad \bigvee_{i=L+1}^{|C|} \bigvee_{j=1}^{m} (E_{i,0} \vee E_{i,1}),
\end{aligned}$$

and let $\overline{C} > 0$ be a polynomial upper bound of the size of C that could ever be chosen by $\mathcal{A}$, then

$$\begin{aligned}
& \Pr[\, E_{i^*,j^*,1} \vee E_{i^*,j^*,2} \mid E, C \,] \geq 1 \,/\, (|C| - L)m \geq 1 \,/\, \overline{C}m \\
\implies \quad & \Pr[\, E_{i^*,j^*,1} \vee E_{i^*,j^*,2} \,] = \mathbb{E}\big[\, \Pr[\, E_{i^*,j^*,1} \vee E_{i^*,j^*,2} \mid C \,] \,\big] \\
& \qquad\qquad \geq \mathbb{E}\,[\, \Pr[\, E \mid C \,] \,/\, \overline{C}m \,] = \Pr[E] \,/\, \overline{C}m \\
\implies \quad & \Pr[E] \leq \overline{C}m \Pr[\, E_{i^*,j^*,1} \vee E_{i^*,j^*,2} \,] \\
& \qquad\qquad \leq \overline{C}m \Pr[E_{i^*,j^*,1}] + \overline{C}m \Pr[E_{i^*,j^*,2}].
\end{aligned}$$

It suffices to show that both $\Pr[E_{i^*,j^*,1}]$ and $\Pr[E_{i^*,j^*,2}]$ are negligible. We prove it by reduction to Lemma 8.6. Let $m' = (\overline{L} + L_{\mathsf{S}} + 2)m$ and define $h : \mathbb{Z}_q^{(n+1)\times m'} \xrightarrow{\$} \mathbb{Z}^{m'}$ as follows. On input $\mathbf{A}$...

1. Sample $C \xleftarrow{\$} \mathcal{A}()$ to obtain $C : \{0,1\}^L \to \{0,1\}^{1\times L'}$. Sample $i^* \xleftarrow{\$} [L+1, |C|]$ and $j^* \xleftarrow{\$} [m]$.

2. Parse $\mathbf{A}$ as $(\mathbf{A}_{\mathsf{attr}}, \mathbf{A}_{\mathsf{circ}}, \dots)$, where $\mathbf{A}_{\mathsf{attr}} \in \mathbb{Z}_q^{(n+1)\times(L+1)m}$ and $\mathbf{A}_{\mathsf{circ}} \in \mathbb{Z}_q^{(n+1)\times(L_{\mathsf{S}}+1)m}$, with the rest ("...") ignored.

3. Run $\mathsf{UEvalC}(\mathbf{A}_{\mathsf{attr}}, \mathbf{A}_{\mathsf{circ}}, C)$ and use the intermediate values to find (low-norm) $\mathbf{H}$ such that $\mathbf{AH} = \mathbf{A}'_{i^*}$. Let i', i'' be the fan-in indices of C_{i^*}, then

$$\mathbf{A}'_{i^*} = (\mathbf{A}_0, \mathbf{A}_{i'}, \mathbf{A}_{i''})\mathbf{H}'_{C_{i^*}}.$$

   Here, $\mathbf{A}_0$ is part of $\mathbf{A}_{\mathsf{attr}}$, and the first third (by rows) of $\mathbf{H}'_{C_{i^*}}$ becomes part of $\mathbf{H}$. When $i' \le L$, the matrix $\mathbf{A}_{i'}$ is again part of $\mathbf{A}_{\mathsf{attr}}$. When $i' > L$, by definition, $\mathbf{A}_{i'} = \mathbf{A}_{\mathsf{circ}}\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_{i'}}$, and $\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_{i'}}$ multiplied by the second third of $\mathbf{H}'_{C_{i^*}}$ contributes to $\mathbf{H}$. Similar handling applies to i'', and most parts of $\mathbf{H}$ are zero-padded.

4. Split $\mathbf{H\Gamma}$ by column into $(\mathbf{h}_1, \dots, \mathbf{h}_m)$ and output $\mathbf{h} = \mathbf{h}_{j^*}$.

Clearly, h is efficient. Recall that B' is $2^{\Omega(\log^6 \lambda)}$ in our choice of parameters. By construction,

$$\|\mathbf{h}^\mathsf{T}\| \le \|\mathbf{H}^\mathsf{T}\| \cdot \|\mathbf{\Gamma}^\mathsf{T}\| \le 3 \cdot 2^{\mathrm{O}(\log^4 \lambda)} \cdot (m+2) \cdot 1 \le B',$$

where the factors come from addition among the three parts, $\mathbf{H}^{\mathsf{RndPad}}_{\mathbf{A}'_{i'}}$, $\mathbf{H}'_{C_{i^*}}$, $\mathbf{\Gamma}$, respectively. Note also that $\mathbf{s}^\mathsf{T}\mathbf{h} = (\mathbf{s}^\mathsf{T}\mathbf{A}'_{i^*}\mathbf{\Gamma})[1, j^*]$. Invoking Lemma 8.6 on h with its (p, B) being our (q, M^2B) and (M, B) yields that under $\mathsf{sLWE}_{n,m',q,\sigma,\sigma'}$,

$$\Pr[E_{i^*,j^*,1}] \le \frac{2M^2B + (2\sigma'\sqrt{\lambda}+1)B'}{q} + \mathrm{negl}(\lambda),$$
$$\Pr[E_{i^*,j^*,2}] \le \frac{2B + (2\sigma'\sqrt{\lambda}+1)B'}{M} + \mathrm{negl}(\lambda),$$

both of which are negligible under our choice of parameters. This completes the proof. □

8.3.4 Stronger Correctness

The primary version in the previous subsections requires that $\mathbf{A}$'s be chosen independently of C. It can be avoided by applying a random shift to each RemoveNoise. We explain the modifications.

- In Construction 8.1, define $\mathsf{RemoveNoise}'(\mathbf{u}^\intercal, \mathbf{w}^\intercal)$ to be

$$\left\lceil \frac{\mathbf{u}^\intercal \mathbf{G}^{-1}(M\mathbf{G}_{\mathrm{L}}, \mathbf{G}_{\mathrm{R}}) + \mathbf{w}^\intercal}{M} \right\rceil \begin{pmatrix} \mathbf{I} & \\ & M\mathbf{I} \end{pmatrix} \mathbf{Q},$$

  and define $\mathsf{RndPad}'_{\mathbf{A},\mathbf{w}}(\mathbf{s})$ to be $\mathsf{RemoveNoise}'(\mathbf{s}^\intercal \mathbf{A}, \mathbf{w}^\intercal)$. In all the succeeding constructions, propositions, and proofs, replace RemoveNoise (resp. RndPad) by RemoveNoise′ (resp. RndPad′).

- In Construction 8.2, the algorithm UEvalC is randomized. It samples $\mathbf{w}_{\mathbf{A}'_i}$ for each $\mathbf{A}'_i$ and outputs the information necessary to recover them. The algorithm UEvalCX additionally takes that information as input so that it can use the same $\mathbf{w}$'s. The $\mathbf{w}$'s can be either truly random or pseudorandom — in the latter case, the information can be just a PRF key of length some fixed $\mathrm{poly}(\lambda)$.

- In Theorems 8.9 and 8.11, change $\mathbf{s}^\intercal \mathbf{A}\boldsymbol{\Gamma}$ to $(\mathbf{s}^\intercal \mathbf{A}\boldsymbol{\Gamma} + \mathbf{w}^\intercal)$.

- In Theorem 8.12, the adversary $\mathcal{A}$ is allowed to choose C and $\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{r}$, but not $\mathbf{w}$'s. The probability is taken over the randomness of $\mathcal{A}$ and $\mathbf{w}$'s.

  - If $\mathbf{w}$'s are truly random, the statement holds unconditionally against worst-case $C, \mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{r}$ of polynomial total length.
  - If $\mathbf{w}$'s are pseudorandom (against uniform adversaries), it holds against uniform $\mathcal{A}$'s.
  - If $\mathbf{w}$'s are pseudorandom against non-uniform adversaries, it holds against worst-case $C, \mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{r}$ of polynomial total length.

The stronger correctness makes the applications *adaptively* (instead of only *selectively*) correct and enables proofs of *selective* (instead of only *very selective*) security.

Comparison with Transformation for Perfect Correctness. In Section 8.5.4, we present a transformation making our KP-ABE (constructed using unbounded homomorphic evaluation) perfectly correct. It is *incomparable* with the method in this section. The technique here works on unbounded homomorphic evaluation itself, whereas Section 8.5.4 is for specific applications. The method in this section enables

proofs of stronger security, but the transformation in Section 8.5.4 does not yield a more secure scheme. (For KP-ABE, there are multiple obstacles limiting the security to be very selective. The point is that the technique in this section removes one of the obstacles but Section 8.5.4 does not remove any.)

8.4 Attribute-Based Laconic Function Evaluation

In this section, we show how our new homomorphic evaluation algorithm helps achieving attribute-based laconic function evaluation for circuits of unbounded depth. See Section 8.6 for its implications.

The AB-LFE scheme of [QWW18] is obtained by modifying the ABE scheme of [BGG+14] and its proof does not rely on the techniques for multi-key security. Essentially the same construction using our unbounded homomorphic evaluation yields AB-LFE for circuits of unbounded depth.

Construction 8.3 (AB-LFE for circuits of unbounded depth). Let (see Definition 8.1)

$$\begin{aligned}\mathsf{Params} &= \left\{ 1^L \,\middle|\, L \in \mathbb{N} \right\},\\ P_{1^L} &= \left\{ f_C \,\middle|\, C : \{0,1\}^L \to \{0,1\}^{1\times L'} \text{ is a circuit for some } L' \in \mathbb{N} \right\},\\ f_C(\mathbf{x})[\ell] &= \neg C(\mathbf{x})[\ell] \qquad \text{for } \mathbf{x} \in \{0,1\}^L,\ \ell \in [L'].\end{aligned}$$

The function f_C is simply represented by C. Our AB-LFE works as follows.

- $\mathsf{GenCRS}(1^L)$ takes the attribute length L as input. It samples

$$\mathbf{A}_{\mathrm{attr}} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(L+1)m}, \qquad \mathbf{A}_{\mathrm{circ}} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(L_{\mathsf{S}}+1)m},$$

  and outputs $\mathsf{crs} = (\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}})$.

- $\mathsf{Compress}(\mathsf{crs}, C)$ takes as input crs and a circuit $C : \{0,1\}^L \to \{0,1\}^{1\times L'}$. The algorithm computes and outputs

$$\mathsf{digest} = (\mathbf{A}_{C:1}, \ldots, \mathbf{A}_{C:L'}) = \mathbf{A}_C \leftarrow \mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C).$$

- $\mathsf{Enc}(\mathsf{crs}, L', \mathsf{digest}, \mathbf{x}, \boldsymbol{\mu})$ takes as input crs, L', digest, an attribute $\mathbf{x} \in \{0,1\}^L$, and a multi-bit message $\boldsymbol{\mu} \in \{0,1\}^{L'}$. It samples $\mathbf{r} \xleftarrow{\$} \mathcal{D}^n_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}$ and sets $\mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal$.

The algorithm creates a circular encryption by

$$\overline{\mathbf{A}}_{\mathsf{fhe}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \qquad \mathbf{e}_{\mathsf{fhe}} \xleftarrow{\$} \mathcal{D}^m_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}, \qquad \mathbf{R} \xleftarrow{\$} \{0,1\}^{m\times m(n+1)\lceil\log_2 q\rceil},$$
$$\mathbf{A}_{\mathsf{fhe}} \leftarrow \begin{pmatrix} \overline{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix}, \qquad \mathbf{S} \leftarrow \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G},$$

and the attribute/circular encodings by

$$\mathbf{e}_{\mathsf{attr}} \xleftarrow{\$} \mathcal{D}^{(L+1)m}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}, \qquad \mathbf{c}_{\mathsf{attr}}^\intercal \leftarrow \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{attr}} - (1,\mathbf{x}^\intercal)\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{attr}}^\intercal,$$
$$\mathbf{e}_{\mathsf{circ}} \xleftarrow{\$} \mathcal{D}^{(L_S+1)m}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}, \qquad \mathbf{c}_{\mathsf{circ}}^\intercal \leftarrow \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1,\mathsf{bits}(\mathbf{S}))\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal.$$

It then samples and sets

$$\mathbf{z}_1,\dots,\mathbf{z}_{L'} \xleftarrow{\$} \mathbb{Z}_q^{n+1}, \qquad \mathbf{A}_{\mathsf{msg}} \leftarrow (\mathbf{A}_{C:1}\mathbf{G}^{-1}(\mathbf{z}_1),\dots,\mathbf{A}_{C:L'}\mathbf{G}^{-1}(\mathbf{z}_{L'})).$$

The algorithm creates the message encoding by

$$\mathbf{e}_{\mathsf{msg}} \xleftarrow{\$} \mathcal{D}^{L'}_{\mathbb{Z},\sigma_{\mathsf{msg}},\le\sigma_{\mathsf{msg}}\sqrt{\lambda}}, \qquad \mathbf{c}_{\mathsf{msg}}^\intercal \leftarrow \mathbf{s}^\intercal\mathbf{A}_{\mathsf{msg}} + \mathbf{e}_{\mathsf{msg}}^\intercal + \lfloor q/2\rceil\cdot\boldsymbol{\mu}^\intercal.$$

It outputs $\mathsf{ct} = (\mathbf{S}, \mathbf{c}_{\mathsf{attr}}, \mathbf{c}_{\mathsf{circ}}, \mathbf{z}_1,\dots,\mathbf{z}_{L'}, \mathbf{c}_{\mathsf{msg}})$.

- $\mathsf{Dec}(\mathsf{crs}, C, \mathsf{digest}, \mathbf{x}, \mathsf{ct})$ evaluates $C(\mathbf{x})$, computes

$$(\mathbf{c}_{C,\mathbf{x}:1}^\intercal,\dots,\mathbf{c}_{C,\mathbf{x}:L'}^\intercal) = \mathbf{c}_{C,\mathbf{x}}^\intercal \leftarrow \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{attr}}, \mathbf{c}_{\mathsf{attr}}^\intercal, \mathbf{A}_{\mathsf{circ}}, \mathbf{c}_{\mathsf{circ}}^\intercal, C, \mathbf{x}, \mathbf{S}),$$
$$(\mathbf{c}')^\intercal \leftarrow \left(\mathbf{c}_{\mathsf{msg}}^\intercal - \left(\mathbf{c}_{C,\mathbf{x}:1}^\intercal\mathbf{G}^{-1}(\mathbf{z}_1),\ \dots,\ \mathbf{c}_{C,\mathbf{x}:L'}^\intercal\mathbf{G}^{-1}(\mathbf{z}_{L'})\right)\right) \bmod q,$$

and sets for all $\ell \in [L']$,

$$\boldsymbol{\mu}'[\ell] \leftarrow \begin{cases} \perp, & \text{if } C(\mathbf{x})[1,\ell] = 1; \\ 0, & \text{if } C(\mathbf{x})[1,\ell] = 0 \text{ and } \mathbf{c}'[\ell] \in [-q/4, q/4); \\ 1, & \text{otherwise.} \end{cases}$$

The algorithm outputs $\boldsymbol{\mu}'$.

This scheme has a deterministic Compress, and satisfies

$$|\mathsf{crs}| = \mathrm{O}(L), \qquad |\mathsf{digest}_C| = \mathrm{O}(L'), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(L+L'),$$
$$T_{\mathsf{GenCRS}} = \mathrm{O}(L), \qquad T_{\mathsf{Compress}}, T_{\mathsf{Dec}} = \mathrm{O}(|C|), \qquad T_{\mathsf{Enc}} = \mathrm{O}(L+L').$$

Theorem 8.13 (¶). *Under* sLWE *(Assumption 8.1) suitable for Theorem 8.12, Construction 8.3 is* computationally f-selectively correct *(Definition 8.2; C must be chosen before seeing* crs*). If the assumption holds against non-uniform adversaries, then the correctness becomes statistical (still f-selective).*

Proof (Theorem 8.13). Let $\mathcal{A}$ be an efficient adversary against $\mathsf{Exp}^{f\text{-sel}}_{\text{AB-LFE}\checkmark}$ and write

$$\mathbf{c}^\intercal_{C,\mathbf{x}} = \mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{C,\mathbf{x}}, \quad \mathbf{c}^\intercal_{C,\mathbf{x}:\ell} = \mathbf{s}^\intercal(\mathbf{A}_{C:\ell} - C(\mathbf{x})[1,\ell]\mathbf{G}) + \mathbf{e}^\intercal_{C,\mathbf{x}:\ell} \text{ for } \ell \in [L'],$$

then $\mathbf{e}^\intercal_{C,\mathbf{x}} = (\mathbf{e}^\intercal_{C,\mathbf{x}:1}, \ldots, \mathbf{e}^\intercal_{C,\mathbf{x}:L'})$. For each $\ell \in [L']$, it suffices to consider the case of $C(\mathbf{x})[1,\ell] = 0$, in which during decryption,

$$\begin{aligned}\mathbf{c}'[\ell] &\equiv (\mathbf{s}^\intercal \mathbf{A}_{C:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{e}_{\text{msg}}[\ell] + \boldsymbol{\mu}[\ell] \cdot \lfloor q/2 \rceil) \\ &\quad - (\mathbf{s}^\intercal(\mathbf{A}_{C:\ell} - C(\mathbf{x})[1,\ell]\mathbf{G}) + \mathbf{e}^\intercal_{C,\mathbf{x}:\ell})\mathbf{G}^{-1}(\mathbf{z}_\ell) \\ &\equiv (\mathbf{e}_{\text{msg}}[\ell] - \mathbf{e}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell)) + \boldsymbol{\mu}[\ell] \cdot \lfloor q/2 \rceil \pmod q.\end{aligned}$$

Suppose $\|\mathbf{e}_{C,\mathbf{x}}\| \le B$, then $\|\mathbf{e}_{C,\mathbf{x}:\ell}\| \le \|\mathbf{e}_{C,\mathbf{x}}\| \le B$ and by our choice of parameters,

$$\begin{aligned}|\mathbf{e}_{\text{msg}}[\ell]| + |\mathbf{e}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell)| + |\boldsymbol{\mu}[\ell] \cdot (q/2 - \lfloor q/2 \rceil)| &\le \sigma_{\text{msg}}\sqrt{\lambda} + \|\mathbf{e}_{C,\mathbf{x}:\ell}\| \cdot \|(\mathbf{G}^{-1}(\mathbf{z}_\ell))^\intercal\| + 1 \\ &\le \sigma_{\text{msg}}\sqrt{\lambda} + B \cdot m + 1 < q/4,\end{aligned}$$

so $\mathbf{c}'[\ell] \in [-q/4, q/4)$ (i.e., $\boldsymbol{\mu}'[\ell] = 0$) if and only if $\boldsymbol{\mu}[\ell] = 0$. It remains to bound $\Pr[\,\|\mathbf{e}_{C,\mathbf{x}:\ell}\| > B\,]$. Note that

$$\|\mathbf{e}_{\text{fhe}}\| \le \sigma\sqrt{\lambda} \le 2^{-\lambda}B, \qquad \|\mathbf{e}_{\text{attr}}\|, \|\mathbf{e}_{\text{circ}}\| \le \sigma'\sqrt{\lambda} \le 2^{-\lambda}B,$$

so applying Theorem 8.12 (corollary part) to the first phase of $\mathcal{A}$ (choosing $1^L, C$) yields the desired bound.

Statistical f-selective correctness follows from non-uniform assumption, because Theorem 8.12 guarantees simultaneous correctness for all $\mathbf{x}$ (not necessarily efficiently computable after choosing C and seeing $\mathbf{A}$'s) with overwhelming probability. □

Theorem 8.14 (¶). *Under* csLWE *(Assumption 8.1) with our choice of parameters, Construction 8.3 is* very selectively secure *(Definition 8.3).*

Proof (Theorem 8.14). The simulator $\mathsf{SimEnc}(\mathsf{crs}, C, \mathsf{digest}, \mathbf{x}, \boldsymbol{\mu}')$, with $\boldsymbol{\mu}'$ over indices ℓ such that $C(\mathbf{x})[1,\ell] = 0$, works as follows.

1. Sample $\mathbf{S}, \mathbf{c}_{\text{attr}}, \mathbf{c}_{\text{circ}}, \mathbf{z}_1, \ldots, \mathbf{z}_{L'}$ uniformly at random and $\mathbf{e}_{\text{msg}} \xleftarrow{\$} \mathcal{D}^{L'}_{\mathbb{Z}, \sigma_{\text{msg}}, \leq \sigma_{\text{msg}}\sqrt{\lambda}}$.

2. Compute

$$(\mathbf{c}^\intercal_{C,\mathbf{x}:1}, \ldots, \mathbf{c}^\intercal_{C,\mathbf{x}:L'}) \leftarrow \mathsf{UEvalCX}(\mathbf{A}_{\text{attr}}, \mathbf{c}^\intercal_{\text{attr}}, \mathbf{A}_{\text{circ}}, \mathbf{c}^\intercal_{\text{circ}}, C, \mathbf{x}, \mathbf{S}),$$

   and for each $\ell \in [L']$,

$$\mathbf{c}_{\text{msg}}[\ell] \begin{cases} \leftarrow \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{e}_{\text{msg}}[\ell] + \boldsymbol{\mu}'[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 0; \\ \xleftarrow{\$} \mathbb{Z}_q, & \text{if } C(\mathbf{x})[1,\ell] = 1. \end{cases}$$

3. Output $(\mathbf{S}, \mathbf{c}_{\text{attr}}, \mathbf{c}_{\text{circ}}, \mathbf{z}_1, \ldots, \mathbf{z}_{L'}, \mathbf{c}_{\text{msg}})$.

We describe the hybrids by their changes to the previous one.

- H_0 is $\mathsf{Exp}^{\text{very-sel,real}}_{\text{AB-LFE}}$, where the ciphertext sent to $\mathcal{A}$ is from Enc. We write

$$\begin{aligned} &(\mathbf{c}^\intercal_{C,\mathbf{x}:1}, \ldots, \mathbf{c}^\intercal_{C,\mathbf{x}:L'}) = \mathsf{UEvalCX}(\mathbf{A}_{\text{attr}}, \mathbf{c}^\intercal_{\text{attr}}, \mathbf{A}_{\text{circ}}, \mathbf{c}^\intercal_{\text{circ}}, C, \mathbf{x}, \mathbf{S}) \\ &= \left(\mathbf{s}^\intercal(\mathbf{A}_{C:1} - C(\mathbf{x})[1,1]\mathbf{G}) + \mathbf{e}^\intercal_{C,\mathbf{x}:1}, \ \ldots, \ \mathbf{s}^\intercal(\mathbf{A}_{C:L'} - C(\mathbf{x})[1,L']\mathbf{G}) + \mathbf{e}^\intercal_{C,\mathbf{x}:L'}\right). \end{aligned}$$

- In H_1, for each $\ell \in [L']$, the component $\mathbf{c}_{\text{msg}}[\ell]$ is computed as

$$\begin{cases} \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) & - \mathbf{e}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{e}_{\text{msg}}[\ell] + \boldsymbol{\mu}'[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 0; \\ \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{s}^\intercal\mathbf{z}_\ell & - \mathbf{e}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{e}_{\text{msg}}[\ell] + \boldsymbol{\mu}[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 1. \end{cases}$$

  Note that $\boldsymbol{\mu}'[\ell] = \boldsymbol{\mu}[\ell]$ when $C(\mathbf{x})[1,\ell] = 0$. This is just rewriting, so $\mathsf{H}_0 \equiv \mathsf{H}_1$.

- In H_2, the experiment aborts if $\|\mathbf{e}_{C,\mathbf{x}}\| > B$. We have $\mathsf{H}_1 \approx \mathsf{H}_2$ by (the proof of) Theorem 8.13.[113]

- In H_3, using $\mathbf{e}_{\text{msg}}$ for flooding, the terms $\mathbf{e}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell)$ are removed from $\mathbf{c}_{\text{msg}}$, and a small noise $\mathbf{e}'_{\text{msg}}[\ell]$ is inserted into $\mathbf{c}_{\text{msg}}[\ell]$ if $C(\mathbf{x})[1,\ell] = 1$, i.e., $\mathbf{c}_{\text{msg}}[\ell]$ is

$$\begin{cases} \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) & + \mathbf{e}_{\text{msg}}[\ell] + \boldsymbol{\mu}'[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 0; \\ \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{s}^\intercal\mathbf{z}_\ell & + \mathbf{e}'_{\text{msg}}[\ell] + \mathbf{e}_{\text{msg}}[\ell] + \boldsymbol{\mu}[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 1; \end{cases}$$

[113] This relies on $\mathcal{A}$ being selective in C.

where $\mathbf{e}'_{\mathrm{msg}} \xleftarrow{\$} \mathcal{D}^{L'}_{\mathbb{Z},\sigma',\leq\sigma'\sqrt{\lambda}}$. We let $\mathbf{w}[\ell] = \mathbf{s}^\intercal \mathbf{z}_\ell + \mathbf{e}'_{\mathrm{msg}}[\ell]$ for $\ell \in [L']$. When the experiment does not abort, we have

$$\begin{aligned} 2^{\lambda+6}(|\mathbf{e}'[\ell]| + |\mathbf{e}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell)|) &\leq 2^{\lambda+6}(|\mathbf{e}'[\ell]| + \|\mathbf{e}_{C,\mathbf{x}}\| \cdot \|(\mathbf{G}^{-1}(\mathbf{z}_\ell))^\intercal\|) \\ &\leq 2^{\lambda+6}(\sigma'\sqrt{\lambda} + Bm) \leq \sigma_{\mathrm{msg}} \end{aligned}$$

by our choice of parameters, so $\mathsf{H}_2 \approx_s \mathsf{H}_3$ by Lemma 2.2.

- In H_4, the experiment no longer tests whether $\|\mathbf{e}_{C,\mathbf{x}}\| > B$ nor aborts. $\mathsf{H}_3 \approx \mathsf{H}_4$ by (the proof of) Theorem 8.13.

- H_5 is $\mathsf{Exp}^{\text{very-sel,sim}}_{\text{AB-LFE}}$, where $\mathbf{S}, \mathbf{c}_{\mathrm{attr}}, \mathbf{c}_{\mathrm{circ}}$ are uniformly random and

$$\mathbf{c}_{\mathrm{msg}}[\ell] = \begin{cases} \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{e}_{\mathrm{msg}}[\ell] + \boldsymbol{\mu}'[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 0; \\ \text{random}, & \text{if } C(\mathbf{x})[1,\ell] = 1. \end{cases}$$

To prove $\mathsf{H}_4 \approx \mathsf{H}_5$, we invoke csLWE while setting[114]

$$(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{z}_1, \ldots, \mathbf{z}_{L'}) = \begin{pmatrix} \overline{\mathbf{A}}' \\ \underline{\mathbf{a}}^\intercal \end{pmatrix} + \left((1, \mathbf{x}^\intercal, 1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G},\ \mathbf{0}_{(n+1)\times L'}\right),$$

where $\overline{\mathbf{A}}', \mathbf{S}$ are public matrix and circular ciphertext received from csLWE, and $\underline{\mathbf{a}}$ is sampled by the reduction algorithm. Note that in H_4,

$$(\mathbf{c}^\intercal_{\mathrm{attr}}, \mathbf{c}^\intercal_{\mathrm{circ}}, \mathbf{w}^\intercal) = \mathbf{r}^\intercal\overline{\mathbf{A}}' + (\mathbf{e}')^\intercal - \underline{\mathbf{a}}^\intercal - \left((1, \mathbf{x}^\intercal, 1, \mathsf{bits}(\mathbf{S})) \otimes \boldsymbol{\iota}^\intercal_{n+1} \otimes \mathbf{g}^\intercal,\ \mathbf{0}_{1\times L'}\right),$$

and for each $\ell \in [L']$,

$$\mathbf{c}_{\mathrm{msg}}[\ell] = \begin{cases} \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{e}_{\mathrm{msg}}[\ell] + \boldsymbol{\mu}'[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 0; \\ \mathbf{c}^\intercal_{C,\mathbf{x}:\ell}\mathbf{G}^{-1}(\mathbf{z}_\ell) + \mathbf{w}[\ell] + \mathbf{e}_{\mathrm{msg}}[\ell] + \boldsymbol{\mu}[\ell] \cdot \lfloor q/2 \rceil, & \text{if } C(\mathbf{x})[1,\ell] = 1. \end{cases}$$

The components change into the distribution in H_5 once csLWE is applied.

By hybrid argument, $\mathsf{Exp}^{\text{very-sel,real}}_{\text{AB-LFE}} \equiv \mathsf{H}_0 \approx \mathsf{H}_5 \equiv \mathsf{Exp}^{\text{very-sel,sim}}_{\text{AB-LFE}}$. □

Stronger Correctness and Security. When using unbounded homomorphic evaluation with stronger correctness, the AB-LFE schemes become *adaptively* correct (computationally or statistically) and *selectively* secure. They will use randomized Compress (see Definition 8.1).

[114] Embedding $\mathbf{x}$ into $\mathbf{A}_{\mathrm{attr}}$ relies on $\mathcal{A}$ being selective in $\mathbf{x}$.

8.5 KP-ABE for Circuits of Unbounded Depth

In this section, we show how to achieve full-fledged ABE with our unbounded homomorphic evaluation algorithms by additionally assuming the evasive (circular small-secret) LWE assumption.

8.5.1 Construction of KP-ABE

Our KP-ABE shares much similarity with AB-LFE (Construction 8.3). There are two major differences. First, an ABE ciphertext is not bound to any policy and is supposed to work with every authorized key. Second, ABE security should hold against unbounded collusion.

Recall that in AB-LFE, the message is blinded by a one-time pad *specific* to the policy. To make the ciphertext potentially work with every policy, we use "another level of indirection".[115] The message is now padded with a policy-independent pad $\mathbf{s}^\intercal\mathbf{z}$. The decryptor can still compute $\mathbf{s}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G})$. To bridge policy evaluation and decryption authorization, a secret key enables computing $\mathbf{s}^\intercal(\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}'_C) + \mathbf{z})$, so that $\mathbf{s}^\intercal\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}'_C)$ can be cancelled to recover $\mathbf{s}^\intercal\mathbf{z}$ given an authorized key, i.e., $C(\mathbf{x}) = 0$. This is implemented by putting $\mathbf{s}^\intercal\mathbf{B}$ in the ciphertext, where $\mathbf{B}$ is a public matrix sampled with trapdoor (Lemma 2.3; the trapdoor is kept in msk), and putting

$$\mathbf{k}_C \xleftarrow{\$} \mathbf{B}^{-1}(\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}'_C) + \mathbf{z}) = \mathsf{SampD}(\mathbf{B}, \dots, \mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}'_C) + \mathbf{z}, \dots)$$

in the secret key for C. During decryption, the bridging value is computed as $\mathbf{s}^\intercal\mathbf{B} \cdot \mathbf{k}_C$.

For security, we rely on evcsLWE (Assumption 8.2) and need to sample fresh $\mathbf{z}'_C$ for each secret key. Using the same argument of security for AB-LFE, we see that $\{\mathbf{s}^\intercal\mathbf{A}_C\mathbf{G}^{-1}(\mathbf{z}'_C)\}_C$ indeed serve as *indirect* one-time pads that jointly protect $\mathbf{s}^\intercal\mathbf{z}$ from all the bridging values when all the keys are unauthorized, i.e., $C(\mathbf{x}) = 1$, and that $\mathbf{s}^\intercal\mathbf{z}$ in turn hides the message.

In the preceding paragraphs, we ignored the difference between $\mathbf{r}, \mathbf{s}$ and matrices with/without bars. The construction below is formally accurate.

[115] "All problems in computer science can be solved by another level of indirection." — David Wheeler.

Construction 8.4 (KP-ABE for circuits of unbounded depth). Let (see Definition 2.3)

$$\mathsf{Params} = \{\, 1^L \mid L \in \mathbb{N} \,\}, \qquad F_{1^L} = \{\, \text{circuit } C \mid C : \{0,1\}^L \to \{0,1\} \,\},$$
$$X_{1^L} = \{0,1\}^L, \qquad P_{1^L}(C, \mathbf{x}) = \neg C(\mathbf{x}).$$

Our KP-ABE works as follows.

- $\mathsf{Setup}(1^L)$ defines appropriate $n, m, q, \sigma, \sigma', \sigma_{-1}, \sigma_{\text{post}}$ as described in Section 8.3, and sets $L_{\mathsf{S}} = m(n+1)^2\lceil \log_2 q \rceil^2$. The algorithm samples

$$\mathbf{A}_{\text{attr}} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(L+1)m}, \qquad (\mathbf{B}, \tau) \xleftarrow{\$} \mathsf{TrapGen}(1^n, 1^m, q),$$
$$\mathbf{A}_{\text{circ}} \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times(L_{\mathsf{S}}+1)m}, \qquad \mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^n.$$

  It outputs $\mathsf{mpk} = (n, m, q, \sigma, \sigma', \sigma_{-1}, \sigma_{\text{post}}, \mathbf{A}_{\text{attr}}, \mathbf{A}_{\text{circ}}, \mathbf{B}, \mathbf{z})$ and $\mathsf{msk} = (\mathsf{mpk}, \tau)$.

- $\mathsf{KeyGen}(\mathsf{msk}, C)$ takes as input msk and some circuit $C : \{0,1\}^L \to \{0,1\}$. It first computes $\mathbf{A}_C \leftarrow \mathsf{UEvalC}(\mathbf{A}_{\text{attr}}, \mathbf{A}_{\text{circ}}, C)$, next samples $\mathbf{z}' \xleftarrow{\$} \mathbb{Z}_q^{n+1}$, and then generates a trapdoor

$$\mathbf{k} \xleftarrow{\$} \mathsf{SampD}(\mathbf{B}, \tau, \overline{\mathbf{A}}_C\mathbf{G}^{-1}(\mathbf{z}') + \mathbf{z}, \sigma_{-1}),$$

  where $\overline{\mathbf{A}}_C \in \mathbb{Z}_q^{n\times m}$ is the first n rows of $\mathbf{A}_C$. The algorithm outputs $\mathsf{sk} = (\mathbf{z}', \mathbf{k})$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ takes as input mpk, attribute $\mathbf{x} \in \{0,1\}^L$, and message $\mu \in \{0,1\}$. It samples $\mathbf{r} \xleftarrow{\$} \mathcal{D}^n_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}$ and sets $\mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal$. The algorithm creates a circular encryption by

$$\overline{\mathbf{A}}_{\text{fhe}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \qquad \mathbf{e}_{\text{fhe}} \xleftarrow{\$} \mathcal{D}^m_{\mathbb{Z},\sigma,\le\sigma\sqrt{\lambda}}, \qquad \mathbf{R} \xleftarrow{\$} \{0,1\}^{m\times m(n+1)\lceil\log_2 q\rceil},$$
$$\mathbf{A}_{\text{fhe}} \leftarrow \begin{pmatrix} \overline{\mathbf{A}}_{\text{fhe}} \\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\text{fhe}} + \mathbf{e}_{\text{fhe}}^\intercal \end{pmatrix}, \qquad \mathbf{S} \leftarrow \mathbf{A}_{\text{fhe}}\mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G},$$

  and the attribute/circular encodings by

$$\mathbf{e}_{\text{attr}} \xleftarrow{\$} \mathcal{D}^{(L+1)m}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}, \qquad \mathbf{c}_{\text{attr}}^\intercal \leftarrow \mathbf{s}^\intercal(\mathbf{A}_{\text{attr}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\text{attr}}^\intercal,$$
$$\mathbf{e}_{\text{circ}} \xleftarrow{\$} \mathcal{D}^{(L_{\mathsf{S}}+1)m}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}, \qquad \mathbf{c}_{\text{circ}}^\intercal \leftarrow \mathbf{s}^\intercal(\mathbf{A}_{\text{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\text{circ}}^\intercal.$$

It also generates the message encoding as

$$\mathbf{e}_{\mathrm{B}} \xleftarrow{\$} \mathcal{D}^{m}_{\mathbb{Z},\sigma_{\mathrm{post}},\leq\sigma_{\mathrm{post}}\sqrt{\lambda}}, \qquad \mathbf{c}_{\mathrm{B}}^{\intercal} \leftarrow \mathbf{r}^{\intercal}\mathbf{B} + \mathbf{e}_{\mathrm{B}}^{\intercal},$$
$$e_{\mathrm{msg}} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma',\leq\sigma'\sqrt{\lambda}}, \qquad c_{\mathrm{msg}} \leftarrow \mathbf{r}^{\intercal}\mathbf{z} + e_{\mathrm{msg}} + \mu \cdot \lfloor q/2 \rceil.$$

The algorithm outputs $\mathsf{ct} = (\mathbf{S}, \mathbf{c}_{\mathrm{attr}}, \mathbf{c}_{\mathrm{circ}}, \mathbf{c}_{\mathrm{B}}, c_{\mathrm{msg}})$.

- $\mathsf{Dec}(\mathsf{mpk}, C, \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ takes $\mathsf{mpk}, C, \mathsf{sk}, \mathbf{x}, \mathsf{ct}$ as input. It computes

$$\begin{pmatrix} \overline{\mathbf{A}}_C \\ \underline{\mathbf{a}}_C^{\intercal} \end{pmatrix} = \mathbf{A}_C \leftarrow \mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C),$$
$$\mathbf{c}_{C,\mathbf{x}}^{\intercal} \leftarrow \mathsf{UEvalCX}(\mathbf{A}_{\mathrm{attr}}, \mathbf{c}_{\mathrm{attr}}^{\intercal}, \mathbf{A}_{\mathrm{circ}}, \mathbf{c}_{\mathrm{circ}}^{\intercal}, C, \mathbf{x}, \mathbf{S}),$$
$$c' \leftarrow c_{\mathrm{msg}} + \mathbf{c}_{C,\mathbf{x}}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}') - (\mathbf{c}_{\mathrm{B}}^{\intercal}\mathbf{k} - \underline{\mathbf{a}}_C^{\intercal}\mathbf{G}^{-1}(\mathbf{z}')).$$

The algorithm outputs $\mu' = 0$ if $c' \in [-q/4, q/4)$, and $\mu' = 1$ otherwise.

The efficiency parameters of the scheme are

$$|\mathsf{mpk}| = \mathrm{O}(L), \qquad |\mathsf{sk}_C| = \mathrm{O}(1), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(L),$$
$$T_{\mathsf{Setup}} = \mathrm{O}(L), \qquad T_{\mathsf{KeyGen}}, T_{\mathsf{Dec}} = \mathrm{O}(|C|), \qquad T_{\mathsf{Enc}} = \mathrm{O}(L).$$

Theorem 8.15 (¶). *Under* sLWE *(Assumption 8.1), Construction 8.4 is* strongly f-selectively correct *(Definition 8.4).*

Proof (Theorem 8.15). The proof resembles that of Theorem 8.13. When $C_j(\mathbf{x}) = 0$, we write

$$\mathbf{c}_{C_j,\mathbf{x}}^{\intercal} = \mathbf{s}^{\intercal}(\mathbf{A}_{C_j} - C_j(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C_j,\mathbf{x}}^{\intercal} = \mathbf{s}^{\intercal}\mathbf{A}_{C_j} + \mathbf{e}_{C_j,\mathbf{x}}^{\intercal}.$$

By definition,

$$\begin{aligned}
\mathbf{c}_{\mathrm{B}}^{\intercal}\mathbf{k}_j - \underline{\mathbf{a}}_{C_j}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}_j') &= \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{C_j}\mathbf{G}^{-1}(\mathbf{z}_j') + \mathbf{r}^{\intercal}\mathbf{z} + \mathbf{e}_{\mathrm{B}}^{\intercal}\mathbf{k}_j - \underline{\mathbf{a}}_{C_j}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}_j') \\
&= \mathbf{s}^{\intercal}\mathbf{A}_{C_j}\mathbf{G}^{-1}(\mathbf{z}_j') + \mathbf{r}^{\intercal}\mathbf{z} + \mathbf{e}_{\mathrm{B}}^{\intercal}\mathbf{k}_j, \\
\Longrightarrow \qquad c' &= c_{\mathrm{msg}} + \mathbf{c}_{C_j,\mathbf{x}}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}_j') - (\mathbf{c}_{\mathrm{B}}^{\intercal}\mathbf{k}_j - \underline{\mathbf{a}}_{C_j}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}_j')) \\
&= (\mathbf{r}^{\intercal}\mathbf{z} + e_{\mathrm{msg}} + \mu \cdot \lfloor q/2 \rceil) + (\mathbf{s}^{\intercal}\mathbf{A}_{C_j}\mathbf{G}^{-1}(\mathbf{z}_j') + \mathbf{e}_{C_j,\mathbf{x}}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}_j')) \\
&\quad - (\mathbf{s}^{\intercal}\mathbf{A}_{C_j}\mathbf{G}^{-1}(\mathbf{z}_j') + \mathbf{r}^{\intercal}\mathbf{z} + \mathbf{e}_{\mathrm{B}}^{\intercal}\mathbf{k}_j) \\
&= \mu \cdot q/2 + (\mathbf{e}_{\mathrm{B}}^{\intercal}\mathbf{k}_j + e_{\mathrm{msg}} + \mathbf{e}_{C_j,\mathbf{x}}^{\intercal}\mathbf{G}^{-1}(\mathbf{z}_j') + \mu \cdot (\lfloor q/2 \rceil - q/2)).
\end{aligned}$$

When $\|\mathbf{e}_{C_j,\mathbf{x}}\| \le B$, according to our choice of parameters, the total error is bounded by

$$\begin{aligned} & m \cdot \|\mathbf{e}_{\mathrm{B}}\| \cdot \|\mathbf{k}_j\| + |e_{\mathrm{msg}}| + \|\mathbf{e}_{C_j,\mathbf{x}}\| \cdot \|(\mathbf{G}^{-1}(\mathbf{z}'_j))^{\intercal}\| + 1 \\ & \qquad \le m \cdot \sigma_{\mathrm{post}}\sqrt{\lambda} \cdot \sigma_{-1}\sqrt{m} + \sigma'\sqrt{\lambda} + B \cdot m + 1 < q/4, \end{aligned}$$

in which case $\mu' = \mu$. From Theorem 8.12, it follows that $\Pr[\, \|\mathbf{e}_{C_j,\mathbf{x}}\| \le B \text{ for all } j \,]$ is overwhelming by a standard guessing reduction. This completes the proof. □

Stronger Correctness. When using unbounded homomorphic evaluation with stronger correctness, our KP-ABE scheme becomes (computationally or statistically) *adaptively* correct. We note that it does *not* become more secure due to the reliance on evcsLWE.

