Efficient Blind Signatures and Threshold Signatures from Pairing-Free Groups and Lattices

Chenzhi Zhu

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

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Stefano Tessaro, Chair

Huijia Lin, Chair

Nirvan Tyagi

Program Authorized to Offer Degree:

Computer Science & Engineering

University of Washington

Abstract

Efficient Blind Signatures and Threshold Signatures from Pairing-Free Groups and Lattices

Chenzhi Zhu

Co-Chairs of the Supervisory Committee:
Stefano Tessaro
Department of Computer Science & Engineering

Huijia Lin
Department of Computer Science & Engineering

Blind signatures and threshold signatures are two fundamental cryptographic primitives that have attracted significant real-world interest, largely due to the development of practically efficient schemes. This thesis focuses on designing efficient constructions in pairing-free groups and lattices, along with rigorous security analyses. Pairing-free constructions are attractive because of their compact key/signature sizes and well-established library support, while lattice-based constructions are of particular interest as they are conjectured to remain secure against quantum adversaries. The thesis makes progress in the following three directions:

Pairing-free blind signatures: The thesis proposes the most efficient pairing-free construction that is provably secure in the concurrent setting, under the discrete logarithm (DL) assumption in the algebraic group model (AGM) and the random oracle model (ROM).

Pairing-free threshold signatures: This thesis introduces a new syntax and security hierarchy for analyzing FROST, the state-of-the-art pairing-free threshold signature scheme. It provides the first security proof of FROST in the ROM under the one-more discrete logarithm (OMDL) assumption, and further proposes variants of FROST that is provably secure under the standard DL assumption. Those are the first partially non-interactive constructions based on the DL assumption.

Lattice-based threshold signatures: This thesis develops new techniques that establish the provable security of one of the state-of-the-art two-round lattice-based threshold signature schemes under standard lattice assumptions, in contrast to prior work which relied on a new non-standard assumption. These techniques also yield a new security analysis for another state-of-the-art two-round scheme with a simpler setup, significantly improving its efficiency.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Stefano Tessaro, for his invaluable guidance throughout my Ph.D. I am especially grateful for his advice on conducting research, writing papers, and giving talks, all of which have been central to my growth as a researcher.

I would also like to thank my co-advisor, Huijia Lin, for her support and feedback, even though we do not have direct research collaborations. I am grateful to Nirvan Tyagi and Gaku Liu for serving on my Ph.D. exam committee.

I would like to thank Ilan Komargodski, who supervised me during my internship at NTT and opened a door into a new research area, even though that work is not included in this thesis. I am also thankful to Melissa Chase and Esha Ghosh for supervising me during my internship at Microsoft Research. I am further grateful to Vipul Goyal, who hosted me during my undergraduate visit to CMU and first introduced me to research in cryptography.

I would like to thank my co-author Mihir Bellare for the opportunity to collaborate and for our insightful discussions on threshold signatures included in this thesis. I am also grateful to my co-author Rutchathon Chairattana-Apirom (Champ) for many useful discussions on lattice-based constructions and secret sharing schemes. I would also like to thank all my other co-authors for their effort, stimulating conversations, and support in preparing talks.

Special thanks go to the other members of the UW cryptography group—Binyi Chen, Pratik Soni, Ashrujit Ghoshal, Ji Luo, Xihu Zhang, Anna Kornfeld Simpson, Tianren Liu, Hanjun Li, Marshall Ball, Yao-Ching Hseih, Marian Dietz, and Andrea Coladangelo—for creating a positive and collaborative atmosphere, organizing engaging events, and supporting me through the difficult times of my Ph.D. journey.

I am thankful to the UW CSE advising team for their assistance with countless matters that made my Ph.D. journey much smoother.

Finally, I owe heartfelt thanks to my family and friends for their unwavering support and encouragement.

Chapter 1

INTRODUCTION

Digital signatures are fundamental cryptographic primitives widely used across various applications. They play a crucial role in authenticating transactions and communications. While many efficient signature schemes exist with fast signing and verification, emerging scenarios demand more advanced protocols for generating digital signatures. Among these, two key areas of focus are blind signatures and threshold signatures.

Blind signature protocols, first introduced by David Chaum in 1982 [43], allow users to obtain signatures from a signer without revealing either the messages being signed or the resulting signatures to the signer. More precisely, their security consists of two key properties: *blindness*, which ensures that the issued messages and signatures remain unlinkable to the signing interaction—even if the signer is malicious, and *one-more unforgeability*, which prevents a malicious user from forging more signatures than the number of signing interactions they engage in. Blind signatures are highly versatile and essential for applications requiring anonymity guarantees. Initially proposed for anonymous cash systems [43, 46, 108, 35], they were later adapted for other privacy-preserving applications, such as anonymous e-voting [45, 66] and anonymous credentials [35, 36, 37, 13]. Furthermore, thanks to the existence of efficient constructions [44, 27, 57], blind signatures have been deployed in various real-world applications, including Private Click Measurement [1], Google's VPN service, and Privacy Pass [52].

Threshold signatures [58, 59] enable the signing key to be distributed among a group of n signers, allowing a valid signature to be issued only if at least a threshold number t of signers are involved. Crucially, any subset of fewer than t signers should be unable to produce a valid signature, even if they collude and deviate from the protocol. The primary motivation for developing practically efficient threshold signatures comes from blockchain systems (e.g., digital wallets [70]), where they are used to mitigate the risk of a single point of failure in key management. Additionally, threshold signatures have recently gained significant attention due to ongoing standardization efforts by NIST [102].

MAIN FOCUSES. The primary focus of this thesis is on constructing efficient cryptographic schemes in pairing-free groups. These constructions treat prime-order cyclic groups as black boxes and do not rely on bilinear pairings. In practice, such groups are typically instantiated using standard elliptic curves. By "efficient," we refer to constructions that avoid heavy cryptographic tools, such as general-purpose non-interactive zero-knowledge proofs.

While efficient constructions based on pairings or RSA exist for blind signatures [44, 27, 57] and threshold signatures [53, 75, 119, 27], pairing-free constructions are preferable for practical applications for the following reasons. Pairing-based constructions suffer from limited library support, making them difficult to deploy in browser applications. RSA-based constructions are generally less efficient than pairing-free alternatives in terms of signature size, signing time, verification time, and communication complexity. This inefficiency arises because RSA-based schemes require significantly larger modulus sizes to ensure the hardness of the RSA problem.

Another motivation for studying pairing-free constructions is that they pave the way for building efficient post-quantum schemes, which is the second focus of this thesis. As the threat of quantum computers looms on the horizon, there is growing interest in developing cryptosystems based on quantum-resistant assumptions. Lattice-based assumptions are among the most promising candidates. The selection of Dilithium [99] and Falcon [111] by NIST for standardization reinforces this belief. A common approach for designing efficient lattice-based schemes is to adapt efficient pairing-free constructions. For example, Dilithium follows the "Fiat–Shamir with Aborts" paradigm [97], which is based on this approach. This paradigm has also led to practical lattice-based schemes in multi-signatures [34] and threshold signatures [54].

SCHNORR SIGNATURES. Most of pairing-free constructions for blind signatures and threshold signatures are all built around Schnorr signatures [115], which is one of the most efficient digital signature schemes in paring-free groups. Denote a group as $\mathbb{G}$ with generator g and order p. The signing key of Schnorr signature is a random scalar $\mathsf{sk} \in \mathbb{Z}_p$ and the corresponding public key is $\mathsf{pk} \leftarrow g^{\mathsf{sk}}$. A Schnorr signature for a message m consists of one group element and one scalar $(R, z) \in \mathbb{G} \times \mathbb{Z}_p$ satisfying

$$g^z = R \cdot \mathsf{pk}^c \,, \tag{1.1}$$

with $c = H(\mathsf{pk}, R, m)$ and H is a cryptographic hash function. We can also use (c, z) as a Schnorr signature since one can compute c from R and also compute $R (= g^z \mathsf{pk}^{-c})$ from c. Then, a Schnorr signature consists of two scalars in $\mathbb{Z}_p$. Schnorr signatures are now supported by major blockchain

systems, including Bitcoin. Pointcheval and Stern [110] shows the security of Schnorr signatures under the discrete logarithm (DL) assumption in the random oracle model (ROM) [18], where H is modeled as a random oracle.

PAIRING-FREE BLIND SIGNATURES. Chaum and Pedersen [47] proposed a highly efficient three-move blind signature scheme for issuing Schnorr signatures, commonly referred to as blind Schnorr. Here, "three-move" means that each signing session consists of three messages exchanged between the signer and the user, with the signer sending the initial message and the user receiving the final message.

However, Schnorr [117] identified a hard problem, called the ROS problem, in the security analysis of blind Schnorr. Specifically, if an adversary can solve the ROS problem, they can break the one-more unforgeability of blind Schnorr in the concurrent setting. The concurrent setting means that there can be arbitrarily many signing sessions happening concurrently with communication from different sessions interleaved arbitrarily. In contrast, the sequential setting requires the signer to complete one signing session before starting another. Consider ring the concurrent setting is crucial in practice, as restricting a scheme to sequential signing makes it vulnerable to denial-of-service (DoS) attacks.

A recent break-through result by Benhamouda et al. [24] shows that there exists a polynomial-time algorithm that solves the ROS problem. Concretely, the attack shows that one can forge $\lceil \log_2(p) \rceil + 1$ valid signatures by engaging in only $\lceil \log_2(p) \rceil$ number of signing sessions, where p denotes the group size. Several attempts [3, 65, 88, 39] have been made to circumvent the ROS problem. However, these solutions either rely on non-standard assumptions [65] or are significantly less efficient compared to blind Schnorr [3, 88, 39]. Therefore, it remains a major open problem to construct an efficient, pairing-free blind signature scheme that is provably secure under standard assumptions.

PAIRING-FREE THRESHOLD SIGNATURES. Currently, the most efficient pairing-free threshold signature scheme is FROST, proposed by Komlo and Goldberg [91]. FROST issues Schnorr signatures and has a two-round signing protocol. More precisely, each signer communicates with a coordinator (which can be one of the signers). In the first round, each signer generates a random token and sends it to the coordinator. In the second round, the coordinator distributes the collected tokens to all signers, who then compute and send partial signatures back to the coordinator. Finally, the coordinator combines them to produce a valid signature.

FROST improves upon prior constructions [120, 72] in terms of both communication and round

complexity at the price of giving up robustness. Additionally, it offers significant usability advantages. The first-round messages are generated independently of both the message being signed and the set of participating signers. This allows the first round to be preprocessed, where the coordinator can collect a batch of tokens in advance. When receiving a signing request for a message m with a designated subset of signers, the coordinator simply selects a token for each designated signer and proceeds with the second-round protocol to generate the final signature.

However, there is no formal security analysis for FROST. The original paper does not provide a formal security definition or proof for the scheme. Moreover, we found that existing syntax and security definitions fail to capture the aforementioned offline-online signing pattern. Prior security definitions, such as the one given in [70], only consider signing protocols where the message to be signed is determined at the start of the execution.

Another issue with prior definitions is that they assume all honest designated signers participate in every signing session. However, in FROST, a malicious coordinator may choose to interact with only a subset of the honest designated signers, and it is unclear whether a signature is considered issued or not in this case. Definitions that account for these more fine-grained signing patterns exist only for non-interactive threshold signatures [119, 124, 29], where the signing protocol consists of a single round of communication between the user and each designated signer. It is unclear how we can define similar security notions for FROST. Also, these definitions are inconsistent on whether a signature is consider issued. The definitions from [124, 29] consider a signature is issued for a message m as long as the adversary receives a partial signature from at least one honest signers, while the definition given by Shoup [119] is stronger, where they consider a signature is issued only if the adversary receives partial signatures from at least $t - k$ honest signers, where k denotes the number of corrupted parties.

LATTICE-BASED CONSTRUCTIONS. For lattice-based constructions, we only focus on threshold signatures in this thesis. The existing lattice-based threshold signatures are not yet satisfactory. Early constructions were based on fully homomorphic encryption (FHE) [29, 8], a heavy cryptographic tool that renders these schemes impractical. A more recent line of work [54, 87, 61, 60] has made significant progress by adapting constructions from pairing-free groups to the lattice setting. In particular, these schemes produce lattice analogs of Schnorr signatures. The high-level idea is to replace the group exponentiation g^x with matrix multiplication $A\boldsymbol{x}$, where A is a random matrix over $\mathbb{Z}_p$ for some integer p and $\boldsymbol{x}$ is a small-norm vector in $\mathbb{Z}_p$. The secret key sk of a lattice-based signature scheme is a random small-norm vector in $\mathbb{Z}_p^m$, and its corresponding public key is com-

puted as $\mathsf{pk} \leftarrow A\mathsf{sk}$, with $A \in \mathbb{Z}_p^{k \times m}$ being part of the public parameter. A signature of a message μ consists of $(\boldsymbol{R} \in \mathbb{Z}_p^k, \boldsymbol{z} \in \mathbb{Z}_p^m)$ satisfying $A\boldsymbol{z} = \boldsymbol{R} + \mathrm{H}(\mathsf{pk}, \boldsymbol{R}, \mu) \cdot \mathsf{pk}$ and $\boldsymbol{z}$ has small norm, where H is a cryptographic hash function.