8.5.2 Security of KP-ABE

Theorem 8.16 (¶). *Under* csLWE *(Assumption 8.1) and* evcsLWE *(Assumption 8.2), Construction 8.4 is* very selectively secure *(Definition 2.3).*

Proof (Theorem 8.16). Let $\mathcal{A}$ be an efficient adversary against the very selective security of Construction 8.4. Consider the following sampler $\mathcal{S}(\mathsf{aux})$.

- Parse aux into (r, r').
- Launch $\mathcal{A}(r)$ to obtain $(1^L, \mathbf{x}, \{C_j\}_{j\in[J]})$, where $\mathbf{x} \in \{0,1\}^L$ and $C_j : \{0,1\}^L \to \{0,1\}$ with $C_j(\mathbf{x}) = 1$ for all $j \in [J]$.
- Pick $n, m, q, \sigma', \sigma', \sigma_{-1}, \sigma_{\mathrm{post}}$ as in $\mathsf{Setup}(1^L)$. Pick an appropriate σ_{pre}.
- Sample $\mathbf{A}_{\mathrm{attr}} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{attr}} \\ \underline{\mathbf{a}}^{\intercal}_{\mathrm{attr}} \end{pmatrix}$, $\mathbf{A}_{\mathrm{circ}} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{circ}} \\ \underline{\mathbf{a}}^{\intercal}_{\mathrm{circ}} \end{pmatrix}$, $\mathbf{z}$, $\{\mathbf{z}'_j\}_{j\in[J]}$ in a straight-forward way[116] using r'.

[116] Precisely speaking, r' conditioned on $\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \mathbf{z}, \{\mathbf{z}'_j\}_{j\in[J]}$ should be efficiently sampleable given them so that r' does not contain a trapdoor breaking LWE for those public matrices.

- Compute and set

$$\begin{pmatrix}\overline{\mathbf{A}}_{C_j}\\ \underline{\mathbf{a}}_{C_j}^{\mathsf{T}}\end{pmatrix} = \mathbf{A}_{C_j} \leftarrow \mathsf{UEvalC}(\mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, C_j), \qquad \mathbf{p}_j \leftarrow \overline{\mathbf{A}}_{C_j}\mathbf{G}^{-1}(\mathbf{z}'_j) + \mathbf{z},$$

$$\overline{\mathbf{A}}' \leftarrow (\overline{\mathbf{A}}_{\mathrm{attr}} - (1, \mathbf{x}^{\mathsf{T}}) \otimes \overline{\mathbf{G}}, \mathbf{z}), \qquad \mathbf{P} \leftarrow (\mathbf{p}_1, \ldots, \mathbf{p}_J),$$

  where $\overline{\mathbf{G}}$ is the first n rows of $\mathbf{G}$.

- Output $(\overline{\mathbf{A}}_{\mathrm{circ}}, \overline{\mathbf{A}}', \mathbf{P}, \sigma, \sigma', \sigma_{-1}, \sigma_{\mathrm{post}}, \sigma_{\mathrm{pre}})$.

We first show that $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{post}}$ implies ABE security against $\mathcal{A}$. Rewriting the view of $\mathcal{A}$ in $\mathsf{Exp}^{\beta}_{\mathrm{ABE}}$, we find

$$\begin{aligned}
\mathsf{mpk} &= (n, m, q, \sigma, \sigma', \sigma_{-1}, \sigma_{\mathrm{post}}, \mathbf{A}_{\mathrm{attr}}, \mathbf{A}_{\mathrm{circ}}, \boxed{\mathbf{B}}, \mathbf{z}),\\
\{\mathsf{sk}_j\}_{j\in[J]} &= (\{\mathbf{z}'_j\}_{j\in[J]}, \boxed{\mathbf{K}}),\\
\mathsf{ct} &= (\boxed{\mathbf{S}}, \mathbf{c}_{\mathrm{attr}}, \mathbf{c}_{\mathrm{circ}}, \boxed{\mathbf{c}_{\mathrm{B}}}, c_{\mathrm{msg}}),
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{c}_{\mathrm{attr}}^{\mathsf{T}} &= \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{attr}} - (1, \mathbf{x}^{\mathsf{T}}) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{attr}}^{\mathsf{T}}\\
&= \boxed{\mathbf{r}^{\mathsf{T}}(\overline{\mathbf{A}}_{\mathrm{attr}} - (1, \mathbf{x}^{\mathsf{T}}) \otimes \overline{\mathbf{G}}) + \mathbf{e}_{\mathrm{attr}}^{\mathsf{T}}} - \underline{\mathbf{a}}_{\mathrm{attr}}^{\mathsf{T}} + (1, \mathbf{x}^{\mathsf{T}}) \otimes \boldsymbol{\iota}_{n+1}^{\mathsf{T}} \otimes \mathbf{g}^{\mathsf{T}},\\
\mathbf{c}_{\mathrm{circ}}^{\mathsf{T}} &= \mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathrm{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{circ}}^{\mathsf{T}}\\
&= \boxed{\mathbf{r}^{\mathsf{T}}(\overline{\mathbf{A}}_{\mathrm{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \overline{\mathbf{G}}) + \mathbf{e}_{\mathrm{circ}}^{\mathsf{T}}} - \underline{\mathbf{a}}_{\mathrm{circ}}^{\mathsf{T}} + (1, \mathsf{bits}(\boxed{\mathbf{S}})) \otimes \boldsymbol{\iota}_{n+1}^{\mathsf{T}} \otimes \mathbf{g}^{\mathsf{T}},\\
c_{\mathrm{msg}} &= \boxed{\mathbf{r}^{\mathsf{T}}\mathbf{z} + e_{\mathrm{msg}}} + \beta \cdot \lfloor q/2 \rceil.
\end{aligned}$$

Here, the boxed terms are from the non-aux part of the first distribution in $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{post}}$. In the right distribution of $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{post}}$, all components of ct are one-time padded by a random value due to the boxed terms. In particular, β is perfectly hidden, implying the very selective security of Construction 8.4.

It remains to prove $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{pre}}$ using csLWE. This is analogous to the proof of Theorem 8.14. We consider the following series of hybrids.

- H_0 is the first distribution of $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{pre}}$, consisting of

$$r,\quad \overbrace{\overline{\mathbf{A}}_{\mathrm{attr}},\quad \underline{\mathbf{a}}_{\mathrm{attr}},\quad \overline{\mathbf{A}}_{\mathrm{circ}},\quad \underline{\mathbf{a}}_{\mathrm{circ}},\quad \mathbf{z},\quad \{\mathbf{z}'_j\}_{j\in[J]},}^{\text{equivalent to } r' \text{ in aux}}\quad \overline{\mathbf{A}}_{\mathrm{fhe}},\quad \mathbf{B},\quad \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}^{\intercal}_{\mathrm{fhe}},$$

$$\mathbf{S}=\begin{pmatrix}\overline{\mathbf{A}}_{\mathrm{fhe}}\\ \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}^{\intercal}_{\mathrm{fhe}}\end{pmatrix}\mathbf{R}-\mathsf{bits}(\mathbf{s})\otimes\mathbf{G},\qquad \mathbf{r}^{\intercal}(\overline{\mathbf{A}}_{\mathrm{circ}}-(1,\mathsf{bits}(\mathbf{S}))\otimes\overline{\mathbf{G}})+\mathbf{e}^{\intercal}_{\mathrm{circ}},$$

$$\mathbf{r}^{\intercal}(\overline{\mathbf{A}}_{\mathrm{attr}}-(1,\mathbf{x}^{\intercal})\otimes\overline{\mathbf{G}})+\mathbf{e}^{\intercal}_{\mathrm{attr}},\quad \mathbf{r}^{\intercal}\mathbf{z}+e_{\mathrm{msg}},\qquad (\text{i.e., } \mathbf{r}^{\intercal}\overline{\mathbf{A}}'+(\mathbf{e}')^{\intercal} \text{ for } \overline{\mathbf{A}}' \text{ of } \mathcal{S})$$

$$\mathbf{r}^{\intercal}\mathbf{B}+\mathbf{e}^{\intercal}_{\mathrm{B}},$$

$$\{\mathbf{r}^{\intercal}(\overline{\mathbf{A}}_{C_j}\mathbf{G}^{-1}(\mathbf{z}'_j)+\mathbf{z})+e_{\mathrm{P},j}\}_{j\in[J]}\qquad (\text{i.e., } \mathbf{r}^{\intercal}\mathbf{P}+\mathbf{e}^{\intercal}_{\mathrm{P}}).$$

  The width of $\mathbf{r},\mathbf{e}_{\mathrm{fhe}}$ (resp. $\mathbf{e}_{\mathrm{circ}},\mathbf{e}_{\mathrm{attr}},e_{\mathrm{msg}}$; $\mathbf{e}_{\mathrm{B}},\{e_{\mathrm{P},j}\}_{j\in[J]}$) is σ (resp. σ'; σ_{pre}).

- In H_1, we write

$$\begin{aligned}\mathbf{c}^{\intercal}_{C_j,\mathbf{x}}&=\mathsf{UEvalCX}(\mathbf{A}_{\mathrm{attr}},\mathbf{c}^{\intercal}_{\mathrm{attr}},\mathbf{A}_{\mathrm{circ}},\mathbf{c}^{\intercal}_{\mathrm{circ}},C_j,\mathbf{x},\mathbf{S})\\ &=\mathbf{r}^{\intercal}(\overline{\mathbf{A}}_{C_j}-C_j(\mathbf{x})\overline{\mathbf{G}})-\underline{\mathbf{a}}^{\intercal}_{C_j}+C_j(\mathbf{x})\mathbf{g}^{\intercal}+\mathbf{e}^{\intercal}_{C_j,\mathbf{x}}\\ (C_j(\mathbf{x})=1)\quad &=\mathbf{r}^{\intercal}(\overline{\mathbf{A}}_{C_j}-\overline{\mathbf{G}})-\underline{\mathbf{a}}^{\intercal}_{C_j}+\boldsymbol{\iota}^{\intercal}_{n+1}\otimes\mathbf{g}^{\intercal}+\mathbf{e}^{\intercal}_{C_j,\mathbf{x}},\end{aligned}$$

  with which the last J components can be rewritten as

$$\mathbf{r}^{\intercal}(\overline{\mathbf{A}}_{C_j}\mathbf{G}^{-1}(\mathbf{z}'_j)+\mathbf{z})+e_{\mathrm{P},j}=(\mathbf{c}_{C_j,\mathbf{x}}+\underline{\mathbf{a}}_{C_j}-\boldsymbol{\iota}_{n+1}\otimes\mathbf{g}-\mathbf{e}_{C_j,\mathbf{x}})^{\intercal}\mathbf{G}^{-1}(\mathbf{z}'_j)+\mathbf{r}^{\intercal}\mathbf{z}'_j+\mathbf{r}^{\intercal}\mathbf{z}+e_{\mathrm{P},j},$$

  causing no change to the distribution. We change the sampling of $\overline{\mathbf{A}}_{\mathrm{circ}},\overline{\mathbf{A}}_{\mathrm{attr}}$ to

$$\overline{\mathbf{A}}_{\mathrm{circ}}=\overline{\mathbf{A}}'_{\mathrm{circ}}+(1,\mathsf{bits}(\mathbf{S}))\otimes\overline{\mathbf{G}},\qquad \overline{\mathbf{A}}_{\mathrm{attr}}=\overline{\mathbf{A}}'_{\mathrm{attr}}+(1,\mathbf{x}^{\intercal})\otimes\overline{\mathbf{G}},$$

  where $\overline{\mathbf{A}}'_{\mathrm{circ}},\overline{\mathbf{A}}'_{\mathrm{attr}}$ are uniformly random matrices of appropriate shapes sampled independently from everything else. The distribution becomes

$$r,\quad \overline{\mathbf{A}}'_{\mathrm{circ}}+(1,\mathsf{bits}(\mathbf{S}))\otimes\overline{\mathbf{G}},\quad \underline{\mathbf{a}}_{\mathrm{circ}},\quad \overline{\mathbf{A}}'_{\mathrm{attr}}+(1,\mathbf{x}^{\intercal})\otimes\overline{\mathbf{G}},\quad \underline{\mathbf{a}}_{\mathrm{attr}},\quad \mathbf{z},\quad \{\mathbf{z}'_j\}_{j\in[J]},$$

$$\overline{\mathbf{A}}_{\mathrm{fhe}},\quad \mathbf{B},\quad \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}^{\intercal}_{\mathrm{fhe}},\quad \mathbf{S}=\begin{pmatrix}\overline{\mathbf{A}}_{\mathrm{fhe}}\\ \mathbf{r}^{\intercal}\overline{\mathbf{A}}_{\mathrm{fhe}}+\mathbf{e}^{\intercal}_{\mathrm{fhe}}\end{pmatrix}\mathbf{R}-\mathsf{bits}(\mathbf{s})\otimes\mathbf{G},$$

$$\mathbf{r}^{\intercal}\overline{\mathbf{A}}'_{\mathrm{circ}}+\mathbf{e}^{\intercal}_{\mathrm{circ}},\quad \mathbf{r}^{\intercal}\overline{\mathbf{A}}'_{\mathrm{attr}}+\mathbf{e}^{\intercal}_{\mathrm{attr}},\quad \mathbf{r}^{\intercal}\mathbf{z}+e_{\mathrm{msg}},\quad \mathbf{r}^{\intercal}\mathbf{B}+\mathbf{e}^{\intercal}_{\mathrm{B}},$$

$$\{(\mathbf{c}_{C_j,\mathbf{x}}+\underline{\mathbf{a}}_{C_j}-\boldsymbol{\iota}_{n+1}\otimes\mathbf{g}-\mathbf{e}_{C_j,\mathbf{x}})^{\intercal}\mathbf{G}^{-1}(\mathbf{z}'_j)+\mathbf{r}^{\intercal}\mathbf{z}'_j+\mathbf{r}^{\intercal}\mathbf{z}+e_{\mathrm{P},j}\}_{j\in[J]}.$$

  Clearly, $\mathsf{H}_0\equiv\mathsf{H}_1$, since $\mathbf{S},\mathbf{x}$ are computed independently of $\overline{\mathbf{A}}_{\mathrm{circ}},\overline{\mathbf{A}}_{\mathrm{attr}},\overline{\mathbf{A}}'_{\mathrm{circ}},\overline{\mathbf{A}}'_{\mathrm{attr}}$.

- In H_2, the last J components have the homomorphic noises truncated:

$$r,\quad \overline{\mathbf{A}}'_{\mathrm{circ}} + (1, \mathsf{bits}(\mathbf{S})) \otimes \overline{\mathbf{G}},\quad \underline{\mathbf{a}}_{\mathrm{circ}},\quad \overline{\mathbf{A}}'_{\mathrm{attr}} + (1, \mathbf{x}^\intercal) \otimes \overline{\mathbf{G}},\quad \underline{\mathbf{a}}_{\mathrm{attr}},\quad \mathbf{z},\quad \{\mathbf{z}'_j\}_{j\in[J]},$$

$$\overline{\mathbf{A}}_{\mathrm{fhe}},\quad \mathbf{B},\quad \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}^\intercal_{\mathrm{fhe}},\quad \mathbf{S} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}^\intercal_{\mathrm{fhe}} \end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G},$$

$$\mathbf{r}^\intercal\overline{\mathbf{A}}'_{\mathrm{circ}} + \mathbf{e}^\intercal_{\mathrm{circ}},\quad \mathbf{r}^\intercal\overline{\mathbf{A}}'_{\mathrm{attr}} + \mathbf{e}^\intercal_{\mathrm{attr}},\quad \mathbf{r}^\intercal\mathbf{z} + e_{\mathrm{msg}},\quad \mathbf{r}^\intercal\mathbf{B} + \mathbf{e}^\intercal_{\mathrm{B}},$$

$$\{(\mathbf{c}_{C_j,\mathbf{x}} + \underline{\mathbf{a}}_{C_j} - \boldsymbol{\iota}_{n+1} \otimes \mathbf{g} - \boxed{\mathbf{e}'_{C_j,\mathbf{x}}})^\intercal\mathbf{G}^{-1}(\mathbf{z}'_j) + \mathbf{r}^\intercal\mathbf{z}'_j + \mathbf{r}^\intercal\mathbf{z} + e_{\mathrm{P},j}\}_{j\in[J]},$$

  where $\mathbf{e}'_{C_j,\mathbf{x}}$ is $\mathbf{e}_{C_j,\mathbf{x}}$ if $\|\mathbf{e}_{C_j,\mathbf{x}}\| \le B$, and $\mathbf{0}$ otherwise. By (the proof of) Theorem 8.15, it holds that $\mathsf{H}_1 \approx \mathsf{H}_2$.

- In H_3, we modify the noises as follows (highlighted in boxes):

$$r,\quad \overline{\mathbf{A}}'_{\mathrm{circ}} + (1, \mathsf{bits}(\mathbf{S})) \otimes \overline{\mathbf{G}},\quad \underline{\mathbf{a}}_{\mathrm{circ}},\quad \overline{\mathbf{A}}'_{\mathrm{attr}} + (1, \mathbf{x}^\intercal) \otimes \overline{\mathbf{G}},\quad \underline{\mathbf{a}}_{\mathrm{attr}},\quad \mathbf{z},\quad \{\mathbf{z}'_j\}_{j\in[J]},$$

$$\overline{\mathbf{A}}_{\mathrm{fhe}},\quad \mathbf{B},\quad \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}^\intercal_{\mathrm{fhe}},\quad \mathbf{S} = \begin{pmatrix} \overline{\mathbf{A}}_{\mathrm{fhe}} \\ \mathbf{r}^\intercal\overline{\mathbf{A}}_{\mathrm{fhe}} + \mathbf{e}^\intercal_{\mathrm{fhe}} \end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G},$$

$$\mathbf{r}^\intercal\overline{\mathbf{A}}'_{\mathrm{circ}} + \mathbf{e}^\intercal_{\mathrm{circ}},\quad \mathbf{r}^\intercal\overline{\mathbf{A}}'_{\mathrm{attr}} + \mathbf{e}^\intercal_{\mathrm{attr}},\quad \mathbf{r}^\intercal\mathbf{z} + e_{\mathrm{msg}},\quad \mathbf{r}^\intercal\mathbf{B} + \boxed{(\mathbf{e}''_0)^\intercal} + \mathbf{e}^\intercal_{\mathrm{B}},$$

$$\{(\mathbf{c}_{C_j,\mathbf{x}} + \underline{\mathbf{a}}_{C_j} - \boldsymbol{\iota}_{n+1} \otimes \mathbf{g} - \boxed{\mathbf{0}})^\intercal\mathbf{G}^{-1}(\mathbf{z}'_j) + \mathbf{r}^\intercal\mathbf{z}'_j + \boxed{e''_j} + \mathbf{r}^\intercal\mathbf{z} + \boxed{e_{\mathrm{msg}}} + e_{\mathrm{P},j}\}_{j\in[J]}.$$

  Here, each entry of $\mathbf{e}''_0, \{e_j\}_{j\in[J]}$ is independently sampled from $\mathcal{D}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}}$. By our choice of parameters that $\sigma_{\mathrm{pre}} \ge 2^{\lambda+6}(Bm + 2\sigma'\sqrt{\lambda})$ and thanks to $\mathbf{e}_{\mathrm{B}}, \{e_{\mathrm{P},j}\}_{j\in[J]}$ (of width σ_{pre}), we have $\mathsf{H}_2 \approx_{\mathrm{s}} \mathsf{H}_3$. Note that in H_3, the values of $\mathbf{e}_{C_j,\mathbf{x}}, \mathbf{e}'_{C_j,\mathbf{x}}$ are no longer used and hence need not be computed.

- In H_4, all $\mathbf{r}$-dependent components are changed to random, arriving at the second distribution of $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{pre}}$. It follows from csLWE (Assumption 8.1) that $\mathsf{H}_3 \approx \mathsf{H}_4$. Roughly speaking, the reduction regards $\overline{\mathbf{A}}'$ (of csLWE, not $\mathcal{S}$) as

$$\left(\overline{\mathbf{A}}'_{\mathrm{circ}}, \overline{\mathbf{A}}'_{\mathrm{attr}}, \mathbf{z}, \mathbf{B}, \mathbf{z}_1, \ldots, \mathbf{z}_J\right)$$

  and samples (including $\mathbf{e}_{\mathrm{B}}, \{e_{\mathrm{P},j}\}_{j\in[J]}$) and computes the required components by itself.

By hybrid argument, $\mathsf{H}_0 \approx \mathsf{H}_4$, i.e., $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathrm{pre}}$ holds. This concludes the proof. □

8.5.3 Attribute-Unbounded Depth-Unbounded KP-ABE

We can apply the result of [GKW16] to obtain KP-ABE unbounded in both attribute length and circuit depth:

Corollary 8.17 (KP-ABE for circuits of unbounded depth and input length). *Under* csLWE *(Assumption 8.1) and* evcsLWE *(Assumption 8.2), there exists a* strongly f-selectively correct *(Definition 8.4),* very selectively secure *(Definition 2.3) KP-ABE scheme for circuits of unbounded depth and input length with*

$$\mathsf{Params} = \{\bot\}, \qquad X = \{0,1\}^{<2^\lambda},$$

$$F = \left\{ \text{circuit } C \;\middle|\; C : \{0,1\}^L \to \{0,1\} \text{ for some } L < 2^\lambda \right\},$$

$$P(C, \mathbf{x}) = \begin{cases} \neg C(\mathbf{x}'), & \text{if } C : \{0,1\}^L \to \{0,1\} \text{ and} \\ & \quad \mathbf{x}' \text{ is the } L\text{-prefix of } \mathbf{x}; \\ 0, & \text{otherwise;} \end{cases}$$

$$|\mathsf{mpk}| = \mathrm{O}(1), \qquad |\mathsf{sk}_C| = \mathrm{O}(L), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(|\mathbf{x}|),$$

$$T_{\mathsf{Setup}} = \mathrm{O}(1), \qquad T_{\mathsf{KeyGen}}, T_{\mathsf{Dec}} = \mathrm{O}(|C|), \qquad T_{\mathsf{Enc}} = \mathrm{O}(|\mathbf{x}|),$$

where L is the input length of C (not necessarily the same as $|\mathbf{x}|$).

The generic transformation of [GKW16] only handles equal-length matching, i.e., $\mathbf{x}$ must be of exactly the input length of C for decryption to succeed. It can be extended to prefix matching when based on Construction 8.4 with two changes, as observed by [BV16].

- For correctness, we use a PRF to generate an exponentially large $\mathbf{A}_{\text{attr}}$ matrix (instead of unrelated $\mathbf{A}_{\text{attr}}$ matrices for different lengths), and use the relevant parts of it in key generation and encryption. This ensures that sk and ct use the same $\mathbf{A}_{\text{attr}}$ for decryption to succeed.
- For security, we transform C and $\mathbf{x}$ into C' and $(L_{\mathrm{x}}, \mathbf{x})$, where $L_{\mathrm{x}} = |\mathbf{x}|$ is the λ-bit binary representation of the length of $\mathbf{x}$ and

$$C'(L_{\mathrm{x}}, \mathbf{x}') = \begin{cases} C(\mathbf{x}'), & \text{if } L_{\mathrm{x}} \geq L; \\ 1, & \text{if } L_{\mathrm{x}} < L; \end{cases}$$

with L being the input length of C and $\mathbf{x}'$ (hopefully) a prefix of $\mathbf{x}$. In the security proof, during reduction to Construction 8.4, we choose $(|\mathbf{x}|, (\mathbf{x}, \mathbf{0}))$ for $\mathbf{0}$ of appropriate length when generating the challenge ciphertext (part of which is thrown away) so that the challenge has the correct length as far as Construction 8.4 is concerned and policy circuits taking overlong input are sure to reject decryption.

When using unbounded homomorphic evaluation with stronger correctness, the scheme becomes (computationally or statistically) *adaptively* correct.

8.5.4 Scheme with Perfect Correctness

The downside of the previous scheme (Construction 8.4) is that it is not perfectly correct. The importance of perfect correctness has been discussed in [GKVW20]. In this section, we show how to obtain a perfectly correct KP-ABE scheme by adding error checking to Construction 8.4. We rely on a perfectly correct predicate encryption (PE) for circuits of bounded depth and a perfectly binding commitment scheme, both known from LWE [GVW15a,BTVW17,Reg05].

The basic idea is as follows. Recall that by Theorem 8.11, decryption errs only when an entry of $\mathbf{s}^\intercal \mathbf{A}'_i \mathbf{\Gamma}$ is close to carry/borrow boundaries ("bad event"), where i runs through all the gates in the circuit. To fix this case, we encrypt the message μ again under PE, with the (hidden) attribute being $\mathbf{s}$, and provide in the ABE key additionally a PE key that checks for and authorizes decryption in the unlikely case of the bad event. Each check takes $\mathrm{O}(\log \lambda)$ depth and aggregating them takes $\mathrm{O}(\log |C|)$ depth. Therefore, it suffices to use a depth-bounded PE scheme. A final catch is that Construction 8.4 does not indicate decryption error and the PE schemes of [GVW15a, BTVW17] does not detect decryption failure (due to policy not authorizing). Given this, how can we know, with absolute certainty, which decryption yields the correct message? Enters the commitment scheme. We commit to the message μ and encrypt its opening r. The decryption result that opens the commitment is always the correct one, due to the commitment scheme being perfectly binding.

The security of the scheme follows from those of its ingredients — PE decryption

is almost never authorized due to the computational correctness of Construction 8.4, so its security applies to hide the opening as well as $\mathbf{s}$, which in turn enables ABE security to completely erase the opening before invoking the hiding property to show that the message is hidden.

We start with the additional preliminaries for this part.

Definition 8.5 (depth-bounded KP-PE). A *key-policy predicate encryption scheme for circuits of bounded depth* consists of four efficient algorithms.

- $\mathsf{Setup}(1^\lambda, 1^L, 1^d)$ takes as input the attribute length L and the maximum depth d. It outputs a pair of master public/secret keys $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, C)$ takes as input msk and a circuit $C : \{0,1\}^L \to \{0,1\}$ of depth at most d. It outputs a secret key sk for C.
- $\mathsf{Enc}(1^\lambda, \mathsf{mpk}, \mathbf{x}, \mu)$ takes as input mpk, an attribute $\mathbf{x} \in \{0,1\}^L$, and some message $\mu \in \{0,1\}$. It outputs a ciphertext ct.
- $\mathsf{Dec}(1^\lambda, \mathsf{mpk}, C, \mathsf{sk}, \mathsf{ct})$ takes as input $\mathsf{mpk}, C, \mathsf{sk}, \mathsf{ct}$ (no $\mathbf{x}$). It is supposed to recover μ if $C(\mathbf{x}) = 0$.

The scheme must be *(perfectly) correct,* i.e., for all $\lambda, L, d \in \mathbb{N}$, circuit $C : \{0,1\}^L \to \{0,1\}$ of depth at most d, attribute $\mathbf{x} \in \{0,1\}^L$, message $\mu \in \{0,1\}$ such that $C(\mathbf{x}) = 0$,

$$\Pr\left[\begin{array}{rcl} (\mathsf{mpk}, \mathsf{msk}) & \xleftarrow{\$} & \mathsf{Setup}(1^\lambda, 1^L, 1^d) \\ \mathsf{sk} & \xleftarrow{\$} & \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, C) \\ \mathsf{ct} & \xleftarrow{\$} & \mathsf{Enc}(1^\lambda, \mathsf{mpk}, \mathbf{x}, \mu) \end{array} \;:\; \mathsf{Dec}(1^\lambda, \mathsf{mpk}, C, \mathsf{sk}, \mathsf{ct}) = \mu\right] = 1.$$

The scheme is *very selectively secure* if $\mathsf{Exp}^0_{\mathrm{PE}} \approx \mathsf{Exp}^1_{\mathrm{PE}}$, where $\mathsf{Exp}^\beta_{\mathrm{PE}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup, Query, Challenge.** Launch $\mathcal{A}(1^\lambda)$ and receive from it 1^L, 1^d, $\{C_j\}_{j\in[J]}$, $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^L$, where $C_j : \{0,1\}^L \to \{0,1\}$ are circuits of depth at most d. Run

$$\begin{aligned} (\mathsf{mpk}, \mathsf{msk}) &\xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^L, 1^d), \\ \mathsf{sk}_j &\xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, C_j) \qquad (\text{for } j \in [J]), \\ \mathsf{ct} &\xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, \mathbf{x}_\beta, \beta), \end{aligned}$$

and send $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{sk}_j\}_{j\in[J]}, \mathsf{ct})$ to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$. If $C_j(\mathbf{x}_0) = C_j(\mathbf{x}_1) = 1$ for all $j \in [J]$, the outcome of the experiment is β'. Otherwise, the outcome is set to 0.

We remark that Definition 8.5, while requiring security when $C(\mathbf{x}) = 1$, does not guarantee perfect detection of $C(\mathbf{x}) = 1$ during decryption, i.e., it is permissible that Dec outputs a random bit when $C(\mathbf{x}) = 1$. This gap between correctness and security is not captured by PHFE per Definition 2.1, so we must define it separately.

Definition 8.6 (perfectly binding commitment). A *commitment scheme* consists of two efficient algorithms.

- $\mathsf{GenCRS}(1^\lambda)$ outputs $\mathsf{crs} = (1^{L_{\mathrm{com}}}, \dots)$, where L_{com} is the length of commitment randomness.
- $\mathsf{Commit}(1^\lambda, \mathsf{crs}, \mu, r)$ takes as input crs, a message μ, and some $r \in \{0,1\}^{L_{\mathrm{com}}}$. It is deterministic and outputs a commitment c.

The scheme is *perfectly binding* if for all $\lambda \in \mathbb{N}$,

$$\Pr_{\mathsf{crs}=(1^{L_{\mathrm{com}}},\dots)\xleftarrow{\$}\mathsf{GenCRS}(1^\lambda)}\left[\begin{array}{l}\text{there exist } r_0, r_1 \in \{0,1\}^{L_{\mathrm{com}}} \text{ such that}\\ \quad \mathsf{Commit}(1^\lambda, \mathsf{crs}, 0, r_0) = \mathsf{Commit}(1^\lambda, \mathsf{crs}, 1, r_1)\end{array}\right] = 0.$$

It is *(computationally) hiding* if $\{(1^\lambda, \mathsf{crs}, c_0)\}_{\lambda\in\mathbb{N}} \approx \{(1^\lambda, \mathsf{crs}, c_1)\}_{\lambda\in\mathbb{N}}$, where

$$\mathsf{crs} = (1^{L_{\mathrm{com}}}, \dots) \xleftarrow{\$} \mathsf{GenCRS}(1^\lambda), \qquad r \xleftarrow{\$} \{0,1\}^{L_{\mathrm{com}}},$$
$$c_\beta \leftarrow \mathsf{Commit}(1^\lambda, \mathsf{crs}, \beta, r) \text{ for } \beta \in \{0,1\}.$$

Multi-Bit Messages for Basic Scheme. We modify Construction 8.4 for better usage as follows. The message is now $\boldsymbol{\mu} \in \{0,1\}^{L'}$, and $1^{L'}$ is part of param. The following replacements are made:

$$\mathbf{z}, \mathbf{z}', \mathbf{k} \quad \text{become} \quad \mathbf{Z}, \mathbf{Z}' \xleftarrow{\$} \mathbb{Z}_q^{(n+1)\times L'},\ \mathbf{K} \in \mathbb{Z}^{m\times L'};$$
$$e_{\mathrm{msg}}, c_{\mathrm{msg}} \quad \text{become} \quad \mathbf{e}_{\mathrm{msg}} \xleftarrow{\$} \mathcal{D}^{L'}_{\mathbb{Z},\sigma',\le\sigma'\sqrt{\lambda}},\ \mathbf{c}_{\mathrm{msg}} \leftarrow \mathbf{r}^\intercal\mathbf{Z} + \mathbf{e}_{\mathrm{msg}} + \lfloor q/2\rceil \cdot \boldsymbol{\mu};$$
$$c', \mu' \quad \text{become} \quad \mathbf{c}' \in \mathbb{Z}_q^{L'},\ \boldsymbol{\mu}' \in \{0,1\}^{L'},\ \boldsymbol{\mu}'[\ell] \leftarrow \begin{cases}0, & \text{if } \mathbf{c}'[\ell] \in [-q/4, q/4);\\ 1, & \text{otherwise.}\end{cases}$$

It can be readily verified that strong f-selective correctness and very selective security still hold.

Ingredients of Construction 8.5. We rely on

- Construction 8.4 modified as above $(\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$,
- a perfectly correct depth-bounded KP-PE scheme $(\mathsf{Setup}'', \mathsf{KeyGen}'', \mathsf{Enc}'', \mathsf{Dec}'')$, and
- a perfectly binding commitment scheme $(\mathsf{GenCRS}, \mathsf{Commit})$.

Construction 8.5 (perfectly correct KP-ABE). The functionality is that of Construction 8.4. The perfectly correct scheme works as follows.

- $\mathsf{Setup}(1^L)$ runs

$$\mathsf{crs} = (1^{L_{\mathrm{com}}}, \dots) \xleftarrow{\$} \mathsf{GenCRS}(), \qquad (\mathsf{mpk}', \mathsf{msk}') \xleftarrow{\$} \mathsf{Setup}'(1^L, 1^{1+L_{\mathrm{com}}}).$$

  It computes a polynomial upper bound d_0 on the depth of circuit

$$\mathsf{CheckOne}_{\mathbf{A}'_i}(\mathbf{s}) = \begin{cases} 0, & \text{if } (\mathbf{s}^\top \mathbf{A}'_i \boldsymbol{\Gamma} \bmod q) \notin [-q/2 + M^2 B, q/2 - M^2 B)^m; \\ 0, & \text{if } (\mathbf{s}^\top \mathbf{A}'_i \boldsymbol{\Gamma} \bmod M) \notin [-M/2 + B, M/2 - B)^m; \\ 1, & \text{otherwise;} \end{cases}$$

  where $\mathbf{s}, \mathbf{A}'_i, \boldsymbol{\Gamma}, q, M, B, m$ are those in (the modified) Construction 8.4. Let L_{s} be the bit-length of $\mathbf{s}$. The algorithm runs

$$(\mathsf{mpk}'', \mathsf{msk}'') \xleftarrow{\$} \mathsf{Setup}''(1^{L_{\mathrm{s}}}, 1^{d_0+\lambda}),$$

  and outputs $\mathsf{mpk} = (\lambda, \mathsf{mpk}', L_{\mathrm{s}}, d_0, \mathsf{mpk}'', \mathsf{crs})$ and $\mathsf{msk} = (\mathsf{mpk}, \mathsf{msk}', \mathsf{msk}'')$.

- $\mathsf{KeyGen}(\mathsf{msk}, C)$ checks whether $|C| \geq 2^\lambda$. If so, it modifies C into the trivially authorizing one, i.e., $C(\mathbf{x}) = 0$ always, before proceeding.[117] The algorithm runs

[117] This truncation is just to adhere to the formalism (Definitions 2.3 and 8.4) that the resultant KP-ABE is correct for all circuits, ensuring that $\mathsf{CheckAll}_C$ is of depth at most $(d_0 + \lambda)$ so the underlying depth-bounded KP-PE works. It does not affect security as the adversary cannot choose a circuit of size 2^λ. Such a huge circuit also need not be considered in practical scenarios.

$\mathsf{sk}' \xleftarrow{\$} \mathsf{KeyGen}'(\mathsf{msk}', C)$ and collects all the $\{\mathbf{A}'_i\}_{i\in[L+1,|C|]}$ matrices produced by KeyGen'. It then creates a circuit

$$\mathsf{CheckAll}_C(\mathbf{s}) = \bigwedge_{i=L+1}^{|C|} \mathsf{CheckOne}_{\mathbf{A}'_i}(\mathbf{s}),$$

and runs $\mathsf{sk}'' \xleftarrow{\$} \mathsf{KeyGen}''(\mathsf{msk}'', \mathsf{CheckAll}_C)$. The algorithm outputs $\mathsf{sk} = (\mathsf{sk}', \mathsf{sk}'')$.

- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ runs

$$r \xleftarrow{\$} \{0,1\}^{L_{\mathrm{com}}}, \qquad c \leftarrow \mathsf{Commit}(\mathsf{crs}, \mu, r),$$
$$\mathsf{ct}' \xleftarrow{\$} \mathsf{Enc}'(\mathsf{mpk}', \mathbf{x}, (\mu, r)), \qquad \mathsf{ct}'' \xleftarrow{\$} \mathsf{Enc}''(\mathsf{mpk}'', \mathbf{s}, \mu),$$

  where the input $\mathbf{s}$ to Enc'' is the one used in Enc'. It outputs $\mathsf{ct} = (c, \mathsf{ct}', \mathsf{ct}'')$.

- $\mathsf{Dec}(\mathsf{mpk}, C, \mathsf{sk}, \mathbf{x}, \mathsf{ct})$ runs

$$(\mu', r') \leftarrow \mathsf{Dec}'(\mathsf{mpk}', C, \mathsf{sk}', \mathbf{x}, \mathsf{ct}'), \qquad \mu'' \leftarrow \mathsf{Dec}''(\mathsf{mpk}'', \mathsf{CheckAll}_C, \mathsf{sk}'', \mathsf{ct}'').$$

  It outputs μ' if $\mathsf{Commit}(\mathsf{crs}, \mu', r') = c$. Otherwise, it outputs μ''.

We note that d_0 can be chosen as $\mathrm{O}(\log \lambda)$. The efficiency parameters of the scheme are still

$$|\mathsf{mpk}| = \mathrm{O}(L), \qquad |\mathsf{sk}_C| = \mathrm{O}(1), \qquad |\mathsf{ct}_{\mathbf{x}}| = \mathrm{O}(L),$$
$$T_{\mathsf{Setup}} = \mathrm{O}(L), \qquad T_{\mathsf{KeyGen}}, T_{\mathsf{Dec}} = \mathrm{O}(|C|), \qquad T_{\mathsf{Enc}} = \mathrm{O}(L).$$

Perfect Correctness. By the perfect binding property of the commitment scheme, if during decryption $\mathsf{Commit}(\mathsf{crs}, \mu', r') = c$, then $\mu' = \mu$ and correctness holds. When the commitment check fails and $C(\mathbf{x}) = 0$, the modified Construction 8.4 must be incorrect, and it follows from Theorem 8.11 that $\mathsf{CheckAll}_C(\mathbf{s}) = 0$ must hold. Note that $\mathsf{CheckAll}_C$ is a conjunction of at most 2^λ circuits, each of depth at most d_0, so it is of depth at most $(d_0 + \lambda)$ and can be handled by the PE scheme. By the perfect correctness of predicate encryption, we have $\mu'' = \mu$, which is also the output of the decryption algorithm in this case.

Theorem 8.18 (¶). *Under* csLWE *(Assumption 8.1) and* evcsLWE *(Assumption 8.2), if the predicate encryption is* very selectively secure *(Definition 8.5) and the commitment scheme is* hiding *(Definition 8.6), then Construction 8.5 is* very selectively secure *(Definition 2.3).*

Proof (Theorem 8.18). We list the hybrids.

- H_0^β is $\mathsf{Exp}_{\mathrm{ABE}}^\beta$, with the ciphertexts being

$$c \leftarrow \mathsf{Commit}(\mathsf{crs}, \beta, r), \quad \mathsf{ct}' \xleftarrow{\$} \mathsf{Enc}'(\mathsf{mpk}', \mathbf{x}, (\beta, r)), \quad \mathsf{ct}'' \xleftarrow{\$} \mathsf{Enc}''(\mathsf{mpk}'', \mathbf{s}, \beta).$$

- In H_1^β, the PE attribute and message are removed, i.e.,

$$c \leftarrow \mathsf{Commit}(\mathsf{crs}, \beta, r), \quad \mathsf{ct}' \xleftarrow{\$} \mathsf{Enc}'(\mathsf{mpk}', \mathbf{x}, (\beta, r)), \quad \mathsf{ct}'' \xleftarrow{\$} \mathsf{Enc}''(\mathsf{mpk}'', \boxed{\mathbf{0}, 0}).$$

  By (the proof of) Theorem 8.15, it holds with overwhelming probability that $\mathsf{CheckAll}_{C_j}(\mathbf{s}) = 1$ for all j. Combining it with the security of PE, we have $\mathsf{H}_0^\beta \approx \mathsf{H}_1^\beta$.

- In H_2^β, the ABE message is removed, i.e.,

$$c \leftarrow \mathsf{Commit}(\mathsf{crs}, \beta, r), \quad \mathsf{ct}' \xleftarrow{\$} \mathsf{Enc}'(\mathsf{mpk}', \mathbf{x}, \boxed{(0, 0^{L_{\mathrm{com}}})}), \quad \mathsf{ct}'' \xleftarrow{\$} \mathsf{Enc}''(\mathsf{mpk}'', \mathbf{0}, 0).$$

  $\mathsf{H}_1^\beta \approx \mathsf{H}_2^\beta$ by the very selective security of ABE (Theorem 8.16 modified for multi-bit messages).

By the hiding property of the commitment scheme, we have $\mathsf{H}_2^0 \approx \mathsf{H}_2^1$. Therefore, $\mathsf{Exp}_{\mathrm{ABE}}^0 \equiv \mathsf{H}_0^0 \approx \mathsf{H}_0^1 \equiv \mathsf{Exp}_{\mathrm{ABE}}^1$ by hybrid argument. □

We remark that our AB-LFE can be similarly modified (using a depth-bounded LFE) for perfect correctness.

8.6 Further Discussions

The results listed below are excluded from this dissertation. They are corollaries of the unbounded homomorphic evaluation procedure. See [HLL23a,HLL25,HLL23b] for details.

Corollary (LFE). *Under the circular LWE assumption, there exists a very selectively secure LFE scheme for circuits of unbounded depth and bounded input/output lengths L, L' with*

$$|\mathsf{crs}| = \mathrm{O}(L), \quad |\mathsf{digest}_C| = \mathrm{O}(L'), \quad T_{\mathsf{Enc}} = \mathrm{O}(L + L'), \quad |\mathsf{ct}(\mathbf{x})| = \mathrm{O}(L + L').$$

Corollary (1-key FE, implying 1-key ABE). *Under the circular LWE assumption, there exists a very selectively 1-key simulation-secure FE scheme for circuits of unbounded depth and bounded input/output lengths L, L' with*

$$|\mathsf{mpk}| = \mathrm{O}(L + L'), \qquad |\mathsf{sk}_C| = \mathrm{O}(L'), \qquad |\mathsf{ct}(\mathbf{x})| = \mathrm{O}(L + L').$$

Corollary (reusable garbled circuits). *Under the circular LWE assumption, there exists a selectively secure reusable garbling scheme for circuits with*

$$|\widehat{C}| = \mathrm{O}(L'), \qquad |\mathsf{pk}| = \mathrm{O}(L + L'), \qquad |\widehat{\mathbf{x}}| = \mathrm{O}(L + L'),$$

where L, L' are the input/output lengths of C.

Corollary (depth-unbounded KP-PE). *Under the evasive circular LWE assumption and the circular LWE assumption, there also exists a very selectively secure PE scheme for circuits of unbounded depth with*

$$|\mathsf{mpk}| = \mathrm{O}(L), \qquad |\mathsf{sk}_C| = \mathrm{O}(1), \qquad |\mathsf{ct}(\mathbf{x})| = \mathrm{O}(L).$$

Chapter 9

On the Optimal Succinctness and Efficiency of ABE

This chapter is a remix of [JLL22,Luo22,JLL23,Luo24].[118]

In this chapter, we push the frontier of ABE on the theoretical side of cryptography. Although the research of ABE has progressed significantly since its conception, we are yet to understand its best-possible efficiency. We initiate a systematic investigation into this basic question, prove inherent space-time trade-offs for ABE, and present several optimal constructions in obfustopia.

9.1 Introduction

Over the past decade, significant progress has been made in establishing the feasibility of ABE, for various classes of computation, with different levels of efficiency and security, and from different assumptions. However, we are yet to understand the asymptotic optimality and theoretical limits of its efficiency. We ask:

What is the best-possible asymptotic efficiency of ABE?
Are there trade-offs among different aspects of efficiency?
Can we construct optimally efficient ABE?

We make progress towards answering the above questions.

For the lower bounds, we prove inherent trade-offs between the ciphertext (or key) size and the decryption time. On the constructive front, we present the first collusion-resistant ABE for RAM solely based on (polynomially secure) collusion-resistant FE for circuits, which in turn can be based on well-studied assumptions [JLS21,JLS22].

---

[118] Aayush Jain, Huijia Lin, Ji Luo:
On the Optimal Succinctness and Efficiency of Functional Encryption and Attribute-Based Encryption,
 DOI 10.1007/978-3-031-30620-4_16.
Ji Luo: *Ad Hoc Broadcast, Trace, and Revoke*, DOI 10.62056/a39qxrxqi.

Our schemes have minimally sized keys and ciphertexts and optimal decryption time, matching our lower bounds.

Dream Efficiency. Before describing our results, we first picture the dream efficiency with respect to three important dimensions. Each dimension has been a consistent research theme across many primitives in cryptography. These dimensions are not disparate but closely related.

For simplicity, we consider key encapsulation mechanism, where the message is λ-bit long. Conversion to regular ABE is standard and ensures best-possible efficiency in all dimensions with respect to the message length.

Efficient Computation Model. Using a realistic and efficient model helps accurately characterizing efficiency. For example, the time of computation in the clear serves as a baseline for decryption time. For this reason, functions should be represented by random-access machines (RAM), the most efficient computation model subsuming both circuits and Turing machines. It is also closer to real-world computers.

We consider a RAM U (fixed[119] at set-up time) with random access to three tapes, a function tape containing f, an input tape containing x, and a working tape. In general, it may produce *arbitrarily long* output, e.g., one bit at *every* step. For the purpose of ABE, it suffices to consider a decision machine, i.e., U outputs one bit right before halting, indicating whether decryption is authorized. This flexible model captures many natural scenarios, e.g., binary search where the database could be part of f, x. It can emulate the evaluation of a circuit C on input x by putting the circuit description on the function tape. In ciphertext-policy ABE, each ciphertext is tied to a function, which can be captured by setting x to be the description of that function. These examples tell us that either or even both of f, x could be long, and we want to optimize the efficiency dependency on their lengths.

Succinctness. Enjoying low communication and storage overhead means having short master public key mpk, secret keys sk_f, and ciphertexts ct_x. At the most basic

[119] We can think of U as a universal RAM.

level, the size of each component should be *polynomial* in the length of its associated information — $|\mathsf{mpk}| = O(1)$,[120] $|\mathsf{sk}_f| = \mathrm{poly}(|f|)$, and $|\mathsf{ct}_x| = \mathrm{poly}(|x|)$, where $|f|, |x|$ are the description lengths of f, x, respectively — referred to as *polynomial efficiency*.[121] However, there is much to be desired beyond this basic level of efficiency. For instance, linear efficiency means $|\mathsf{sk}_f| = O(|f|)$ and $|\mathsf{ct}| = O(|x|)$, and rate-1 efficiency means $|\mathsf{sk}_f| = |f| + O(1)$ and $|\mathsf{ct}| = |x| + O(1)$.

In fact, even smaller parameters are possible. Since ABE does not aim to hide the function f, it is allowed to put the description of f in the clear in the secret key, and the *non-trivial* part of the secret key may be shorter than f. In this case, the right measure of efficiency *could* be the size of the non-trivial part (i.e., the overhead), which we now view as *the* secret key. We can aim for secret keys of size *independent* of that of the function — i.e., $|\mathsf{sk}_f| = O(1)$ — referred to as *constant-size* keys. The same observation applies to the public input x tied to the ciphertext and we can hope for ciphertexts of size *independent* of $|x|$, i.e., $|\mathsf{ct}_x| = O(1)$ — for long messages, it becomes $|\mathsf{ct}_x(\mu)| = |\mu| + O(1)$. In summary:

Ideal Component Sizes. $|\mathsf{mpk}| = O(1), \quad |\mathsf{sk}_f| = O(1), \quad |\mathsf{ct}_x| = O(1).$

Note that the ideal component sizes are completely independent of the running time of the computation.

Decryption Time. Decryption is also a RAM computation, $\mathsf{Dec}^{\mathsf{mpk},f,\mathsf{sk}_f,x,\mathsf{ct}_x}()$, which with random access to $\mathsf{mpk}, f, \mathsf{sk}_f, x, \mathsf{ct}_x$, recovers the message if $U^{f,x}()$ accepts. We want decryption to be efficient, ideally taking time linear in the *instance* running time T of the RAM computation in the clear:

Ideal Decryption Time. $T_{\mathsf{Dec}} = O(T)$.

[120] In this introduction, $O(\cdot)$ hides a multiplicative factor of $\mathrm{poly}(\lambda)$.

[121] It may appear that polynomial efficiency is the bare minimum. However, it is possible to consider components whose sizes depend on an upper bound of the length of some information *not* tied to them. Many early schemes are as such, e.g., the celebrated ABE scheme by [BGG+14] has $|\mathsf{mpk}|, |\mathsf{ct}|$ growing polynomially with the maximum depth of the computation. When a scheme requires fixing an upper bound on parameter Z (e.g., input length, depth, or size), it is said to be Z-bounded.

Note that since decryption necessarily has to verify that $U^{f,x}()$ accepts (otherwise there is no security), due to the time-hierarchy theorems for RAM [CR73,Iva82], we cannot expect Dec to run in time $O(T^{1-\varepsilon})$ for any constant $\varepsilon > 0$.

9.1.1 Our Results

Is the dream efficiency attainable simultaneously in all of the three dimensions? Towards understanding this, we present both new constructions and lower bounds.

New ABE for RAM with Optimal Succinctness. Starting from polynomially secure bounded FE for circuits, i.e., all of $|\mathsf{mpk}|, |\mathsf{sk}_f|, |\mathsf{ct}(y)|$ are just $\mathrm{poly}(|f|, |y|)$, which in turn can be constructed from well-studied assumptions [JLS21,JLS22], we construct an adaptively secure (unbounded) ABE for RAM with optimal succinctness.

Theorem 9.12. *Assuming polynomially secure FE for circuits, there exists an adaptively secure ABE for RAM with*

$$|\mathsf{mpk}| = O(1), \qquad |\mathsf{sk}_f| = O(1), \qquad |\mathsf{ct}_x| = O(1), \qquad T_{\mathsf{Dec}} = O(T + |f| + |x|).$$

Our construction gives the first ABE for RAM from falsifiable assumptions, and also the first ABE for any model of computation (e.g., circuit or TM) with optimal succinctness (both keys and ciphertexts are constant-size). The only prior construction of ABE for RAM by [GKP+13a] relies on non-falsifiable assumptions like SNARK and differing-input obfuscation. In terms of succinctness, existing schemes achieve *either* constant-size keys *or* constant-size ciphertexts (see [ALdP11,YAHK14,Tak14,Att16,ZGT+16,AT20] and Chapters 5 and 6). Achieving constant-size keys and ciphertexts *simultaneously* has been an important theoretical open question (see discussion in Chapter 6). The state-of-the-art is summarized in Table 9.1. We further discuss related works in Section 9.1.4.

Lower Bounds on Component Sizes and Decryption Time. The decryption time of our ABE *appears* suboptimal. In addition to the necessary linear dependency on T, it also grows linearly with $|f|, |x|$. It turns out that *ideal succinctness and ideal decryption time are at conflict*.

Theorem 9.4. *For all moderately expressive secure ABE scheme, it holds that*

$$|\mathsf{ct}_x| \cdot T_{\mathsf{Dec}} = \Omega(|x|), \qquad |\mathsf{sk}_f| \cdot T_{\mathsf{Dec}} = \Omega(|f|).$$

We note that the latter bound (about sk) is not formally proven. Its proof is similar to that of the former (about ct). Our ABE scheme matches the lower bounds — it is Pareto-optimal with respect to the dependency on $|f|, |x|$.

Table 9.1. Comparison among select KP-ABE schemes.

reference	functionality	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	T_{Dec}	adaptive	assumption
[GGH+13a]	circuit	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{poly}(\lvert x\rvert)$	$\mathrm{poly}(\lvert C\rvert)$		$i\mathcal{O}$
[KNTY19]	circuit	$\mathrm{poly}(\lvert C\rvert)$	$\mathrm{poly}(\lvert x\rvert)$	$\mathrm{poly}(\lvert C\rvert)$	✓	1-key sls FE
[GWZ22]	circuit	$\mathrm{poly}(\lvert C\rvert)$	$\lvert x\rvert + \mathrm{O}(1)$	$\mathrm{poly}(\lvert C\rvert)$		$i\mathcal{O}$
[AS16]	TM	$\mathrm{poly}(\lvert f\rvert)$	$\mathrm{poly}(\lvert x\rvert)$	$T\,\mathrm{poly}(\lvert f\rvert, \lvert x\rvert)$	✓	$i\mathcal{O}$
[AJS17]	TM	$c\lvert f\rvert + \mathrm{O}(1)$	$c\lvert x\rvert + \mathrm{O}(1)$	$T\,\mathrm{poly}(\lvert f\rvert, \lvert x\rvert)$	✓	subexp $i\mathcal{O}$
[AM18]	TM	$\mathrm{poly}(\lvert f\rvert)$	$\mathrm{O}(\lvert x\rvert)$	$T\,\mathrm{poly}(\lvert f\rvert, \lvert x\rvert)$	✓	FE
[KNTY19]	TM	$\mathrm{poly}(\lvert f\rvert)$	$\mathrm{poly}(\lvert x\rvert)$	$T\,\mathrm{poly}(\lvert f\rvert, \lvert x\rvert)$		1-key sls FE
[ACFQ22]	RAM	$\mathrm{poly}(\lvert f\rvert)$	$\mathrm{poly}(\lvert x\rvert)$	$T\,\mathrm{poly}(\lvert f\rvert)$		PK-DE-PIR & FE
Chapter 5	ABP	$\mathrm{O}(\lvert C\rvert \cdot \lvert x\rvert)$	$\mathrm{O}(1)$	$\mathrm{O}(\lvert C\rvert \cdot \lvert x\rvert)$	✓	k-Lin
[BGG+14]	circuit	$\mathrm{poly}(d)$	$\lvert x\rvert\,\mathrm{poly}(d)$	$\lvert C\rvert\,\mathrm{poly}(d)$		LWE
Chapter 6	circuit	$\mathrm{O}(1)$	$\mathrm{poly}(d)$	$\lvert C\rvert\,\mathrm{poly}(d)$	✓	GGM & LWE
[GKP+13a]	RAM	$\mathrm{O}(1)$	$\mathrm{poly}(\lvert x\rvert)$	$\mathrm{O}(T + \lvert f\rvert + \lvert x\rvert)$		SNARK & $di\mathcal{O}$
Theorem 9.12	RAM	$\mathrm{O}(1)$	$\mathrm{O}(1)$	$\mathrm{O}(T + \lvert f\rvert + \lvert x\rvert)$	✓	FE
Corollary 9.15	RAM	$\lvert f\rvert + \mathrm{O}(1)$	$\mathrm{O}(1)$	$\mathrm{O}(T + \lvert x\rvert)$	✓	FE
Corollary 9.15	RAM	$\mathrm{O}(1)$	$\lvert x\rvert + \mathrm{O}(1)$	$\mathrm{O}(T + \lvert f\rvert)$	✓	FE
Corollary 9.15	RAM	$\lvert f\rvert + \mathrm{O}(1)$	$\lvert x\rvert + \mathrm{O}(1)$	$\mathrm{O}(T)$	✓	FE

The first group of works study FE, and the rows show them used as ABE. The second group of works study ABE directly. All the schemes encrypt at least a single bit (maybe more, but not considered in this table). C is the circuit. T is the instance running time of TM/RAM. ABP means arithmetic branching programs (also denoted by C). All $\mathrm{poly}(\cdot)$ and $\mathrm{O}(\cdot)$ implicitly contains λ. For assumptions, "$i\mathcal{O}$" mean indistinguishability obfuscation, "FE" means functional encryption for circuits, "1-key sls" means 1-key sublinearly succinct, "subexp" means subexponentially secure, "PK-DE-PIR" means public-key doubly efficient private information retrieval, "k-Lin" means k-Linear in pairing groups, "LWE" means learning with errors, "GGM" means generic pairing group model, "SNARK" means succinct non-interactive argument of knowledge, and "$di\mathcal{O}$" means differing-input obfuscation.

ABE for RAM with Optimal Decyrption Time. By tweaking the construction, we obtain several other Pareto-optimal ABE schemes.

Corollary 9.15. *Assuming polynomially secure FE for circuits, there exist adaptively secure ABE schemes for RAM with*

$$|\mathsf{mpk}| = \mathrm{O}(1), \quad |\mathsf{sk}_f| = |f|^A + \mathrm{O}(1), \quad |\mathsf{ct}_x| = |x|^B + \mathrm{O}(1), \quad T_{\mathsf{Dec}} = \mathrm{O}(T + |f|^{1-A} + |x|^{1-B}).$$

All of the four combinations of $(A, B) \in \{0,1\}^2$ *are possible.*

Succinct Garbled RAM. The main tool in our construction of ABE for RAM is succinct garbled RAM (GRAM). Initiated by [KLW15,BGL+15,CHJV15], a sequence of works have constructed succinct garbled RAM [CH16,CCC+16,ACC+16,CCHR16] based on subexponentially secure FE for circuits and succinct garbled Turing machines [KLW15, AJS17,AL18,GS18b] based on polynomially secure FE for circuits.

In this chapter, we formulate a new notion of succinct GRAM geared for building highly efficient ABE for RAM, and construct it based on polynomially secure FE for circuits. Previously, succinct GRAM (our or the standard notion) was only known from subexponential hardness assumptions.

9.1.2 What's Next?

Recent developments of ABE for RAM leave several interesting questions open for future research.

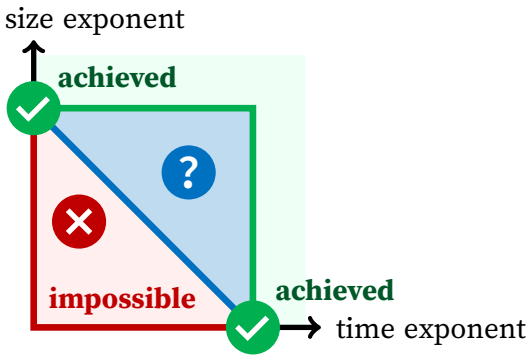


Figure 9.1. Trade-offs between ABE component sizes and decryption time (dependency on f or x).

Exact Pareto Frontier of ABE Efficiency. The efficiency trade-offs characterized in this chapter are depicted in Figure 9.1. There is a triangular area where neither lower bounds nor constructions are known. On the one hand, the lower bound only uses broadcast functionality, a rather simple ABE whose predicate can be decided in constant time. It is well possible that ABE for general RAM is subject to stronger lower bounds. On the other hand, it is also possible that a more clever construction achieves trade-offs other than the two corners of the impossibility triangle. Of course, both could be true. It remains to pin down the exact Pareto frontier of ABE efficiency.

Rethinking Formulation of ABE Efficiency.[122] ABE per Definition 2.3 gives verbatim copies of f, x to the decryption algorithm for free, and our lower bound (on component sizes) applies to the overhead in keys and ciphertexts. This has become the norm in the literature of ABE, possibly in the hope that discounting the non-hidden f, x would provide a more accurate characterization of ABE efficiency. Let's call this the "verbatim-for-free" model.

Consider an alternative (or rather, the original [GPSW06]) formulation, where f, x are not explicitly given to Dec. Instead, f, x must be encoded in sk, ct. Let's call it the "custom-encoded" model. In this model, the sizes of keys and ciphertexts are at least the lengths of policies and attributes, respectively (as long as the functionality is sufficiently expressive and dependent on f, x). Clearly, putting a copy of f, x in sk, ct is one way to encode them, so any verbatim-for-free ABE can be converted to a custom-encoded ABE. If the verbatim-for-free ABE has constant-size components, the converted custom-encoded ABE has minimally sized components, but its decryption time is also linear in the component sizes. However, our lower bounds do not apply to custom-encoded ABE in general, i.e., it is well possible that there is a custom-encoded ABE, which cannot be obtained by conversion from verbatim-for-free ABE, that has minimally sized components and $|f|$- and/or $|x|$-independent decryption time simultaneously. If such highly efficient custom-encoded ABE exists, we should use

[122] This is a new thought during the preparation of this dissertation. Previously, the lower bound was taken to suggest rethinking whether succinctness is indeed the pursuit of ABE. See [Luo24] for the discussion on that aspect.

the custom-encoded model as the standard accounting model for ABE efficiency.

We note that the succinctness of custom-encoded ABE is only meaningful at a very fine grain. See Section 9.8 for further discussion.

9.1.3 Technical Overview

We start with an overview of the lower bound.

YOU CAN (NOT) OPTIMIZE. As introduced earlier, we show that $|\mathsf{ct}_x| \cdot T_{\mathsf{Dec}} = \Omega(|x|)$ is inherent for mildly expressive secure ABE schemes. For the bound to hold, the ABE only needs to support what we call the *restricted broadcast* functionality. In restricted broadcast encryption, there are $2N$ users, each with their own key and paired into N groups. The users are indexed by (j, s) for $j \in [N]$ (group index) and $s \in \{0,1\}$ (indicating which user in group j), i.e., (j, s) is the policy. The encryptor chooses exactly one user from each group, described by the attribute $R \in \{0,1\}^N$, so that the users $\{(i, R[i])\}_{i\in[N]}$ can, and only they can, decrypt the ciphertext. This is a restricted form of broadcast encryption as it constrains how the recipient set can be chosen. Formally, we also only require very weak security, but this detail is not important for now. We will prove our lower bound for restricted BE. Interestingly, the adversary used in the proof simply runs the decryption algorithm with a *non-decrypting* key (while *lying* about the recipient set), so the bound holds as long as the scheme is not *blatantly* insecure.

We explain the ideas of our proof based on a corollary[123] of a result [Unr07] dealing with random oracles in the presence of non-uniform advice. Let $S, T \geq 0$ be such that ST is much smaller than N. The corollary says that for any adversary learning any S-bit function (advice) of a random string $R \xleftarrow{\$} \{0,1\}^N$ and additionally (adaptively) querying at most T bits in R, it is "indistinguishable" to flip a bit in R at a random location after the advice is computed (using the non-flipped R) and before queries are answered, even if the index of the potentially flipped bit is revealed to the adversary after the advice is computed.

[123] This corollary is also a lower bound of a probabilistic variant of Yao's box problem [Yao90] (generalized and studied in [CHK22]), on which our proof can be alternatively based.

Back to restricted BE. Consider a ciphertext ct encrypting a random plaintext with a random string R as the attribute and regard ct as the advice. Suppose Y is either R itself or R flipped at index $i^* \xleftarrow{\$} [N]$. Let's try decrypting ct using $\mathsf{sk}_{i^*,Y[i^*]}$ while *pretending* that ct is generated for Y. By way of contradiction, suppose $|\mathsf{ct}| \cdot T_{\mathsf{Dec}}$ is much smaller than N, which would translate to ST being much smaller than N in the corollary. By the correctness of restricted BE, when Y is R itself, the attempted decryption should successfully recover the plaintext. From the corollary it follows that the other case (Y is R flipped at i^*) should also lead to successful recovery. But if $Y[i^*] = \neg R[i^*]$, by the security of restricted BE, the attempted decryption must fail to recover the plaintext except for negligible probability, yielding a contradiction.

Overview of Our ABE for RAM. At a very high level, we compose FE for circuits with *succinct* garbled RAM (GRAM) (for conditional disclosure of secrets) to obtain ABE for RAM. The latter can be viewed as a 1-key, 1-ciphertext, secret-key ABE for RAM, where succinctness means that the running time of key generation and encryption is independent of the running time of the RAM computation. A (collusion-resistant) FE for circuits then lifts one-time security to many-time. This high-level approach first appeared in [AS16] for building FE for TM and can be regarded as a powered-up version of the paradigm in Part I. In this chapter, towards nearly optimally efficient ABE for RAM, we first observe that existing definitions and constructions of succinct GRAM [BGL+15,CHJV15,CH16,CCC+16,ACC+16,CCHR16] are insufficient. Therefore, we formulate a new variant of succinct GRAM, termed *laconic GRAM*, then construct it. Along the way, we also weaken the assumption needed for succinct/laconic GRAM schemes from $i\mathcal{O}$, which is considered an inherently subexponential assumption, to polynomially secure FE for circuits.

Let's first review the syntax and security of standard GRAM schemes. They consist of the following algorithms. The encoding algorithm encodes a database D into $\widehat{D}$ and outputs a private state k. The garbling algorithms uses k to garble a RAM M into $\widehat{M}$ and outputs a collection of input labels $\{L_{i,b}\}_{i,b}$. The evaluation algorithm, given the garbled RAM $\widehat{M}$, one set of labels corresponding to an input w, and random access to $\widehat{D}$, returns the output $M^D(w)$ of the RAM computation. Simulation-based

security ensures that $\widehat{D}, \widehat{M}, \{L_{i,w[i]}\}_i$ can be simulated using only the output $M^D(w)$. The efficiency requirements of those algorithms are

$$(\widehat{D}, k) \xleftarrow{\$} \mathsf{Encode}(D), \quad (\widehat{M}, \{L_{i,b}\}_{i,b}) \xleftarrow{\$} \mathsf{Garble}(M, k), \quad M^D(w) \leftarrow \mathsf{Eval}^{\widehat{D}}(\widehat{M}, \{L_{i,w[i]}\}_i),$$
$$|\widehat{D}| = \mathrm{O}(|D|), \qquad |\widehat{M}| = \mathrm{poly}(|M|), \qquad T_{\mathsf{Eval}} = T\,\mathrm{poly}(|M|).$$

Unfortunately, the standard notion falls short for our purpose of building highly efficient ABE for RAM. We elaborate and explain how to address those issues.

- *multi-tape* instead of *single-tape*. To begin, we consider RAM computation with multiple tapes, $M^{f,x}(\mu)$, instead of $M^D(w)$ with a single tape. GRAM schemes directly constructed often have (inevitably bad) polynomial efficiency dependency on $|M|$, which is inherited by the ABE. By making M a universal machine and letting f, x be the tape contents, it is easier to characterize and achieve the desired efficiency.

- *public tape* instead of *private tape*. The tape contents D or parts of them (such as f, x) are public. The standard GRAM is defined only for private tapes, making the encoding size at least $|D|$. In ABE, we provide verbatim copies of f, x for free and only count the *overhead* in sk, ct, so we must not use standard GRAM when aiming for $|\mathsf{sk}|, |\mathsf{ct}|$ independent of $|f|, |x|$.

- *compressing* instead of *encoding*. Our notion requires a compression algorithm that hashes public tapes down to short digests, and the garbling algorithm "binds" the hashes to the garbled program. Lastly, the evaluation algorithm gets random access to the tape contents in the clear, like ABE decryption. Our GRAM only handles public tapes and the hiding of μ is provided by the (short) input to M, which can then be used as an encapsulated key for the actual message.

- *reusable* digests instead of *one-time* encoding. The encoded $\widehat{D}$ in standard GRAM can only be used once for a single garbled program generated using the secret k.[124] When lifted to multi-time security, each decryption of ABE for RAM requires generating a fresh $\widehat{D}$ *inside* the FE for circuits, inheriting its bad efficiency. To

[124] We remark that $\widehat{D}$ is often sequentially reusable, meaning that it can be updated by one execution,

get rid of it, we require the digests be reusable, so that decryption does not have to recompute them.

- *relaxed* evaluation time. Reusability comes at a cost. In our notion, the evaluation time is $(T + |f| + |x|)\,\mathrm{poly}(|M|)$, whereas in standard GRAM it is independent of the tape lengths $|f|, |x|$. Nevertheless, our negative results imply that the dependencies are impossible (subject to the laconic requirement) to get around.

Putting the above pieces together,[125] we formulate our laconic GRAM as in Figure 9.2.

- $\mathsf{Compress}(\tau, D_\tau)$ compresses the τ^{th} public tape D_τ into a short hash digest_τ of constant length. It runs in time $\mathrm{O}(|D_\tau|)$.
- $\mathsf{Garble}(M, \{\mathsf{digest}_\tau\}_\tau)$ outputs a garbled program $\widehat{M}$ bound to the public tapes via their hashes, and pairs of labels $\{L_{i,b}\}_{i,b}$. It runs in time $\mathrm{poly}(|M|)$.
- $\mathsf{Eval}^{D_1,\ldots,D_T}(M, \{\mathsf{digest}_\tau\}_\tau, \widehat{M}, \{L_{i,w[i]}\}_i)$ returns the output of RAM computation $M^{D_1,\ldots,D_T}(w)$. It runs in time $\mathrm{O}(T + \sum_\tau |D_\tau|)\,\mathrm{poly}(|M|)$.
- Security guarantees that if $M^{D_1,\ldots,D_T}(w_0)$ and $M^{D_1,\ldots,D_T}(w_1)$ have identical outputs and running time, the distributions of $(\widehat{M}, \{\mathsf{digest}_\tau\}_\tau, \{L_{i,w_0[i]}\}_i)$ and $(\widehat{M}, \{\mathsf{digest}_\tau\}_\tau, \{L_{i,w_1[i]}\}_i)$ are indistinguishable. This holds when the tape contents $\{D_\tau\}_\tau$ are chosen adaptively, dependent on the hashes of previously chosen tape contents, before the program M and inputs w_0, w_1 are chosen. (For ABE, only fixed-memory security is needed, i.e., the addresses and the memory contents are identical for security to hold.)

Figure 9.2. An overview of the notion of laconic GRAM.

from where the next execution resumes. However, in ABE, we need $\widehat{D}$ to be parallel reusable because multiple decryptions start from the same tape contents and can be done in no particular order.

[125] See [JLL22] for the treatment of laconic GRAM for constructing PHFE. The overview in this dissertation handles the private input (message) in ABE-specific ways and does not consider long output. Formal definitions later in this chapter might still have remnants of the general treatment.

OUR CONSTRUCTION OF LACONIC GRAM. One approach towards constructing laconic GRAM for RAM is to first obtain a non-succinct GRAM for RAM (with garbling size proportional to the worst-case running time) with tape compression from laconic OT techniques, then further compress the GRAM. First introduced in [BGL+15], the second step uses $i\mathcal{O}$ to obfuscate a circuit that on input an index t outputs the t^{th} component of the non-succinct GRAM. If each component can be locally generated using a small circuit of size poly($|M|$), then the obfuscated circuit is also of size poly($|M|$) and can be used as the succinct garbled program. To prove security, [BGL+15] identified that the non-succinct garbling scheme must satisfy another property, articulated later by [AL18], called *local simulation*. Informally, the non-succinct garbling must be proven secure via a sequence of hybrids where the components of every hybrid garbled program can be locally generated using a small circuit, and in neighboring hybrids, only a few components change. Beyond succinct garbling, local simulation has also found applications in achieving adaptive security [BHR12] of garbling schemes. A sequence of works developed local simulation strategies for garbled circuits [HJO+16, GS18a], Turing machines [GS18b,AL18], and RAM [GOS18]. Most notably, the work of [GS18a] took advantage of a clever pebbling technique.

Our construction of laconic GRAM proceeds in multiple steps. First, we use the techniques of [GS18a] to obtain a non-succinct GRAM with local simulation proof for a weak security called fixed-memory security. Indistinguishability holds when the two RAM computations have identical memory access pattern, memory content, outputs, and running time. Albeit weak, it already suffices for ABE. Then, by the same approach of [BGL+15,GS18b,AL18], we turn it into a succinct one, still with only *fixed-memory security*, relying on $i\mathcal{O}$ for polynomial-size domains, which is implied by polynomially secure FE for circuits.

For the full treatment of laconic GRAM (needed for general PHFE), see [JLL22].

9.1.4 Related Works

Our new constructions significantly improve upon the efficiency of prior ABE schemes. The state-of-the-art is summarized in Table 9.1. For discussion of PHFE-themed works, see [JLL22]. Below, we compare with prior works related to ABE in more detail.

FE for Circuits and Turing Machines. Many works (including an underlying work [JLL22,JLL23] of this chapter) study functional encryption (FE), a stronger primitive than ABE. We list those works (excluding ours) used as ABE in the first group of Table 9.1.

The first construction of collusion-resistant FE for polynomial-size circuits is by [GGH+13a] and based on $i\mathcal{O}$, which in turn relies on subexponential hardness. Later works [GS16,LM16,KNT18,KNTY19] improved the assumption from $i\mathcal{O}$ to 1-key FE with sublinearly compact ciphertext, $|\mathsf{ct}(y)| = |f|^{1-\varepsilon}\,\mathrm{poly}(|y|)$, where ε is a positive constant and $|f|$ is the maximum circuit size of f. The latter has been recently constructed by [JLS21,JLS22] from the polynomial hardness of three well-studied assumptions. However, these FE schemes for circuits only enjoy polynomial efficiency. The only exception is the recent construction due to [GWZ22], which has rate-1 ciphertexts, i.e., $|\mathsf{ct}(y)| = |y| + \mathrm{O}(1)$, yet large secret keys with $|\mathsf{sk}_f| = \mathrm{poly}(|f|)$.

Several works constructed FE for Turing machines with arbitrary-length inputs, first from $i\mathcal{O}$ [AS16], then from FE for circuits [AM18], and more recently from 1-key sublinearly compact FE [KNTY19]. The scheme of [AS16] relies on a 1-key 1-ciphertext secret-key FE for TM, which is essentially a succinct garbling scheme for TM with indistinguishability-based security. They constructed it by modifying the succinct garbling for TM of [KLW15]. Later, the works of [AL18,GS18b] improved and simplified the garbling construction. The work of [KNTY19] improved the assumption to 1-key sublinearly succinct FE, and showed that their garbling scheme can be combined with [AS16] to obtain FE for TM. On the other hand, the work of [AM18] presented an alternative direct approach to FE for TM from FE for circuits without going through succinct garbling for TM. Prior constructions of FE for TM [AS16,AM18,KNTY19] put more focus on weakening the underlying assumptions, and only show polynomial efficiency.

ABE for Circuits and Turing Machines. The literature on ABE focuses on constructing ABE from weaker assumptions and achieving better efficiency, among others. The celebrated works of [GVW13,BGG+14] showed that ABE for *bounded-depth* circuits can be constructed from the learning with errors (LWE) assumption. Parameters

of these schemes, however, depend polynomially on the maximum depth d of the supported circuits, namely, $|\mathsf{mpk}| = \mathrm{poly}(d)$, $|\mathsf{sk}_f| = \mathrm{poly}(d)$, $|\mathsf{ct}_x| = |x|\,\mathrm{poly}(d)$, and the decryption time is $T_{\mathsf{Dec}} = |f|\,\mathrm{poly}(d)$. In Chapter 6, we improved it to obtain constant-size keys while keeping the sizes of master public key and ciphertext intact, but at the cost of additionally relying on the generic (pairing) group model (GGM). ABE for low-depth computation such as NC^1 or (arithmetic) branching programs can be constructed using pairing groups, where several schemes have either constant-size keys *or* constant-size ciphertexts, but never both (see [ALdP11,YAHK14,Tak14,Att16, ZGT+16,AT20] and Chapter 5).

The work of [GKP+13a] constructed ABE for Turing machines and RAM with constant-size secret keys $|\mathsf{sk}_f| = \mathrm{O}(1)$, yet large ciphertexts $|\mathsf{ct}_x| = \mathrm{poly}(|x|)$. Their scheme uses SNARK and differing-input obfuscation, which cannot be based on falsifiable assumptions. Another work [AFS19] tries to construct ABE for RAM from LWE, at the cost of making the master public key, secret keys, and ciphertexts all grow polynomially with the maximum running time, i.e., it is an ABE for bounded-time RAM.

In summary, we give the first ABE schemes for RAM from falsifiable assumptions, simultaneously having constant- or linear-size secret keys, constant- or linear-size ciphertexts, and the best-possible decryption times matching the lower bound, also proven in this chapter. We summarize the ABE-specific works and our results in the second and third groups of Table 9.1.

Lower Bounds. Previous works [BC95,LS98,KYDB98,AK08,KY09,GKW15,DLY21] show a few efficiency lower bounds related to ABE, yet they only apply to information-theoretically secure primitives and even specific construction techniques. Moreover, all of them prove space (ciphertext or secret key size, or their trade-off) lower bounds, whereas we prove space-time trade-offs. The two underlying works of this chapter are the first systematic study of inherent efficiency trade-offs of advanced forms of encryption.

Concurrent and Independent Work on FE for RAM. Concurrently and independently of the research of this chapter, the recent work by Ananth, Chung, Fan, and

Qian [ACFQ22] also considers the question of FE for RAM. Despite an apparent overlap between the two works, there are many differences. The two works start with different motivations. Our goal is to understand the optimal succinctness and efficiency of PHFE (only ABE in this dissertation), both constructively and from a lower-bound perspective, whereas [ACFQ22] aims to construct FE for RAM with optimal decryption time $T_{\mathsf{Dec}} = \mathrm{O}(T)$. Consequently, the two works obtain mostly complementary results. The detailed comparison is best discussed in the context of PHFE, for which we refer the reader to [JLL22].

9.2 Preliminaries

Let Σ be a set and n a natural number, $\Sigma^{\leq n}$ is the set of *non-empty* strings of length at most n over the alphabet Σ. For a string x, its i^{th} symbol is denoted by $x[i]$. As an example, the third *bit* in some sufficiently long $x \in (\{0,1\}^2)^{\leq 3}$ is $x[2][1]$.

Symbols. Table 9.2 explains select single-letter symbols used in this chapter.

9.2.1 Multi-Tape Random-Access Machine

We use a model of *multi-tape* random-access machines with several read-only input tapes and one read/write working tape. In addition to the tapes, the machine also has a short input and maintains a (small) state.[126] The behavior of a RAM is described by its step circuit

$$(\ell_1, \ldots, \ell_T, w, \mathsf{oldst}, \mathsf{rdata}) \xmapsto{\mathsf{CPU}} (\mathsf{done}, \mathsf{newst}, \tau, i, \mathsf{wdata}, \mathsf{out}).$$

The step circuit takes as input *i)* the input tape lengths $\ell_1, \ldots, \ell_T$ and a short input w, which stay the same throughout the computation, and *ii)* the previous state oldst and the last string rdata read from the tapes, which change from step to step during the computation. It outputs whether the machine should halt in the flag done. If the machine does not halt, it outputs the next state newst, and the next tape τ and

[126] The distinction between input and state is arbitrary, and many works formulate them as a single entity. We choose to separate them for clearer semantics.

Table 9.2. Select single-letter symbols in this chapter.

context	symbol	meaning
general	$\lambda, \mathcal{A}, \beta', q$	security parameter, adversary, output of adversary, query index
	β, β', b	challenge bit, output of adversary, choice bit
	B, β', b	choice bit (parameter, name), choice bit (argument, value)
	ℓ, r	length, randomness
	J, J'	object in constructed scheme, object J in underlying scheme
RAM	$\mathcal{T}, \tau, D$	tape count, tape index, tape content
	i, j	address (index into tape content / of cell), index into cell content
	M, w, t, T	RAM, short input, time step, instance running time
	$\overline{T}$	instance running time bound (for security)
	$T_{\max}$	time / time and space bound (for correctness, $T \le \overline{T} \le T_{\max}$)
LGRAM	$M, \widehat{M}, L$	program, garbled program, label
	b, i	short input bit (choosing L), index into short input
	$\mathfrak{t}, \mathfrak{s}$	hybrid index (time step), hybrid index (pebbles)
	$\mathfrak{t}^\star$	pebble to be changed (time step, $\mathfrak{t}^\star < \mathfrak{t}$)
	$\widetilde{M}, \widetilde{L}$	encrypted underlying garbling, encrypted underlying label
	K, W	SKE key, short input (parameters, names)
	k, w	SKE key, short input (arguments, values)
	E	logarithm of attempted running time (parameter, name)
	e	logarithm of attempted running time (argument, value)
	$\overline{e}, \mathfrak{e}$	upper bound of attempted e, hybrid index related to e and $\overline{e}$
PHFE and ABE	φ, Φ, P	functionality, functionality family, predicate (family)
	f, F	function description, function description space
	x, X, y, Y	public input, public input space, private input, private input space
	z, Z	output, output space
	$T, \overline{T}$	baseline of decryption efficiency, upper bound of T (for security)
	$\mathsf{idx}, \mathsf{idx}'$	index, encrypted index (proof progress)
	k, s, k'	PPRF key, PPRF input, SKE key
	$\mathfrak{q}$	hybrid index (PHFE key being operated)
	β, w, k	choice bit, double encryption, PRF key
$i\mathcal{O}$	$C, \widetilde{C}$	circuit, obfuscated circuit
LOT	$D, \widehat{D}, h$	database, processed database, hash
	m, b	message, cell content / hash bit (choosing m)
	$\widetilde{\mathsf{rct}}, \widetilde{\mathsf{wct}}$	simulated rct, simulated wct
GC	$C, \widehat{C}, L$	circuit, garbled circuit, label
	b, i	input bit (choosing L), index into input
	$\pi, \widehat{x}$	point-and-permute string, permuted input
PPRF	$k, \mathring{k}, \mathring{k}_{\mathfrak{p}}$	key, punctured key, key punctured over $\mathfrak{p}$

address i to read from. Additionally, if τ specifies the working tape, then the step circuit also outputs a string wdata, which gets written at address i of the working tape. Each step also optionally generates out, one bit of output.

The execution of a machine M with $D_1, \ldots, D_{\mathcal{T}}$ on the input tapes and w as the short input begins by zero-initializing[127] the working tape, the state, and the last-read string. Throughout the process, the short input stays unchanged and the state is updated by the machine. At each step, the requested location (determined by the output of the previous step) is read, whose value is passed into the last-read string of the current step, before that location is overwritten. For simplicity, we insist that overwriting happen if and only if the requested tape is the working tape (the input tapes are read-only). The whole sequence of out's, including their timing (at which step each output bit is produced), is the output of the execution. This process is denoted by $M^{D_1,\ldots,D_{\mathcal{T}}}(w)$. We now give the formal definition.

Definition 9.1 (multi-tape RAM). Let $\mathcal{T} \in \mathbb{N}$. A *$\mathcal{T}$-tape RAM* M is specified by its step circuit

$$\mathsf{CPU} : (\ell_1, \ldots, \ell_{\mathcal{T}}, w, \mathsf{oldst}, \mathsf{rdata}) \mapsto (\mathsf{done}, \mathsf{newst}, \tau, i, \mathsf{wdata}, \mathsf{out}),$$

which is subject to the constraints below.