Despite this progress, it remains an open problem whether we can construct an efficient 2-round lattice-based analog of FROST that is provably secure under standard lattice assumptions. Existing approaches either require a higher number of rounds [54, 87, 61], relies on non-standard assumptions [60] or has significantly larger signature size and communication complexity [41].

OUR CONTRIBUTIONS.

Blind signatures in paring-free groups. We present the first blind signature schemes in pairing-free groups that simultaneously achieve: (1) concurrent security, (2) provable security in either the generic group model (GGM) [104, 118] or the algebraic group model (AGM) [64] under the DL assumption additionally assuming random oracles, and (3) efficiency comparable to blind Schnorr signatures.

Our GGM-based construction (BS_1) produces signatures of size 3 scalars—just one more than a standard Schnorr signature—and requires 2 group elements and 3 scalars in communication. Our AGM-based construction (BS_2) has the same signature size and adds one additional scalar to the communication.

Previously, the only scheme satisfying the first two properties was due to Abe [3, 85], but it is significantly less efficient: Abe's signatures consist of 2 group elements and 6 scalars, with a communication cost of 3 group elements, 6 scalars, and an additional κ bits, where κ denotes the security parameter.

Moreover, we show that BS_2 can be naturally extended to support partially blind signing without incurring any additional cost in signature size or communication complexity. In this setting, a portion of the message to be signed is publicly known to both the signer and the user. This feature is particularly useful in real-world applications, as it allows binding public information, such as the date of issuance, to the signature. This part of the contributions is presented in Part I.

Threshold signatures in paring-free groups. In this thesis, we formalize a notion called partially non-interactive threshold signatures, which captures schemes like FROST, and develop a security hierarchy for partially non-interactive schemes. These schemes consist of a message-

independent pre-processing round followed by a non-interactive signing round. In the pre-processing round, each signer generates a token and sends it to a leader (coordinator). In the signing round, the leader constructs a leader request lr, which includes the message m to be signed, the set of participating signers S, and an aggregated message derived from the signers' tokens. For FROST, the aggregated message is $\{(i, R_i, S_i)\}_{i \in S}$. The leader then sends lr to the involved signers, who respond with their partial signatures. Finally, the leader aggregates these partial signatures to produce the final signature.

In our unforgeability game, the adversary can corrupt the leader and up to $t-1$ signers. It also has access to two oracles: a pre-processing oracle, which allows the adversary to request pre-processing tokens from any honest party, and a signing oracle, which allows the adversary to request partial signatures from any honest party on any leader request. The adversary wins if it can forge a signature that is not "considered issued". We propose different ways of defining "considered issued", leading to a security hierarchy TS-UF-0 to TS-UF-4, where TS-UF-$(i+1)$ is stronger than TS-UF-i.

The weakest security, TS-UF-0, considers a signature for a message m to be issued if and only if the adversary has obtained any partial signature from an honest party on a leader request for signing message m. This is exactly the security notion considered by the prior work on FROST [50]. TS-UF-1 considers a signature for a message m to be issued if and only if the adversary has obtained partial signature from at least $t - |CS|$ honest signers for signing message m, where CS denotes the set of corrupted signers. A similar notion is considered in the case of non-interactive threshold signatures [119, 94].

For partially non-interactive schemes, it is possible to consider stronger security notions. The stronger notions in the hierarchy aim to guarantee that the partial signatures obtained for different leader requests cannot be combined. TS-UF-2 considers a signature for m to be issued if and only if there exists a leader request lr for signing m such that the adversary has obtained partial signatures for lr from at least $t - |CS|$ signers. The condition of an issued signature in TS-UF-3 additionally requires the adversary to obtain partial signatures from each honest signer $i \in lr.S$ that the token (R_i, S_i) in lr is generated by i. The strongest security in our hierarchy, TS-UF-4, considers a signature for a message m to be issued if and only if there exists a leader request lr for signing m such that the adversary has obtained partial signatures from all honest signers involved in $lr.S$.

We also analyze FROST1 (the original FROST [91]) and two variants FROST2 [50] and FROST3 [113], using our security hierarchy. In particular, we show TS-UF-3 of FROST1, TS-UF-2 of FROST2, and TS-UF-1 of FROST3 under the AOMDL assumption [105] in the ROM. We also give counterexamples showing that they do not guarantee stronger notions in the hierarchy.

Our second main contribution here is a new approach that can transform FROST into threshold signature schemes whose security relies on weaker assumptions. Specifically, while all three FROST variants rely on the AOMDL assumption in the ROM, our new schemes are provably secure under the plain DL assumption in the ROM. The resulting schemes produce Okamoto signatures [107], which are just one scalar larger than Schnorr signatures. This part of the contributions is presented in Part II.

Lattice-based threshold signatures. We establish the security of the 2-round lattice-based threshold signatures proposed by Espitau et al. [60] (referred to as the EKT scheme) under the MSIS and MLWE assumptions in the ROM. Previously, the security of EKT relied on the hardness of the algebraic one-more MLWE (AOM-MLWE) problem, a new assumption introduced in their work.

To achieves this, we propose a modification of the AOM-MLWE problem, referred to as the algebraic one-more MISIS problem (AOM-MISIS) problem. The new problem is no harder than the original problem, meaning that the hardness of AOM-MISIS implies the hardness of AOM-MLWE. We then show the hardness of the AOM-MISIS problem is implied by the MSIS and MLWE assumptions. The reduction is inspired by our techinques for removing the AOMDL assumption.

We also present a new security analysis of EKT directly based on the hardness of the AOM-MISIS problem, which yields a tighter reduction. The concrete parameters derived from our reduction is slightly weaker than those in their work, as their estimates are based their cryptanalysis of the AOM-MLWE problem rather than security reductions.