Addresses and Tapes. An *(input-tape) address* is an ℓ_{addr}-bit string indexing a *cell*. Each input tape consists of at most $2^{\ell_{\mathrm{addr}}}$ cells and each cell is ℓ_{cell}-bit long, i.e., the content D of each input tape is in $(\{0,1\}^{\ell_{\mathrm{cell}}})^{\leq 2^{\ell_{\mathrm{addr}}}}$. For technical convenience in [JLL23, JLL22], the working tape uses address and cell lengths (ℓ_{ADDR} and ℓ_{CELL}) different from the input tapes. Without loss of generality (efficiency- and security-wise), we assume $\ell_{\mathrm{ADDR}} \geq \ell_{\mathrm{addr}}$ and $\ell_{\mathrm{CELL}} \geq \ell_{\mathrm{cell}}$. Conceptually, the working tape has exactly $2^{\ell_{\mathrm{ADDR}}}$ cells.

Inputs and Outputs of CPU. The inputs include the following.

- $\ell_\tau \in [2^{\ell_{\mathrm{addr}}}]$ is the length (in cells) of the τ^{th} input tape.
- $w \in \{0,1\}^{\ell_{\mathrm{in}}}$ is the short input of length ℓ_{in}.

[127] We do not consider persistent memory in this chapter.

- oldst $\in \{0,1\}^{\ell_{\text{st}}}$ is the state of length ℓ_{st}.
- rdata $\in \{0,1\}^{\ell_{\text{CELL}}}$ is the string that was read.

Among the outputs, done is a flag indicating whether the machine should halt. If done is set, all of newst, τ, i, wdata, out should be $\perp$. Otherwise, they are as follows.

- newst $\in \{0,1\}^{\ell_{\text{st}}}$ is the new state.
- $\tau \in [\mathcal{T}] \cup \{\text{work}\}$ is the tape to read from (and possibly write to).
  - If $\tau \in [\mathcal{T}]$, then $i \in [\ell_\tau]$ is the address to read from on the τ^{th} input tape, and wdata $= \perp$.
  - If $\tau = \text{work}$, then $i \in [2^{\ell_{\text{ADDR}}}]$ is the address to read from and write to on the working tape, and wdata $\in \{0,1\}^{\ell_{\text{CELL}}}$ is the string to be written.
- out $\in \{0,1,\perp\}$ is an optional output bit.

Executing RAM. We now formally present the steps involved in RAM execution. Let $D_1, \ldots, D_{\mathcal{T}} \in (\{0,1\}^{\ell_{\text{cell}}})^{\leq 2^{\ell_{\text{addr}}}}$ be the input tape contents and $w \in \{0,1\}^{\ell_{\text{in}}}$ the short input. To execute $M^{D_1,\ldots,D_{\mathcal{T}}}(w)$...

1. Let $\text{st}_0 \leftarrow 0^{\ell_{\text{st}}}$, $\text{rdata}_0 \leftarrow 0^{\ell_{\text{CELL}}}$, $D_{\text{work},0} \leftarrow (0^{\ell_{\text{CELL}}})^{2^{\ell_{\text{ADDR}}}}$, $t \leftarrow 1$.
2. Let $(\text{done}_t, \text{st}_t, \tau_t, i_t, \text{wdata}_t, \text{out}_t) \leftarrow \text{CPU}(|D_1|, \ldots, |D_{\mathcal{T}}|, w, \text{st}_{t-1}, \text{rdata}_{t-1})$.
3. If done_t is set, the execution halts.
4. If $\tau_t \in [\mathcal{T}]$, let $\text{rdata}_t \leftarrow D_\tau[i_t] \| 0^{\ell_{\text{CELL}} - \ell_{\text{cell}}}$ and $D_{\text{work},t} \leftarrow D_{\text{work},t-1}$.
5. Otherwise, $\tau_t = \text{work}$, let $\text{rdata}_t \leftarrow D_{\text{work},t-1}[i_t]$ and $D_{\text{work},t}$ be $D_{\text{work},t-1}$ with $D_{\text{work},t}[i_t]$ replaced by wdata_t.
6. Let $t \leftarrow t+1$ and go back to Step 2.

We assume, without loss of generality, that all those constraints are respected during all executions.[128] For a halting execution,

[128] For example, memory access should never be out of range. Given a step circuit, it is efficient to wrap it inside another circuit checking all the constraints and correcting any error (e.g., by halting immediately when a constraint is violated).

- its *running time* $T = \mathsf{time}(M, D_1, \ldots, D_{\mathcal{T}}, w)$ is the value of t when it halts;[129]
- its *running space* is $\mathsf{space}(\cdots) = \max_{t<T, \tau_t = \mathsf{work}} i_t$;[130]
- its *state sequence* is $\mathsf{stS}(\cdots) = (\mathsf{st}_1, \ldots, \mathsf{st}_{T-1})$;
- its *address sequence* is $\mathsf{addrS}(\cdots) = (\tau_1, i_1, \ldots, \tau_{T-1}, i_{T-1})$;
- its *write sequence* is $\mathsf{writeS}(\cdots) = (\mathsf{wdata}_1, \ldots, \mathsf{wdata}_{T-1})$;
- its *output sequence* is $\mathsf{outS}(\cdots) = (\mathsf{out}_1, \ldots, \mathsf{out}_{T-1})$.

Remark 9.1 (output sequence). The machine does *not* necessarily produce non-$\perp$ outputs at the end or even consecutively. In this chapter, we do not care whether the output sequence is in some particular format. When defining secure evaluation of RAM, we simply require that all information be hidden except the output sequence, which implies that the running time and the timing of each non-$\perp$ output are not supposed to be hidden as such information is incorporated in the output sequence.

Remark 9.2 (running space). A better name for $\mathsf{space}(\cdots)$ is *address bound*. Jumping ahead, the running space must be polynomially bounded in bounded laconic garbled RAM schemes, but can be exponentially large (and is hidden) in (full-fledged) LGRAM.

9.2.2 Laconic Garbled RAM

Our notion of garbling RAM laconically involves two steps. First, a reusable short digest is created for each input tape. The digest has length independent of that of the tape content and must be computable in linear time. Second, the RAM and the short digests are put together to produce a garbled program and the labels. This procedure runs in time poly-logarithmic in the RAM running time. Given a garbled program and one set of labels (selected by the bits of the short input), the evaluation procedure computes the output sequence in time linear in the RAM running time.

We consider indistinguishability-based security for the short input. The input tape contents can be chosen adaptively, but the short input cannot depend on the garbled

[129] If the execution does not halt, we say $\mathsf{time}(\cdots) = +\infty$.

[130] To be complete, $\mathsf{space}(\cdots)$ is defined to be 1 if the working tape is never accessed.

program (i.e., selectiveness). We also consider several weaker notions, where the running space is bounded and more information about the execution is allowed to be revealed.

Definition 9.2 (LGRAM). Let $\mathcal{T}$ be a natural number. A *laconic garbling scheme* for $\mathcal{T}$-tape RAM consists of three algorithms.

- $\mathsf{Compress}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, \tau, D_\tau)$ takes as input a cell length ℓ_{cell}, an address length ℓ_{addr}, an input tape index $\tau \in [\mathcal{T}]$, and its content $D_\tau \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$. It outputs a short digest digest_τ. The algorithm runs in time $|D_\tau| \operatorname{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}})$ and its output length is $\operatorname{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}})$.
- $\mathsf{Garble}(1^\lambda, T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]})$ takes as input a time bound $T_{\max} \in \mathbb{N}_+$, a $\mathcal{T}$-tape RAM M, and $\mathcal{T}$ input tape digests. It outputs a garbled program $\widehat{M}$ and ℓ_{in} pairs of labels $\{L_{i,b}\}_{i\in[\ell_{\mathsf{in}}],b\in\{0,1\}}$. The algorithm runs in polynomial time.
- $\mathsf{Eval}^{D_1,\ldots,D_\mathcal{T}}(1^\lambda, T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]}, \widehat{M}, \{L_i\}_{i\in[\ell_{\mathsf{in}}]})$ takes as input $T_{\max}$, M, the input tape digests, $\widehat{M}$, and one set of labels. Given random access to the input tapes, it is supposed to compute the output sequence. The algorithm runs in time
$$\left(\min\left\{T_{\max}, \mathsf{time}(M, D_1, \ldots, D_\mathcal{T}, w)\right\} + \sum_{i=1}^{\mathcal{T}} |D_\tau|\right) \operatorname{poly}(\lambda, |M|, \log T_{\max}),$$
where w is the short input corresponding to the labels.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $\ell_{\mathsf{cell}}, \ell_{\mathsf{addr}} \in \mathbb{N}_+$, $T_{\max} \in \mathbb{N}_+$, $\ell_{\mathsf{in}} \in \mathbb{N}$, $\mathcal{T}$-tape RAM M, input tape contents $D_1, \ldots, D_\mathcal{T} \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$, short input $w \in \{0,1\}^{\ell_{\mathsf{in}}}$ such that $M^{D_1,\ldots,D_\mathcal{T}}(w)$ halts in time at most $T_{\max}$, it holds that

$$\Pr\left[\begin{array}{r} \mathsf{digest}_\tau \xleftarrow{\$} \mathsf{Compress}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, \tau, D_\tau) \qquad \text{for all } \tau \in [\mathcal{T}] \\ (\widehat{M}, \{L_{i,b}\}_{i\in[\ell_{\mathsf{in}}],b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Garble}(1^\lambda, T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]}) \\ : \quad \mathsf{Eval}^{D_1,\ldots,D_\mathcal{T}}(1^\lambda, T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]}, \widehat{M}, \{L_{i,w[i]}\}_{i\in[\ell_{\mathsf{in}}]}) \\ = \mathsf{outS}(M, D_1, \ldots, D_\mathcal{T}, w) \end{array}\right] = 1.$$

Remark 9.3 (unboundedness). Our notion of LGRAM is *unbounded*, i.e., it is not necessary to know a polynomial upper bound of the instance running time upon garbling. By

choosing an exponentially large $T_{\max}$, one garbling works for all polynomial-time computation. In contrast is a *bounded* scheme for all polynomial-time computation, where $T_{\max}$ can be *any* polynomial, but it *must* be a polynomial, hence every garbling is restricted to *some* polynomial time bound upon creation. Unboundedness is reflected in both efficiency and security (below), where $T_{\max}$ is written in binary.

See also Remark 9.2 on RAM running space.

Definition 9.3 (LGRAM security). An LGRAM scheme per Definition 9.2 is *(tape-adaptively, indistinguishability-based) secure* if $\mathsf{Exp}^0_{\mathrm{LGRAM}} \approx \mathsf{Exp}^1_{\mathrm{LGRAM}}$, where the experiment $\mathsf{Exp}^{\beta}_{\mathrm{LGRAM}}(1^\lambda, \mathcal{A})$ proceeds as follows.

- **Setup.** Launch $\mathcal{A}(1^\lambda)$ and receive from it $1^{\ell_{\mathrm{cell}}}$ and $1^{\ell_{\mathrm{addr}}}$.
- **Tape Choices.** Repeat the following for $\mathcal{T}$ rounds. In each round, $\mathcal{A}$ chooses $\tau \in [\mathcal{T}]$ and $D_\tau \in (\{0,1\}^{\ell_{\mathrm{cell}}})^{\leq 2^{\ell_{\mathrm{addr}}}}$. Upon receiving such choice, run

$$\mathsf{digest}_\tau \xleftarrow{\$} \mathsf{Compress}(1^\lambda, 1^{\ell_{\mathrm{cell}}}, 1^{\ell_{\mathrm{addr}}}, \tau, D_\tau)$$

  and send digest_τ to $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ chooses an instance running time bound $1^{\overline{T}}$ (in unary), a time bound $T_{\max}$ (in binary), a $\mathcal{T}$-tape RAM M, and two inputs (w_0, w_1). Run

$$(\widehat{M}, \{L_{i,b}\}_{i\in[\ell_{\mathrm{in}}], b\in\{0,1\}}) \xleftarrow{\$} \mathsf{Garble}(1^\lambda, T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]})$$

  and send $(\widehat{M}, \{L_{i,w_\beta[i]}\}_{i\in[\ell_{\mathrm{in}}]})$ to $\mathcal{A}$.
- **Guess.** $\mathcal{A}$ outputs a bit β'. The outcome of the experiment is β' if all of the following conditions hold.
  - The τ's in all rounds of **Tape Choices** are distinct.
  - Both $M^{D_1,\ldots,D_{\mathcal{T}}}(w_0)$ and $M^{D_1,\ldots,D_{\mathcal{T}}}(w_1)$ halt in time $T \leq \overline{T} \leq T_{\max}$ with identical *output sequences* $\mathsf{outS}(\cdots)$.

  Otherwise, the outcome is set to 0.

Weaker security notions are obtained by strengthening the second condition in **Guess**.

- *Fixed-address security*. "... with identical $\mathsf{outS}(\cdots)$ and $\mathsf{addrS}(\cdots)$."
- *Fixed-memory security*. "... with identical $\mathsf{outS}(\cdots)$, $\mathsf{addrS}(\cdots)$, and $\mathsf{writeS}(\cdots)$."

Remark 9.4 (polynomial security). Although $T_{\max}$ can be exponentially large, we only require security for polynomially large *instance* running time, which is captured by the requirement that the adversary must produce $1^{\overline{T}}$, an upper bound of the instance running time in unary.

Bounded LGRAM. As an intermediate primitive, we consider *bounded* LGRAM.

Definition 9.4 (bounded LGRAM and security). The notion of *bounded LGRAM* is obtained by modifying Definition 9.2 as follows.

- The evaluation procedure runs in time
$$\left(T_{\max} + \sum_{i=1}^{\mathcal{T}} |D_\tau|\right) \mathrm{poly}(\lambda, |M|, \log T_{\max}).$$
- Correctness is required only if $M^{D_1,\ldots,D_{\mathcal{T}}}(w)$ halts using space at most $T_{\max}$ in time at most $T_{\max}$.

Its security notions are obtained by modifying Definition 9.3 as follows.

- In **Challenge**, the adversary chooses $1^{T_{\max}}$ (instead of $1^{\overline{T}}$ and $T_{\max}$).
- In **Guess**, the second condition is strengthened to "... halt using space at most $T_{\max}$ in time $T \le T_{\max}$ with identical..."

Remark 9.5 (efficiency and security). Although evaluation efficiency is relaxed and security is restricted to polynomially large $T_{\max}$, the garbling procedure of a bounded LGRAM scheme still runs in time poly-logarithmic in $T_{\max}$. This is important for its bootstrapping to (unbounded) LGRAM.

9.2.3 PHFE Efficiency and FE for Circuits

Definition 9.5 (PHFE efficiency). For a PHFE scheme, on top of the basic efficiency required in Definition 2.1, the following time-efficiency properties are considered.

- It has *linear-time* KeyGen if KeyGen runs in time $|f|\,\mathrm{poly}(\lambda, |\mathsf{param}|)$; for Enc, it is $(|x| + |y|)\,\mathrm{poly}(\lambda, |\mathsf{param}|)$; for Dec, it is $(T + |f| + |x| + |y|)\,\mathrm{poly}(\lambda, |\mathsf{param}|)$.
- It has *f-fast* Dec if Dec runs in time $(T + |x| + |y|)\,\mathrm{poly}(\lambda, |\mathsf{param}|)$; for *$x$-fastness*, it is $(T + |f| + |y|)\,\mathrm{poly}(\lambda, |\mathsf{param}|)$; for *$y$-fastness*, it is $(T + |f| + |x|)\,\mathrm{poly}(\lambda, |\mathsf{param}|)$.

The following size-efficiency properties are considered.

- It is *f-succinct* if $|\mathsf{sk}_f| = \mathrm{poly}(\lambda, |\mathsf{param}|)$, independent of $|f|$.
- It is *x-succinct* if $|\mathsf{ct}_x| = \mathrm{poly}(\lambda, |\mathsf{param}|, |y|)$, independent of $|x|$.
- It has *rate-c ciphertext* for some constant c if $|\mathsf{ct}_x| = c|y| + \mathrm{poly}(\lambda, |\mathsf{param}|)$.

FE for Circuits. We will use FE for circuits as a building block:

Definition 9.6 (FE for circuits). A *functional encryption scheme for circuits* is a PHFE scheme (Definition 2.1) for

$$\begin{aligned}
\mathsf{Params}_\lambda &= \{\, (1^\ell, 1^s) \mid \ell, s \in \mathbb{N}_+ \,\},\\
F_{\lambda,1^\ell,1^s} &= \{\, \text{circuits of input length } \ell \text{ and size at most } s \,\},\\
X_{\lambda,1^\ell,1^s} &= \{\bot\}, \qquad Y_{\lambda,1^\ell,1^s} = \{0,1\}^\ell, \qquad Z_{\lambda,1^\ell,1^s} = \{0,1\}^\star,\\
&\varphi_{\lambda,1^\ell,1^s}(f, \bot, y) = \big(1, f(y)\big).
\end{aligned}$$

Remark 9.6. The first output of $\varphi_{\lambda,1^\ell,1^s}$ is just a placeholder value and all efficiency parameters (Definition 9.5) are always allowed arbitrary polynomial dependency on λ, ℓ, s by our choice of $\mathsf{param} = (1^\ell, 1^s)$. This is intended as we use FE for circuits as a building block and do not wish to start with too strong a scheme.

Thanks to a long line of beautiful prior works, we can weaken the assumption of FE for circuits all the way down to an "obfuscation-minimum FE" with only polynomial security, summarized in the lemma below.

Lemma 9.1 ([KNTY19]). *Assuming the existence of "weakly selectively secure, 1-key, sub-linearly succinct FE for circuits" (per definitions in [KNTY19]), i.e., a so-called "obfuscation-minimum FE with polynomial security", there exists an* FE scheme for circuits *(Definition 9.6) with* adaptive IND-CPA security *(Definition 2.2).*

9.2.4 Universal RAM and ABE for RAM

In this section, we define ABE for RAM after explaining some rationales behind certain subtleties in our formulation. (For a treatment of PHFE for RAM, see [JLL22].)

To obtain ABE for RAM, we will employ the standard transformation [QWW18] of using FE for circuits to compute LGRAM (with fixed-memory security). However, in LGRAM (Definition 9.2), the running time of Garble depends on the machine size. This dependency is inherited by all the efficiency parameters of the resultant ABE for RAM if we associate each key with a RAM. To get rid of this dependency, we fix some universal RAM U of size $\mathrm{poly}(\lambda)$[131] upon setting up the scheme, and associate with each key a piece of code interpreted by U.

The other issue is that LGRAM puts an upper bound on the running time and its incorrectness in case of exceeding the time limit is propagated to the ABE scheme. We avoid it[132] by defining $\varphi = \bot$ if the running time exceeds some super-polynomial value prescribed upon set-up.

The above explains the intended usage of ABE for RAM, yet we define it for general machines. Moreover, we make it handle λ-bit messages (it can be used as an AB-KEM for arbitrary messages).

Definition 9.7 (ABE for RAM). An *ABE scheme for RAM* is an ABE scheme (Definition 2.3) for

$$\mathsf{Params}_\lambda = \left\{ (M', T'_{\max}) \,\middle|\, M' \text{ is a 2-tape RAM and } T'_{\max} \in \mathbb{N}_+ \right\},$$

$$F_{\lambda,M',T'_{\max}} = X_{\lambda,M',T'_{\max}} = (\{0,1\}^{\ell_{\mathrm{cell}}})^{\leq 2^{\ell_{\mathrm{addr}}}}, \qquad Y_{\lambda,M',T'_{\max}} = \{0,1\}^\lambda,$$

$$P_{\lambda,M',T'_{\max}}(f,x) = \begin{cases} (T,b), & \text{if } \mathsf{time}(M',f,x) = T \in [2, T'_{\max}] \text{ and} \\ & \quad \mathsf{outS}(M',f,x)[T-1] = b \in \{0,1\}; \\ \bot, & \text{otherwise.} \end{cases}$$

Here, we are using M' as a decision machine and consider the computation *accepting* if and only if its last-step output is 1.

---

[131] U is not the same RAM across different values of λ — its input address length should be $\omega(\log \lambda)$ to accommodate all polynomially long input.

[132] An alternative solution is to blatantly reveal the message when the running time is too large —

Remark 9.7 (unbounded scheme and polynomial security). When Definitions 2.1 and 2.2 are instantiated into ABE for RAM (Definition 9.7), the scheme is *unbounded*, meaning that $T_{\max}$ can be exponentially large, yet security only holds for polynomially bounded instance running time. See also Remarks 9.3 and 9.4.

Remark 9.8 (falsifiability and efficient reduction). In Definition 2.2, limiting T to be polynomially bounded ensures that PHFE security is falsifiable [Nao03,GW11]. Roughly speaking, falsifiability means that there is an efficient procedure to verify with high confidence that a given adversary breaks the property, if it indeed does so. We want to base the security of our schemes on falsifiable assumptions by efficient reductions, but such proofs often only prove falsifiable properties, so it is important to make the security definition itself falsifiable.

Concretely, running $\mathsf{Exp}^{\beta}_{\mathrm{PHFE}}$ involves checking whether decryption is unauthorized for all the keys requested by the adversary. Due to time hierarchy theorems, the check cannot be done significantly faster than performing the RAM computations, so the policy evaluation time must be polynomially bounded. From the perspective of proofs, the security loss incurred by the reductions (from ABE to the underlying ingredients) grows polynomially with T, again prompting us to require a polynomial upper bound.

9.2.5 Indistinguishability Obfuscation

We will use indistinguishability obfuscation for circuits *with polynomial-size domains* as a building block.

Definition 9.8 ($i\mathcal{O}$). An *indistinguishability obfuscator* is an efficient algorithm $i\mathcal{O}(1^\lambda, C)$ taking a circuit C as input. It outputs an obfuscated circuit $\widetilde{C}$. The obfuscator must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, circuit $C : \{0,1\}^n \to \{0,1\}^m$, and input $x \in \{0,1\}^n$,

$$\Pr\left[i\mathcal{O}(1^\lambda, C)(x) = C(x)\right] = 1.$$

The $i\mathcal{O}$ is *secure for polynomial-size domains* if for all polynomial-size $(1^{2^n}, C_0, C_1)$ such

---

security is not affected because the adversary is not allowed to choose keys and ciphertexts exhibiting super-polynomial instance running time. The current treatment follows that for general PHFE, where non-halting computation is best dealt as $\perp$.

that $C_0, C_1 : \{0,1\}^n \to \{0,1\}^m$ have identical truth tables and sizes, it holds that

$$\left\{(1^\lambda, C_0, C_1, i\mathcal{O}(1^\lambda, C_0))\right\}_{\lambda\in\mathbb{N}} \approx \left\{(1^\lambda, C_0, C_1, i\mathcal{O}(1^\lambda, C_1))\right\}_{\lambda\in\mathbb{N}}.$$

Remark 9.9. The above definition does not allow the obfuscator to work by simply outputting the truth table, as the constraint of having polynomial-size domains only applies to security, not efficiency nor correctness. See also Remarks 9.3, 9.4, and 9.7.

Secure $i\mathcal{O}$ for polynomial-size domains is implied by polynomially secure FE for circuits, via either a tight security reduction [LT17] of FE-to-$i\mathcal{O}$ transformation or decomposable obfuscation [LZ17].

Lemma 9.2 ([LT17,LZ17]). *Assuming the existence of* secure FE for circuits *(Definitions 9.6 and 2.2), there exists an $i\mathcal{O}$ for circuits that is* secure for polynomial-size domains *(Definition 9.8).*

9.2.6 Laconic Oblivious Transfer

We will use selectively secure laconic oblivious transfer as a building block.

Definition 9.9 ((updatable) LOT [CDG+17]). A *laconic oblivious transfer scheme* consists of four algorithms.

- $\mathsf{HashGen}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}})$ takes as input a cell length ℓ_{cell} and an address length ℓ_{addr}. It outputs a hash key hk in polynomial time.
- $\mathsf{Hash}(1^\lambda, \mathsf{hk}, D)$ takes as input hk and a database $D \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$. It *deterministically* outputs a hash h and a processed database $\widehat{D}$ in time $|D|\,\mathrm{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}})$, with the hash length being $\ell_{\mathsf{hash}} = \mathrm{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}})$.
- $\mathsf{SendRead}(1^\lambda, \mathsf{hk}, h, i, \{m_j^b\}_{j\in[\ell_{\mathsf{cell}}]}^{b\in\{0,1\}})$ takes as input hk, h, an address $i \in [2^{\ell_{\mathsf{addr}}}]$, and ℓ_{cell} pairs of messages of identical length. It outputs a read ciphertext rct. The algorithm runs in polynomial time.
- $\mathsf{RecvRead}^{\widehat{D}}(1^\lambda, \mathsf{hk}, h, i, \mathsf{rct})$ takes as input $\mathsf{hk}, h, i, \mathsf{rct}$. Given random access to $\widehat{D}$, it is supposed to recover $\{m_j^{D[i][j]}\}_{j\in[\ell_{\mathsf{cell}}]}$. It runs in time $\mathrm{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}}, |\mathsf{rct}|)$.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $\ell_{\mathsf{cell}}, \ell_{\mathsf{addr}} \in \mathbb{N}_+$, $D \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$, $i \in [|D|]$, $\{m_j^b\}_{j\in[\ell_{\mathsf{cell}}]}^{b\in\{0,1\}}$ of identical length, it holds that

$$\Pr\left[\begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}) \\ (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D) \\ \mathsf{rct} \xleftarrow{\$} \mathsf{SendRead}(1^\lambda, \mathsf{hk}, h, i, \{m_j^b\}_{j\in[\ell_{\mathsf{cell}}]}^{b\in\{0,1\}}) \\ \qquad : \quad \mathsf{RecvRead}^{\widehat{D}}(1^\lambda, \mathsf{hk}, h, i, \mathsf{rct}) = \{m_j^{D[i][j]}\}_{j\in[\ell_{\mathsf{cell}}]} \end{array}\right] = 1.$$

An *updatable* LOT has two additional algorithms.

- $\mathsf{SendWrite}(1^\lambda, \mathsf{hk}, h, i, \mathsf{wdata}, \{m_j^b\}_{j\in[\ell_{\mathsf{hash}}]}^{b\in\{0,1\}})$ takes as input hk, h, i, an overwriting string $\mathsf{wdata} \in \{0,1\}^{\ell_{\mathsf{cell}}}$, and ℓ_{hash} pairs of messages of identical length. It outputs a write ciphertext wct in polynomial time.
- $\mathsf{RecvWrite}^{\widehat{D}}(1^\lambda, \mathsf{hk}, h, i, \mathsf{wdata}, \mathsf{wct})$ takes as input $\mathsf{hk}, h, i, \mathsf{wdata}, \mathsf{wct}$. Given random access to $\widehat{D}$, it is supposed to update $\widehat{D}$ to $\widehat{D}'$ and recover $\{m_j^{h'[j]}\}_{j\in[\ell_{\mathsf{hash}}]}$, where D' is D with $D'[i]$ replaced by wdata and h' is the hash of D'. The algorithm runs in time $\mathrm{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}}, |\mathsf{wct}|)$.

Correctness requires that for all $\lambda \in \mathbb{N}$, $\ell_{\mathsf{cell}}, \ell_{\mathsf{addr}} \in \mathbb{N}_+$, $D \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$, $i \in [|D|]$, $\mathsf{wdata} \in \{0,1\}^{\ell_{\mathsf{cell}}}$, $\{m_j^b\}_{j\in[\ell_{\mathsf{hash}}]}^{b\in\{0,1\}}$ of identical length,

$$\Pr\left[\begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}) \\ (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D) \qquad (h', \widehat{D}') \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D') \\ \mathsf{wct} \xleftarrow{\$} \mathsf{SendWrite}(1^\lambda, \mathsf{hk}, h, i, \mathsf{wdata}, \{m_j^b\}_{j\in[\ell_{\mathsf{hash}}]}^{b\in\{0,1\}}) \\ \qquad : \begin{array}{l} \mathsf{RecvWrite}^{\widehat{D}}(1^\lambda, \mathsf{hk}, h, i, \mathsf{wdata}, \mathsf{wct}) = \{m_j^{h'[j]}\}_{j\in[\ell_{\mathsf{hash}}]} \\ \quad \text{and } \widehat{D} \text{ is updated to } \widehat{D}' \text{ by RecvWrite} \end{array} \end{array}\right] = 1,$$

where D' is D with $D'[i]$ replaced by wdata.

Part of our LGRAM garbling procedure involves hashing the all-zero string (the initial working tape) under a newly generated LOT hash key. It must be done very efficiently.

Definition 9.10 (LOT fast initialization). An LOT scheme (Definition 9.9) has *fast initialization* if there is an efficient algorithm $\mathsf{Hash0s}(1^\lambda, \mathsf{hk}, S)$ computing the hash

of $(0^{\ell_{\mathsf{cell}}})^S$. More precisely, for all $\lambda \in \mathbb{N}$, $\ell_{\mathsf{cell}}, \ell_{\mathsf{addr}} \in \mathbb{N}_+$, $S \in [2^{\ell_{\mathsf{addr}}}]$, it holds that

$$\Pr\left[\begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}) \\ (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, (0^{\ell_{\mathsf{cell}}})^S) \end{array} : \mathsf{Hash0s}(1^\lambda, \mathsf{hk}, S) = h\right] = 1.$$

The generic bootstrapping procedure [CDG+17,AL18,KNTY19] for LOT using Merkle tree always yields a scheme with fast initialization, because the Merkle hash of an all-zero string can be computed in time $\mathrm{poly}(\lambda, \ell_{\mathsf{cell}}, \ell_{\mathsf{addr}})$.

Security. Following [KNTY19], we consider database-selective security for LOT.

Definition 9.11 (LOT read security [CDG+17]). An LOT scheme (Definition 9.9) is *read-secure* if there exists an efficient simulator SimRead such that for all polynomial-size $1^{\ell_{\mathsf{cell}}}$, $1^{\ell_{\mathsf{addr}}}$, $D \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$, $i \in [|D|]$, $\{m_j^b\}_{j\in[\ell_{\mathsf{cell}}]}^{b\in\{0,1\}}$ of identical length, it holds that

$$\begin{aligned} &\left\{\left(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, \mathsf{hk}, D, i, \{m_j^b\}_{j\in[\ell_{\mathsf{cell}}]}^{b\in\{0,1\}}, \mathsf{rct}\right)\right\}_{\lambda\in\mathbb{N}} \approx \left\{\left(\cdots, \widetilde{\mathsf{rct}}\right)\right\}_{\lambda\in\mathbb{N}}, \text{ where} \\ &\quad \mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}), \quad (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D), \\ &\quad \mathsf{rct} \xleftarrow{\$} \mathsf{SendRead}(1^\lambda, \mathsf{hk}, h, i, \{m_j^b\}_{j\in[\ell_{\mathsf{cell}}]}^{b\in\{0,1\}}), \\ &\quad \widetilde{\mathsf{rct}} \xleftarrow{\$} \mathsf{SimRead}(1^\lambda, \mathsf{hk}, D, i, \{m_j^{D[i][j]}\}_{j\in[\ell_{\mathsf{cell}}]}). \end{aligned}$$

Definition 9.12 (LOT write security [CDG+17]). An updatable LOT scheme (Definition 9.9) is *write-secure* if there exists an efficient simulator SimWrite such that for all polynomial-size $1^{\ell_{\mathsf{cell}}}$, $1^{\ell_{\mathsf{addr}}}$, $D \in (\{0,1\}^{\ell_{\mathsf{cell}}})^{\leq 2^{\ell_{\mathsf{addr}}}}$, $i \in [|D|]$, $\mathsf{wdata} \in \{0,1\}^{\ell_{\mathsf{cell}}}$, $\{m_j^b\}_{j\in[\ell_{\mathsf{hash}}]}^{b\in\{0,1\}}$ of identical length, letting D' be D with $D'[i]$ replaced by wdata, it holds that

$$\begin{aligned} &\left\{\left(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, \mathsf{hk}, D, i, \mathsf{wdata}, \{m_j^b\}_{j\in[\ell_{\mathsf{hash}}]}^{b\in\{0,1\}}, \mathsf{wct}\right)\right\}_{\lambda\in\mathbb{N}} \approx \left\{\left(\cdots, \widetilde{\mathsf{wct}}\right)\right\}_{\lambda\in\mathbb{N}}, \text{ where} \\ &\quad \mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}(1^\lambda, 1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}), \\ &\quad (h, \widehat{D}) \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D), \quad (h', \widehat{D}') \leftarrow \mathsf{Hash}(1^\lambda, \mathsf{hk}, D'), \\ &\quad \mathsf{wct} \xleftarrow{\$} \mathsf{SendWrite}\left(1^\lambda, \mathsf{hk}, h, i, \mathsf{wdata}, \{m_j^b\}_{j\in[\ell_{\mathsf{hash}}]}^{b\in\{0,1\}}\right), \\ &\quad \widetilde{\mathsf{wct}} \xleftarrow{\$} \mathsf{SimWrite}\left(1^\lambda, \mathsf{hk}, D, i, \mathsf{wdata}, \{m_j^{h'[j]}\}_{j\in[\ell_{\mathsf{hash}}]}\right). \end{aligned}$$

Remark 9.10 (partially adaptive security). The above definitions are index- and message-selective. Yet, without loss of generality, we may assume that security holds even when the adversary is allowed to choose i and m_j^b's adaptively (dependent on hk). Adaptive

security with respect to i can be obtained by a standard guessing argument as noted in [GOS18], and that with respect to m_j^b's by using the scheme as a key encapsulation mechanism (or by encrypting bit by bit and following a standard hybrid argument).

Lemma 9.3 ([LZ17,CDG+17,AL18,KNTY19]). *Assuming the existence of* secure FE for circuits *(Definitions 9.6 and 2.2), there exists a read- and write-secure* updatable LOT *with fast initialization (Definitions 9.9, 9.10, 9.11, and 9.12).*

9.2.7 Garbled Circuits

Following the formulation in [GS18a], we make the garbling procedure take the labels as input instead of letting it generate them. However, their formulation cannot be perfectly correct,[133] which we fix by incorporating the point-and-permute [BMR90] technique.

Definition 9.13 (garbled circuits [Yao82]). A *circuit garbling scheme* with label length $\ell_L(\lambda) \le \mathrm{poly}(\lambda)$ consists of two efficient algorithms.

- $\mathsf{Garble}(1^\lambda, C, \pi, \{L_{i,b}\}_{i\in[n],b\in\{0,1\}})$ takes as input a circuit C of input length n, a point-and-permute string $\pi \in \{0,1\}^n$, and n pairs of labels (each label of length $\ell_L(\lambda)$). It outputs a garbled circuit $\widehat{C}$.
- $\mathsf{Eval}(1^\lambda, \widehat{C}, \widehat{x}, \{L_i\}_{i\in[n]})$ takes as input $\widehat{C}$, the permuted input, and one set of labels. It is supposed to compute $C(x)$.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $C : \{0,1\}^n \to \{0,1\}^\star$, $x \in \{0,1\}^n$,

$$\Pr\left[\begin{array}{l} \pi \xleftarrow{\$} \{0,1\}^n \\ L_{i,b} \xleftarrow{\$} \{0,1\}^{\ell_L(\lambda)} \text{ for all } i \in [n], b \in \{0,1\} \\ \widehat{C} \xleftarrow{\$} \mathsf{Garble}(1^\lambda, C, \pi, \{L_{i,b}\}_{i\in[n],b\in\{0,1\}}) \\ y \leftarrow \mathsf{Eval}(1^\lambda, \widehat{C}, x \oplus \pi, \{L_{i,x[i]}\}_{i\in[n]}) \end{array} : \quad y = C(x)\right] = 1.$$

[133] The inevitable correctness error of [GS18a] arises from the absence of point-and-permute string and the (albeit unlikely) possibility of two labels for one input bit being the same.

The scheme is *secure* if there exists an efficient simulator $\widetilde{\mathsf{Garble}}$ such that for all polynomial-size (C, x),

$$\begin{aligned}&\left\{\left(1^\lambda, C, x, \mathsf{Garble}(1^\lambda, C, \quad\quad\quad\quad \pi, \{L_{i,b}\quad\}_{i\in[n],b\in\{0,1\}}), x\oplus\pi, \{L_{i,x[i]}\}_{i\in[n]}\right)\right\}_{\lambda\in\mathbb{N}}\\ \approx &\left\{\left(1^\lambda, C, x, \widetilde{\mathsf{Garble}}(1^\lambda, 1^{|C|}, C(x), x\oplus\pi, \{L_{i,x[i]}\}_{i\in[n]}\quad\quad\quad), x\oplus\pi, \{L_{i,x[i]}\}_{i\in[n]}\right)\right\}_{\lambda\in\mathbb{N}},\end{aligned}$$

where $\pi \xleftarrow{\$} \{0,1\}^n$ and $L_{i,b} \xleftarrow{\$} \{0,1\}^{\ell_L(\lambda)}$ for all $i \in [n], b \in \{0,1\}$.

9.2.8 Puncturable Pseudorandom Function

We need puncturable pseudorandom functions as a building block.[134]

Definition 9.14 (PPRF [BW13]). A *puncturable pseudorandom function* with key (resp. input, output) length $\ell_{\mathrm{key}}(\lambda)$ (resp. $\ell_{\mathrm{in}}(\lambda)$, $\ell_{\mathrm{out}}(\lambda)$; all poly$(\lambda)$-bounded) consists of two efficient algorithms.

- $\mathsf{Eval}(1^\lambda, k, x)$ takes as input a (punctured or not) key k and an input $x \in \{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$. It *deterministically* outputs a bit-string of length $\ell_{\mathrm{out}}(\lambda)$.

- $\mathsf{Puncture}(1^\lambda, k, \mathfrak{p})$ takes as input a non-punctured key $k \in \{0,1\}^{\ell_{\mathrm{key}}(\lambda)}$ and some set $\mathfrak{p} \subseteq \{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$. It outputs a punctured key $\mathring{k}_{\mathfrak{p}}$.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, $k \in \{0,1\}^{\ell_{\mathrm{key}}(\lambda)}$, $\mathfrak{p} \subseteq \{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$, and input $x \in \{0,1\}^{\ell_{\mathrm{in}}(\lambda)} \setminus \mathfrak{p}$, it holds that

$$\Pr\left[\mathsf{Eval}(1^\lambda, \mathsf{Puncture}(1^\lambda, k, \mathfrak{p}), x) = \mathsf{Eval}(1^\lambda, k, x)\right] = 1.$$

The scheme is *secure* if for all polynomial-size $\mathfrak{p} \subseteq \{0,1\}^{\ell_{\mathrm{in}}(\lambda)}$, it holds that

$$\left\{\left(1^\lambda, \mathfrak{p}, \mathring{k}_{\mathfrak{p}}, \{\mathsf{Eval}(1^\lambda, k, x)\}_{x\in\mathfrak{p}}\right)\right\}_{\lambda\in\mathbb{N}} \approx \left\{\left(1^\lambda, \mathfrak{p}, \mathring{k}_{\mathfrak{p}}, \{r_x\}_{x\in\mathfrak{p}}\right)\right\}_{\lambda\in\mathbb{N}},$$

where $k \xleftarrow{\$} \{0,1\}^{\ell_{\mathrm{key}}(\lambda)}$, $\mathring{k}_{\mathfrak{p}} \xleftarrow{\$} \mathsf{Puncture}(1^\lambda, k, \mathfrak{p})$, and $r_x \xleftarrow{\$} \{0,1\}^{\ell_{\mathrm{out}}(\lambda)}$ for all $x \in \mathfrak{p}$.

[134] We also use selectively secure usual PRF, yet omit it here as it is implied by PPRF.

9.2.9 Secret-Key Encryption

We will use secret-key encryption as a building block.

Definition 9.15 (SKE). A *secret-key encryption scheme* consists of three efficient algorithms.

- $\mathsf{Gen}(1^\lambda)$ outputs a key k.
- $\mathsf{Enc}(1^\lambda, k, m)$ takes as input k and a message m of arbitrary length, and outputs a ciphertext c.
- $\mathsf{Dec}(1^\lambda, k, c)$ takes as input k, c. It is supposed to recover the message.

The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$ and $m \in \{0,1\}^\star$, it holds that

$$\Pr\left[k \xleftarrow{\$} \mathsf{Gen}(1^\lambda) \ : \ \mathsf{Dec}\big(1^\lambda, \mathsf{Enc}(1^\lambda, k, m)\big) = m \right] = 1.$$

It has *pseudorandom ciphertexts* if the ciphertext length is a function of λ and the message length (independent of key generation and encryption randomness) and for all $\lambda \in \mathbb{N}$ and polynomially bounded $\{m_q\}_{q\in[Q]}$, it holds that

$$\left\{\left(1^\lambda, 1^Q, \{m_q, c_q\}_{q\in[Q]}\right)\right\}_{\lambda\in\mathbb{N}} \approx \left\{\left(1^\lambda, 1^Q, \{m_q, r_q\}_{q\in[Q]}\right)\right\}_{\lambda\in\mathbb{N}},$$

where $k \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, and $c_q \xleftarrow{\$} \mathsf{Enc}(1^\lambda, k, m_q)$, $r_q \xleftarrow{\$} \{0,1\}^{|c_q|}$ for all $q \in [Q]$.

9.2.10 Primitive Related to Lower Bound

We formally define the class of ABE to which the lower bound applies. The primitive is *restricted* from broadcast encryption, in the sense that the users are paired and each ciphertext can be decrypted by exactly one user in each pair. The required security notion is also weak — it does not consider collusion among multiple keys nor adaptive attacks.

Definition 9.16 (restricted BE and security). A *restricted broadcast encryption scheme* is an ABE scheme (Definition 2.3) with

$$\mathsf{Params}_\lambda = \{\,1^N \mid N \in \mathbb{N}\,\}, \qquad (i,r) \in F_{\lambda,1^N} = [N] \times \{0,1\}, \qquad R \in X_{\lambda,1^N} = \{0,1\}^N,$$

$$\mu \in Y_{\lambda,1^N} = \{0,1\}^\lambda, \qquad P_{\lambda,1^N}((i,r),R) = \neg(R[i] \oplus r) = \begin{cases} 1, & \text{if } R[i] = r; \\ 0, & \text{otherwise.} \end{cases}$$

The scheme is *1-key secure for random challenge against uniform adversaries* (or simply *weakly 1-key secure*) if

$$\left\{(1^\lambda, 1^N, \mathsf{mpk}, R, i^*, \mu_0, \mathsf{sk}_{i^*,\neg R[i^*]}, \boxed{\mathsf{ct}_0})\right\}_{\lambda\in\mathbb{N}} \approx \left\{(1^\lambda, 1^N, \mathsf{mpk}, R, i^*, \mu_0, \mathsf{sk}_{i^*,\neg R[i^*]}, \boxed{\mathsf{ct}_1})\right\}_{\lambda\in\mathbb{N}}$$

with the components being (β is 0 or 1)

$$(\mathsf{mpk},\mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^N), \qquad R \xleftarrow{\$} \{0,1\}^N, \qquad i^* \xleftarrow{\$} [N], \qquad \mu_\beta \xleftarrow{\$} \{0,1\}^\lambda,$$

$$\mathsf{sk}_{i^*,\neg R[i^*]} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, (i^*, \neg R[i^*])), \qquad \mathsf{ct}_\beta \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, R, \mu_\beta),$$

for all polynomially bounded $N = N(\lambda)$,[135] where the computational indistinguishability only has to hold against *uniform* adversaries.

9.3 Lower Bound on Ciphertext Size and Decryption Time

We start with the negative result.

Theorem 9.4 (¶). *For all* weakly 1-key secure restricted BE *(Definition 9.16),*

$$\max|\mathsf{ct}| \cdot \max T_{\mathsf{Dec}} \geq \frac{N}{1000}$$

for all polynomially bounded $N = N(\lambda)$ and sufficiently large λ, where ct *runs through all possible ciphertexts and T_{Dec} the time to probe R and produce output by* Dec*, both for R of length N.*

We remark that "for sufficiently large λ" is necessary because asymptotic security, by definition, is a tail property unaffected by finitely many λ's. The bound starts to hold once the scheme starts to be secure against the adversary used in the proof. While

[135] N need not be a computable function of λ. This does not make the security definition "non-uniformy", as a standard guessing argument (with advantage sign correction) applies to an interactive formulation in which the uniform and efficient $\mathcal{A}$ chooses N.

the statement and the proof here apply to perfectly correct schemes with polynomial security, it is straight-forward to adapt them for schemes with sufficient (say, constant) gap between correctness and security.

To prove Theorem 9.4, we need the following lemma:

Lemma 9.5 (adapted from [Unr07; Theorem 2]). *For all $N, P \in \mathbb{N}$ with $1 \le P \le N$, distribution D supported over some finite set Z, function $W : Z \times \{0,1\}^N \to \{0,1\}^S$, there exists a function $G : Z \times \{0,1\}^N \to \{0,1,\perp\}^N$ such that*

$$|\{j \in [N] : G(z,R)[j] \neq \perp\}| \le P \qquad \text{for all } z \in Z \text{ and } R \in \{0,1\}^N$$

and for all[136] *oracle (randomized) algorithm $\mathcal{B}^Y$ making at most T queries to Y,*

$$\left|\Pr\left[\mathcal{B}^R(z, W(z,R)) \to 1\right] - \Pr\left[\mathcal{B}^H(z, W(z,R)) \to 1\right]\right| \le \sqrt{\frac{ST}{2P}},$$

where

$$R \xleftarrow{\$} \{0,1\}^N, \qquad z \xleftarrow{\$} D, \qquad H[j] \begin{cases} = G(z,R)[j], & \text{if } G(z,R)[j] \neq \perp; \\ \xleftarrow{\$} \{0,1\}, & \text{if } G(z,R)[j] = \perp. \end{cases}$$

Proof (Theorem 9.4). Define

$$S = 1 + \max|\mathsf{ct}|, \qquad T = 1 + \max\{\text{number of bits in } R \text{ probed by } \mathsf{Dec}\}.$$

For $\lambda, N \ge 1$, it is necessary that $\max|\mathsf{ct}| \ge 1$ because ct can encode any string μ of length λ, and that $\max T_{\mathsf{Dec}} \ge T$ because Dec performs all the probes then produces at least 1 bit of output. Therefore,

$$\max|\mathsf{ct}| \cdot \max T_{\mathsf{Dec}} \ge \frac{\max|\mathsf{ct}| + 1}{2} \cdot \max T_{\mathsf{Dec}} \ge \frac{ST}{2}.$$

It remains to prove $ST \ge \frac{2N}{1000}$ for sufficiently large λ. It suffices to consider the case when $N \ge 2$ and $ST \le 2N$.

[136] Here, $\mathcal{B}^Y$ need not be efficient for the lemma to hold. The particular $\mathcal{B}^Y$ used below is efficient.

We prepare for Lemma 9.5. Let P be determined later, and[137]

$$z = (\mu, z_{\mathsf{Enc}}, \mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} D = \left\{ \begin{array}{c} \mu \xleftarrow{\$} \{0,1\}^\lambda \\ z_{\mathsf{Enc}} : \text{randomness for } \mathsf{Enc} \\ (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^N) \end{array} \right\},$$

$$w = W(z, R) = 0^{S-|\mathsf{ct}|-1} \| 1 \| \mathsf{ct}, \qquad \text{where} \quad \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, R, \mu; z_{\mathsf{Enc}}).$$

Let G be the function guaranteed by Lemma 9.5 and make $\mathcal{B}^Y(z, w)$ do the following.

- Sample $i^* \xleftarrow{\$} [N]$ and query $r^* \leftarrow Y[i^*]$.
- Read $\mu, \mathsf{mpk}, \mathsf{msk}$ from z. Read ct from w.
- Run $\mathsf{sk}_{i^*,r^*} \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, (i^*, r^*))$.
- Run $\mu' \xleftarrow{\$} \mathsf{Dec}^{\mathsf{mpk},(i^*,r^*),\mathsf{sk}_{i^*,r^*},Y,\mathsf{ct}}()$.
- Output 1 if and only if $\mu = \mu'$.

Indeed, $\mathcal{B}$ makes at most T queries to Y, the first to obtain r^*, the rest to run Dec.

For $k \in \{1,2,3,4,5\}$, write p_k for $\Pr[\mathcal{B}^{Y_k}(z, w; i^*) \to 1]$, where

$$R \xleftarrow{\$} \{0,1\}^N, \qquad z \xleftarrow{\$} D, \qquad w = W(z, R), \qquad i^* \xleftarrow{\$} [N], \qquad Y_1 = R,$$

$$Y_2[j] \begin{cases} = G(z,R)[j], & \text{if } G(z,R)[j] \neq \bot; \\ \xleftarrow{\$} \{0,1\}, & \text{if } G(z,R)[j] = \bot; \end{cases}$$

$$Y_3[j] \begin{cases} = G(z,R)[j], & \text{if } j \neq i^* \text{ and } G(z,R)[j] \neq \bot; \\ \xleftarrow{\$} \{0,1\}, & \text{if } j \neq i^* \text{ and } G(z,R)[j] = \bot; \\ \xleftarrow{\$} \{0,1\}, & \text{if } j = i^*; \end{cases}$$

$$Y_4[j] \begin{cases} = R[j], & \text{if } j \neq i^*; \\ \xleftarrow{\$} \{0,1\}, & \text{if } j = i^*; \end{cases} \qquad Y_5[j] \begin{cases} = R[j], & \text{if } j \neq i^*; \\ = \neg R[i^*], & \text{if } j = i^*. \end{cases}$$

By the correctness of the restricted BE scheme, $p_1 = 1$.

[137] $\mathcal{B}$ does not use z_{Enc}. It is included in z only to "provide randomness" for W.

From Lemma 9.5,

$$|p_1 - p_2| \le \sqrt{\frac{ST}{2P}}, \qquad |p_4 - p_3| \le \sqrt{\frac{ST}{2P}}.$$

Here, the second inequality is obtained by applying the lemma to

$$\mathcal{C}^Y(z,w) = \mathcal{B}^{Y'}(z,w;i^*), \qquad \text{where} \quad i^* \xleftarrow{\$} [N], \quad Y'[j] \begin{cases} = Y[j], & \text{if } j \neq i^*; \\ \xleftarrow{\$} \{0,1\}, & \text{if } j = i^*. \end{cases}$$

Clearly, $|p_2 - p_3| \le \frac{P}{N}$. Setting $P = \left\lceil \sqrt[3]{\frac{STN^2}{2}} \right\rceil$, we have

$$\begin{aligned} |p_1 - p_4| &\le |p_1 - p_2| + |p_2 - p_3| + |p_3 - p_4| \\ &\le \sqrt{\frac{ST}{2P}} + \frac{P}{N} + \sqrt{\frac{ST}{2P}} \le 3\sqrt[3]{\frac{ST}{2N}} + \frac{1}{N} < 4\sqrt[3]{\frac{ST}{2N}}, \end{aligned}$$

where the last inequality follows from $N \ge 2$. By how $Y[i^*]$ is set,

$$p_4 = \frac{p_1 + p_5}{2} \qquad \Longrightarrow \qquad p_5 = p_1 - 2(p_1 - p_4) \ge p_1 - 2|p_1 - p_4| > 1 - 8\sqrt[3]{\frac{ST}{2N}}.$$

Consider the following adversary $\mathcal{A}(\mathsf{mpk}, R, i^*, \mu_0, \mathsf{sk}_{i^*,\neg R[i^*]}, \mathsf{ct})$ against the weak 1-key security of the restricted BE scheme.

- Construct Y_5 from R and let $r^* \leftarrow Y_5[i^*] = \neg R[i^*]$.
- Run $\mu' \xleftarrow{\$} \mathsf{Dec}^{\mathsf{mpk},(i^*,r^*),\mathsf{sk}_{i^*,r^*},Y_5,\mathsf{ct}}()$, i.e., pretend $R[i^*]$ were $\neg R[i^*]$ and attempt to decrypt using the (supposedly non-decrypting) key given to $\mathcal{A}$.
- Output 1 if and only if $\mu' = \mu_0$.

If $\mathsf{ct} = \mathsf{ct}_1$ is an encryption of μ_1, then μ_0 is uniformly random and independent of everything else, hence

$$\Pr[\mathcal{A}(\cdots) \to 1 \text{ with } \mathsf{ct} = \mathsf{ct}_1] \le 2^{-\lambda}.$$

Note that $\mathcal{A}$ is a uniform adversary. By the security of the restricted BE scheme,

$$p_5 = \Pr[\mathcal{B}^{Y_5}(z,w;i^*) \to 1] = \Pr[\mathcal{A}(\cdots) \to 1 \text{ with } \mathsf{ct} = \mathsf{ct}_0] \le 2^{-\lambda} + \mathrm{negl}(\lambda) < \frac{1}{5}$$

for sufficiently large λ, which gives

$$1 - 8\sqrt[3]{\frac{ST}{2N}} < p_5 < \frac{1}{5} \qquad \Longrightarrow \qquad ST > \frac{2N}{1000}.$$

□

9.4 Bounded LGRAM with Fixed-Memory Security

In this section, we present our LGRAM with fixed-memory security. The construction is based on the works of [GS18a,GOS18,AL18,KNTY19].

Roughly speaking, [GS18a] constructs a circuit garbling scheme from *adaptively* secure LOT with *local* proof of security and lifts it to an adaptively secure scheme using somewhere equivocal encryption, which is further developed to obtain a garbled RAM scheme with fixed-memory security (or unprotected memory access) [GOS18]. The work of [AL18] modularizes the construction and shows how to obtain a succinct garbling scheme using obfuscation for circuits with polynomial-size domains, which is in turn implied by FE for circuits. The work of [KNTY19] shows that *selectively* secure LOT suffices and that such an LOT is implied by FE for circuits.

All of the works consider garbling schemes with no public input, and the security notions for their final products are simulation-based. Consequently, the length of the garbling necessarily [AIKW13] grows linearly with the input and output lengths. In contrast, our notion of garbling has a short private input and we are interested in indistinguishability and succinctness. (The contrast between long and short outputs do not show up here. See [JLL22] for the case when it matters.)

9.4.1 Construction

Ingredients of Construction 9.1. Let

- $i\mathcal{O}$ be a circuit obfuscator (Definition 9.8),
- LOT an updatable LOT with fast initialization (Definitions 9.9 and 9.10),

$$\mathsf{LOT} = (\mathsf{LOT.HashGen}, \mathsf{LOT.Hash}, \mathsf{LOT.SendRead}, \mathsf{LOT.RecvRead},$$
$$\mathsf{LOT.SendWrite}, \mathsf{LOT.RecvWrite}, \mathsf{LOT.Hash0s}),$$

- $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$ a circuit garbling scheme (Definition 9.13), and
- $\mathsf{PPRF} = (\mathsf{PPRF.Puncture}, \mathsf{PPRF.Eval})$ a puncturable PRF (Definition 9.14).

Construction 9.1 (bounded LGRAM with fixed-memory security). Our bounded LGRAM works as follows.

$\mathsf{GenAugCPU}[\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, k](t)$

Hardwired. $\mathsf{hk}_\tau, \mathsf{hk}_{\mathsf{work}}$, LOT hash keys for the tapes;
$h_\tau, |D_\tau|$, LOT hashes and lengths of the input tapes;
k, PPRF key.

Input. $t \in [T_{\max}]$, current time (step number).

Output. $\widehat{\mathsf{AugCPU}}_t$, the garbled "augmented step circuit" at time t, computed as follows.

1. Obtain randomness from PPRF:

   $(\pi_t, \{L_{t,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}, r_t^{\mathsf{GC}}) \leftarrow \mathsf{PPRF.Eval}(k, t)$

   $(\pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}) \leftarrow \mathsf{PPRF.Eval}(k, t+1)$ with $r_{t+1}^{\mathsf{LOT}}, r_{t+1}^{\mathsf{GC}}$ removed

2. Garble AugCPU (Figure 9.4):

   $$\mathsf{AugCPU}_t \leftarrow \mathsf{AugCPU}\left[\begin{array}{l}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}\end{array}\right]$$

   output $\widehat{\mathsf{AugCPU}}_t \leftarrow \mathsf{GC.Garble}(\mathsf{AugCPU}_t, \pi_t, \{L_{t,i,b}\}_{i,b}; r_t^{\mathsf{GC}})$

Figure 9.3. The circuit GenAugCPU in Construction 9.1.

- $\mathsf{Compress}(1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, \tau, D_\tau)$ takes as input the cell length, the address length, the tape index, and the tape content. It runs

  $$\mathsf{hk}_\tau \xleftarrow{\$} \mathsf{LOT.HashGen}(1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}), \qquad (h_\tau, \widehat{D}_\tau) \leftarrow \mathsf{LOT.Hash}(\mathsf{hk}_\tau, D_\tau),$$

  and outputs $\mathsf{digest}_\tau = (\mathsf{hk}_\tau, h_\tau, |D_\tau|)$.

- $\mathsf{Garble}(T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]})$ takes as input an upper bound of the running time, the machine, and the input tape digests. It runs

  $$\mathsf{hk}_{\mathsf{work}} \xleftarrow{\$} \mathsf{LOT.HashGen}(1^{\ell_{\mathrm{CELL}}}, 1^{\ell_{\mathrm{ADDR}}}), \qquad h_{\mathsf{work},0} \leftarrow \mathsf{LOT.Hash0s}(\mathsf{hk}_{\mathsf{work}}, T_{\max}),$$

  samples PPRF key k, prepares GenAugCPU using M (Figure 9.3), and runs

  $$\widehat{\mathsf{GenAugCPU}} \xleftarrow{\$} i\mathcal{O}(\mathsf{GenAugCPU}[k, \{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}]).$$

GenAugCPU is padded to size $\mathrm{poly}(\lambda, |M|, \log T_{\max})$ for the security proof to work. The algorithm computes using k and splits π, L for $\widetilde{\mathsf{AugCPU}}_1$ as defined in Figure 9.4,

$$\begin{aligned} \pi_1 :&\quad \pi_1^w, \pi_1^{\mathsf{st}}, \pi_1^{\mathsf{rdata}}, \pi_1^h, \\ \{L_{1,i,b}\}_{i,b} :&\quad \{L^w_{1,i,b}\}_{i\in[\ell_{\mathrm{in}}],b}, \{L^{\mathsf{st}}_{1,i,b}\}_{i\in[\ell_{\mathsf{st}}],b}, \{L^{\mathsf{rdata}}_{1,i,b}\}_{i\in[\ell_{\mathrm{CELL}}],b}, \{L^h_{1,i,b}\}_{i\in[\ell_{\mathrm{HASH}}],b}, \end{aligned}$$

sets $\mathsf{st}_0 = 0^{\ell_{\mathsf{st}}}$, $\mathsf{rdata}_0 = 0^{\ell_{\mathrm{CELL}}}$, and outputs

$$\widehat{M} = \left(\begin{array}{l} \mathsf{hk}_{\mathsf{work}}, \widetilde{\mathsf{GenAugCPU}}, \mathsf{st}_0 \oplus \pi_1^{\mathsf{st}}, \mathsf{rdata}_0 \oplus \pi_1^{\mathsf{rdata}}, h_{\mathsf{work},0} \oplus \pi_1^h, \\ \{L^{\mathsf{st}}_{1,i,\mathsf{st}_0[i]}\}_{i\in[\ell_{\mathsf{st}}]}, \{L^{\mathsf{rdata}}_{1,i,\mathsf{rdata}_0[i]}\}_{i\in[\ell_{\mathrm{CELL}}]}, \{L^h_{1,i,h_{\mathsf{work},0}[i]}\}_{i\in[\ell_{\mathrm{HASH}}]}, \end{array} \right),$$

$$\{L_{i,b} = (b \oplus \pi_1^w[i]) \| L^w_{1,i,b}\}_{i\in[\ell_{\mathrm{in}}],b\in\{0,1\}}.$$

- $\mathsf{Eval}^{D_1,\ldots,D_{\mathcal{T}}}(T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]}, \widehat{M}, \{L_i\}_{i\in[\ell_{\mathrm{in}}]})$ takes as input the upper bound of the running time, the machine, the input tape digests, the garbled machine, and a set of labels. It parses $\widehat{M}$ as specified in Garble, and does the following.

  1. Initialize by

$$\begin{aligned} (h_\tau, \widehat{D}_\tau) &\leftarrow \mathsf{LOT.Hash}(\mathsf{hk}_\tau, D_\tau) \qquad \text{for } \tau \in [\mathcal{T}], \\ (h_{\mathsf{work},0}, \widehat{D}_{\mathsf{work},0}) &\leftarrow \mathsf{LOT.Hash}(\mathsf{hk}_{\mathsf{work}}, (0^{\ell_{\mathrm{CELL}}})^{T_{\max}}), \\ \mathsf{st}_0 &\leftarrow 0^{\ell_{\mathsf{st}}}, \qquad \mathsf{rdata}_0 \leftarrow 0^{\ell_{\mathrm{CELL}}}, \qquad t \leftarrow 1. \end{aligned}$$

  2. Writing

$$\begin{aligned} X_t &= w \| \mathsf{st}_{t-1} \| \mathsf{rdata}_{t-1} \| h_{\mathsf{work},t-1}, \\ Y_t &= \text{parts of } (X_{t+1} \oplus \pi_{t+1}) \text{ and } L_{t+1} \text{ and } \mathsf{rct}_t, \mathsf{wct}_t, \end{aligned}$$

  run[138]

$$\begin{aligned} \widetilde{\mathsf{AugCPU}}_t &\leftarrow \widetilde{\mathsf{GenAugCPU}}(t), \\ (\mathsf{done}_t, \tau_t, i_t, \mathsf{wdata}_t, \mathsf{out}_t, Y_t) &\leftarrow \mathsf{GC.Eval}(\widetilde{\mathsf{AugCPU}}_t, X_t \oplus \pi_t, \{L_{t,i,X_t[i]}\}_i), \end{aligned}$$

  and halt if done_t is set. Otherwise, output out_t and continue.

[138] X_t itself is not needed for evaluation and is not supposed to be efficiently computable. The values of $(X_t \oplus \pi_t)$ and $\{L_{t,i,X_t[i]}\}_i$ for $t = 1$ are read from $\widehat{M}$, and those for $t > 1$ are computed by evaluating $\widetilde{\mathsf{AugCPU}}_{t-1}$ as explained later.

$$\mathsf{AugCPU}_t: \quad \mathsf{AugCPU}\left[\begin{matrix}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}\end{matrix}\right](w, \mathsf{st}_{t-1}, \mathsf{rdata}_{t-1}, h_{\mathsf{work},t-1})$$

Hardwired. $\mathsf{hk}_\tau, \mathsf{hk}_{\mathsf{work}}, h_\tau, |D_\tau|$, see Figure 9.3;
$\pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}$, to be selected for evaluating $\widehat{\mathsf{AugCPU}}_{t+1}$;
r_t^{LOT}, LOT randomness to select π, L for $\widehat{\mathsf{AugCPU}}_{t+1}$.

Input. $w, \mathsf{st}_{t-1}, \mathsf{rdata}_{t-1}$, short input, old state, last-read cell content;
$h_{\mathsf{work},t-1}$, LOT hash of the old working tape.

Output. $\mathsf{done}_t, \tau_t, i_t, \mathsf{wdata}_t, \mathsf{out}_t$, produced by $M^{D_1,\ldots,D_{\mathcal{T}}}(w)$ at time t;
selected (or LOT ciphertexts to select) π, L for $\widehat{\mathsf{AugCPU}}_{t+1}$;
computed as follows.

1. Execute one step of M:

   $(\mathsf{done}_t, \mathsf{st}_t, \tau_t, i_t, \mathsf{wdata}_t, \mathsf{out}_t) \leftarrow \mathsf{CPU}(|D_1|, \ldots, |D_{\mathcal{T}}|, w, \mathsf{st}_{t-1}, \mathsf{rdata}_{t-1})$

   output $\mathsf{done}_t, \tau_t, i_t, \mathsf{wdata}_t, \mathsf{out}_t$

2. Split r_t^{LOT}, and $\pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}$ by the input components of AugCPU:

$$\left(\begin{matrix} r^{\mathsf{LOT}}_{t,\mathsf{rct}}, r^{\mathsf{LOT}}_{t,\mathsf{wct}}, & \pi^{w}_{t+1}, \pi^{\mathsf{st}}_{t+1}, \pi^{\mathsf{rdata}}_{t+1}, \pi^{h}_{t+1},\\ \{L^{w}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{in}}],b}, & \{L^{\mathsf{st}}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{st}}],b},\\ \{L^{\mathsf{rdata}}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{CELL}}],b}, & \{L^{h}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{HASH}}],b}\end{matrix}\right) \leftarrow (r_t^{\mathsf{LOT}}, \pi_{t+1}, \{L_{t+1,i,b}\}_{i,b})$$

3. Select π, L for w, st_t if $\neg\mathsf{done}_t$:

   output $w \oplus \pi^{w}_{t+1}, \{L^{w}_{t+1,i,w[i]}\}_{i\in[\ell_{\mathsf{in}}]}, \quad \mathsf{st}_t \oplus \pi^{\mathsf{st}}_{t+1}, \{L^{\mathsf{st}}_{t+1,i,\mathsf{st}_t[i]}\}_{i\in[\ell_{\mathsf{st}}]}$

4a. If $\tau_t \in [\mathcal{T}]$, select π, L for $D_{\tau_t}[i_t]\|0^{\ell_{\mathsf{CELL}}-\ell_{\mathsf{cell}}}$ and $h_{\mathsf{work},t} = h_{\mathsf{work},t-1}$:

$$\textbf{output } \mathsf{rct}_t \leftarrow \mathsf{LOT.SendRead}\left(\begin{matrix}\mathsf{hk}_{\tau_t}, h_{\tau_t}, i_t,\\ \{(b \oplus \pi^{\mathsf{rdata}}_{t+1}[i])\|L^{\mathsf{rdata}}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{cell}}],b}\end{matrix}; r^{\mathsf{LOT}}_{t,\mathsf{rct}}\right)$$

and $\{(0 \oplus \pi^{\mathsf{rdata}}_{t+1}[i])\|L^{\mathsf{rdata}}_{t+1,i,0}\}_{i\in[\ell_{\mathsf{CELL}}]\setminus[\ell_{\mathsf{cell}}]}$

and $h_{\mathsf{work},t} \oplus \pi^{h}_{t+1}, \{L^{h}_{t+1,i,h_{\mathsf{work},t}[i]}\}_{i\in[\ell_{\mathsf{HASH}}]}$

4b. If $\tau_t = \mathsf{work}$, select π, L for $D_{\mathsf{work},t-1}[i_t]$ and updated $h_{\mathsf{work},t}$:

$$\textbf{output } \mathsf{rct}_t \leftarrow \mathsf{LOT.SendRead}\left(\begin{matrix}\mathsf{hk}_{\mathsf{work}}, h_{\mathsf{work},t-1}, i_t,\\ \{(b \oplus \pi^{\mathsf{rdata}}_{t+1}[i])\|L^{\mathsf{rdata}}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{CELL}}],b}\end{matrix}; r^{\mathsf{LOT}}_{t,\mathsf{rct}}\right)$$

and $$\mathsf{wct}_t \leftarrow \mathsf{LOT.SendWrite}\left(\begin{matrix}\mathsf{hk}_{\mathsf{work}}, h_{\mathsf{work},t-1}, i_t, \mathsf{wdata}_t,\\ \{(b \oplus \pi^{h}_{t+1}[i])\|L^{h}_{t+1,i,b}\}_{i\in[\ell_{\mathsf{HASH}}],b}\end{matrix}; r^{\mathsf{LOT}}_{t,\mathsf{wct}}\right)$$

Figure 9.4. The circuit AugCPU in Construction 9.1.

3. If $\tau_t \in [\mathcal{T}]$, pick rct_t from Y_t and run

$$\{(\mathsf{rdata}_t[i] \oplus \pi^{\mathsf{rdata}}_{t+1,i}) \| L^{\mathsf{rdata}}_{t+1,i,\mathsf{rdata}_t[i]}\}_{i\in[\ell_{\mathrm{cell}}]} \leftarrow \mathsf{LOT.RecvRead}^{\widehat{D}_{\tau_t}}(\mathsf{hk}_{\tau_t}, h_{\tau_t}, i_t, \mathsf{rct}_t).$$

Combine it with parts of $(X_{t+1} \oplus \pi_{t+1})$ and L_{t+1} already in Y_t to obtain $(X_{t+1} \oplus \pi_{t+1})$ and $\{L_{t+1,i,X_{t+1}[i]}\}_i$.

4. Otherwise, $\tau_t = \mathsf{work}$, then pick $\mathsf{rct}_t, \mathsf{wct}_t$ from Y_t, run

$$\{(\mathsf{rdata}_t[i] \oplus \pi^{\mathsf{rdata}}_{t+1,i}) \| L^{\mathsf{rdata}}_{t+1,i,\mathsf{rdata}_t[i]}\}_{i\in[\ell_{\mathrm{CELL}}]} \leftarrow \mathsf{LOT.RecvRead}^{\widehat{D}_{\mathsf{work},t-1}}(\mathsf{hk}_{\mathsf{work}}, h_{\mathsf{work},t-1}, i_t, \mathsf{rct}_t),$$

$$\{(h_{\mathsf{work},t} \oplus \pi^{h}_{t+1,i}) \| L^{h}_{t+1,i,h_{\mathsf{work},t}[i]}\}_{i\in[\ell_{\mathrm{HASH}}]} \leftarrow \mathsf{LOT.RecvWrite}^{\widehat{D}_{\mathsf{work},t-1}}\begin{pmatrix}\mathsf{hk}_{\mathsf{work}}, h_{\mathsf{work},t-1}, & \\ i_t, \mathsf{wdata}_t, & \mathsf{wct}_t\end{pmatrix},$$

updating $\widehat{D}_{\mathsf{work},t-1}$ into $\widehat{D}_{\mathsf{work},t}$. Again, combine it with parts of $(X_{t+1} \oplus \pi_{t+1})$ and L_{t+1} already in Y_t to obtain $(X_{t+1} \oplus \pi_{t+1})$ and $\{L_{t+1,i,X_{t+1}[i]}\}_i$.

5. Let $t \leftarrow t+1$ and go back to Step 2.

Correctness and Efficiency. They follow from those of all the ingredients and the invariant that AugCPU_t is evaluated on $w, \mathsf{st}_{t-1}, \mathsf{rdata}_{t-1}, h_{\mathsf{work},t-1}$ coinciding with those from the execution of M. We remark that the evaluation time is *not* instance-specific, because processing the empty working tape for LOT already takes time $T_{\max}\,\mathrm{poly}(\lambda, |M|)$.

9.4.2 Security

Theorem 9.6 (¶). *Suppose in Construction 9.1, $i\mathcal{O}$ is* secure for polynomial-size domains *(Definition 9.8),* LOT *is* read- *and* write-secure *(Definitions 9.11 and 9.12), and* GC, PPRF *are* secure *(Definitions 9.13 and 9.14), then the constructed scheme is* fixed-memory secure *(Definition 9.4).*

The proof follows the pebbling strategy in [GS18a]. Recall that in $\mathsf{Exp}^{\beta}_{\mathrm{LGRAM}}$, the LGRAM labels are selected for w_β, and each AugCPU_t performs a step of $M^{D_1,\ldots,D_{\mathcal{T}}}(w_\beta)$, for which we write $M^{\cdots}(w_\beta)$ hereafter in this proof. Along the hybrids from $\mathsf{Exp}^{0}_{\mathrm{LGRAM}}$ to $\mathsf{Exp}^{1}_{\mathrm{LGRAM}}$, we gradually switch the trailing steps from those for $M^{\cdots}(w_0)$ to those for $M^{\cdots}(w_1)$.

Specification of Hybrids. Each hybrid $\mathsf{H}_{\mathfrak{t},\mathfrak{s}}$ is indexed by an integer $0 \le \mathfrak{t} \le T_{\max} + 1$ and a set $\mathfrak{s} \subseteq [T_{\max}]$. The indices $\mathfrak{t}, \mathfrak{s}$ specify how AugCPU_t is garbled and what it does:

- when $t < \mathfrak{t}$, it runs the t^{th} step of $M^{\cdots}(w_0)$;
  - if $t \notin \mathfrak{s}$, the step is garbled normally;
  - if $t \in \mathfrak{s}$, the step is simulated with true randomness for labels and LOT encryption that selects the labels for AugCPU_{t+1};
- when $t = \mathfrak{t}$, it is simulated and outputs the labels so that $\mathsf{AugCPU}_{\mathfrak{t}+1}$ runs the $(\mathfrak{t}+1)^{\text{st}}$ step of $M^{\cdots}(w_1)$, i.e., it switches the execution from $M^{\cdots}(w_0)$ to $M^{\cdots}(w_1)$;
- when $t > \mathfrak{t}$, it is garbled normally but runs the t^{th} step of $M^{\cdots}(w_1)$, due to the behavior of $\mathsf{AugCPU}_{\mathfrak{t}}$.

There are two special cases:

- when $\mathfrak{t} = 0$, we give the LGRAM labels for w_1 (think of them as the output of the zeroth step circuit) to the adversary so that AugCPU_1 runs the first step of $M^{\cdots}(w_1)$ when the LGRAM is evaluated normally;
- when $t > T$ for $t \in \{\mathfrak{t}\} \cup \mathfrak{s}$, neither execution has a t^{th} step, and we simulate this step as if it were the last step of the execution, whose output would not select any label for the $(t+1)^{\text{st}}$ step.[139]

In the parlance of [GS18a], consider a line graph where vertex t represents step t,

- a gray pebble is placed on $t \in \mathfrak{s}$ if $t < \mathfrak{t}$ (the step is being simulated), and
- a black pebble is placed on $t \ge \mathfrak{t}$ (the step is done for good).

[139] This pretention trick unifies the usage of garbled circuit security. Since an out-of-range t^{th} step is pretended to repeat the last existent step, simulating it removes all the labels for the $(t+1)^{\text{st}}$ step, and consequently (and inductively), we can pretend that the set of labels for the $(t+1)^{\text{st}}$ step corresponding to the input to the last existent step were revealed, so that the $(t+1)^{\text{st}}$ step can also be pretended to repeat the last step. Without pretending, we will need another security notion of garbled circuits, namely that it can be simulated without any output if no input labels are given. This notion holds for many known constructions, but is not implied by the usual version.

However, the first black pebble (on t) is special for us as step t facilitates the transition from $M^{\cdots}(w_0)$ to $M^{\cdots}(w_1)$. We will use an optimized pebbling sequence [GS18a] to connect the hybrids.

Formally, let

$$\begin{aligned}
\mathsf{stS}(M, D_1, \ldots, D_{\mathcal{T}}, w_0) &= (\mathsf{st}_{0,1}, \ldots, \mathsf{st}_{0,T-1}),\\
\mathsf{stS}(M, D_1, \ldots, D_{\mathcal{T}}, w_1) &= (\mathsf{st}_{1,1}, \ldots, \mathsf{st}_{1,T-1}),\\
\mathsf{addrS}(\ldots, w_0) = \mathsf{addrS}(\ldots, w_1) &= (\tau_1, i_1, \ldots, \tau_{T-1}, i_{T-1}),\\
\mathsf{writeS}(\ldots, w_0) = \mathsf{writeS}(\ldots, w_1) &= (\mathsf{wdata}_1, \ldots, \mathsf{wdata}_{T-1}),\\
\mathsf{outS}(\ldots, w_0) = \mathsf{outS}(\ldots, w_1) &= (\mathsf{out}_1, \ldots, \mathsf{out}_{T-1}).
\end{aligned}$$

We also write $D_{\mathsf{work},t}$ for the working tape content, rdata_t for the read cell content, and $h_\tau, h_{\mathsf{work},t}$ for the LOT hashes — due to the constraint of fixed-memory security, the values of them in the two executions coincide. Define

$$X_{b,t} = w_b \| \mathsf{st}_{b,t-1} \| \mathsf{rdata}_{t-1} \| h_{\mathsf{work},t-1} \qquad \forall b \in \{0,1\}.$$

In $\mathsf{H}_{\mathsf{t},\mathfrak{s}}$, we change $\widetilde{\mathsf{GenAugCPU}}$ in $\widehat{M}$ to

$$i\mathcal{O}\left(\mathsf{GenAugCPU}'\left[\begin{array}{l}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \mathsf{t}, \mathfrak{s}, \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}}, \{\pi_t, \{L_{t,i,b}\}_{i,b}, \widetilde{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}\end{array}\right]\right),$$

where

- $\mathsf{GenAugCPU}'$ is shown in Figure 9.5,
- $\mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}}$ is a PPRF key punctured at $\{\mathsf{t}\}\cup\mathfrak{s}$,
- π_t's and $L_{t,i,b}$'s are random strings, and
- $\widetilde{\mathsf{AugCPU}}_t$'s are the simulated garbled step circuits.

We still use $\pi_t, L_{t,i,b}$ for part of the PPRF output when $t \notin \{\mathsf{t}\}\cup\mathfrak{s}$.

$\widetilde{\mathsf{AugCPU}}_t$'s are generated by

$$\mathsf{GC}.\widetilde{\mathsf{Garble}}(1^{|\mathsf{AugCPU}|}, Z_{b,t}, X_{0,t}\oplus\pi_t, \{L_{t,i,X_{0,t}[i]}\}_i) \text{ with } b = \begin{cases}0, & \text{if } t < \mathsf{t} \text{ and } t \in \mathfrak{s};\\ 1, & \text{if } t = \mathsf{t};\end{cases}$$

$$\mathsf{GenAugCPU}'\left[\begin{array}{l}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \mathsf{t}, \mathfrak{s}, \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}}, \{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}\end{array}\right](t)$$

Hardwired. $\mathsf{hk}_\tau, \mathsf{hk}_{\mathsf{work}}, h_\tau, |D_\tau|$, see Figure 9.3;
$\mathsf{t}, \mathfrak{s}$, switching time, hardwired step numbers;
$\mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}}$, punctured PPRF key;
$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}$, hardwired randomness and garbled steps.

Input. $t \in [T_{\max}]$, current time (step number).

Output. $\widehat{\mathsf{AugCPU}}_t$, computed as follows.

1. Check for hardwired steps, if $t \in \{\mathsf{t}\} \cup \mathfrak{s}$:

   output hardwired $\widehat{\mathsf{AugCPU}}_t$ and **terminate**

2. Otherwise, obtain randomness, either hardwired or from PPRF:

$$(\pi_t, \{L_{t,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}, r_t^{\mathsf{GC}}) \leftarrow \mathsf{PPRF.Eval}(\mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}}, t)$$

$$(\pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}) \leftarrow \begin{cases}\text{hardwired}, & \text{if } t+1 \in \{\mathsf{t}\}\cup\mathfrak{s};\\ \mathsf{PPRF.Eval}(\mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}}, t+1) \text{ with } r_{t+1}^{\mathsf{LOT}}, r_{t+1}^{\mathsf{GC}} \text{ removed}, & \\ & \text{if } t+1 \notin \{\mathsf{t}\}\cup\mathfrak{s};\end{cases}$$

3. Garble AugCPU (Figure 9.4) as done in GenAugCPU (Figure 9.3):

$$\mathsf{AugCPU}_t \leftarrow \mathsf{AugCPU}\left[\begin{array}{l}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}\end{array}\right]$$

   output $\widehat{\mathsf{AugCPU}}_t \leftarrow \mathsf{GC.Garble}(\mathsf{AugCPU}_t, \pi_t, \{L_{t,i,b}\}_{i,b}; r_t^{\mathsf{GC}})$

Figure 9.5. The circuit GenAugCPU′ in the proof of Theorem 9.6.

where $\mathsf{GC}.\widetilde{\mathsf{Garble}}$ is a simulator for GC, and $Z_{b,t}$ for $b \in \{0,1\}$ consists of $(\mathsf{done}_t, \tau_t, i_t, \mathsf{wdata}_t, \mathsf{out}_t)$, parts of $(X_{b,t+1} \oplus \pi_{t+1})$ and L_{t+1}, and $\widetilde{\mathsf{rct}}_t, \widetilde{\mathsf{wct}}_t$ (see Figure 9.4).

- If $\neg\mathsf{done}_t$, then $Z_{b,t}$ contains (dependent on b)

$$w_b \oplus \pi^w_{t+1}, \{L^w_{t+1,i,w_b[i]}\}_{i\in[\ell_{\mathrm{in}}]}, \quad \mathsf{st}_{b,t} \oplus \pi^w_{t+1}, \{L^{\mathsf{st}}_{t+1,i,\mathsf{st}_{b,t}[i]}\}_{i\in[\ell_{\mathrm{st}}]}.$$

- If $\tau_t \in [\mathcal{T}]$, then $Z_{b,t}$ contains (independent of b)

$$\widetilde{\mathsf{rct}}_t \xleftarrow{\$} \mathsf{LOT.SimRead}\left(\begin{array}{l}\mathsf{hk}_{\tau_t}, D_{\tau_t}, i_t,\\ \{(\mathsf{rdata}_t[i] \oplus \pi^{\mathsf{rdata}}_{t+1}[i]) \| L^{\mathsf{rdata}}_{t+1,i,\mathsf{rdata}_t[i]}\}_{i\in[\ell_{\mathrm{cell}}]}\end{array}\right),$$

$$\{(0 \oplus \pi^{\mathsf{rdata}}_{t+1}[i]) \| L^{\mathsf{rdata}}_{t+1,i,0}\}_{i\in[\ell_{\mathrm{CELL}}]\setminus[\ell_{\mathrm{cell}}]}, \quad h_{\mathsf{work},t} \oplus \pi^h_{t+1}, \{L^h_{t+1,i,h_{\mathsf{work},t}[i]}\}_{i\in[\ell_{\mathrm{HASH}}]}.$$

- If $\tau_t = \mathsf{work}$, then $Z_{b,t}$ contains (independent of b)

$$\widetilde{\mathsf{rct}}_t \xleftarrow{\$} \mathsf{LOT.SimRead}\left(\begin{array}{l}\mathsf{hk}_{\mathsf{work}}, D_{\mathsf{work},t-1}, i_t,\\ \{(\mathsf{rdata}_t[i] \oplus \pi^{\mathsf{rdata}}_{t+1}[i]) \| L^{\mathsf{rdata}}_{t+1,i,\mathsf{rdata}_t[i]}\}_{i\in[\ell_{\mathrm{CELL}}]}\end{array}\right),$$

$$\widetilde{\mathsf{wct}}_t \xleftarrow{\$} \mathsf{LOT.SimWrite}\left(\begin{array}{l}\mathsf{hk}_{\mathsf{work}}, D_{\mathsf{work},t-1}, i_t, \mathsf{wdata}_t,\\ \{(h_{\mathsf{work},t}[i] \oplus \pi^h_{t+1}[i]) \| L^h_{t+1,i,h_{\mathsf{work},t}[i]}\}_{i\in[\ell_{\mathrm{HASH}}]}\end{array}\right).$$

Connecting the Hybrids. We start with the experiments in Definition 9.4:

Claim 9.7 (¶). $\mathsf{Exp}^0_{\mathrm{LGRAM}} \approx \mathsf{H}_{T_{\max}+1,\varnothing}$ *and* $\mathsf{Exp}^1_{\mathrm{LGRAM}} \approx \mathsf{H}_{0,\varnothing}$.

It remains to connect $\mathsf{H}_{T_{\max}+1,\varnothing}$ and $\mathsf{H}_{0,\varnothing}$, for which we employ the pebbling strategy of [GOS18]. We state our claims in parallel with the pebbling rules.