Our techniques also lead to a new security analysis of another efficient 2-round lattice-based threshold scheme [41] (referred to as the CATZ scheme), which significantly improves its concrete parameters. Compared to EKT, the main advantage of CATZ is that it avoids the need for setting up pairwise secret keys between signers during distributed key generation.

Instead, similar to FROST, each signer only holds its own share of the secret key. This part of the contributions is presented in Part III.

Chapter 2

NOTATION

INTEGERS AND VECTORS. If $b \geqslant a \geqslant 1$ are positive integers, then $\mathbb{Z}_a$ denotes the set $\{0, \ldots, a-1\}$ and $[a..b]$ denotes the set $\{a, \ldots, b\}$. We write $\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\}$. If $\boldsymbol{x}$ is a vector then $|\boldsymbol{x}|$ is its length (the number of its coordinates), x_i is its i-th coordinate. For a sequence of variables $x_1, \ldots, x_\ell$, we use $x_{[i]}$ to denote $(x_1, \ldots, x_j)$ and $x_{[i..j]}$ to denote $(x_i, \ldots, x_j)$.

SETS AND DISTRIBUTIONS. Let S be a finite set. The size of a set S is denoted $|S|$. We let $x \leftarrow\!\!\$\, S$ denote sampling an element uniformly at random from S and assigning it to x. For a distribution $\mathcal{D}$, denote $\mathrm{Supp}(\mathcal{D})$ as the support of $\mathcal{D}$, and $x \leftarrow\!\!\$\, \mathcal{D}$ denotes sampling x according to $\mathcal{D}$. For any $y \in \mathrm{Supp}(\mathcal{D})$, denote $\mathcal{D}(y) := \mathsf{Pr}_{x \leftarrow\!\!\$\, \mathcal{D}}[x = y]$, and for $S \subseteq \mathrm{Supp}(\mathcal{D})$, denote $\mathcal{D}(S) := \mathsf{Pr}_{x \leftarrow\!\!\$\, \mathcal{D}}[x \in S]$. For any set T and any function $F : \mathrm{Supp}(\mathcal{D}) \to T$, denote $F(\mathcal{D})$ as the distribution of $F(x)$ for x sampled from $\mathcal{D}$.

VECTOR SPACES. For any vector space V over a field F and a set $S \in V$, we denote $\mathsf{Span}_F(S)$ as the F-span of S, which is the smallest F-subspace of V that contains S. In particular, we omit F from the subscript if $F = \mathbb{R}$. For a finite set $S = \{v_1, \ldots, v_n\} \subseteq V$, we say S is F-linearly independent if and only if for any non-zero $(a_1, \ldots, a_n) \in F^n$, $\sum_{i \in [n]} a_i v_i \neq 0$. We say S is a F-basis of V if and only if S is F-linearly independent and $\mathsf{Span}_F(S) = V$. When F is not specified, we assume $F = \mathbb{R}$. The dimension of V is equal to the size of S.

ALGORITHMS. We let $y \leftarrow A^{\mathsf{O}_1, \cdots}(x_1, \ldots; r)$ denote executing algorithm A on inputs $x_1, \ldots$ and coins r with access to oracles $\mathsf{O}_1, \ldots$ and letting y be the result. We let $y \leftarrow\!\!\$\, A^{\mathsf{O}_1, \cdots}(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A^{\mathsf{O}_1, \cdots}(x_1, \ldots; r)$. Algorithms are randomized unless otherwise indicated. The running time is the worst case. We use κ to denote the security parameter.

GROUPS. The group generation algorithm $\mathsf{GGen}(1^\kappa)$ takes the security parameter as input and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group with prime size $p \geqslant 2^\kappa$ and generator g. We let $1_{\mathbb{G}}$ denote the identity element of $\mathbb{G}$. We let $\mathbb{G}^* = \mathbb{G} \backslash \{1_{\mathbb{G}}\}$ denote the set of non-identity elements, which is the set of generators of $\mathbb{G}$ if the latter has prime order. If $g \in \mathbb{G}^*$ is a generator and $X \in \mathbb{G}$, the discrete logarithm base g of X is denoted $\mathrm{DLog}_{\mathbb{G},g}(X)$, and it is in the set $\mathbb{Z}_{|\mathbb{G}|}$. The

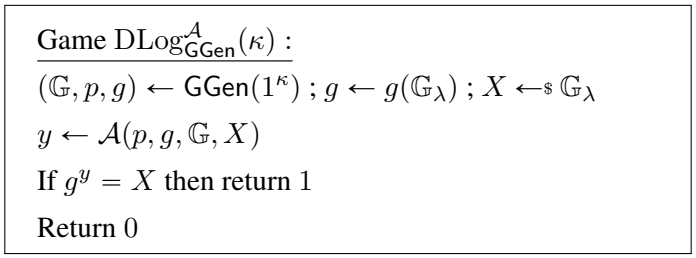

Figure 2.1: The DLog game.

---

discrete logarithm game is defined in Figure 2.1. For a DL adversary $\mathcal{A}$, its advantage is denoted as $\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{A}, 1^\kappa) := \mathsf{Pr}[\mathrm{DLog}^{\mathcal{A}}_{\mathsf{GGen}}(\kappa) = 1]$.

MODULES. For any ring R with multiplicative identity 1 and any abelian group $(M, +)$, we say M is an R-module if there exists an operation $\cdot : R \times M \to M$ such that for any $a, b \in R$ and any $x, y \in M$, (i) $a \cdot (x + y) = a \cdot x + a \cdot y$, (ii) $(a + b) \cdot x = a \cdot x + b \cdot x$, (iii) $(ab) \cdot x = a \cdot (b \cdot x)$, (iv) $1 \cdot x = x$. Also, we use 0 to denote the identity of M.

MODULE HOMOMORPHISMS. For any R-modules M and N, a map $f : M \to N$ is a homomorphism of R-modules if for any $r \in R$ and $x, y \in M$, $f(x + r \cdot y) = f(x) + r \cdot f(y)$. We say a homomorphism f is an epimorphism if f is a surjection. We say a homomorphism f is a monomorphism if f is an injection.