Claim 9.8 (Rule A; ¶). *A gray pebble can be placed on or removed from $t^\star$ if $t^\star = 1$ or a gray pebble is placed on $(t^\star - 1)$. Formally, $\mathsf{H}_{\mathfrak{t},\mathfrak{s}} \approx \mathsf{H}_{\mathfrak{t},\{t^\star\}\cup\mathfrak{s}}$ for $t^\star \notin \mathfrak{s}$ and $t^\star < \mathfrak{t}$ if $t^\star = 1$ or $t^\star - 1 \in \mathfrak{s}$.*

Claim 9.9 (Rule B; ¶). *A gray pebble on $t^\star$ can be replaced by a black pebble if black pebbles are placed on all vertices greater than $t^\star$. Formally, $\mathsf{H}_{\mathfrak{t},\mathfrak{s}} \approx \mathsf{H}_{\mathfrak{t}-1,\mathfrak{s}\setminus\{\mathfrak{t}-1\}}$ if $\mathfrak{t}-1 \in \mathfrak{s}$ (here, $t^\star = \mathfrak{t}-1$). Moreover, $\mathsf{H}_{1,\varnothing} \approx \mathsf{H}_{0,\varnothing}$.*

Claim 9.9 is different from rule B in [GOS18] (no need for a gray pebble on $(t^\star - 1)$ for us) because the step corresponding to the first black pebble is always simulated.

It is helpful to consider a virtual vertex 0 (corresponding to $\widehat{M}$ and the LGRAM labels), where a gray pebble is initially placed, and put 0 into $\mathfrak{s}$. With the virtual vertex, the separate case $t^\star = 1$ in rule A is just a special case of $t^\star - 1 = 0 \in \mathfrak{s}$, and the separate case $\mathsf{H}_{1,\varnothing} \approx \mathsf{H}_{0,\varnothing}$ in rule B becomes a special case of the general rule (which would read $\mathsf{H}_{1,\{0\}} \approx \mathsf{H}_{0,\varnothing}$ instead). However, for consistency with existing literature, we will go without the virtual vertex in the text below.

The hybrid sequence can be built using an optimized pebbling strategy.

Lemma 9.10 ([Ben89,GS18a]). *There exists an efficient algorithm that on input* $1^{T_{\max}}$ *computes a pebbling sequence for a line graph of size* $T_{\max}$ *with* $\mathrm{O}(\log T_{\max})$ *gray pebbles at any time during the* $\mathrm{poly}(T_{\max})$ *moves using rule A or B, starting with no pebbles and ending with black pebbles on all vertices. As a corollary, the efficient algorithm can compute*

$$\mathsf{t}_0 = T_{\max} + 1, \mathfrak{s}_0 = \varnothing, \qquad c_1, \mathsf{t}_1, \mathfrak{s}_1, \qquad c_2, \mathsf{t}_2, \mathfrak{s}_2, \qquad \ldots,$$
$$c_{\mathrm{poly}(T_{\max})-1}, \mathsf{t}_{\mathrm{poly}(T_{\max})-1} = 1, \mathfrak{s}_{\mathrm{poly}(T_{\max})-1} = \varnothing, \qquad c_{\mathrm{poly}(T_{\max})}, \mathsf{t}_{\mathrm{poly}(T_{\max})} = 0, \mathfrak{s}_{\mathrm{poly}(T_{\max})} = \varnothing,$$

such that $\max_j |\mathfrak{s}_j| = \mathrm{O}(\log T_{\max})$ *and* $\mathsf{H}_{\mathsf{t}_{j-1},\mathfrak{s}_{j-1}} \approx \mathsf{H}_{\mathsf{t}_j,\mathfrak{s}_j}$ *by Claim* $c_j \in \{9.8, 9.9\}$.

A typical sequence of hybrid transitions is depicted in Figure 9.6.

Proof (Theorem 9.6). By Claims 9.7, 9.8, 9.9, Lemma 9.10, $\mathsf{Exp}^0_{\mathrm{LGRAM}} \approx \mathsf{Exp}^1_{\mathrm{LGRAM}}$. □

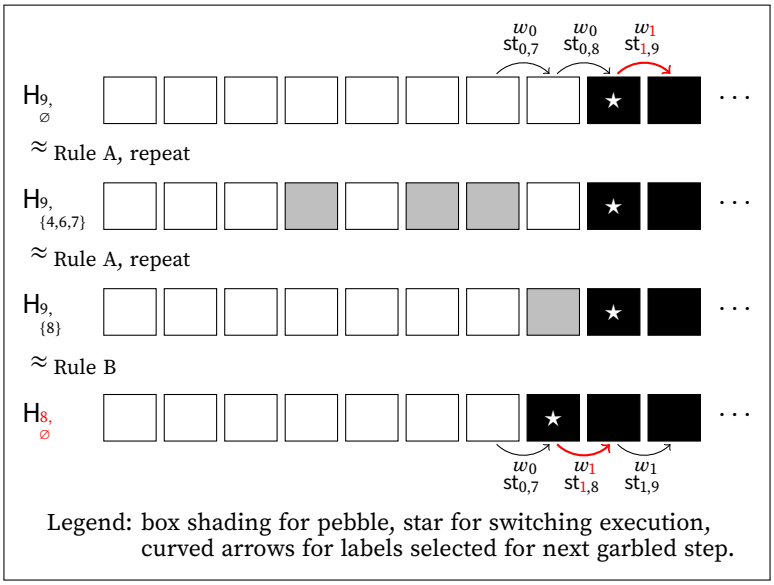


Figure 9.6. Typical hybrid transitions in the proof of Theorem 9.6.

We remark that for full rigor, $T_{\max}$ is a random variable, not a fixed number for each λ. It cannot be derandomized even in the non-uniform setting, because $\mathcal{A}$, given hk's returned for the input tapes, can choose $T_{\max}$ adaptively, whose randomness originates from that of the LGRAM scheme. This phenomenon is not uncommon and can be solved by either guessing $T_{\max}$ (or targeting a specific $T_{\max}$ in the non-uniform setting), or considering a polynomial upper bound of $T_{\max}$ and supplying appropriate definitions for hybrids with out-of-range indices.

Proof (Claim 9.7). Both follow from the security of $i\mathcal{O}$ for polynomial-size domains. □

Proof (Claim 9.8). Let $\mathsf{t}, \mathfrak{s}, t^\star$ be such that $t^\star \notin \mathfrak{s}$ and $t^\star < \mathsf{t}$ and either $t^\star = 1$ or $t^\star - 1 \in \mathfrak{s}$. The hybrids below are mostly identical to $\mathsf{H}_{\mathsf{t},\mathfrak{s}}$ except the contents hardwired into GenAugCPU′ are modified. We specify them.

- G_0. This is just $\mathsf{H}_{\mathsf{t},\mathfrak{s}}$, where the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}.$$

- G_1. In this hybrid, the PPRF key k is punctured at $\{t^\star, \mathsf{t}\} \cup \mathfrak{s}$ instead of $\{\mathsf{t}\} \cup \mathfrak{s}$, and the $(t^\star)^{\text{th}}$ garbled step (but not its randomness) is hardwired into GenAugCPU′, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \boxed{\{t^\star\}\cup\mathfrak{s}}, \quad \boxed{\mathring{k}_{\{\mathsf{t},t^\star\}\cup\mathfrak{s}}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}, \quad \boxed{\widehat{\mathsf{AugCPU}}_{t^\star},}$$

  where the newly hardwired information is computed using (Figure 9.5 Step 3)

$$(\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}, r^{\mathsf{LOT}}_{t^\star}, r^{\mathsf{GC}}_{t^\star}) \leftarrow \mathsf{PPRF.Eval}(k, t^\star),$$
$$\mathsf{AugCPU}_{t^\star} \leftarrow \mathsf{AugCPU}\left[\begin{matrix}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t^\star+1}, \{L_{t^\star+1,i,b}\}_{i,b}, r^{\mathsf{LOT}}_{t^\star}\end{matrix}\right],$$
$$\widehat{\mathsf{AugCPU}}_{t^\star} \leftarrow \mathsf{GC.Garble}(\mathsf{AugCPU}_{t^\star}, \pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}; r^{\mathsf{GC}}_{t^\star}).$$

  This is different from $\widehat{\mathsf{AugCPU}}_t$'s for $t \in \{\mathsf{t}\} \cup \mathfrak{s}$, which are simulated. The absence of $\pi_{t^\star}$ and $\{L_{t^\star,i,b}\}_{i,b}$ in GenAugCPU′ is fine, because they are only used in Step 2

(Figure 9.5) for $t = t^\star - 1$ and $t = t^\star$. In G_1, that branch is not taken for those two values of t as both of them are handled by Step 1. Therefore, the two versions of GenAugCPU′ in G_0 and G_1 have identical truth tables (and are padded appropriately to the same size), and $\mathsf{G}_0 \approx \mathsf{G}_1$ follows from the security of $i\mathcal{O}$ for polynomial-size domains.

- G_2. In this hybrid, $\mathsf{PPRF.Eval}(k, t^\star)$ is replaced by a uniformly random string, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \{t^\star\} \cup \mathsf{s}, \quad \mathring{k}_{\{\mathsf{t},t^\star\}\cup\mathsf{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathsf{s}}, \quad \widehat{\mathsf{AugCPU}}_{t^\star},$$

where

$$(\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}, r^{\mathsf{LOT}}_{t^\star}, r^{\mathsf{GC}}_{t^\star}) \leftarrow \boxed{\text{uniformly random}},$$
$$\mathsf{AugCPU}_{t^\star} \leftarrow \mathsf{AugCPU}\left[\begin{matrix}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t^\star+1}, \{L_{t^\star+1,i,b}\}_{i,b}, r^{\mathsf{LOT}}_{t^\star}\end{matrix}\right],$$
$$\widehat{\mathsf{AugCPU}}_{t^\star} \leftarrow \mathsf{GC.Garble}(\mathsf{AugCPU}_{t^\star}, \pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}; r^{\mathsf{GC}}_{t^\star}).$$

$\mathsf{G}_1 \approx \mathsf{G}_2$ follows from the security of PPRF.

- G_3. In this hybrid, $\widehat{\mathsf{AugCPU}}_{t^\star}$ is simulated instead of being normally generated, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \{t^\star\} \cup \mathsf{s}, \quad \mathring{k}_{\{\mathsf{t},t^\star\}\cup\mathsf{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathsf{s}}, \quad \widehat{\mathsf{AugCPU}}_{t^\star},$$

where

$$(\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}, r^{\mathsf{LOT}}_{t^\star}, \boxed{r^{\mathsf{GC}}_{t^\star}}) \leftarrow \text{uniformly random},$$
$$\widehat{\mathsf{AugCPU}}_{t^\star} \xleftarrow{\$} \boxed{\widehat{\mathsf{GC.Garble}}(1^{|\mathsf{AugCPU}|}, Z_{0,t^\star}, X_{0,t^\star} \oplus \pi_{t^\star}, \{L_{t^\star,i,X_{0,t^\star}[i]}\}_i)},$$
$$Z_{0,t^\star} : \pi_{t^\star+1}, \{L_{t^\star+1,i,b}\}_{i,b} \begin{cases} \text{in the clear,} \\ \text{in } \mathsf{rct}_{t^\star} \leftarrow \mathsf{LOT.SendRead}(\cdots; r^{\mathsf{LOT}}_{t^\star,\mathsf{rct}}), \\ \text{in } \mathsf{wct}_{t^\star} \leftarrow \mathsf{LOT.SendWrite}(\cdots; r^{\mathsf{LOT}}_{t^\star,\mathsf{wct}}). \end{cases}$$

This is different from $Z_{b,t}$'s for $t \in \{\mathsf{t}\} \cup \mathfrak{s}$, which might contain simulated $\widetilde{\mathsf{rct}}_t$'s and $\widetilde{\mathsf{wct}}_t$'s. In G_2, the augmented step circuit $\mathsf{AugCPU}_{t^\star}$ is garbled with truly random $\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}, r^{\mathsf{GC}}_{t^\star}$. If $t^\star = 1$, then $\pi_1, \{L_{1,i,b}\}_{i,b}$ only additional appear in $\widehat{M}$ and the LGRAM labels given to the adversary as $(X_{0,1} \oplus \pi_1)$ and $\{L_{1,i,X_{0,1}[i]}\}_i$. If $t^\star - 1 \in \mathfrak{s}$, then they are only additionally used to simulate

$$\widetilde{\mathsf{AugCPU}}_{t^\star-1} \xleftarrow{\$} \mathsf{GC}.\widetilde{\mathsf{Garble}}(\ldots, Z_{0,t^\star-1}, \ldots),$$

where $Z_{0,t^\star-1}$ contains only $(X_{0,t^\star} \oplus \pi_{t^\star})$ and $\{L_{t^\star,i,X_{0,t^\star}[i]}\}_i$, potentially in the clear, in $\widetilde{\mathsf{rct}}_{t^\star-1}$, in $\widetilde{\mathsf{wct}}_{t^\star-1}$, and by imagination.[140] Also, $\mathsf{AugCPU}_t(X_{0,t^\star}) = Z_{0,t^\star}$. It follows from the security of GC that $\mathsf{G}_2 \approx \mathsf{G}_3$.

- G_4. In this hybrid, $Z_{0,t^\star}$ contains $\widetilde{\mathsf{rct}}_{t^\star}, \widetilde{\mathsf{wct}}_{t^\star}$ simulated using LOT.SimRead and LOT.SimWrite as needed, instead of normally generated ones, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \{t^\star\} \cup \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t},t^\star\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widetilde{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}, \quad \widetilde{\mathsf{AugCPU}}_{t^\star},$$

where

$$(\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}, r^{\mathsf{LOT}}_{t^\star}) \leftarrow \text{uniformly random},$$
$$\widetilde{\mathsf{AugCPU}}_{t^\star} \xleftarrow{\$} \mathsf{GC}.\widetilde{\mathsf{Garble}}(1^{|\mathsf{AugCPU}|}, Z_{0,t^\star}, X_{0,t^\star} \oplus \pi_{t^\star}, \{L_{t^\star,i,X_{0,t^\star}[i]}\}_i),$$
$$Z_{0,t^\star} : \boxed{X_{0,t^\star+1} \oplus \pi_{t^\star+1}, \{L_{t^\star+1,i,X_{0,t^\star+1}[i]}\}_i} \begin{cases} \text{in the clear,} \\ \text{in } \widetilde{\mathsf{rct}}_{t^\star} \xleftarrow{\$} \boxed{\mathsf{LOT.SimRead}(\cdots)}, \\ \text{in } \widetilde{\mathsf{wct}}_{t^\star} \xleftarrow{\$} \boxed{\mathsf{LOT.SimWrite}(\cdots)}. \end{cases}$$

The LOT ciphertexts $\mathsf{rct}_{t^\star}, \mathsf{wct}_{t^\star}$ are encrypted using truly random $r^{\mathsf{LOT}}_{t^\star}$ in G_3. Each database is known before its LOT hash key is provided to the adversary: for an input tape, hk_τ is provided after D_τ is chosen; for the working tape, $D_{\mathsf{work},t^\star-1}$ and $i_{t^\star}, \mathsf{wdata}_{t^\star}$ are known once the adversary commits to the challenge M, w_0, w_1,

[140] When $t^\star > T$, none of $\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}$ appears in $Z_{0,t^\star-1}$. We pretend $X_{0,t^\star} = X_{0,T}$ and that $(X_{0,t^\star} \oplus \pi_{t^\star})$ and $\{t^\star, i, X_{0,t^\star}[i]\}_i$ are revealed, imagining that the $(t^\star)^{\text{th}}$ step repeated the last existent step.

after which $\mathsf{hk}_{\mathsf{work}}$ is provided. Therefore, it follows from the read/write security of LOT that $\mathsf{G}_3 \approx \mathsf{G}_4$.

- G_5. In this hybrid, we hardwire $\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b}$ into GenAugCPU′, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \{t^\star\} \cup \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t},t^\star\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}, \quad \boxed{\pi_{t^\star}, \{L_{t^\star,i,b}\}_{i,b},} \widehat{\mathsf{AugCPU}}_{t^\star}.$$

  $\mathsf{G}_4 \approx \mathsf{G}_5$ follows from the security of $i\mathcal{O}$ for polynomial-size domains.

By inspection, G_5 is just $\mathsf{H}_{\mathsf{t},\{t^\star\}\cup\mathfrak{s}}$. Therefore, $\mathsf{H}_{\mathsf{t},\mathfrak{s}} \equiv \mathsf{G}_0 \approx \mathsf{G}_5 \equiv \mathsf{H}_{\mathsf{t},\{t^\star\}\cup\mathfrak{s}}$. □

Proof (Claim 9.9). Let $\mathsf{t}, \mathfrak{s}$ be such that $\mathsf{t} = 1$ or $\mathsf{t} - 1 \in \mathfrak{s}$. The hybrids are mostly identical to $\mathsf{H}_{\mathsf{t},\mathfrak{s}}$ except for the contents hardwired into GenAugCPU′.

- G_0. This is just $\mathsf{H}_{\mathsf{t},\mathfrak{s}}$, where the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\{\mathsf{t}\}\cup\mathfrak{s}}.$$

- G_1. In this hybrid, we no longer directly hardwire $\pi_t, \{L_{t,i,b}\}_{i,b}$ into GenAugCPU′ for $t = \mathsf{t}$, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{\boxed{t\in\mathfrak{s}}}, \quad \boxed{\pi_\mathsf{t}, \{L_{\mathsf{t},i,b}\}_{i,b},} \widehat{\mathsf{AugCPU}}_\mathsf{t},$$

  where

$$(\pi_\mathsf{t}, \{L_{\mathsf{t},i,b}\}_{i,b}) \xleftarrow{\$} \text{uniformly random};$$
$$\begin{cases} \widehat{M}, \text{LGRAM labels} \leftarrow X_{0,1} \oplus \pi_1, \{L_{1,i,X_{0,1}[i]}\}_i, \ldots, & \text{if } \mathsf{t} = 1; \\ \widehat{\mathsf{AugCPU}}_{\mathsf{t}-1} \xleftarrow{\$} \mathsf{GC}.\widetilde{\mathsf{Garble}}\left(1^{|\mathsf{AugCPU}|}, Z_{0,\mathsf{t}-1}, X_{0,\mathsf{t}-1} \oplus \pi_{\mathsf{t}-1}, \{L_{\mathsf{t}-1,i,X_{0,\mathsf{t}-1}[i]}\}_i\right), & \\ Z_{0,\mathsf{t}-1} : X_{0,\mathsf{t}} \oplus \pi_\mathsf{t}, \{L_{\mathsf{t},i,X_{0,\mathsf{t}}[i]}\}_i \text{ in the clear}, \widetilde{\mathsf{rct}}_{\mathsf{t}-1}, \widetilde{\mathsf{wct}}_{\mathsf{t}-1}, & \text{if } \mathsf{t}-1 \in \mathfrak{s}; \end{cases}$$

$$\widetilde{\mathsf{AugCPU}}_{\mathsf{t}} \xleftarrow{\$} \widetilde{\mathsf{GC.Garble}}\left(\begin{array}{l} 1^{|\mathsf{AugCPU}|}, Z_{1,\mathsf{t}}, \\ X_{0,\mathsf{t}} \oplus \pi_{\mathsf{t}}, \{L_{\mathsf{t},i,X_{0,\mathsf{t}}[i]}\}_i \end{array}\right),$$

$$Z_{1,\mathsf{t}} : X_{1,\mathsf{t}+1} \oplus \pi_{\mathsf{t}+1}, \{L_{\mathsf{t}+1,i,X_{1,\mathsf{t}+1}[i]}\}_i \text{ in the clear}, \widetilde{\mathsf{rct}}_{\mathsf{t}}, \widetilde{\mathsf{wct}}_{\mathsf{t}}, \quad \text{if } \mathsf{t} \le T_{\max}.$$

Note that GenAugCPU′ still indirectly contains $\pi_{\mathsf{t}}, \{L_{\mathsf{t},i,b}\}_{i,b}$ via $\widehat{M}$ and LGRAM labels, or $Z_{0,\mathsf{t}-1}$. Removal of their direct hardwiring does not alter the truth table of GenAugCPU′ as they are only used in Step 2 (Figure 9.5) for $t = \mathsf{t}$ and $t = \mathsf{t}-1$, a branch never taken for those values of t because they are handled by Step 1. By the security of $i\mathcal{O}$ for polynomial-size domains, $\mathsf{G}_0 \approx \mathsf{G}_1$.

- G_2. In this hybrid, we change the input of $\mathsf{AugCPU}_{\mathsf{t}}$ from $X_{0,\mathsf{t}}$ to $X_{1,\mathsf{t}}$ by modifying the permuted input and the labels selected for $\widetilde{\mathsf{AugCPU}}_{\mathsf{t}}$, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widetilde{\mathsf{AugCPU}}_t\}_{t\in\mathfrak{s}}, \quad \widetilde{\mathsf{AugCPU}}_{\mathsf{t}},$$

where

$$(\pi_{\mathsf{t}}, \{L_{\mathsf{t},i,b}\}_{i,b}) \xleftarrow{\$} \text{uniformly random};$$

$$\begin{cases} \widehat{M}, \text{LGRAM labels} \leftarrow \boxed{X_{1,1} \oplus \pi_1, \{L_{1,i,X_{1,1}[i]}\}_i}, \ldots, & \text{if } \mathsf{t} = 1; \\ \widetilde{\mathsf{AugCPU}}_{\mathsf{t}-1} \xleftarrow{\$} \widetilde{\mathsf{GC.Garble}}\left(\begin{array}{l} 1^{|\mathsf{AugCPU}|}, Z_{0,\mathsf{t}-1}, \\ X_{0,\mathsf{t}-1} \oplus \pi_{\mathsf{t}-1}, \{L_{\mathsf{t}-1,i,X_{0,\mathsf{t}-1}[i]}\}_i \end{array}\right), & \\ Z_{0,\mathsf{t}-1} : \boxed{X_{1,\mathsf{t}} \oplus \pi_1, \{L_{\mathsf{t},i,X_{1,\mathsf{t}}[i]}\}_i} \text{ in the clear}, \widetilde{\mathsf{rct}}_{\mathsf{t}-1}, \widetilde{\mathsf{wct}}_{\mathsf{t}-1}, & \text{if } \mathsf{t}-1 \in \mathfrak{s}; \end{cases}$$

$$\widetilde{\mathsf{AugCPU}}_{\mathsf{t}} \xleftarrow{\$} \widetilde{\mathsf{GC.Garble}}\left(\begin{array}{l} 1^{|\mathsf{AugCPU}|}, Z_{1,\mathsf{t}}, \\ \boxed{X_{1,1} \oplus \pi_1, \{L_{1,i,X_{1,1}[i]}\}_i} \end{array}\right),$$

$$Z_{1,\mathsf{t}} : X_{1,\mathsf{t}+1} \oplus \pi_{\mathsf{t}+1}, \{L_{\mathsf{t}+1,i,X_{1,\mathsf{t}+1}[i]}\}_i \text{ in the clear}, \widetilde{\mathsf{rct}}_{\mathsf{t}}, \widetilde{\mathsf{wct}}_{\mathsf{t}}, \quad \text{if } \mathsf{t} \le T_{\max}.$$

This change does not alter the distribution, i.e., $\mathsf{G}_1 \equiv \mathsf{G}_2$, because π_{t} is a one-time pad that hides $X_{0,\mathsf{t}}$ or $X_{1,\mathsf{t}}$ and either set of $\{L_{\mathsf{t},i,b}\}_{i,b}$ chosen by $X_{0,\mathsf{t}}$ and $X_{1,\mathsf{t}}$ is simply random.

- G_3. In this hybrid, we undo the simulation of $\widetilde{\mathsf{rct}}_\mathsf{t}$ and $\widetilde{\mathsf{wct}}_\mathsf{t}$, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\mathfrak{s}}, \quad \widehat{\mathsf{AugCPU}}_\mathsf{t},$$

where

$$(\pi_\mathsf{t}, \{L_{\mathsf{t},i,b}\}_{i,b} \boxed{, r_\mathsf{t}^{\mathsf{LOT}}}) \xleftarrow{\$} \text{uniformly random};$$

$$\begin{cases} \widehat{M}, \text{LGRAM labels} \leftarrow X_{1,1}\oplus\pi_1, \{L_{1,i,X_{1,1}[i]}\}_i, \ldots, & \text{if } \mathsf{t}=1; \\ \widehat{\mathsf{AugCPU}}_{\mathsf{t}-1} \xleftarrow{\$} \mathsf{GC}.\widetilde{\mathsf{Garble}}\left(\begin{matrix} 1^{|\mathsf{AugCPU}|}, Z_{0,\mathsf{t}-1}, \\ X_{0,\mathsf{t}-1}\oplus\pi_{\mathsf{t}-1}, \{L_{\mathsf{t}-1,i,X_{0,\mathsf{t}-1}[i]}\}_i \end{matrix}\right), & \\ Z_{0,\mathsf{t}-1} : X_{1,\mathsf{t}}\oplus\pi_1, \{L_{\mathsf{t},i,X_{1,\mathsf{t}}[i]}\}_i \text{ in the clear}, \widetilde{\mathsf{rct}}_{\mathsf{t}-1}, \widetilde{\mathsf{wct}}_{\mathsf{t}-1}, & \text{if } \mathsf{t}-1\in\mathfrak{s}; \\ \widehat{\mathsf{AugCPU}}_\mathsf{t} \xleftarrow{\$} \mathsf{GC}.\widetilde{\mathsf{Garble}}\left(\begin{matrix} 1^{|\mathsf{AugCPU}|}, Z_{1,\mathsf{t}}, \\ X_{1,1}\oplus\pi_1, \{L_{1,i,X_{1,1}[i]}\}_i \end{matrix}\right), & \\ Z_{1,\mathsf{t}} : \boxed{\pi_{\mathsf{t}+1}, \{L_{\mathsf{t}+1,i,b}\}_{i,b}} \text{ in the clear}, \boxed{\mathsf{rct}_\mathsf{t}, \mathsf{wct}_\mathsf{t}}, & \text{if } \mathsf{t}\le T_{\max}. \end{cases}$$

$\mathsf{G}_2 \approx \mathsf{G}_3$ by the read/write security of LOT.

- G_4. In this hybrid, we undo the simulation of $\widehat{\mathsf{AugCPU}}_\mathsf{t}$, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad \mathsf{t}, \quad \mathfrak{s}, \quad \mathring{k}_{\{\mathsf{t}\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\mathfrak{s}}, \quad \widehat{\mathsf{AugCPU}}_\mathsf{t},$$

where

$$(\pi_\mathsf{t}, \{L_{\mathsf{t},i,b}\}_{i,b}, r_\mathsf{t}^{\mathsf{LOT}} \boxed{, r_\mathsf{t}^{\mathsf{GC}}}) \xleftarrow{\$} \text{uniformly random};$$

$$\begin{cases} \widehat{M}, \text{LGRAM labels} \leftarrow X_{1,1}\oplus\pi_1, \{L_{1,i,X_{1,1}[i]}\}_i, \ldots, & \text{if } \mathsf{t}=1; \\ \widehat{\mathsf{AugCPU}}_{\mathsf{t}-1} \xleftarrow{\$} \mathsf{GC}.\widetilde{\mathsf{Garble}}\left(\begin{matrix} 1^{|\mathsf{AugCPU}|}, Z_{0,\mathsf{t}-1}, \\ X_{0,\mathsf{t}-1}\oplus\pi_{\mathsf{t}-1}, \{L_{\mathsf{t}-1,i,X_{0,\mathsf{t}-1}[i]}\}_i \end{matrix}\right), & \\ Z_{0,\mathsf{t}-1} : X_{1,\mathsf{t}}\oplus\pi_1, \{L_{\mathsf{t},i,X_{1,\mathsf{t}}[i]}\}_i \text{ in the clear}, \widetilde{\mathsf{rct}}_{\mathsf{t}-1}, \widetilde{\mathsf{wct}}_{\mathsf{t}-1}, & \text{if } \mathsf{t}-1\in\mathfrak{s}; \end{cases}$$

$$\mathsf{AugCPU}_t \leftarrow \mathsf{AugCPU}\left[\begin{matrix}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}\end{matrix}\right],$$

$$\widehat{\mathsf{AugCPU}}_t \leftarrow \boxed{\mathsf{GC.Garble}(\mathsf{AugCPU}_t, \pi_t, \{L_{t,i,b}\}_{i,b}; r_t^{\mathsf{GC}})}, \qquad \text{if } t \le T_{\max}.$$

Since $\mathsf{AugCPU}_t(X_{1,t}) = Z_{1,t}$, it follows from the security of GC that $\mathsf{G}_3 \approx \mathsf{G}_4$.

- G_5. In this hybrid, we change the randomness for step t to the PPRF evaluation, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad t, \quad \mathfrak{s}, \quad \mathring{k}_{\{t\}\cup\mathfrak{s}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{t\in\mathfrak{s}}, \quad \widehat{\mathsf{AugCPU}}_t,$$

where

$$(\pi_t, \{L_{t,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}, r_t^{\mathsf{GC}}) \xleftarrow{\$} \boxed{\mathsf{PPRF.Eval}(k, t)};$$

$$\begin{cases} \widehat{M}, \text{ LGRAM labels} \leftarrow X_{1,1} \oplus \pi_1, \{L_{1,i,X_{1,1}[i]}\}_i, \ldots, & \text{if } t = 1;\\ \widetilde{\mathsf{AugCPU}}_{t-1} \xleftarrow{\$} \widetilde{\mathsf{GC.Garble}}\left(\begin{matrix}1^{|\mathsf{AugCPU}|}, Z_{0,t-1},\\ X_{0,t-1} \oplus \pi_{t-1}, \{L_{t-1,i,X_{0,t-1}[i]}\}_i\end{matrix}\right), & \\ \quad Z_{0,t-1} : X_{1,t} \oplus \pi_1, \{L_{t,i,X_{1,t}[i]}\}_i \text{ in the clear}, \widetilde{\mathsf{rct}}_{t-1}, \widetilde{\mathsf{wct}}_{t-1}, & \text{if } t-1 \in \mathfrak{s};\\ \mathsf{AugCPU}_t \leftarrow \mathsf{AugCPU}\left[\begin{matrix}\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}},\\ \pi_{t+1}, \{L_{t+1,i,b}\}_{i,b}, r_t^{\mathsf{LOT}}\end{matrix}\right], & \\ \widehat{\mathsf{AugCPU}}_t \leftarrow \mathsf{GC.Garble}(\mathsf{AugCPU}_t, \pi_t, \{L_{t,i,b}\}_{i,b}; r_t^{\mathsf{GC}}), & \text{if } t \le T_{\max}. \end{cases}$$

$\mathsf{G}_4 \approx \mathsf{G}_5$ by the security of PPRF.

- G_6. In this hybrid, we no longer puncture the PPRF key at t and no longer hardwire the randomness nor the garbled step for t, i.e., the hardwired information is

$$\{\mathsf{hk}_\tau, h_\tau, |D_\tau|\}_{\tau\in[\mathcal{T}]}, \mathsf{hk}_{\mathsf{work}}, \quad t-1, \quad \mathfrak{s}\setminus\{t-1\}, \quad \boxed{\mathring{k}_{\{t-1\}\cup(\mathfrak{s}\setminus\{t-1\})}},$$
$$\{\pi_t, \{L_{t,i,b}\}_{i,b}, \widehat{\mathsf{AugCPU}}_t\}_{\boxed{t\in\{t-1\}\cup(\mathfrak{s}\setminus\{t-1\})}},$$

and $\widehat{M}$ and the LGRAM labels given to the adversary contain $(X_{1,1} \oplus \pi_1)$ and $\{L_{1,i,X_{1,1}[i]}\}_i$ if $t = 1$. By the security of $i\mathcal{O}$ for polynomial-size domains, $\mathsf{G}_5 \approx \mathsf{G}_6$.

By inspection, G_6 is just $\mathsf{H}_{t-1,\mathfrak{s}\setminus\{t-1\}}$. Therefore, $\mathsf{H}_{t,\mathfrak{s}} \equiv \mathsf{G}_0 \approx \mathsf{G}_6 \equiv \mathsf{H}_{t-1,\mathfrak{s}\setminus\{t-1\}}$. □

9.5 From Bounded LGRAM to Unbounded LGRAM

In this section, we employ the powers-of-two transformation [AL18] to construct an unbounded LGRAM from a bounded one.[141]

Construction 9.1 is a bounded LGRAM. Recall that a bounded scheme has the following three deficiencies: *i)* its evaluation time scales with the time bound $T_{\max}$ instead of the instance running time T; *ii)* it only works if the maximum address accessed by the machine is within $T_{\max}$; and *iii)* its security only holds if $T_{\max}$ is polynomially bounded. To get rid of these restrictions, we apply the transformation due to [AL18]. The idea is to generate a series of LGRAM garblings with $T'_{\max,e} = 2^e$ for $e \in [\log T_{\max}]$, each encrypted under a different key. The e^{th} garbling simulates the execution, but if the time exceeds 2^e, it outputs the secret key that decrypts the next garbling, and the evaluation can be retried. This avoids the exponential security loss because we can apply the LGRAM security to the garblings with $T'_{\max} < 2T$ and argue that any larger garblings are hidden by encryption, removing the need to apply the LGRAM security on those large garblings.

Ingredients of Construction 9.2. Let

- $\mathsf{LGRAM}' = (\mathsf{Compress}', \mathsf{Garble}', \mathsf{Eval}')$ be a bounded LGRAM (Definition 9.4), and
- $\mathsf{SKE} = (\mathsf{SKE.Gen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ an SKE scheme (Definition 9.15).

Construction 9.2 (unbounded LGRAM). Our scheme works as follows.

- $\mathsf{Compress}(1^{\ell_{\text{addr}}}, 1^{\ell_{\text{cell}}}, \tau, D_\tau)$ is the same as $\mathsf{Compress}'$.
- $\mathsf{Garble}(T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]})$ prepares M' from M and $T_{\max}$ as described in Figure 9.7. For $e \in [\lceil \log_2 T_{\max} \rceil]$, it samples SKE key k_e and runs

$$(\widehat{M}'_e, \{L'_{e,i,b}\}_{i,b}) \xleftarrow{\$} \mathsf{Garble}'(T'_{\max,e}, M', \{\mathsf{digest}_\tau\}_{\tau\in[\mathcal{T}]}) \qquad \text{for } e \in [\lceil \log_2 T_{\max} \rceil],$$

[141] See the full version of the original paper [JLL22] for other transformations for fixed-address or full security.

Machine M' Transformed from M and $T_{\max}$

Lengths. $\ell'_{\text{in}} = \ell_{\text{in}} + \text{poly}(\lambda)$, $\ell'_{\text{st}} = \ell_{\text{st}} + \text{O}(\log T_{\max}) + \text{poly}(\lambda, |M|)$,
$\ell'_{\text{addr}} = \ell_{\text{addr}}$, $\ell'_{\text{ADDR}} = \text{poly}(\lambda, |M|, \log T_{\max})$,
$\ell'_{\text{cell}} = \ell_{\text{cell}}$, $\ell'_{\text{CELL}} = \text{poly}(\lambda, |M|, \log T_{\max})$.

Input w'. W, short input to M;
E, attempted time limit of M;
K, secret key of SKE.

State st'. $t - 1$, current time period minus one (initially 0);
$t_0 - 1$, current time step minus one (initially 0) modulo the period;
st, state of M.

Steps. Each step of M is simulated by one period of M'.

1. When $t \le 2^E$:

   1 step for one step of M

   $\text{poly}(\lambda, |M|, \log T_{\max})$ steps for the memory operation of M (using a deterministic balanced binary tree of capacity 2^E to implement a virtual sparse memory for M)

   halt if M halts

2. When $2^E < t < 2^E + \text{poly}(\lambda)$:

   report running time exceeded

   output K

Figure 9.7. The machine M' in Construction 9.2.

where $T'_{\max,e} = 2^e + \mathrm{poly}(\lambda, |M|, \log T_{\max})$. The algorithm encrypts $\widehat{M}'_e$ and $L'_{e,i,b}$ by

$$\widetilde{M}'_e \xleftarrow{\$} \mathsf{SKE.Enc}(k_e, \widehat{M}_e), \qquad \widetilde{L}'_{e,i,b} \xleftarrow{\$} \mathsf{SKE.Enc}(k_e, L'_{e,i,b}),$$

and splits the encrypted labels into $\{\widetilde{L}^W_{e,i,b}\}_{e,i,b}, \{\widetilde{L}^E_{e,i,b}\}_{e,i,b}, \{\widetilde{L}^K_{e,i,b}\}_{e,i,b}$ by the portion of input to M' they correspond to. Defining $k_{\lceil \log_2 T_{\max} \rceil + 1} = \bot$, it sets and outputs

$$\widehat{M} = \left(k_1, \left\{\widetilde{M}'_e, \widetilde{L}^E_{e,i,b}\text{'s and } \widetilde{L}^K_{e,i,b}\text{'s for } E = e, K = k_{e+1}\right\}_{e \in [\lceil \log_2 T_{\max} \rceil]}\right),$$
$$L_{i,b} = \widetilde{L}^W_{1,i,b} \| \cdots \| \widetilde{L}^W_{\lceil \log_2 T_{\max} \rceil, i, b}.$$

- $\mathsf{Eval}^{D_1,\ldots,D_\mathcal{T}}(T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau \in [\mathcal{T}]}, \widehat{M}, \{L_i\}_i)$ parses $\widehat{M}$ and L_i's as defined by Garble. For $e = 1, \ldots, \lceil \log_2 T_{\max} \rceil$, it retrieves k_e, either from $\widehat{M}$ if $e = 1$, or from the previous call to Eval′ if $e > 1$, and runs

$$(\mathsf{Eval}')^{D_1,\ldots,D_\mathcal{T}}\left(T_{\max}, M, \{\mathsf{digest}_\tau\}_{\tau \in [\mathcal{T}]}, \mathsf{SKE.Dec}(k_e, \widetilde{M}'_e), \{\mathsf{SKE.Dec}(k_e, \widetilde{L}_{e,i})\}_i\right)$$
$$\to \begin{cases} \mathsf{outS}, & \text{if running time is not exceeded;} \\ (\text{prefix of } \mathsf{outS}, k_{e+1}), & \text{if running time is exceeded.} \end{cases}$$

  The algorithm removes the empty, placeholder output from outS, use the stripped version as the output, and halt, if the running time is not exceeded. Otherwise, it attempts the next e.

Correctness and Efficiency. Those are clear by inspection.

Theorem 9.11 (¶). *Suppose in Construction 9.2,* LGRAM′ *is a (fixed-memory, fixed-address, fully)* secure bounded LGRAM *(Definition 9.4) and* SKE *is* secure *(Definition 9.15), then the constructed scheme is an* unbounded LGRAM *(Definition 9.3) with the same level of security as* LGRAM′.

Proof (Theorem 9.11). Since M' uses a deterministic balanced binary tree to implement the virtual sparse memory for M, it preserves the constraints of fixed-memory, fixed-address, or full security. Let T be the instance running time and $\bar{e} = \lceil \log_2 T \rceil$. We consider the following hybrids.

- H^{β}_{-1}. This is just $\mathsf{Exp}^{\beta}_{\mathrm{LGRAM}}$ for the constructed scheme, where

$$\widehat{M} : k_1,\ \left\{\widetilde{M}'_e,\ \widetilde{L}^{E}_{e,i,b}\text{'s and } \widetilde{L}^{K}_{e,i,b}\text{'s for } E = e, K = k_{e+1}\right\}_{e\in[\lceil \log_2 T_{\max}\rceil]},$$
$$L_i : \quad \left\{\widetilde{L}^{W}_{e,i,b}\text{'s for } W = w_\beta\right\}_{e\in[\lceil \log_2 T_{\max}\rceil]}.$$

- $\mathsf{H}^{\beta}_{\mathfrak{e}}$ for $\mathfrak{e} = 0, \ldots, \lceil \log_2 T_{\max}\rceil$. In this hybrid, we remove $k_{\bar{e}+1}$ as well as all the unused garblings up to time bound $2^{\mathfrak{e}}$ as follows.

$$\widehat{M} : k_1, \begin{cases} \widetilde{M}'_e,\ \widetilde{L}^{E}_{e,i,b}\text{'s and } \widetilde{L}^{K}_{e,i,b}\text{'s for } E = e, K = k_{e+1}, & \text{if } e < \bar{e};\\ \widetilde{M}'_{\bar{e}},\ \widetilde{L}^{E}_{\bar{e},i,b}\text{'s and } \widetilde{L}^{K}_{\bar{e},i,b}\text{'s for } E = \bar{e}, K = \boxed{\bot}, & \text{if } e = \bar{e};\\ \boxed{\text{random}}, & \text{if } e > \bar{e} \text{ and } e \le \mathfrak{e};\\ \widetilde{M}'_e,\ \widetilde{L}^{E}_{e,i,b}\text{'s and } \widetilde{L}^{K}_{e,i,b}\text{'s for } E = e, K = k_{e+1}, & \text{if } e > \bar{e} \text{ and } e > \mathfrak{e}; \end{cases}$$

$$L_i : \begin{cases} \widetilde{L}^{W}_{e,i,b}\text{'s for } W = w_\beta, & \text{if } e \le \bar{e};\\ \boxed{\text{random}}, & \text{if } e > \bar{e} \text{ and } e \le \mathfrak{e};\\ \widetilde{L}^{W}_{e,i,b}\text{'s for } W = w_\beta, & \text{if } e > \bar{e} \text{ and } e > \mathfrak{e}. \end{cases}$$

To see indistinguishability...

- $\mathsf{H}^{\beta}_{-1} \approx \mathsf{H}^{\beta}_{0}$. Compared to H^{β}_{-1}, in H^{β}_{0}, the garbling $\widehat{M}'_{\bar{e}}$ has K in its input changed from $k_{\bar{e}+1}$ to $\bot$. Since $2^{\bar{e}} \ge T$, the execution of M in that garbling does not output K and this change satisfies the constraint of LGRAM security. Moreover, $T'_{\max,\bar{e}} = 2^{\bar{e}} \le 2T$ is polynomially bounded. Therefore, $\mathsf{H}^{\beta}_{-1} \approx \mathsf{H}^{\beta}_{0}$ by the bounded security of LGRAM′.