CHARACTERISTIC OF A FIELD. For any field $\mathbb{F}$, the characteristic of $\mathbb{F}$, denoted by $\mathrm{char}(\mathbb{F})$, is the smallest positive number k such that $k \cdot \mathbf{1} = \sum_{i=1}^{k} \mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ denotes the multiplicative identity of $\mathbb{F}$ and $\mathbf{0}$ denotes the additive identity of $\mathbb{F}$. If k does not exist, we say the characteristic of F is 0.

Part I

BLIND SIGNATURES IN PAIRING-FREE GROUPS

Chapter 3

BLIND SIGNATURES: INTRODUCTION

Blind signatures [42] allow a *user* to interact with a *signer* to produce a valid signature that cannot be linked back by the signer to the interaction that produced it. Blind signatures are used in several applications, such as e-cash systems [42, 46], anonymous credentials (e.g., [37]), privacy-preserving ad-click measurement [1], and various forms of anonymous tokens [82, 2]. They are also covered by an RFC draft [56].

This thesis develops the first practical pairing-free three-move blind signature schemes that (1) are concurrently secure, (2) produce short signatures (i.e., *three* or *four* group elements/scalars), and (3) are provably secure either in the *generic group model* (GGM) [118, 101] or in the *algebraic group model* (AGM) [64] under the discrete logarithm (DL) or the one-more discrete logarithm (OMDL) assumption (in addition to assuming *random oracles* [18]). Our DL-based scheme also admits a *partially blind* version [4], roughly following a paradigm by Abe and Okamoto [5], that targets applications where signatures need to depend on some public input (e.g., an issuing date) known to the signer. An overview of our schemes is given in Table 3.1.

Unlike blind Schnorr [47], Okamoto-Schnorr [110], and other generic constructions based on identification schemes [80], we do not rely on the hardness of the ROS problem, for which a polynomial-time attack has recently been presented [24]. Also, unlike Clause Blind Schnorr (CBS) signatures [65], we do not rely on the assumed hardness of the mROS problem, which is subject to (mildly) sub-exponential attacks and we can thus support smaller group sizes.[1] In fact, our schemes all admit tight bounds, and this suggests that they can achieve $(\lambda/2)$-bit of security on λ-bit elliptic curves, supporting an instantiation with 256-bit curves. Our security proofs rely on a reduction to a new variant of the ROS problem, called *weighted fractional* ROS (WFROS), for which we prove an exponential, unconditional lower bound. Therefore, another benefit over CBS, beyond concrete parameters, is that we do not need to rely on an additional assumption.

[1] The best known attack against mROS [65] runs in time $2^{\ell+\log(\ell+1)+\lambda/(1+\log(\ell+1))}$, where λ is the security parameter and ℓ corresponds to the number of concurrent sessions. The worst ℓ gives a $2^{O(\lambda/\log\lambda)}$ attack, and in practice, this suggests a choice of $\lambda = 512$ to achieve 128-bit security for all ℓ's.

Perhaps as a testament of the unsatisfactory status of pairing-free schemes, the *only* other scheme known to achieve exponential, concurrent, security is Abe's scheme [3]. Although its original (standard-model) proof was found to be flawed, proofs were then given both in the GGM [106] and the AGM [85], along with a proof for the restricted setting of sequential security [11]. Still, it produces longer signatures and public keys, and is overall less efficient. Also, it only offers computational blindness (under DDH), whereas our scheme provides perfect blindness.

DISCRETE-LOGARITHM BASED BLIND SIGNATURES. We stress that our focus here is making pairing-free schemes as practical and as secure as possible. Indeed, very simple pairing-based blind signature schemes in the ROM can be obtained from BLS signatures [31, 27]. Blind BLS offers a different trade-off: signatures are short (i.e., one group element) and signing requires only *two* moves, but signature verification requires a more expensive (and more complex) pairing evaluation. Indeed, the current blind signature RFC draft [56] favors RSA over BLS, also due to lesser availability of pairings implementations. In particular, several envisioned applications of blind signatures are inherently browser-based, and the available cryptographic libraries (e.g., NSS for Firefox and BoringSSL for Chrome) do not yet offer pairing-friendly curve implementations.

In contrast, (non-blind) Schnorr signatures [114, 116] (such as EdDSA [25]) are short, can rely on standard libraries, and outperform RSA. Though their blind evaluation requires three rounds, this may be less concerning in applications where verification cost is the dominating factor and the signing application can easily keep state. Indeed, [56] identifies CBS as the only plausible alternative to RSA, and our schemes improve upon CBS by avoiding the mROS assumption. Once the group order is adjusted to resist sub-exponential attacks, we achieve comparable signature size, more efficient signing, and accommodate for partial blindness. (No partially blind version of CBS is known to the best of our knowledge.)

Finally, note that it is easier to prove security of pairing-free schemes under sequential access to the signer. For example, Kastner et al. [85] prove that plain blind Schnorr signatures are secure in this case, in the AGM, assuming the hardness of OMDL. Also, Baldimtsi and Lysyanskaya [11] (implicitly) prove sequential security of Abe's scheme. However, many applications, like PCM, easily enable concurrent attacks.

ON IDEAL MODELS. The use of the AGM or the GGM, along with the ROM, still appears necessary for the most practical pairing-free schemes with concurrent security. As of now, solutions solely assuming the ROM can only handle bounded concurrency [80] or, alternatively, their communication and computation costs grow with the number of signing sessions [88, 40, 123].

Scheme	PK size	Sig. size	Assumption	Communication
BS_1 (Section 5.1)	1 $\mathbb{G}$	3 $\mathbb{Z}_p$	GGM	2 $\mathbb{G}$ + 3 $\mathbb{Z}_p$
BS_2 (Section 6.1)	2 $\mathbb{G}$	4 $\mathbb{Z}_p$	DL	2 $\mathbb{G}$ + 4 $\mathbb{Z}_p$
PBS (Section 6.3)	1 $\mathbb{G}$	4 $\mathbb{Z}_p$	DL	2 $\mathbb{G}$ + 4 $\mathbb{Z}_p$
Blind Schnorr [65]	1 $\mathbb{G}$	2 $\mathbb{Z}_p$	OMDL + ROS	1 $\mathbb{G}$ + 2 $\mathbb{Z}_p$
Clause Blind Schnorr [65]	1 $\mathbb{G}$	2 $\mathbb{Z}_p$	OMDL + mROS	2 $\mathbb{G}$ + 4 $\mathbb{Z}_p$
Abe [3, 85]	3 $\mathbb{G}$	2 $\mathbb{G}$ + 6 $\mathbb{Z}_p$	DL	λ bits + 3 $\mathbb{G}$ + 6 $\mathbb{Z}_p$

Table 3.1: **Overview of our results.** The four schemes proposed in this paper compared to pairing-free schemes that admit GGM/AGM security proofs in the literature. All schemes are three-move and secure assuming the ROM; All schemes except BS_1 admit AGM security proofs; further $p = |\mathbb{G}|$. As in plain Schnorr signatures, most schemes allow replacing one element in $\mathbb{Z}_p$ with a group element in the signature. The ROS assumption can be broken in polynomial time unless the scheme is restricted to tolerate only a very small number of sessions. Also, the mROS assumption admits sub-exponential attacks, which require the choice of a larger order p over all schemes (roughly 512-bit for 128-bit security [65]).

A number of other schemes [68, 26, 67, 63, 62, 77, 86] partially or completely avoid ideal models, some of which are fairly practical. However, they do not yet appear suitable for at-scale deployment.

3.1 A Scheme in the GGM

Our simplest scheme only admits a proof in the generic-group model (GGM) but best illustrates our ideas, in particular, how we bypass ROS-style attacks. It is slightly less efficient than Schnorr signatures, i.e., a signature that consists of *three* scalars mod p (or alternatively, two scalars and a group element). Nonetheless, it has a very similar flavor (in particular, signature verification can be built on top of a suitable implementation of Schnorr signatures in a black-box way).

PREFACE: BLIND SCHNORR SIGNATURES AND ROS. Recall that we seek an interactive scheme (1) that is one-more unforgeable (i.e., no adversary should be able to generate $\ell + 1$ signatures by interacting only ℓ times with the signer), and (2) for which interaction can be blinded. It is helpful to illustrate the main technical barrier behind proving (1) for *interactive* Schnorr signatures. Recall

that the verification key is $X = g^x$ for a generator g of a cyclic group $\mathbb{G}$ of prime order p, and a signing key x. The signer starts the session by sending $A = g^a$, for a random $a \in \mathbb{Z}_p$. Then, the user sends a challenge $c = H(A, m)$ for a hash function H and a message m to be signed. Finally, the signer responds with $s = a + c \cdot x$, and the signature is $\sigma = (c, s)$.

Let us now consider an adversary that obtains ℓ initial messages $A_1, \ldots, A_\ell$ from the signer, where $A_i = g^{a_i}$. By solving the so-called *ROS problem* [117, 80, 65], the attacker can find $\ell + 1$ vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{\ell+1} \in \mathbb{Z}_p^\ell$ and a vector $(c_1, \ldots, c_\ell) \in \mathbb{Z}_p^\ell$ such that

$$\sum_{j=1}^{\ell} \alpha_i^{(j)} \cdot c_j = c_i^* \tag{3.1}$$

for all $i \in [\ell + 1]$, where $c_i^* = H(\prod_{j=1}^{\ell} A_j^{\alpha_i^{(j)}}, m_i^*)$, for some message $m_i^* \in \{0,1\}^*$. (Here, $\alpha_i^{(j)}$ is the j-th component of $\boldsymbol{\alpha}_i$.) Then, the attacker can obtain $s_j = a_j + c_j x$ from the signer for all $j \in [\ell]$ by completing the ℓ signing sessions. It is now easy to verify that (c_i^*, s_i^*) is a valid signature for m_i^* for all $i \in [\ell + 1]$, where $s_i^* = \sum_{j=1}^{\ell} \alpha_i^{(j)} \cdot s_j$. Benhamouda et al. [24] recently gave a simple polynomial-time algorithm to solve the ROS problem for the case $\ell > \log(p)$, which thus breaks one-more unforgeability.[2]

Fuchsbauer et al. [65] propose a different interactive signing process for Schnorr signatures that is one-more unforgeable (in the AGM + ROM) assuming that a variant of the ROS problem, called mROS, is hard. The mROS problem, however, admits sub-exponential attacks, and as it gives approximately only 70 bits of security from an implementation on a 256-bit curve, it effectively forces the use of 512-bit curves.[3]

OUR FIRST SCHEME. We take a different path which completely avoids the ROS and mROS problems to obtain our first scheme, BS_1. Again, we present a non-blind version – the scheme can be made blind via fairly standard tricks, as we explain in the body of the paper below. Again, the public key is $X = g^x$ for a secret key x. Then, the signer and the user engage in the following protocol to sign $m \in \{0,1\}^*$:

1. The signer sends $A = g^a$ and $Y = X^y$ for random $a, y \in \mathbb{Z}_p$.

[2] Many envisioned implementations allow for $\ell > \log(p)$. Still, is worth noting that the scheme retains some security for $\ell < \log(p)$ even in the standard model [80].

[3] mROS depends on a parameter ℓ, with a similar role as in ROS – sub-exponential attacks require $\ell < \log(p)$, but a one-more unforgeability attack for a small ℓ implies one for any $\ell' > \ell$ simply by generating $(\ell' - \ell)$ additional valid signatures.

2. The user responds with $c = H(A, Y, m)$

3. The signer returns a pair (s, y), where $s = a + cxy$.

4. The user accepts the signature $\sigma = (c, s, y)$ iff $g^s = A \cdot Y^c$ and $Y = X^y$.

Verification simply checks that $H(g^s X^{-yc}, X^y, M) = c$. In particular, note that (c, s) is a valid Schnorr signature with respect to the public-key X^y – this can be leveraged to implement the verification algorithm on top of an existing implementation of basic Schnorr signatures that also hash the public key (EdDSA does exactly this).[4] Further, as in Schnorr signatures, we could replace c with A in σ, and our results would be unaffected.