- $\mathsf{H}^{\beta}_{\mathfrak{e}-1} \approx \mathsf{H}^{\beta}_{\mathfrak{e}}$. The two hybrids are different only when $\mathfrak{e} > \bar{e}$, in which case the ciphertexts under $k_{\mathfrak{e}}$ changes into random strings and $k_{\mathfrak{e}}$ is not used in the two hybrids. Therefore, $\mathsf{H}^{\beta}_{\mathfrak{e}-1} \approx \mathsf{H}^{\beta}_{\mathfrak{e}}$ by the ciphertext pseudorandomness of SKE.

- $\mathsf{H}^{0}_{\lceil \log_2 T_{\max}\rceil} \approx \mathsf{H}^{1}_{\lceil \log_2 T_{\max}\rceil}$. The difference is that the $\bar{e}$ non-erased garblings have W in their inputs changed between w_0 and w_1. By how M' works, this change satisfies the constraint of LGRAM security. Moreover, all of them have $T'_{\max,e} \le 2T$ polynomially bounded. Therefore, $\mathsf{H}^{0}_{\lceil \log_2 T_{\max}\rceil} \approx \mathsf{H}^{1}_{\lceil \log_2 T_{\max}\rceil}$ follows from a standard hybrid argument by the bounded security of LGRAM′.

We conclude that $\mathsf{Exp}^{0}_{\mathrm{LGRAM}} \equiv \mathsf{H}^{0}_{-1} \approx \mathsf{H}^{1}_{-1} \equiv \mathsf{Exp}^{1}_{\mathrm{LGRAM}}$. □

9.6 ABE for RAM

In this section, we build an ABE for RAM that is f- and x-succinct and has linear-time KeyGen and Enc and whose Dec runs in time $O(T + |f| + |x|)$, ignoring polynomial factors in the security parameter. (See Definitions 2.1, 2.2, and 9.7.)

Recall that in ABE for RAM, the functionality (resp. key, ciphertext) is associated with some RAM M and (up to exponential) time bound $T_{\max}$ (resp. f of arbitrary length, x of arbitrary length and μ of λ bits) and decryption recovers μ if $M^{f,x}()$ accepts in time at most $T_{\max}$.

Ingredients of Construction 9.3. Let

- $\mathsf{ckt} = (\mathsf{ckt.Setup}, \mathsf{ckt.KeyGen}, \mathsf{ckt.Enc}, \mathsf{ckt.Dec})$ be an FE scheme for circuits (Definition 9.6),
- $\mathsf{LGRAM} = (\mathsf{LGRAM.Compress}, \mathsf{LGRAM.Garble}, \mathsf{LGRAM.Eval})$ a 2-tape LGRAM scheme (Definition 9.2; fixed-memory security suffices),
- $\mathsf{PPRF} = (\mathsf{PPRF.Puncture}, \mathsf{PPRF.Eval})$ a puncturable PRF (Definition 9.14), and
- $\mathsf{SKE} = (\mathsf{SKE.Gen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ an SKE scheme (Definition 9.15).

Construction 9.3 (ABE for RAM). It works as follows.

- $\mathsf{Setup}(M', T'_{\max})$ defines $T_{\max} = T'_{\max} + \lambda$. It writes down a machine $M^{f,x}(\mu)$ that simulates $(M')^{f,x}()$ step-for-step, for at most $T'_{\max}$ steps, and if the simulation reaches acceptance, uses another λ steps to output the λ bits of μ (otherwise, M halts without further output).[142] The algorithm runs

$$(\mathsf{ckt.mpk}, \mathsf{ckt.msk}) \xleftarrow{\$} \mathsf{ckt.Setup}(1^{\cdots}, 1^{\cdots}),$$

  and outputs $(\mathsf{mpk}, \mathsf{msk}) = \big((T_{\max}, M, \mathsf{ckt.mpk}), \mathsf{ckt.msk}\big)$, where the input/circuit sizes given to ckt.Setup are appropriately chosen (see below).

[142] Note that the memory of M is fixed for fixed f, x. This allows us to rely on fixed-memory security of LGRAM in the security proof.

$$\mathsf{GenLGRAM}\left[\begin{matrix}\mathsf{digest}_f, s_{\mathsf{PPRF}},\\ \mathsf{idx}'_{\mathsf{sk}}, \mathsf{lgram}'_{\mathsf{sk}}\end{matrix}\right]\left(\begin{matrix}\mathsf{digest}_x, \mu, k_{\mathsf{PPRF}}, \mathsf{mode},\\ \mu', \mathsf{idx}_{\mathsf{ct}}, \mathsf{idx}_{\mathsf{H}}, k'_{\mathsf{idx}}, k'_{\mathsf{lgram}}, \mathsf{lgram}_{\mathsf{ct}}\end{matrix}\right)$$

Hardwired.
- digest_f, LGRAM digest of f (tape 1);
- s_{PPRF}, PPRF input for LGRAM randomness;
- $\mathsf{idx}'_{\mathsf{sk}}$, encrypted ordinal ($q^{\text{th}}$ query) of this key;
- $\mathsf{lgram}'_{\mathsf{sk}}$, encrypted hardwired LGRAM garbling.

Input.
- digest_x, μ, LGRAM digest of x (tape 2), message;
- k_{PPRF}, PPRF key for LGRAM randomness;
- mode, mode of ciphertext and security proof;
- μ', alternative message;
- $\mathsf{idx}_{\mathsf{ct}}$, ordinal of the challenge ciphertext (number of pre-challenge keys);
- $\mathsf{idx}_{\mathsf{H}}$, progress of security proof (number of keys using μ');
- $k'_{\mathsf{idx}}, k'_{\mathsf{lgram}}$, SKE keys to decrypt $\mathsf{idx}'_{\mathsf{sk}}, \mathsf{lgram}'_{\mathsf{sk}}$;
- $\mathsf{lgram}_{\mathsf{ct}}$, hardwired LGRAM garbling.

Output. lgram, LGRAM garbling, computed as follows.

1. If $\mathsf{mode} = \mathsf{normal}$ (garbling is for μ):

$$(\widehat{M}, \{L_{i,b}\}_{i,b}) \leftarrow \mathsf{LGRAM.Garble}\left(\begin{matrix}T_{\max}, M, \mathsf{digest}_f, \mathsf{digest}_x;\\ \mathsf{PPRF.Eval}(k_{\mathsf{PPRF}}, s_{\mathsf{PPRF}})\end{matrix}\right)$$

output $\mathsf{lgram} = (\widehat{M}, \{L_{i,\mu[i]}\}_i)$

2. Otherwise, this key is decrypting the challenge ciphertext:

$\mathsf{idx}_{\mathsf{sk}} \leftarrow \mathsf{SKE.Dec}(k'_{\mathsf{idx}}, \mathsf{idx}'_{\mathsf{sk}})$

3a. If $\mathsf{mode} = \mathsf{hybrid}$ or $\mathsf{idx}_{\mathsf{sk}} \neq \mathsf{idx}_{\mathsf{H}}$ (garbling is for μ or μ'):

$$(\widehat{M}, \{L_{i,b}\}_{i,b}) \leftarrow \mathsf{LGRAM.Garble}\left(\begin{matrix}T_{\max}, M, \mathsf{digest}_f, \mathsf{digest}_x;\\ \mathsf{PPRF.Eval}(k_{\mathsf{PPRF}}, s_{\mathsf{PPRF}})\end{matrix}\right)$$

$$\textbf{output } \mathsf{lgram} = \begin{cases}(\widehat{M}, \{L_{i,\mu[i]}\}_i), & \text{if } \mathsf{idx}_{\mathsf{H}} < \mathsf{idx}_{\mathsf{sk}};\\ (\widehat{M}, \{L_{i,\boxed{\mu'}[i]}\}_i), & \text{if } \mathsf{idx}_{\mathsf{H}} \geq \mathsf{idx}_{\mathsf{sk}}.\end{cases}$$

3b. If $\mathsf{mode} = \mathsf{hardwire}$ and $\mathsf{idx}_{\mathsf{sk}} = \mathsf{idx}_{\mathsf{H}}$ (garbling is hardwired):

$$\textbf{output } \mathsf{lgram} = \begin{cases}\mathsf{lgram}_{\mathsf{ct}}, & \text{if } \mathsf{idx}_{\mathsf{ct}} \geq \mathsf{idx}_{\mathsf{sk}};\\ \mathsf{SKE.Dec}(k'_{\mathsf{lgram}}, \mathsf{lgram}'_{\mathsf{sk}}), & \text{if } \mathsf{idx}_{\mathsf{ct}} < \mathsf{idx}_{\mathsf{sk}}.\end{cases}$$

Figure 9.8. The circuit GenLGRAM in Construction 9.3.

- $\mathsf{KeyGen}(\mathsf{msk}, f)$ samples random strings $s_{\mathsf{PPRF}}, \mathsf{idx}'_{\mathsf{sk}}, \mathsf{lgram}'_{\mathsf{sk}}$, runs

$$\mathsf{digest}_f \xleftarrow{\$} \mathsf{LGRAM.Compress}(1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, 1, f),$$
$$\mathsf{ckt.sk} \xleftarrow{\$} \mathsf{ckt.KeyGen}(\mathsf{ckt.msk}, \mathsf{GenLGRAM}[\mathsf{digest}_f, s_{\mathsf{PPRF}}, \mathsf{idx}'_{\mathsf{sk}}, \mathsf{lgram}'_{\mathsf{sk}}]),$$

  and outputs $\mathsf{sk} = (\mathsf{digest}_f, s_{\mathsf{PPRF}}, \mathsf{idx}'_{\mathsf{sk}}, \mathsf{lgram}'_{\mathsf{sk}}, \mathsf{ckt.sk})$, where GenLGRAM is defined in Figure 9.8.

- $\mathsf{Enc}(\mathsf{mpk}, x, \mu)$ samples random string k_{PPRF}, runs

$$\mathsf{digest}_x \xleftarrow{\$} \mathsf{LGRAM.Compress}(1^{\ell_{\mathsf{cell}}}, 1^{\ell_{\mathsf{addr}}}, 2, x),$$
$$\mathsf{ckt.ct} \xleftarrow{\$} \mathsf{ckt.Enc}\big(\mathsf{ckt.mpk}, \bot, (\mathsf{digest}_x, \mu, k_{\mathsf{PPRF}}, \mathsf{normal}, \overbrace{\bot, \bot, \bot, \bot, \bot, \bot}^{\mu', \mathsf{idx}_{\mathsf{ct}}, \mathsf{idx}_{\mathsf{H}}, k'_{\mathsf{idx}}, k'_{\mathsf{lgram}}, \mathsf{lgram}_{\mathsf{ct}}})\big),$$

  and outputs $\mathsf{ct} = (\mathsf{digest}_x, \mathsf{ckt.ct})$.

- $\mathsf{Dec}^{\mathsf{mpk}, f, \mathsf{sk}, x, \mathsf{ct}}()$ checks whether $(M')^{f,x}()$ accepts. If not, it outputs $\bot$ and terminates. Otherwise, the algorithm parses $\mathsf{mpk}, \mathsf{sk}, \mathsf{ct}$ as defined earlier, runs

$$(\widehat{M}, \{L_i\}_i) \xleftarrow{\$} \mathsf{ckt.Dec}^{\mathsf{ckt.mpk}, \mathsf{GenLGRAM}[\mathsf{digest}_f, s_{\mathsf{PPRF}}, \mathsf{idx}'_{\mathsf{sk}}, \mathsf{lgram}'_{\mathsf{sk}}], \mathsf{ckt.sk}, \bot, \mathsf{ckt.ct}}(),$$

  then $\mathsf{LGRAM.Eval}^{f,x}(T_{\max}, M, \mathsf{digest}_f, \mathsf{digest}_x, \widehat{M}, \{L_i\}_i)$, from which it obtains and outputs μ.

Correctness and Efficiency. To ensure correctness, it suffices to set the input/circuit sizes of the underlying FE for circuits to be $\mathrm{poly}(\lambda, M, \log T_{\max})$ — the normal branch of GenLGRAM is just LGRAM.Eval, which runs in time $\mathrm{poly}(\lambda, M, \log T_{\max})$. This (order of) value is also sufficient for the security proof to go through. By the efficiency guarantees of its ingredients, Construction 9.3 is f- and x-succinct, has linear-time KeyGen and Enc, and its Dec runs in time $(T + |f| + |x|)\,\mathrm{poly}(\lambda, M, T_{\max})$.

Theorem 9.12 (¶). *Suppose in Construction 9.3,* ckt, PPRF *are* secure *(Definitions 2.2 and 9.14),* LGRAM *is* fixed-memory secure *(Definition 9.3), and* SKE *has* pseudorandom ciphertexts *(Definition 9.15), then the constructed ABE scheme is* secure *(Definition 2.2).*

We prove security by hybridizing over the keys. We denote by

$$\mathsf{digest}_{f_q}, s_{\mathsf{PPRF},q}, \mathsf{idx}'_{\mathsf{sk},q}, \mathsf{lgram}'_{\mathsf{sk},q}$$

the content hardwired into GenLGRAM for the q^{th} secret key sk_q. In the hybrids,

$$\mu', \mathsf{idx}_{\mathsf{ct}}, \mathsf{idx}_{\mathsf{H}}, \mathsf{lgram}_{\mathsf{ct}}, k'_{\mathsf{idx}}, k'_{\mathsf{lgram}}$$

in the challenge ciphertext ct are used, and its decryption behavior is controlled by mode and those values. The strategies of handling pre- and post-challenge keys are different. At a high level, they work as follows.

- $\mathsf{idx}'_{\mathsf{sk},q}$ is an encryption of q so that each key "knows" its ordinal number.
- $\mathsf{idx}_{\mathsf{ct}}$ is the ordinal number of the challenge ciphertext.
- $\mathsf{idx}_{\mathsf{H}}$ indicates the progress of the proof and increases from 0 to Q:
  - initially, $q > \mathsf{idx}_{\mathsf{H}}$, decrypting ct by sk_q yields an LGRAM garbling for μ_0;
  - when transitioning, $q = \mathsf{idx}_{\mathsf{H}}$, the behavior depends on mode and the relative timing of ct and sk_q,
    - when mode = hardwire, the decryption takes the hardwired LGRAM garbling from either $\mathsf{lgram}'_{\mathsf{sk},q}$ (in sk_q) or $\mathsf{lgram}_{\mathsf{ct}}$ (in ct), whichever is generated later the security game (decided by comparing $\mathsf{idx}_{\mathsf{ct}}$ and q);
    - when mode = hybrid, the behavior is the same as when $q < \mathsf{idx}_{\mathsf{H}}$;
  - eventually, $q < \mathsf{idx}_{\mathsf{H}}$, decrypting ct by sk_q yields an LGRAM garbling for μ_1.

Proof (Theorem 9.12). Let Q_1, Q_2 be the number of secret key queries in **Query I** and **Query II**, respectively. Writing k_{idx} for a random key of SKE and k for a random (non-punctured) key of PPRF, we list our hybrids.

- H_0. This is just $\mathsf{Exp}^0_{\mathsf{PHFE}}$, described as

$$\begin{array}{lccc}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{random}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}, & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \text{normal}, \\
& \mu' \leftarrow \bot, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \bot, & \mathsf{idx}_{\mathsf{H}} \leftarrow \bot, \\
& k'_{\mathsf{idx}} \leftarrow \bot, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

- H_1. In this hybrid, we sample $s_{\mathsf{PPRF},q}$'s without replacement, i.e.,

$$\mathsf{sk}_q: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \boxed{\text{distinct}}, \quad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}, \quad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};$$

$$\begin{array}{lccc}
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \mathsf{normal}, \\
& \mu' \leftarrow \bot, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \bot, & \mathsf{idx}_{\mathsf{H}} \leftarrow \bot, \\
& k'_{\mathsf{idx}} \leftarrow \bot, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

H_0 and H_1 are statistically indistinguishable.

- H_2. In this hybrid, we change $\mathsf{idx}'_{\mathsf{sk},q}$ into an encryption of q (padded to some fixed $\mathrm{poly}(\lambda)$ bits), i.e.,

$$\begin{array}{lccc}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \boxed{\mathsf{SKE.Enc}(k_{\mathsf{idx}}, q)}, & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \mathsf{normal}, \\
& \mu' \leftarrow \bot, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \bot, & \mathsf{idx}_{\mathsf{H}} \leftarrow \bot, \\
& k'_{\mathsf{idx}} \leftarrow \bot, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

  $\mathsf{H}_1 \approx \mathsf{H}_2$ follows from the ciphertext pseudorandomness of SKE, as the key k_{idx} is not used anywhere else.

- H_3. In this hybrid, we prepare for hybridizing over the keys. H_3 is described as

$$\begin{array}{lccc}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \boxed{\mathsf{hybrid}}, \\
& \mu' \leftarrow \boxed{\mu_1}, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \boxed{Q_1}, & \mathsf{idx}_{\mathsf{H}} \leftarrow \boxed{0}, \\
& k'_{\mathsf{idx}} \leftarrow \boxed{k_{\mathsf{idx}}}, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

  It is readily verified that the decryption outcome (as perceived by ckt) does not change. Therefore, $\mathsf{H}_2 \approx \mathsf{H}_3$ follows from the security of ckt.

- $\mathsf{H}_{4,\mathfrak{q}}$ for $q = 0, \ldots, Q_1 + Q_2$. In this hybrid, $\mathsf{idx}_{\mathsf{H}}$ is incremented to q, i.e.,

$$\begin{array}{lccc}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \mathsf{hybrid}, \\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, & \mathsf{idx}_{\mathsf{H}} \leftarrow \boxed{q}, \\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

  $\mathsf{H}_{4,0}$ is just H_3. The proofs of indistinguishability between $\mathsf{H}_{4,\mathfrak{q}-1}$ and $\mathsf{H}_{4,\mathfrak{q}}$ depend on whether $\mathfrak{q} \le Q_1$ or $\mathfrak{q} > Q_1$.

Claim 9.13 (¶). $\mathsf{H}_{4,\mathfrak{q}-1} \approx \mathsf{H}_{4,\mathfrak{q}}$ *for* $\mathfrak{q} = 1, \ldots, Q_1$.

Claim 9.14 (¶). $\mathsf{H}_{4,\mathfrak{q}-1} \approx \mathsf{H}_{4,\mathfrak{q}}$ *for* $\mathfrak{q} = Q_1 + 1, \ldots, Q_1 + Q_2$.

- H_5. In this hybrid, we finish the hybrid argument over the keys. H_5 is described as

$$
\begin{array}{lllll}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \text{hybrid}, \\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, & \mathsf{idx}_{\mathsf{H}} \leftarrow \boxed{Q_1 + Q_2}, \\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}
$$

  H_5 is just H_{4,Q_1+Q_2}.

- H_6. In this hybrid, the challenge ciphertext becomes a normal one for μ_1, i.e.,

$$
\begin{array}{lllll}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \boxed{\mu_1}, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \boxed{\text{normal}}, \\
& \mu' \leftarrow \boxed{\bot}, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \boxed{\bot}, & \mathsf{idx}_{\mathsf{H}} \leftarrow \boxed{\bot}, \\
& k'_{\mathsf{idx}} \leftarrow \boxed{\bot}, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}
$$

  $\mathsf{H}_5 \approx \mathsf{H}_6$ follows from the security of ckt, analogously to $\mathsf{H}_2 \approx \mathsf{H}_3$.

- H_7. In this hybrid, we revert $\mathsf{idx}'_{\mathsf{sk},q}$ to random, i.e.,

$$
\begin{array}{lllll}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \boxed{\text{random}}, & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_1, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \text{normal}, \\
& \mu' \leftarrow \bot, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \bot, & \mathsf{idx}_{\mathsf{H}} \leftarrow \bot, \\
& k'_{\mathsf{idx}} \leftarrow \bot, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}
$$

  $\mathsf{H}_6 \approx \mathsf{H}_7$ by the ciphertext pseudorandomness of SKE, analogously to $\mathsf{H}_1 \approx \mathsf{H}_2$.

- H_8. In this hybrid, $s_{\mathsf{PPRF},q}$'s are sampled as specified by KeyGen, i.e.,

$$
\mathsf{sk}_q: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \boxed{\text{random}}, \quad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}, \quad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};
$$

$$\begin{array}{lllll}
\mathsf{ct}: & \mu \leftarrow \mu_1, & k_{\mathsf{PPRF}} \leftarrow k, & & \mathsf{mode} \leftarrow \mathsf{normal}, \\
& \mu' \leftarrow \bot, & \mathsf{idx}_{\mathsf{ct}} \leftarrow \bot, & & \mathsf{idx}_{\mathsf{H}} \leftarrow \bot, \\
& k'_{\mathsf{idx}} \leftarrow \bot, & k'_{\mathsf{lgram}} \leftarrow \bot, & & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

H_8 is statistically indistinguishable from H_7.

By inspection, H_8 is just $\mathsf{Exp}^1_{\mathrm{PHFE}}$, and therefore, $\mathsf{Exp}^0_{\mathrm{PHFE}} \equiv \mathsf{H}_0 \approx \mathsf{H}_8 \equiv \mathsf{Exp}^1_{\mathrm{PHFE}}$. □

Proof (Claim 9.13). To show indistinguishability between $\mathsf{H}_{4,\mathfrak{q}-1}$ and $\mathsf{H}_{4,\mathfrak{q}}$ when $\boxed{\mathfrak{q} \le Q_1}$, we temporarily hardwire the LGRAM garbling yielded by decryption ct using $\mathsf{sk}_{\mathfrak{q}}$ into $\boxed{\mathsf{lgram}_{\mathsf{ct}}}$, which is generated when both $f_{\mathfrak{q}}$ and x, μ_0, μ_1 are known.

Let $\mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}$ be k punctured at the singleton set $\{s_{\mathsf{PPRF},\mathfrak{q}}\}$. We specify the hybrids.

- G_0. This is just $\mathsf{H}_{4,\mathfrak{q}-1}$, described as

$$\begin{array}{lllll}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & & \mathsf{mode} \leftarrow \mathsf{hybrid}, \\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, & & \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q}-1, \\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} \leftarrow \bot, & & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot.
\end{array}$$

- G_1. In this hybrid, we puncture k_{PPRF}, hardwire the decryption result of ct by $\mathsf{sk}_{\mathfrak{q}}$ into ct at $\mathsf{lgram}_{\mathsf{ct}}$, indicating hardwiring using mode, and increment $\mathsf{idx}_{\mathsf{H}}$, i.e.,

$$\begin{array}{lllll}
\mathsf{sk}_q: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow \boxed{\mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}}, & & \mathsf{mode} \leftarrow \boxed{\mathsf{hardwire}}, \\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, & & \mathsf{idx}_{\mathsf{H}} \leftarrow \boxed{\mathfrak{q}}, \\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} \leftarrow \bot, & &
\end{array}$$

$$\mathsf{lgram}_{\mathsf{ct}} \leftarrow \boxed{(\widehat{M}, \{L_{i,\mu_0[i]}\}_i) \text{ from } \mathsf{LGRAM.Garble}\left(T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})\right)}.$$

$\mathsf{G}_0 \approx \mathsf{G}_1$ follows from the security of ckt.

- G_2. In this hybrid, we change the LGRAM randomness from $\mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})$ to true randomness, i.e.,

$$\mathsf{sk}_q: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, \quad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), \quad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};$$

$$\mathsf{ct}: \quad \mu \leftarrow \mu_0, \quad k_{\mathsf{PPRF}} \leftarrow \mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}, \quad \mathsf{mode} \leftarrow \mathsf{hardwire},$$
$$\mu' \leftarrow \mu_1, \quad \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, \quad \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},$$
$$k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, \quad k'_{\mathsf{lgram}} \leftarrow \bot,$$
$$\mathsf{lgram}_{\mathsf{ct}} \leftarrow (\widehat{M}, \{L_{i,\mu_0[i]}\}_i) \text{ from } \mathsf{LGRAM.Garble}\left(\begin{matrix} T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \\ \boxed{\text{random}} \end{matrix}\right).$$

$\mathsf{G}_1 \approx \mathsf{G}_2$ follows from the security of PPRF.

- G_3. In this hybrid, we change the hardwired LGRAM garbling into one for μ_1, i.e.,

$$\mathsf{sk}_q: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, \quad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), \quad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};$$
$$\mathsf{ct}: \quad \mu \leftarrow \mu_0, \quad k_{\mathsf{PPRF}} \leftarrow \mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}, \quad \mathsf{mode} \leftarrow \mathsf{hardwire},$$
$$\mu' \leftarrow \mu_1, \quad \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, \quad \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},$$
$$k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, \quad k'_{\mathsf{lgram}} \leftarrow \bot,$$
$$\mathsf{lgram}_{\mathsf{ct}} \leftarrow (\widehat{M}, \{L_{i,\boxed{\mu_1}[i]}\}_i) \text{ from } \mathsf{LGRAM.Garble}\left(\begin{matrix} T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \\ \text{random} \end{matrix}\right).$$

$\mathsf{G}_2 \approx \mathsf{G}_3$ follows from the fixed-memory security of LGRAM.

- G_4. In this hybrid, the hardwired LGRAM garbling is reverted to be generated with pseudorandomness $\mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})$, i.e.,

$$\mathsf{sk}_q: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, \quad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), \quad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};$$
$$\mathsf{ct}: \quad \mu \leftarrow \mu_0, \quad k_{\mathsf{PPRF}} \leftarrow \mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}, \quad \mathsf{mode} \leftarrow \mathsf{hardwire},$$
$$\mu' \leftarrow \mu_1, \quad \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, \quad \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},$$
$$k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, \quad k'_{\mathsf{lgram}} \leftarrow \bot,$$
$$\mathsf{lgram}_{\mathsf{ct}} \leftarrow (\widehat{M}, \{L_{i,\mu_1[i]}\}_i) \text{ from } \mathsf{LGRAM.Garble}\left(\begin{matrix} T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \\ \boxed{\mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})} \end{matrix}\right).$$

$\mathsf{G}_3 \approx \mathsf{G}_4$ follows from the security of PPRF.

- G_5. In this hybrid, k_{PPRF} is no longer punctured and hardwiring is undone, i.e.,

$$\mathsf{sk}_q: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, \quad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), \quad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};$$

$$\begin{aligned}
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} &\leftarrow \boxed{k}, & \mathsf{mode} &\leftarrow \mathsf{hybrid},\\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} &\leftarrow Q_1, & \mathsf{idx}_{\mathsf{H}} &\leftarrow \mathfrak{q},\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} &\leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} &\leftarrow \boxed{\bot}.
\end{aligned}$$

$\mathsf{G}_4 \approx \mathsf{G}_5$ follows from the security of ckt.

By inspection, G_5 is exactly $\mathsf{H}_{4,\mathfrak{q}}$. Therefore, $\mathsf{H}_{4,\mathfrak{q}-1} \equiv \mathsf{G}_0 \approx \mathsf{G}_5 \equiv \mathsf{H}_{4,\mathfrak{q}}$. □

Proof (Claim 9.14). To show indistinguishability between $\mathsf{H}_{4,\mathfrak{q}-1}$ and $\mathsf{H}_{4,\mathfrak{q}}$ when $\boxed{\mathfrak{q} > Q_1}$, we temporarily hardwire the LGRAM garbling yielded by decryption ct using $\mathsf{sk}_{\mathfrak{q}}$ into $\boxed{\mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}}}$, generated when both x, μ_0, μ_1 and $f_{\mathfrak{q}}$ are known. Let $\mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}$ be k punctured at $\{s_{\mathsf{PPRF},\mathfrak{q}}\}$ and k_{lgram} a random secret key of SKE. We specify the hybrids.

- G_0. This is just $\mathsf{H}_{4,\mathfrak{q}-1}$, described as

$$\begin{aligned}
\mathsf{sk}_{q\neq\mathfrak{q}}: \quad & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} &\xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} &\xleftarrow{\$} \text{random};\\
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} &\leftarrow k, & \mathsf{mode} &\leftarrow \mathsf{hybrid},\\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} &\leftarrow Q_1, & \mathsf{idx}_{\mathsf{H}} &\leftarrow \mathfrak{q}-1,\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} &\leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} &\leftarrow \bot;\\
\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} &\xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), & \mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} &\xleftarrow{\$} \text{random}.
\end{aligned}$$

- G_1. In this hybrid, we encrypt the LGRAM garbling into $\mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}}$, i.e.,

$$\begin{aligned}
\mathsf{sk}_{q\neq\mathfrak{q}}: \quad & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} &\xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} &\xleftarrow{\$} \text{random};\\
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} &\leftarrow k, & \mathsf{mode} &\leftarrow \mathsf{hybrid},\\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} &\leftarrow Q_1, & \mathsf{idx}_{\mathsf{H}} &\leftarrow \mathfrak{q}-1,\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} &\leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} &\leftarrow \bot;\\
\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} &\xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), & & \\
& \mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \boxed{\mathsf{SKE.Enc}\Big(k_{\mathsf{lgram}}, & & (\widehat{M}, \{L_{i,\mu_0[i]}\}_i) \text{ from } \mathsf{LGRAM.Garble}\Big(T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})\Big)\Big)}.
\end{aligned}$$

$\mathsf{G}_0 \approx \mathsf{G}_1$ by the ciphertext pseudorandomness of SKE.

- G_2. In this hybrid, we puncture k at $\{s_{\mathsf{PPRF},\mathfrak{q}}\}$, activate the hardwiring using k'_{lgram} and mode, and increment $\mathsf{idx}_{\mathsf{H}}$, i.e.,

$$\begin{aligned}
\mathsf{sk}_{q\neq\mathfrak{q}}: \quad & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), && \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};\\
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, && k_{\mathsf{PPRF}} \leftarrow \boxed{\mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}}, && \mathsf{mode} \leftarrow \boxed{\text{hardwire}},\\
& \mu' \leftarrow \mu_1, && \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, && \mathsf{idx}_{\mathsf{H}} \leftarrow \boxed{\mathfrak{q}},\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, && k'_{\mathsf{lgram}} \leftarrow \boxed{k_{\mathsf{lgram}}}, && \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot;\\
\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), &&
\end{aligned}$$

$$\mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}\left(k_{\mathsf{lgram}}, \begin{array}{c}(\widehat{M}, \{L_{i,\mu_0[i]}\}_i) \text{ from}\\ \mathsf{LGRAM.Garble}\left(T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \right.\\ \left.\mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})\right)\end{array}\right).$$

$\mathsf{G}_1 \approx \mathsf{G}_2$ follows from the security of ckt.

- G_3. In this hybrid, we generate the hardwired garbling with true randomness instead of $\mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})$, i.e.,

$$\begin{aligned}
\mathsf{sk}_{q\neq\mathfrak{q}}: \quad & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), && \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};\\
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, && k_{\mathsf{PPRF}} \leftarrow \mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}, && \mathsf{mode} \leftarrow \text{hardwire},\\
& \mu' \leftarrow \mu_1, && \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, && \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, && k'_{\mathsf{lgram}} \leftarrow k_{\mathsf{lgram}}, && \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot;\\
\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), &&
\end{aligned}$$

$$\mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}\left(k_{\mathsf{lgram}}, \begin{array}{c}(\widehat{M}, \{L_{i,\mu_0[i]}\}_i) \text{ from}\\ \mathsf{LGRAM.Garble}\left(T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \right.\\ \left.\boxed{\text{random}}\right)\end{array}\right).$$

$\mathsf{G}_2 \approx \mathsf{G}_3$ follows from the security of PPRF.

- G_4. In this hybrid, the hardwired garbling becomes one for μ_1, i.e.,

$$\mathsf{sk}_{q\neq\mathfrak{q}}: \quad s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, \qquad \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), \qquad \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};$$

$$
\begin{aligned}
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, && k_{\mathsf{PPRF}} \leftarrow \mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}, && \mathsf{mode} \leftarrow \mathsf{hardwire},\\
& \mu' \leftarrow \mu_1, && \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, && \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, && k'_{\mathsf{lgram}} \leftarrow k_{\mathsf{lgram}}, && \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot;\\
\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), &&\\
& \mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}\left(k_{\mathsf{lgram}}, \begin{array}{c}(\widehat{M}, \{L_{i,\boxed{\mu_1}[i]}\}_i) \text{ from}\\ \mathsf{LGRAM.Garble}\left(\begin{array}{c}T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x;\\ \text{random}\end{array}\right)\end{array}\right).&&&&
\end{aligned}
$$

$\mathsf{G}_3 \approx \mathsf{G}_4$ follows from the fixed-memory security of LGRAM.

- G_5. In this hybrid, the randomness for generating the hardwired LGRAM garbling is reverted to be pseudorandom, i.e.,

$$
\begin{aligned}
\mathsf{sk}_{q\neq\mathfrak{q}}: \quad & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), && \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};\\
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, && k_{\mathsf{PPRF}} \leftarrow \mathring{k}_{\{s_{\mathsf{PPRF},\mathfrak{q}}\}}, && \mathsf{mode} \leftarrow \mathsf{hardwire},\\
& \mu' \leftarrow \mu_1, && \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, && \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, && k'_{\mathsf{lgram}} \leftarrow k_{\mathsf{lgram}}, && \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot;\\
\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), &&\\
& \mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}\left(k_{\mathsf{lgram}}, \begin{array}{c}(\widehat{M}, \{L_{i,\mu_1[i]}\}_i) \text{ from}\\ \mathsf{LGRAM.Garble}\left(\begin{array}{c}T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x;\\ \boxed{\mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})}\end{array}\right)\end{array}\right).&&&&
\end{aligned}
$$

$\mathsf{G}_4 \approx \mathsf{G}_5$ follows from the security of PPRF.

- G_6. In this hybrid, the hardwiring is deactivated by reverting most of the changes made in the transition from G_1 to G_2, i.e.,

$$
\begin{aligned}
\mathsf{sk}_{q\neq\mathfrak{q}}: \quad & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, && \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), && \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random};\\
\mathsf{ct}: \quad & \mu \leftarrow \mu_0, && k_{\mathsf{PPRF}} \leftarrow \boxed{k}, && \mathsf{mode} \leftarrow \boxed{\mathsf{hybrid}},\\
& \mu' \leftarrow \mu_1, && \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, && \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q},\\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, && k'_{\mathsf{lgram}} \leftarrow \boxed{\bot}, && \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot;
\end{aligned}
$$

$$\mathsf{sk}_{\mathfrak{q}>Q_1}: \quad s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, \qquad \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}),$$
$$\mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}\left(k_{\mathsf{lgram}}, \begin{array}{c}(\widehat{M}, \{L_{i,\mu_1[i]}\}_i) \text{ from} \\ \mathsf{LGRAM.Garble}\left(T_{\max}, M, \mathsf{digest}_{f_{\mathfrak{q}}}, \mathsf{digest}_x; \right. \\ \left. \mathsf{PPRF.Eval}(k, s_{\mathsf{PPRF},\mathfrak{q}})\right)\end{array}\right).$$

$\mathsf{G}_5 \approx \mathsf{G}_6$ follows from the security of ckt.

- G_7. In this hybrid, we revert $\mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}}$ to random, i.e.,

$$\begin{array}{rccccc}
\mathsf{sk}_{q\neq\mathfrak{q}}: & s_{\mathsf{PPRF},q} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},q} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, q), & \mathsf{lgram}'_{\mathsf{sk},q} \xleftarrow{\$} \text{random}; \\
\mathsf{ct}: & \mu \leftarrow \mu_0, & k_{\mathsf{PPRF}} \leftarrow k, & \mathsf{mode} \leftarrow \mathsf{hybrid}, \\
& \mu' \leftarrow \mu_1, & \mathsf{idx}_{\mathsf{ct}} \leftarrow Q_1, & \mathsf{idx}_{\mathsf{H}} \leftarrow \mathfrak{q}, \\
& k'_{\mathsf{idx}} \leftarrow k_{\mathsf{idx}}, & k'_{\mathsf{lgram}} \leftarrow \bot, & \mathsf{lgram}_{\mathsf{ct}} \leftarrow \bot; \\
\mathsf{sk}_{\mathfrak{q}>Q_1}: & s_{\mathsf{PPRF},\mathfrak{q}} \xleftarrow{\$} \text{distinct}, & \mathsf{idx}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \mathsf{SKE.Enc}(k_{\mathsf{idx}}, \mathfrak{q}), & \mathsf{lgram}'_{\mathsf{sk},\mathfrak{q}} \xleftarrow{\$} \boxed{\bot}.
\end{array}$$

$\mathsf{G}_6 \approx \mathsf{G}_7$ by the ciphertext pseudorandomness of SKE.

By inspection, G_7 is exactly $\mathsf{H}_{4,\mathfrak{q}}$. Therefore, $\mathsf{H}_{4,\mathfrak{q}-1} \equiv \mathsf{G}_0 \approx \mathsf{G}_7 \equiv \mathsf{H}_{4,\mathfrak{q}}$. □

9.7 Trading Short Components for Fast Decryption

Theorem 9.12 matches the lower bound in Section 9.3. By breaking abstraction and slightly tweaking our construction, we obtain additional trade-offs matching the lower bound:

Corollary 9.15 (¶). *Assuming* FE for circuits *(Definition 9.6), for all* $A, B \in \{0,1\}$*, there exists* ABE for RAM *(Definition 9.7) with*

$$T_{\mathsf{KeyGen}} = |f|\,\mathrm{poly}(\lambda, |M|, \log T_{\max}), \quad |\mathsf{sk}| = |f|^A + \mathrm{poly}(\lambda, |M|, \log T_{\max}),$$
$$T_{\mathsf{Enc}} = |x|\,\mathrm{poly}(\lambda, |M|, \log T_{\max}), \quad |\mathsf{ct}| = |x|^B + \mathrm{poly}(\lambda, |M|, \log T_{\max}),$$
$$|\mathsf{mpk}| = \mathrm{poly}(\lambda, |M|, \log T_{\max}), \quad T_{\mathsf{Dec}} = (T + |f|^{1-A} + |x|^{1-B})\,\mathrm{poly}(\lambda, |M|, \log T_{\max}).$$

Proof Sketch (Corollary 9.15). When Construction 9.3 is instantiated with Construc-

tions 9.2 and 9.1, the linear dependency of T_{Dec} on $|f|, |x|$ comes solely from recomputing the LOT processed databases (for f, x). In the proof of Lemma 9.3, the processed database $\widehat{D}$ is the Merkle hash tree of D, whose bit-length can be made to be that of D.[143] Therefore, by choosing to store the Merkle tree of f, x (or not) in sk, ct, we achieve the four promised trade-offs. □

9.8 Further Discussions

Here are some further comments on the topic of this chapter, after which some results excluded from this dissertation are listed.

Concreteness in Space-Time Trade-Offs. I hold the opinion that the characterization of *storage* complexity should be down-to-constant in the leading terms, whereas that of time/*space* complexity can be asymptotic. For storage, the bottleneck is often the amount of necessary information. For time/space complexity, the bounds depend on the minor details of computation model. As an example, for PKE, having $|\mathsf{ct}| = |\mu| + \mathrm{poly}(\lambda)$, $\mathrm{poly}(\lambda)$-time decryption for any specific bit in the plaintext, and $|\mu|\,\mathrm{poly}(\lambda)$-time decryption for the entire ciphertext can be considered optimal — we require strict storage dependency on μ, but multiplicative $\mathrm{poly}(\lambda)$ slow down in reading bits in μ from ct compared to direct reading (i.e., 1 step for every bit).

Trade-Offs in Theorem 9.4 and Corollary 9.15. We do not achieve all the trade-offs still possible subject to Theorem 9.4 in Corollary 9.15. One straight-forward idea is to store a part of the LOT Merkle tree and reconstruct missing parts on demand, say by omitting the last L levels. The simplest implementation will blow up the time to evaluate each step of the garbling by $\mathrm{poly}(\lambda)\min(2^L, |x|)$, the time to reconstruct one root-to-leaf path, while reducing the size dependency by 2^L.

If $L = \omega(\log \lambda)$, we would have $|ct| = \mathrm{poly}(\lambda)$ but $T_{\mathsf{Dec}} \geq T|x|$ as $|x| \leq \mathrm{poly}(\lambda) < 2^L$, which is worse than $T_{\mathsf{Dec}} \sim T + |x|$. If $L = \lfloor \log \sqrt{|x|} \rceil$, then $|\mathsf{ct}| = \sqrt{|x|} + \mathrm{poly}(\lambda)$ but

[143] The standard Merkle tree is twice as long as the database as the leaves constitute the database itself. Recall that in the syntax of PHFE (Definition 2.1), the databases f, x are given in the clear for free, so we can exclude them from *the* Merkle trees stored in sk, ct.

$T_{\mathsf{Dec}} \geq T\sqrt{|x|}$. This is a strange trade-off and often worse than $T_{\mathsf{Dec}} \sim T + |x|$. However, when $T = \mathrm{poly}(\lambda)$, we do achieve all the possible trade-offs by tuning L dependent on $|x|$. A simple version of this fact is reflected in [Luo24].

Setting $L = \Theta(\log \lambda)$, one can achieve $|\mathsf{ct}| = \frac{1}{\lambda^k}|x| + \mathrm{poly}(\lambda)$ and T_{Dec} independent of $|x|$, for any constant k. (There is an expense of larger $\mathrm{poly}(\lambda)$ hidden in front of T in T_{Dec}.) This is not satisfactory to me. For one, $|x|$ should be thought of as varying independently of λ (polynomially bounded for security, but there is no fixed $\mathrm{poly}(\lambda)$ bounding $|x|$). Otherwise, this gives an iffy feeling of "playing with" trade-offs.

Custom-Encoded ABE and Fine-Grained Rate Characterization. Continuing the discussion of accounting models of ABE efficiency from Section 9.1.2, the iffy trade-offs imply one can achieve rate $\left(1 + \frac{1}{\mathrm{poly}(\lambda)}\right)$ component sizes in the custom-encoded model (recall that in this model, x, f are to be arbitrarily encoded in ct, sk and not provided for free during decryption, unlike the verbatim-for-free model used in this dissertation). This means the interesting characterization of optimal rate must be finer-grained, e.g., whether one can achieve $|\mathsf{ct}| = |x| + \mathrm{poly}(\lambda)$ or $|\mathsf{ct}| = |x| + \sqrt{|x|} + \mathrm{poly}(\lambda)$ while maintaining T_{Dec} independent of $|x|$.