SECURITY INTUITION. To gather initial insights about the security of BS_1, it is instructive to *attempt* an ROS-style attack. The attacker opens ℓ sessions and obtains pairs $(A_1, Y_1), \ldots, (A_\ell, Y_\ell)$, where $A_i = g^{a_i}$ and $Y_i = X^{y_i} = g^{xy_i}$ for all $i \in [\ell]$. One natural extension of the ROS attack is to find $\ell + 1$ vectors $\boldsymbol{\alpha}_i \in \mathbb{Z}_p^\ell$ along with messages $m_1^*, m_2^*, \ldots \in \{0,1\}^*$ such that

$$c_i^* = H\left(\prod_{j=1}^{\ell} A_j^{\alpha_i^{(j)}}, \prod_{j=1}^{\ell} Y_j^{\alpha_i^{(j)}}, m_i^*\right)$$

for all $i \in [\ell + 1]$ and then find $(c_1, \ldots, c_\ell) \in \mathbb{Z}_p^\ell$ such that

$$\sum_{j=1}^{\ell} \alpha_i^{(j)} \cdot y_j \cdot c_j = c_i^* \cdot \sum_{j=1}^{\ell} \alpha_i^{(j)} \cdot y_j \ , \tag{3.2}$$

for all $i \in [\ell + 1]$. Indeed, if this succeeded, the adversary could complete the ℓ sessions to learn (s_j, y_j) by inputting c_j, where y_j is random and $s_j = a_j + c_j \cdot x \cdot y_j$. One could generate $\ell + 1$ signatures (c_i^*, s_i^*, y_i^*) for $i \in [\ell + 1]$ by setting $s_i^* = \sum_{j=1}^{\ell} \alpha_i^{(j)} s_j$ and $y_i^* = \sum_{j=1}^{\ell} \alpha_i^{(j)} \cdot y_j$. These would be valid because

$$\begin{aligned} g^{s_i^*} &= g^{\sum_{j=1}^{\ell} \alpha_i^{(j)}(a_j + c_j x y_j)} \\ &= \prod_{j=1}^{\ell} A_j^{\alpha_i^{(j)}} \cdot X^{\sum_{j=1}^{\ell} \alpha_i^{(j)} c_j y_j} \overset{(3.2)}{=} \prod_{j=1}^{\ell} A_j^{\alpha_i^{(j)}} \cdot \left(\prod_{j=1}^{\ell} Y_j^{\alpha_i^{(j)}}\right)^{c_i^*} . \end{aligned}$$

[4] Note that this only superficially resembles key-blinding for Schnorr signatures [83]. Here, the "blinding" y is actually public and part of the signature.

However, finding $(c_1, \ldots, c_\ell)$ that satisfy (3.2) for $\ell + 1$ i's simultaneously is *much harder* than ROS. An initial intuition here is that X^y *completely hides* y to the point where y is revealed later in the session, where it appears like a random and fresh weight in the sum, *independent of* c_i. This intuition is however not correct, as an attacker can use the group element X^y and can try to gain information about y, but our proof will show (among other things) that in the GGM no useful information is obtained about y, and y is (close to) uniform when it is later revealed.

THE WFROS PROBLEM. The above attack paradigm is in fact generalized in terms of a new ROS-like problem that we call WFROS (this stands for *Weighted Fractional ROS*), for which we prove an unconditional lower bound. WFROS considers a game with two oracles that can be invoked adaptively in an interleaved way:

- The first oracle, H, accepts as input a pair of vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^{2\ell+1}$, which are then associated with a random $\delta \in \mathbb{Z}_p^*$.

- The second oracle, S, allows to bind, for some $i \in [\ell]$, chosen input $c_i \in \mathbb{Z}_p$ with a random *weight* $y_i \in \mathbb{Z}_p^*$. During the course of the game, this latter oracle must be called *exactly* once for each $i \in [\ell]$.

The adversary finally commits to a subset of $\ell + 1$ prior H queries and wins if for each query in the subset, which has defined a pair of vector $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and returned δ, we have $A/B = \delta$, where

$$A = \alpha^{(0)} + \sum_{i \in [\ell]} y_i(\alpha^{(2i-1)} + c_i \cdot \alpha^{(2i)}) \ , \quad B = \beta^{(0)} + \sum_{i \in [\ell]} y_i(\beta^{(2i-1)} + c_i \cdot \beta^{(2i)}) \ .$$

Here, $v^{(i)}$ denotes the i-th component of vector $\boldsymbol{v}$. Our main result (Theorem 4.1.1) says that no adversary making Q_{H} queries to H can win this game with probability better than $(Q_{\mathrm{H}}^2 + 2\ell Q_{\mathrm{H}})/(p-1)$, or, in other words, $Q_{\mathrm{H}} \geqslant \min\{\sqrt{p}, p/\ell\}$ is needed to win with constant probability. Note that $\ell \ll \sqrt{p}$ is generally true, as for our usage, ℓ is bounded by the number of signing sessions.

Our GGM proof for BS_1 transforms any generic attacker into one breaking the WFROS problem. This transformation is actually not immediate because a one-more unforgeability attacker can learn functions of the secret key x when obtaining the second message from the signer. A similar challenge occurs in proving hardness of the OMDL problem in the GGM, which was recently resolved by Bauer et al. [12], and we rely on their techniques.

3.2 AGM Security and Partial Blindness

The Algebraic Group Model (AGM) [64] can be seen as a weaker idealization than the GGM. In particular, AGM proofs deal with actual groups (as opposed to representing group elements with random labels) and proceed via *reductions* that apply only to "algebraic adversaries", which provide representation of the group elements they output to the reduction. AGM has become a very popular model for validating security of a number of practical group-based protocols.

The main barrier to proving one-more unforgeability of BS_1 in the AGM is that the representation of X^y could leak some information about y that would not be available in the GGM, and thus we would not be able to apply our argument showing that y is still (close to) random looking when it is later revealed – our reduction in the GGM security proof crucially relies on this. To overcome this issue, for BS_2, we replace X^y with a *hiding* commitment to y:

Scheme BS_2. Here, $g^t X^y$ is replaced by $g^t Z^y$, where Z is an extra random group element included in the verification key.

We can prove security of BS_2 in the AGM based *solely* on the DL assumption. While BS_2 requires a longer key than BS_1, one could mitigate this by obtaining Z as the output of a hash function (assumed to be a random oracle) evaluated on some public input. The proof of security for BS_2 consists of showing that any adversary breaking one-more unforgeability can be transformed into one breaking either OMDL or DL (depending on the scheme) *or* into one breaking the WFROS problem. For the latter, however, we can resort to our unconditional hardness lower bound (Theorem 4.1.1).

ADDING PARTIAL BLINDNESS. Finally, we note that it is not too hard to add partial blindness to BS_3, which is another reason to consider this scheme. In particular, to obtain the resulting PBS scheme, we can adopt a framework by Abe and Okamoto [5]. The main idea is simply to use a hash function (modeled as a random oracle) to generate the extra group element Z in a way that is dependent on a public input upon which the signature depends. We target in particular a stronger notion of one-more unforgeability, which shows that if the protocol is run ℓ times for a public input, then no $\ell + 1$ signatures can be generated for that public input regardless of how many signatures have been generated for *different* public inputs. We defer more details to Section 6.3.