Results Excluded Here. See [JLL23,JLL22] for details.

Theorem (PHFE for RAM, nearly optimal succinctness, linear decryption efficiency). *Assuming polynomially secure FE for circuits, there exists an adaptively secure PHFE for RAM with*

$$|\mathsf{mpk}| = O(1), \quad |\mathsf{sk}_f| = O(1), \quad |\mathsf{ct}_x(y)| = 2|y| + O(1), \qquad T_{\mathsf{Dec}} = O(T + |f| + |x| + |y|),$$

and a semi-adaptively secure PHFE for decisional (1-bit output) RAM with

$$|\mathsf{mpk}| = O(1), \quad |\mathsf{sk}_f| = O(1), \quad |\mathsf{ct}_x(y)| = |y| + O(1), \qquad T_{\mathsf{Dec}} = O(T + |f| + |x| + |y|).$$

Note that the first PHFE implies an ABE with the same efficiency and security characteristics as Construction 9.3, but the latter is *not* the scheme obtained from the former by using it as ABE. In general, ABE should be studied separately due to the theorem below, which subjects PHFE to stricter lower bounds than currently known for ABE (Theorem 9.4).

Theorem (PHFE time-space trade-offs). *For a secret-key 1-key 1-ciphertext selectively secure moderately expressive PHFE with*

$$\begin{array}{rlll} \text{either} & |\mathsf{sk}| = \mathrm{O}(|f|^A) & \text{and} & T_{\mathsf{Dec}} = (T + |f|^B + |y|)\,\mathrm{poly}(|x|), \\ \text{or} & |\mathsf{ct}| = |x|^A\,\mathrm{poly}(|y|) & \text{and} & T_{\mathsf{Dec}} = (T + |f| + |x|^B)\,\mathrm{poly}(|y|), \end{array}$$

it must hold that $A \geq 1$ or $B \geq 1$. In the first case, the same (without x) is true for FE.

The lower bound above is stricter than Theorem 9.4 (for ABE). In particular, the blue unknown area in Figure 9.1 is impossible for PHFE.

Theorem (PHFE and DE-PIR). *Suppose a secret-key moderately expressive PHFE with selective security has*

$$\begin{array}{rlll} \text{either} & |\mathsf{sk}_f| = \mathrm{O}(|f|^A) & \text{and} & T_{\mathsf{Dec}} = |f|^B\,\mathrm{poly}(T, |x|, |y|), \\ \text{or} & |\mathsf{ct}_x| = |x|^A\,\mathrm{poly}(|y|) & \text{and} & T_{\mathsf{Dec}} = |x|^B\,\mathrm{poly}(T, |f|, |y|), \\ \text{or} & |\mathsf{ct}_x| = |y|^A\,\mathrm{poly}(|x|) & \text{and} & T_{\mathsf{Dec}} = |y|^B\,\mathrm{poly}(T, |f|, |x|), \end{array}$$

for constants A and $0 \leq B < 1$, then there exists a secret-key doubly efficient private information retrieval (DE-PIR) with

$$|\widetilde{D}| = |D| + \mathrm{O}(|D|^A), \qquad T_{\mathsf{CreateQuery}} = \mathrm{O}(1),$$
$$T_{\mathsf{CreateResponse}} = \mathrm{O}(|D|^B), \qquad T_{\mathsf{DecryptResponse}} = \mathrm{O}(1).$$

In the first case, the PHFE only has to be 1-key secure, and if it is public-key, then so can be the resultant DE-PIR.

DE-PIR [BIPW17,CHR17,BHMW21,LMW23] enables clients to query servers for bits in a long string encoded and stored on the server while keeping the query indices hidden from the server. A PHFE scheme with decryption time independent of $|y|$ and ciphertext size linear in $|y|$ implies a DE-PIR with $|\widetilde{D}| = \mathrm{O}(|D|)$ and constant time complexity per query. Such efficiency is currently considered open (see [JLL22] for elaboration on this point), so the corresponding PHFE is difficult to construct.

Theorem (constant-overhead $i\mathcal{O}$). *Assuming subexponentially secure FE for circuits, there exists a subexponentially secure $i\mathcal{O}$ for RAM, where the obfuscated program is of size $(2|f| + \mathrm{poly}(\lambda, N))$ for input length N.*

Bibliography

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010. DOI: 10.1007/978-3-642-13190-5_28. (Cited on page 292.)

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Berlin, Heidelberg, March / April 2015. DOI: 10.1007/978-3-662-46447-2_33. (Cited on pages 145, 190, 202, 203, 211, 240, 257, 265, 266, 272, 280.)

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. URL: https://eprint.iacr.org/2013/689. (Cited on page 363.)

[AC17] Shashank Agrawal and Melissa Chase. Simplifying design and analysis of complex predicate encryption schemes. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Cham, April / May 2017. DOI: 10.1007/978-3-319-56620-7_22. (Cited on pages 25, 28, 29, 48.)

[ACC+16] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 3–30. Springer, Berlin, Heidelberg, October / November 2016. DOI: 10.1007/978-3-662-53644-5_1. (Cited on pages 29, 428, 431.)

[ACF+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Cham, August 2018. DOI: 10.1007/978-3-319-96884-1_20. (Cited on pages 165, 286.)

[ACFQ22] Prabhanjan Ananth, Kai-Min Chung, Xiong Fan, and Luowen Qian. Collusion-resistant functional encryption for RAMs. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 160–194. Springer, Cham, December 2022. DOI: 10.1007/978-3-031-22963-3_6. (Cited on pages 364, 427, 437.)

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Berlin, Heidelberg, August 2009. DOI: 10.1007/978-3-642-03356-8_35. (Cited on pages 216, 217, 228, 379.)

[AFS19] Prabhanjan Ananth, Xiong Fan, and Elaine Shi. Towards attribute-based encryption for RAMs from LWE: Sub-linear decryption, and more. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 112–141. Springer, Cham, December 2019. DOI: 10.1007/978-3-030-34578-5_5. (Cited on pages 48, 436.)

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Cham, May 2019. DOI: 10.1007/978-3-030-17653-2_7. (Cited on pages 264, 266, 280.)

[AGW20] Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*,

pages 685–716. Springer, Cham, August 2020. DOI: 10.1007/978-3-030-56784-2_23. (Cited on page 149.)

[AHY15] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 575–601. Springer, Berlin, Heidelberg, November / December 2015. DOI: 10.1007/978-3-662-48797-6_24. (Cited on pages 138, 149.)

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004. DOI: 10.1109/FOCS.2004.20. (Cited on page 31.)

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011. DOI: 10.1109/FOCS.2011.40. (Cited on pages 20, 30, 31, 34, 40, 59, 90, 259, 269, 304, 306, 311.)

[AIKW13] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 166–184. Springer, Berlin, Heidelberg, August 2013. DOI: 10.1007/978-3-642-40084-1_10. (Cited on page 458.)

[AJS17] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation for Turing machines: Constant overhead and amortization. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 252–279. Springer, Cham, August 2017. DOI: 10.1007/978-3-319-63715-0_9. (Cited on pages 364, 427, 428.)

[AK08] Per Austrin and Gunnar Kreitz. Lower bounds for subset cover based broadcast encryption. In Serge Vaudenay, editor, *AFRICACRYPT 08*,

volume 5023 of *LNCS*, pages 343–356. Springer, Berlin, Heidelberg, June 2008. DOI: 10.1007/978-3-540-68164-9_23. (Cited on page 436.)

[AKY24a] Shweta Agrawal, Simran Kumari, and Shota Yamada. Attribute based encryption for turing machines from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 352–386. Springer, Cham, August 2024. DOI: 10.1007/978-3-031-68382-4_11. (Cited on pages 365, 366.)

[AKY24b] Shweta Agrawal, Simran Kumari, and Shota Yamada. Pseudorandom multi-input functional encryption and applications. Cryptology ePrint Archive, Report 2024/1720, 2024. URL: https://eprint.iacr.org/2024/1720. (Cited on pages 365, 366, 382.)

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 455–472. Springer, Cham, November 2018. DOI: 10.1007/978-3-030-03810-6_17. (Cited on pages 428, 434, 435, 450, 451, 458, 475.)

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, Berlin, Heidelberg, March 2011. DOI: 10.1007/978-3-642-19379-8_6. (Cited on pages 138, 149, 196, 197, 426, 436.)

[ALMT20] Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 34–64. Springer, Cham, May 2020. DOI: 10.1007/978-3-030-45374-9_2. (Cited on pages 157, 162, 165, 286.)

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Berlin, Heidelberg, August 2016. DOI: 10.1007/978-3-662-53015-3_12. (Cited on pages 58, 135, 145, 148, 151, 161, 162, 203, 209, 210, 220, 266, 272, 280.)

[AM18] Shweta Agrawal and Monosij Maitra. FE and iO for Turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 473–512. Springer, Cham, November 2018. DOI: 10.1007/978-3-030-03810-6_18. (Cited on pages 25, 27, 48, 364, 427, 435.)

[AMR25a] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: Fully succinct randomised encodings and more. Cryptology ePrint Archive, Report 2025/339, 2025. URL: https://eprint.iacr.org/2025/339. (Cited on page 349.)

[AMR25b] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor hashing for all circuits. Cryptology ePrint Archive, Report 2025/336, 2025. URL: https://eprint.iacr.org/2025/336. (Cited on pages 254, 360.)

[AMY19a] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 765–797. Springer, Cham, August 2019. DOI: 10.1007/978-3-030-26951-7_26. (Cited on pages 25, 28, 48, 260, 264.)

[AMY19b] Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption for deterministic finite automata from DLIN. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of

LNCS, pages 91–117. Springer, Cham, December 2019. DOI: 10.1007/978-3-030-36033-7_4. (Cited on pages 25, 28, 48.)

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 110–140. Springer, Cham, May 2020. DOI: 10.1007/978-3-030-45721-1_5. (Cited on pages 264, 266, 280.)

[ARS24] Damiano Abram, Lawrence Roy, and Mark Simkin. Time-based cryptography from weaker assumptions: Randomness beacons, delay functions and more. Cryptology ePrint Archive, Report 2024/769, 2024. URL: https://eprint.iacr.org/2024/769. (Cited on page 365.)

[ARYY23] Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 532–564. Springer, Cham, August 2023. DOI: 10.1007/978-3-031-38551-3_17. (Cited on page 381.)

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for Turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Berlin, Heidelberg, January 2016. DOI: 10.1007/978-3-662-49096-9_6. (Cited on pages 27, 48, 364, 427, 431, 435.)

[AS17a] Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from learning with errors. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *ICALP 2017*, volume 80 of *LIPIcs*, pages 36:1–36:13. Schloss Dagstuhl, July 2017. DOI: 10.4230/LIPIcs.ICALP.2017.36. (Cited on pages 260, 264.)

[AS17b] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear

maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Cham, April / May 2017. DOI: 10.1007/978-3-319-56620-7_6. (Cited on page 25.)

[AT20] Nuttapong Attrapadung and Junichi Tomida. Unbounded dynamic predicate compositions in ABE from standard assumptions. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 405–436. Springer, Cham, December 2020. DOI: 10.1007/978-3-030-64840-4_14. (Cited on pages 138, 139, 140, 141, 149, 193, 196, 197, 426, 436.)

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Berlin, Heidelberg, May 2014. DOI: 10.1007/978-3-642-55220-5_31. (Cited on pages 1, 19, 25, 28, 29, 48.)

[Att16] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Berlin, Heidelberg, December 2016. DOI: 10.1007/978-3-662-53890-6_20. (Cited on pages 25, 28, 29, 48, 56, 138, 149, 196, 197, 220, 426, 436.)

[Att17] Nuttapong Attrapadung. Dual system framework in multilinear settings and applications to fully secure (compact) ABE for unbounded-size circuits. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 3–35. Springer, Berlin, Heidelberg, March 2017. DOI: 10.1007/978-3-662-54388-7_1. (Cited on page 25.)

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass

and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 149–178. Springer, Cham, November 2020. DOI: 10.1007/978-3-030-64375-1_6. (Cited on pages 11, 196, 197, 199, 201, 202, 203, 208, 221, 222, 223, 224, 243, 259, 262, 263, 271, 349.)

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Cham, May 2020. DOI: 10.1007/978-3-030-45721-1_2. (Cited on pages 141, 196, 197, 198, 199, 201, 202, 203, 208, 221, 259, 262, 263, 349.)

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th ACM STOC*, pages 1–5. ACM Press, May 1986. DOI: 10.1145/12130.12131. (Cited on page 213.)

[BC95] Carlo Blundo and Antonella Cresti. Space requirements for broadcast encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 287–298. Springer, Berlin, Heidelberg, May 1995. DOI: 10.1007/BFb0053444. (Cited on page 436.)

[BDHM92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-MOD class. *Mathematical Systems Theory*, 25(3):223–237, September 1992. DOI: 10.1007/BF01374526. (Cited on pages 128, 221, 259, 266.)

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Technion–Israel Institute of Technology, 1996. (Cited on page 221.)

[Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989. DOI: 10.1137/0218053. (Cited on page 467.)

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*,

18(3):147–150, 1984. DOI: 10.1016/0020-0190(84)90018-8. (Cited on pages 221, 259, 266.)

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, Berlin, Heidelberg, May 2011. DOI: 10.1007/978-3-642-20465-4_10. (Cited on page 354.)

[BG99] Amos Beimel and Anna Gál. On arithmetic branching programs. *Journal of Computer and System Sciences*, 59(2):195–220, 1999. DOI: 10.1006/jcss.1999.1648. (Cited on pages 52, 55.)

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. DOI: 10.1007/978-3-642-55220-5_30. (Cited on pages 19, 25, 30, 31, 40, 90, 141, 195, 196, 198, 199, 213, 214, 221, 225, 245, 255, 260, 263, 270, 338, 339, 340, 349, 354, 357, 359, 360, 361, 363, 364, 365, 366, 367, 368, 369, 370, 386, 403, 425, 427, 435.)

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001. DOI: 10.1007/3-540-44647-8_1. (Cited on page 354.)

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 439–448. ACM Press, June 2015. DOI: 10.1145/2746539.2746574. (Cited on pages 28, 428, 431, 434.)

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. DOI: 10.1145/2090236.2090262. (Cited on pages 356, 362.)

[BHMW21] Elette Boyle, Justin Holmgren, Fermi Ma, and Mor Weiss. On the security of doubly efficient PIR. Cryptology ePrint Archive, Report 2021/1113, 2021. URL: https://eprint.iacr.org/2021/1113. (Cited on page 493.)

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012. DOI: 10.1145/2382196.2382279. (Cited on page 434.)

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 662–693. Springer, Cham, November 2017. DOI: 10.1007/978-3-319-70503-3_22. (Cited on page 493.)

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Berlin, Heidelberg, November / December 2015. DOI: 10.1007/978-3-662-48797-6_20. (Cited on pages 30, 31.)

[BL15] Xavier Boyen and Qinyi Li. Attribute-based encryption for finite automata from LWE. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 247–267. Springer, Cham, November 2015. DOI: 10.1007/978-3-319-26059-4_14. (Cited on page 48.)

[BM24] Zvika Brakerski and Stav Medina. Limits on adaptive security for attribute-based encryption. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 91–123. Springer,

Cham, December 2024. DOI: 10.1007/978-3-031-78020-2_4. (Cited on page 360.)

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990. DOI: 10.1145/100216.100287. (Cited on page 451.)

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Cham, August 2019. DOI: 10.1007/978-3-030-26951-7_27. (Cited on page 56.)

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007. DOI: 10.1109/SP.2007.11. (Cited on page 29.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Berlin, Heidelberg, March 2011. DOI: 10.1007/978-3-642-19571-6_16. (Cited on pages 211, 353.)

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 264–302. Springer, Cham, November 2017. DOI: 10.1007/978-3-319-70500-2_10. (Cited on pages 213, 370, 384, 386, 387, 416.)

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages

418–449. Springer, Singapore, December 2024. DOI: 10.1007/978-981-96-0894-2_14. (Cited on page 383.)

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. DOI: 10.1109/FOCS.2011.12. (Cited on page 354.)

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Berlin, Heidelberg, March 2015. DOI: 10.1007/978-3-662-46497-7_1. (Cited on pages 213, 354.)

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384. Springer, Berlin, Heidelberg, August 2016. DOI: 10.1007/978-3-662-53015-3_13. (Cited on pages 355, 364, 365, 415.)

[BV20] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. Cryptology ePrint Archive, Report 2020/191, 2020. URL: https://eprint.iacr.org/2020/191. (Cited on page 349.)

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. In Mark Braverman, editor, *ITCS 2022*, volume 215, pages 28:1–28:20. LIPIcs, January / February 2022. DOI: 10.4230/LIPIcs.ITCS.2022.28. (Cited on page 349.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer,

Berlin, Heidelberg, December 2013. DOI: 10.1007/978-3-642-42045-0_15. (Cited on pages 354, 452.)

[CCC+16] Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Cryptography for parallel RAM from indistinguishability obfuscation. In Madhu Sudan, editor, *ITCS 2016*, pages 179–190. ACM, January 2016. DOI: 10.1145/2840728.2840769. (Cited on pages 428, 431.)

[CCHR16] Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Adaptive succinct garbled RAM or: How to delegate your database. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 61–90. Springer, Berlin, Heidelberg, October / November 2016. DOI: 10.1007/978-3-662-53644-5_3. (Cited on pages 428, 431.)

[CCL18] Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 371–390. Springer, Cham, April / May 2018. DOI: 10.1007/978-3-319-78372-7_12. (Cited on pages 202, 207.)

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Cham, August 2017. DOI: 10.1007/978-3-319-63715-0_2. (Cited on pages 448, 450, 451.)

[CGKW18a] Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 503–534. Springer, Cham, April / May 2018. DOI: 10.1007/978-3-319-78381-9_19. (Cited on pages 25, 27, 35, 48, 169.)

[CGKW18b] Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. Cryptology ePrint Archive, Report 2018/116, 2018. URL: https://eprint.iacr.org/2018/116. (Cited on page 193.)

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Berlin, Heidelberg, April 2015. DOI: 10.1007/978-3-662-46803-6_20. (Cited on pages 29, 194.)

[CGW18] Jie Chen, Junqing Gong, and Hoeteck Wee. Improved inner-product encryption with adaptive security and full attribute-hiding. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 673–702. Springer, Cham, December 2018. DOI: 10.1007/978-3-030-03329-3_23. (Cited on page 31.)

[CH16] Ran Canetti and Justin Holmgren. Fully succinct garbled RAM. In Madhu Sudan, editor, *ITCS 2016*, pages 169–178. ACM, January 2016. DOI: 10.1145/2840728.2840765. (Cited on pages 428, 431.)

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 429–437. ACM Press, June 2015. DOI: 10.1145/2746539.2746621. (Cited on pages 28, 428, 431.)

[CHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 3–33. Springer, Cham, May / June 2022. DOI: 10.1007/978-3-031-07085-3_1. (Cited on page 430.)

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Berlin, Heidelberg, April 2015. DOI: 10.1007/978-3-662-46800-5_1. (Cited on pages 256, 359.)

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 694–726. Springer, Cham, November 2017. DOI: 10.1007/978-3-319-70503-3_23. (Cited on page 493.)

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Berlin, Heidelberg, March 2015. DOI: 10.1007/978-3-662-46497-7_19. (Cited on page 354.)

[CR73] Stephen A. Cook and Robert A. Reckhow. Time bounded random access machines. *Journal of Computer and System Sciences*, 7(4):354–375, 1973. DOI: 10.1016/S0022-0000(73)80029-7. (Cited on page 426.)

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002. DOI: 10.1007/3-540-46035-7_4. (Cited on page 151.)

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Cham, August 2018. DOI: 10.1007/978-3-319-96881-0_20. (Cited on pages 256, 359.)

[CW23] Valerio Cini and Hoeteck Wee. ABE for circuits with poly (λ)-sized keys from LWE. In *64th FOCS*, pages 435–446. IEEE Computer Society Press, November 2023. DOI: 10.1109/FOCS57990.2023.00032. (Cited on pages 349, 355, 365.)

[CW24] Valerio Cini and Hoeteck Wee. Unbounded ABE for circuits from LWE, revisited. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 238–267. Springer, Singapore, December 2024. DOI: 10.1007/978-981-96-0894-2_8. (Cited on pages 349, 355.)

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 164–195. Springer, Berlin, Heidelberg, March 2016. DOI: 10.1007/978-3-662-49384-7_7. (Cited on pages 31, 140.)

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. DOI: 10.1109/TIT.1976.1055638. (Cited on page 354.)

[DHM+24] Fangqi Dong, Zihan Hao, Ethan Mook, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and ABE for RAMs from (ring-)LWE. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 107–142. Springer, Cham, August 2024. DOI: 10.1007/978-3-031-68382-4_4. (Cited on page 365.)

[DLY21] Ivan Bjerre Damgård, Kasper Green Larsen, and Sophia Yakoubov. Broadcast secret-sharing, bounds and applications. In Stefano Tessaro, editor, *ITC 2021*, volume 199 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl, July 2021. DOI: 10.4230/LIPIcs.ITC.2021.10. (Cited on page 436.)

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 256–287. Springer, Cham, November 2021. DOI: 10.1007/978-3-030-90453-1_9. (Cited on page 359.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013. DOI: 10.1007/978-3-642-40084-1_8. (Cited on pages 13, 56, 151.)

[EMN+09] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In Feng Bao, Hui Li, and Guilin Wang, editors, *Information Security Practice and Experience*, pages 13–23, Berlin, Heidelberg, 2009. Springer. (Cited on page 137.)

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009. DOI: 10.1145/1536414.1536440. (Cited on pages 353, 354, 355, 362.)

[GGH+13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. DOI: 10.1109/FOCS.2013.13. (Cited on pages 197, 256, 354, 363, 364, 427, 435.)

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Berlin, Heidelberg, August 2013. DOI: 10.1007/978-3-642-40084-1_27. (Cited on page 364.)

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. DOI: 10.1145/2488608.2488667. (Cited on page 197.)

[GKP+13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Berlin, Heidelberg, August 2013. DOI: 10.1007/978-3-642-40084-1_30. (Cited on pages 27, 197, 256, 363, 364, 426, 427, 436.)

[GKP+13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013. DOI: 10.1145/2488608.2488678. (Cited on pages 48, 354, 355, 357.)

[GKVW20] Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 229–259. Springer, Cham, November 2020. DOI: 10.1007/978-3-030-64375-1_9. (Cited on page 416.)

[GKW15] Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 485–502. Springer, Berlin, Heidelberg, August 2015. DOI: 10.1007/978-3-662-48000-7_24. (Cited on page 436.)

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages

361–388. Springer, Berlin, Heidelberg, October / November 2016. DOI: 10.1007/978-3-662-53644-5_14. (Cited on pages 291, 355, 358, 365, 415.)

[GOS18] Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 515–544. Springer, Cham, August 2018. DOI: 10.1007/978-3-319-96878-0_18. (Cited on pages 434, 451, 458, 466.)

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 736–749. ACM Press, June 2021. DOI: 10.1145/3406325.3451070. (Cited on page 359.)

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. DOI: 10.1145/1180405.1180418. (Cited on pages 1, 25, 220, 353, 429.)

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 419–442. Springer, Berlin, Heidelberg, October / November 2016. DOI: 10.1007/978-3-662-53644-5_16. (Cited on page 435.)

[GS18a] Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565. Springer, Cham, April / May 2018. DOI: 10.1007/978-3-319-78375-8_18. (Cited on pages 434, 451, 458, 462, 463, 464, 467.)

[GS18b] Sanjam Garg and Akshayaram Srinivasan. A simple construction of iO for Turing machines. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 425–454. Springer, Cham, November 2018. DOI: 10.1007/978-3-030-03810-6_16. (Cited on pages 428, 434, 435.)

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Berlin, Heidelberg, August 2013. DOI: 10.1007/978-3-642-40041-4_5. (Cited on pages 213, 356, 359, 370, 371, 379, 383.)

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Berlin, Heidelberg, November / December 2015. DOI: 10.1007/978-3-662-48797-6_23. (Cited on pages 25, 209, 213, 214, 221, 245.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Berlin, Heidelberg, August 2012. DOI: 10.1007/978-3-642-32009-5_11. (Cited on pages 199, 355.)

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013. DOI: 10.1145/2488608.2488677. (Cited on pages 19, 25, 27, 199, 255, 354, 360, 362, 363, 364, 435.)

[GVW15a] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B.

Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Berlin, Heidelberg, August 2015. DOI: 10.1007/978-3-662-48000-7_25. (Cited on pages 354, 416.)

[GVW15b] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015. DOI: 10.1145/2746539.2746576. (Cited on page 354.)

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. DOI: 10.1145/1993636.1993651. (Cited on page 447.)

[GW20a] Junqing Gong and Hoeteck Wee. Adaptively secure ABE for DFA from k-Lin and more. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 278–308. Springer, Cham, May 2020. DOI: 10.1007/978-3-030-45727-3_10. (Cited on pages 49, 133, 140, 151, 264.)

[GW20b] Junqing Gong and Hoeteck Wee. Adaptively secure ABE for DFA from k-lin and more. Cryptology ePrint Archive, Report 2020/194, 2020. URL: https://eprint.iacr.org/2020/194. (Cited on page 193.)

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. ABE for DFA from k-Lin. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 732–764. Springer, Cham, August 2019. DOI: 10.1007/978-3-030-26951-7_25. (Cited on pages 25, 28, 30, 48, 264.)

[GWZ22] Jiaxin Guan, Daniel Wichs, and Mark Zhandry. Incompressible cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 700–730. Springer,

Cham, May / June 2022. DOI: 10.1007/978-3-031-06944-4_24. (Cited on pages 364, 427, 435.)

[HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying iO. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 673–700, Virtual Event, August 2021. Springer, Cham. DOI: 10.1007/978-3-030-84245-1_23. (Cited on pages 256, 359.)

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Berlin, Heidelberg, August 2016. DOI: 10.1007/978-3-662-53015-3_6. (Cited on page 434.)

[HKS15] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 799–822. Springer, Berlin, Heidelberg, March / April 2015. DOI: 10.1007/978-3-662-46447-2_36. (Cited on page 48.)

[HLL23a] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023. DOI: 10.1109/FOCS57990.2023.00031. (Cited on pages 353, 373, 390, 421.)

[HLL23b] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices: Garbled circuits of optimal size, laconic functional evaluation, and more. Cryptology ePrint Archive, Report 2023/1716, 2023. URL: https://eprint.iacr.org/2023/1716. (Cited on pages 353, 373, 377, 421.)

[HLL24a] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. A general framework for lattice-based ABE using evasive inner-product functional encryption. In Marc

Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 433–464. Springer, Cham, May 2024. DOI: 10.1007/978-3-031-58723-8_15. (Cited on pages 255, 302, 304, 308, 311, 321.)

[HLL24b] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. A general framework for lattice-based ABE using evasive inner-product functional encryption. Cryptology ePrint Archive, Report 2024/821, 2024. URL: https://eprint.iacr.org/2024/821. (Cited on page 255.)

[HLL25] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices: Garbled circuits of optimal size, laconic functional evaluation, and more. *SIAM Journal on Computing*, 0(0):FOCS23-90–FOCS23-146, 2025. Published electronically on 17 April 2025. DOI: 10.1137/24M1629316. (Cited on pages 353, 373, 421.)

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 174–183, 1997. DOI: 10.1109/ISTCS.1997.595170. (Cited on page 52.)

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000. DOI: 10.1109/SFCS.2000.892118. (Cited on pages 52, 64.)

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Berlin, Heidelberg, July 2002. DOI: 10.1007/3-540-45465-9_22. (Cited on pages 31, 52, 53, 64.)

[Iva82] A. G. Ivanov. Theorems on the time hierarchy for random access machines. *Journal of Soviet Mathematics*, 20(4):2299–2304, 1982. DOI: 10.1007/BF01629438. (Cited on page 426.)

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Berlin, Heidelberg, July 2014. DOI: 10.1007/978-3-662-43948-7_54. (Cited on pages 20, 25, 27, 30, 31, 34, 35, 52, 59, 60, 61, 64, 65, 67, 90, 142, 304, 311.)

[JKK+17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Cham, August 2017. DOI: 10.1007/978-3-319-63688-7_5. (Cited on page 29.)

[JLL22] Aayush Jain, Huijia Lin, and Ji Luo. On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. Cryptology ePrint Archive, Report 2022/1317, 2022. URL: https://eprint.iacr.org/2022/1317. (Cited on pages 423, 433, 434, 435, 437, 439, 446, 458, 475, 492, 493.)

[JLL23] Aayush Jain, Huijia Lin, and Ji Luo. On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 479–510. Springer, Cham, April 2023. DOI: 10.1007/978-3-031-30620-4_16. (Cited on pages 423, 435, 439, 492.)

[JLLS23] Aayush Jain, Huijia Lin, Paul Lou, and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum $i\mathcal{O}$. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 205–235. Springer, Cham, April 2023. DOI: 10.1007/978-3-031-30545-0_8. (Cited on pages 256, 359.)

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021. DOI: 10.1145/3406325.3451093. (Cited on pages 354, 423, 426, 435.)

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Cham, May / June 2022. DOI: 10.1007/978-3-031-06944-4_23. (Cited on pages 354, 423, 426, 435.)

[JP14] Dimitar Jetchev and Krzysztof Pietrzak. How to fake auxiliary input. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 566–590. Springer, Berlin, Heidelberg, February 2014. DOI: 10.1007/978-3-642-54242-8_24. (Cited on pages 202, 207.)

[KLM+18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 544–562. Springer, Cham, September 2018. DOI: 10.1007/978-3-319-98113-0_29. (Cited on page 30.)

[KLMM19] Lucas Kowalczyk, Jiahui Liu, Tal Malkin, and Kailash Meiyappan. Mitigating the one-use restriction in attribute-based encryption. In Kwangsu Lee, editor, *ICISC 18*, volume 11396 of *LNCS*, pages 23–36. Springer, Cham, November 2019. DOI: 10.1007/978-3-030-12146-4_2. (Cited on page 27.)

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for Turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015. DOI: 10.1145/2746539.2746614. (Cited on pages 28, 428, 435.)

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 187–217. Springer, Cham, March 2018. DOI: 10.1007/978-3-319-76581-5_7. (Cited on page 435.)

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Cham, August 2019. DOI: 10.1007/978-3-030-26954-8_17. (Cited on pages 27, 28, 48, 197, 199, 364, 427, 435, 445, 450, 451, 458.)

[KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Journal of Cryptology*, 26(2):191–224, April 2013. DOI: 10.1007/s00145-012-9119-4. (Cited on page 31.)

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of Structures in Complexity Theory*, pages 102–111. IEEE, May 1993. DOI: 10.1109/SCT.1993.336536. (Cited on pages 135, 221, 259, 266.)

[KW19] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Cham, May 2019. DOI: 10.1007/978-3-030-17653-2_1. (Cited on pages 25, 26, 29, 30, 35, 140, 141, 151, 172, 193, 220.)

[KW20] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. *Journal of Cryptology*, 33(3):954–1002, July 2020. DOI: 10.1007/s00145-019-09335-x. (Cited on pages 192, 193, 194.)

[KY09] Jonathan Katz and Arkady Yerukhimovich. On black-box constructions of predicate encryption from trapdoor permutations. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 197–213. Springer, Berlin, Heidelberg, December 2009. DOI: 10.1007/978-3-642-10366-7_12. (Cited on page 436.)

[KYDB98] Kaoru Kurosawa, Takuya Yoshida, Yvo Desmedt, and Mike Burmester. Some bounds and a construction for secure broadcast encryption. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 420–433. Springer, Berlin, Heidelberg, October 1998. DOI: 10.1007/3-540-49649-1_33. (Cited on page 436.)

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Cham, August 2017. DOI: 10.1007/978-3-319-63688-7_20. (Cited on pages 30, 31, 37, 56, 58, 135.)

[LL20a] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 247–277. Springer, Cham, May 2020. DOI: 10.1007/978-3-030-45727-3_9. (Cited on pages 25, 33, 62, 134, 155.)

[LL20b] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC1 and towards NL. Cryptology ePrint Archive, Report 2020/318, 2020. URL: https://eprint.iacr.org/2020/318. (Cited on pages 25, 33, 62, 134, 135, 155.)

[LL20c] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-Lin. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 437–466. Springer, Cham, December 2020. DOI: 10.1007/978-3-030-64840-4_15. (Cited on pages 137, 190.)

[LL20d] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-Lin. Cryptology ePrint Archive, Report 2020/1139, 2020. URL: https://eprint.iacr.org/2020/1139. (Cited on pages 137, 190.)

[LLL22a] Hanjun Li, Huijia Lin, and Ji Luo. ABE for circuits with constant-size secret keys and adaptive security. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 680–710. Springer, Cham, November 2022. DOI: 10.1007/978-3-031-22318-1_24. (Cited on pages 195, 202, 207, 224, 225, 227, 229, 247.)

[LLL22b] Hanjun Li, Huijia Lin, and Ji Luo. ABE for circuits with constant-size secret keys and adaptive security. Cryptology ePrint Archive, Report 2022/659, 2022. URL: https://eprint.iacr.org/2022/659. (Cited on page 195.)

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 443–468. Springer, Berlin, Heidelberg, October / November 2016. DOI: 10.1007/978-3-662-53644-5_17. (Cited on page 435.)

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 595–608. ACM Press, June 2023. DOI: 10.1145/3564246.3585175. (Cited on page 493.)

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Berlin, Heidelberg, May / June 2010. DOI: 10.1007/978-3-642-13190-5_4. (Cited on pages 25, 29, 220.)

[LS98] Michael Luby and Jessica Staddon. Combinatorial bounds for broadcast encryption. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 512–526. Springer, Berlin, Heidelberg, May / June 1998. DOI: 10.1007/BFb0054150. (Cited on page 436.)

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Cham, August 2017. DOI: 10.1007/978-3-319-63688-7_21. (Cited on page 448.)

[Luo20] Ji Luo. 属性加密（番外）：分段安全性的故事 (Attribute-based encryption: Another story of piecewise security), 2020. URL: https://adversary.blog/posts/ll20a-pcwssec-more/#thm-akgs-abp-tight. (Cited on page 135.)

[Luo22] Ji Luo. Ad hoc broadcast, trace, and revoke — plus time-space trade-offs for attribute-based encryption. Cryptology ePrint Archive, Report 2022/925, 2022. URL: https://eprint.iacr.org/2022/925. (Cited on page 423.)

[Luo24] Ji Luo. Ad hoc broadcast, trace, and revoke: Plus time-space trade-offs for attribute-based encryption. *CiC*, 1(2):15, 2024. DOI: 10.62056/a39qxrxqi. (Cited on pages 423, 429, 492.)

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016. DOI: 10.1109/FOCS.2016.11. (Cited on pages 31, 32, 37, 56, 58, 135, 140.)

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479.

Springer, Berlin, Heidelberg, February 2010. DOI: 10.1007/978-3-642-11799-2_27. (Cited on pages 48, 56.)

[LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Berlin, Heidelberg, May 2011. DOI: 10.1007/978-3-642-20465-4_30. (Cited on pages 25, 48.)

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Berlin, Heidelberg, August 2012. DOI: 10.1007/978-3-642-32009-5_12. (Cited on pages 25, 29, 220.)

[LZ17] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 138–169. Springer, Cham, November 2017. DOI: 10.1007/978-3-319-70500-2_6. (Cited on pages 448, 451.)

[MP11] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. Cryptology ePrint Archive, Report 2011/501, 2011. URL: https://eprint.iacr.org/2011/501. (Cited on page 14.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. DOI: 10.1007/978-3-642-29011-4_41. (Cited on pages 14, 358.)

[Mul87] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987. DOI: 10.1007/BF02579205. (Cited on pages 221, 259, 266.)

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Berlin, Heidelberg, August 2003. DOI: 10.1007/978-3-540-45146-4_6. (Cited on page 447.)

[Nis91] Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *23rd ACM STOC*, pages 410–418. ACM Press, May 1991. DOI: 10.1145/103418.103462. (Cited on page 52.)

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. URL: https://eprint.iacr.org/2010/556. (Cited on page 353.)

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231. Springer, Berlin, Heidelberg, December 2009. DOI: 10.1007/978-3-642-10366-7_13. (Cited on page 31.)

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Berlin, Heidelberg, August 2010. DOI: 10.1007/978-3-642-14623-7_11. (Cited on pages 25, 31, 220.)

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Berlin, Heidelberg, December 2012. DOI: 10.1007/978-3-642-34961-4_22. (Cited on pages 25, 31, 48.)

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*,

pages 422–439. Springer, Berlin, Heidelberg, March 2012. DOI: 10.1007/978-3-642-28914-9_24. (Cited on page 1.)

[Ps16] Rafael Pass and abhi shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 3–17. Springer, Berlin, Heidelberg, January 2016. DOI: 10.1007/978-3-662-49096-9_1. (Cited on pages 217, 218.)

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018. DOI: 10.1109/FOCS.2018.00086. (Cited on pages 199, 200, 201, 202, 204, 207, 212, 214, 225, 245, 259, 263, 354, 355, 357, 363, 373, 403, 446.)

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–180, 1978. (Cited on page 353.)

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. DOI: 10.1145/1060590.1060603. (Cited on pages 15, 416.)

[RW15] Yannis Rouselakis and Brent Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 315–332. Springer, Berlin, Heidelberg, January 2015. DOI: 10.1007/978-3-662-47854-7_19. (Cited on page 11.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. DOI: 10.1145/1866307.1866359. (Cited on page 355.)

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, Berlin, Heidelberg, March 2009. DOI: 10.1007/978-3-642-00457-5_27. (Cited on page 31.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Berlin, Heidelberg, May 2005. DOI: 10.1007/11426639_27. (Cited on pages 1, 353.)

[Tak14] Katsuyuki Takashima. Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 298–317. Springer, Cham, September 2014. DOI: 10.1007/978-3-319-10879-7_17. (Cited on pages 138, 149, 196, 197, 426, 436.)

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In Matt Bishop and Anderson C. A. Nascimento, editors, *ISC 2016*, volume 9866 of *LNCS*, pages 408–425. Springer, Cham, September 2016. DOI: 10.1007/978-3-319-45871-7_24. (Cited on page 30.)

[TCH12] Terence Tao, Ernest Croot, and Harald Helfgott. Deterministic methods to find primes. *Mathematics of Computation*, 81(278):1233–1246, 2012. URL: http://www.jstor.org/stable/23267994. (Cited on pages 15, 216, 302.)

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Cham, August 2019. DOI: 10.1007/978-3-030-26948-7_3. (Cited on pages 199, 349.)

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022. DOI: 10.1007/978-3-031-15802-5_19. (Cited on pages 256, 265, 276, 349, 355, 358, 381, 382.)

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Berlin, Heidelberg, August 2007. DOI: 10.1007/978-3-540-74143-5_12. (Cited on pages 430, 455.)

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Cham, December 2022. DOI: 10.1007/978-3-031-22963-3_7. (Cited on pages 256, 276, 359, 361, 381, 382.)

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Berlin, Heidelberg, August 2009. DOI: 10.1007/978-3-642-03356-8_36. (Cited on pages 1, 19, 29, 38, 48, 56, 151.)

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Berlin, Heidelberg, August 2012. DOI: 10.1007/978-3-642-32009-5_14. (Cited on pages 25, 28, 48, 133, 264, 271, 327.)

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Berlin, Heidelberg, August 2015. DOI: 10.1007/978-3-662-48000-7_33. (Cited on page 199.)

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Berlin, Heidelberg, February 2014. DOI: 10.1007/978-3-642-54242-8_26. (Cited on pages 1, 19, 29, 151.)

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Cham, November 2017. DOI: 10.1007/978-3-319-70500-2_8. (Cited on pages 32, 58, 135, 151, 165, 219, 220, 280, 286.)

[Wee21] Hoeteck Wee. ABE for DFA from LWE against bounded collusions, revisited. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 288–309. Springer, Cham, November 2021. DOI: 10.1007/978-3-030-90453-1_10. (Cited on pages 260, 264, 265.)

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Cham, May / June 2022. DOI: 10.1007/978-3-031-07085-3_8. (Cited on pages 4, 256, 259, 265, 276, 282, 349, 355, 357, 358, 359, 360, 363, 381, 382.)

[Wee24] Hoeteck Wee. Circuit ABE with poly(depth, λ)-sized ciphertexts and keys from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 178–209. Springer, Cham, August 2024. DOI: 10.1007/978-3-031-68382-4_6. (Cited on pages 349, 365.)

[Wee25] Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. Cryptology ePrint Archive, Report 2025/509, 2025. URL: https://eprint.iacr.org/2025/509. (Cited on page 349.)

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 127–156. Springer,

Cham, October 2021. DOI: 10.1007/978-3-030-77883-5_5. (Cited on page 359.)

[WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 651–679. Springer, Cham, November 2022. DOI: 10.1007/978-3-031-22318-1_23. (Cited on pages 256, 276, 381, 382.)

[YAHK14] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 275–292. Springer, Berlin, Heidelberg, March 2014. DOI: 10.1007/978-3-642-54631-0_16. (Cited on pages 138, 149, 196, 197, 426, 436.)

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. DOI: 10.1109/SFCS.1982.38. (Cited on pages 20, 451.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. DOI: 10.1109/SFCS.1986.25. (Cited on pages 20, 31.)

[Yao90] Andrew Chi-Chih Yao. Coherent functions and program checkers (extended abstract). In *22nd ACM STOC*, pages 84–94. ACM Press, May 1990. DOI: 10.1145/100216.100226. (Cited on page 430.)

[ZGT+16] Kai Zhang, Junqing Gong, Shaohua Tang, Jie Chen, Xiangxue Li, Haifeng Qian, and Zhenfu Cao. Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *ASIACCS 16*, pages 269–279. ACM Press, May / June 2016. DOI: 10.1145/2897845.2897858. (Cited on pages 138, 149, 196, 197, 426, 436.)