Outline of Part I

Section 3.3 will introduce some basic preliminaries. Chapter 4 will then introduce the WFROS problem, and prove a lower bound for it. We will then discuss our GGM-based scheme in Chapter 5, whereas variants secure in the AGM are presented in Chapter 6. In particular, we give a partially blind instantiation of our AGM scheme in Section 6.3.

3.3 Preliminaries

BLIND SIGNATURES. This paper focuses on *three-move* blind signature schemes, and our notation is similar to that of prior works (e.g., [80, 65]). Formally, a (three-move) *blind signature scheme* BS is a tuple of efficient (randomized) algorithms

$$\mathsf{BS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{S}_1, \mathsf{S}_2, \mathsf{U}_1, \mathsf{U}_2, \mathsf{Ver}) \ ,$$

with the following behavior:

- The *parameter generation* algorithm $\mathsf{Setup}(1^\lambda)$ outputs a string of parameters par, whereas the *key generation* algorithm $\mathsf{KeyGen}(par)$ outputs a key-pair (sk, pk), where sk is the *secret* (or *signing*) key and pk is the *public* (or *verification*) key.

- The interaction between the user and the signer to sign a message $m \in \{0,1\}^*$ with key-pair (pk, sk) is defined by the following experiment:

$$\begin{aligned} (\mathsf{st}^s, \mathsf{msg}_1) &\leftarrow \mathsf{BS}.\mathsf{S}_1(sk) \ , \ \ (\mathsf{st}^u, \mathsf{chl}) \leftarrow \mathsf{BS}.\mathsf{U}_1(pk, \mathsf{msg}_1, m) \ , \\ \mathsf{msg}_2 &\leftarrow \mathsf{BS}.\mathsf{S}_2(\mathsf{st}^s, \mathsf{chl}) \ , \ \ \sigma \leftarrow \mathsf{BS}.\mathsf{U}_2(\mathsf{st}^u, \mathsf{msg}_2) \ . \end{aligned} \tag{3.3}$$

  Here, σ is either the resulting *signature* or an *error message* $\perp$.

- The (deterministic) *verification algorithm* outputs a bit $\mathsf{BS}.\mathsf{Ver}(pk, \sigma, m)$.

We say that BS is (perfectly) *correct* if for every message $m \in \{0,1\}^*$, with probability one over the sampling of parameters and the key pair (pk, sk), the experiment in (3.3) returns σ such that $\mathsf{BS}.\mathsf{Ver}(pk, \sigma, m) = 1$. All of our schemes are going to be perfectly correct.

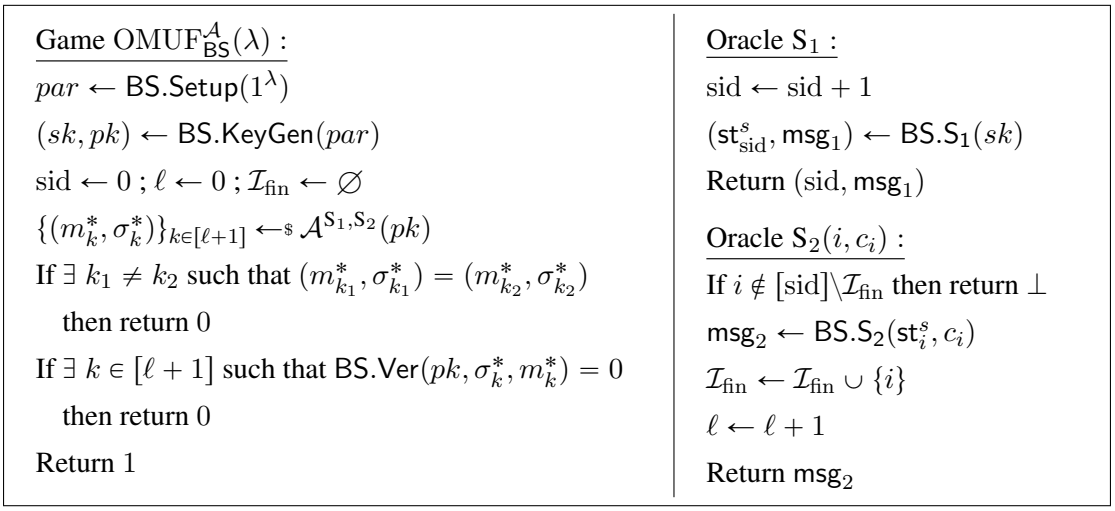

Figure 3.1: The OMUF security game for a blind signature scheme BS.

ONE-MORE UNFORGEABILITY. The standard notion of security for blind signatures is *one-more unforgeability* (OMUF). OMUF ensures that no adversary playing the role of a user interacting with the signer ℓ times, in an arbitrarily concurrent fashion, can issue $\ell + 1$ signatures (or more, of course). The $\mathrm{OMUF}^{\mathcal{A}}_{\mathsf{BS}}$ game for a blind signature scheme BS is defined in Figure 3.1. The corresponding advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{BS}}(\mathcal{A}, \lambda) := \mathsf{Pr}[\mathrm{OMUF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda) = 1]$. All of our analyses will further assume one or more random oracles, which are modeled as an additional oracle to which the adversary $\mathcal{A}$ is given access.

BLINDNESS. We also consider the standard notion of blindness against a malicious server that can, in particular, attempt to publish a malformed public key. The corresponding game $\mathrm{Blind}^{\mathcal{A}}_{\mathsf{BS}}$ is defined in Figure 3.2, and for any adversary $\mathcal{A}$, we define its advantage as $\mathsf{Adv}^{\mathrm{blind}}_{\mathsf{BS}}(\mathcal{A}, \lambda) := \left|\mathsf{Pr}[\mathrm{Blind}^{\mathcal{A}}_{\mathsf{BS}}(\lambda) = 1] - \frac{1}{2}\right|$. We say the scheme is perfectly blind if and only if $\mathsf{Adv}^{\mathrm{blind}}_{\mathsf{BS}}(\mathcal{A}, \lambda) = 0$ for any $\mathcal{A}$ and all λ.

GAME-PLAYING PROOFS. Several of our proofs adopt a lightweight variant of the standard "Game-Playing Framework" by Bellare and Rogaway [19].

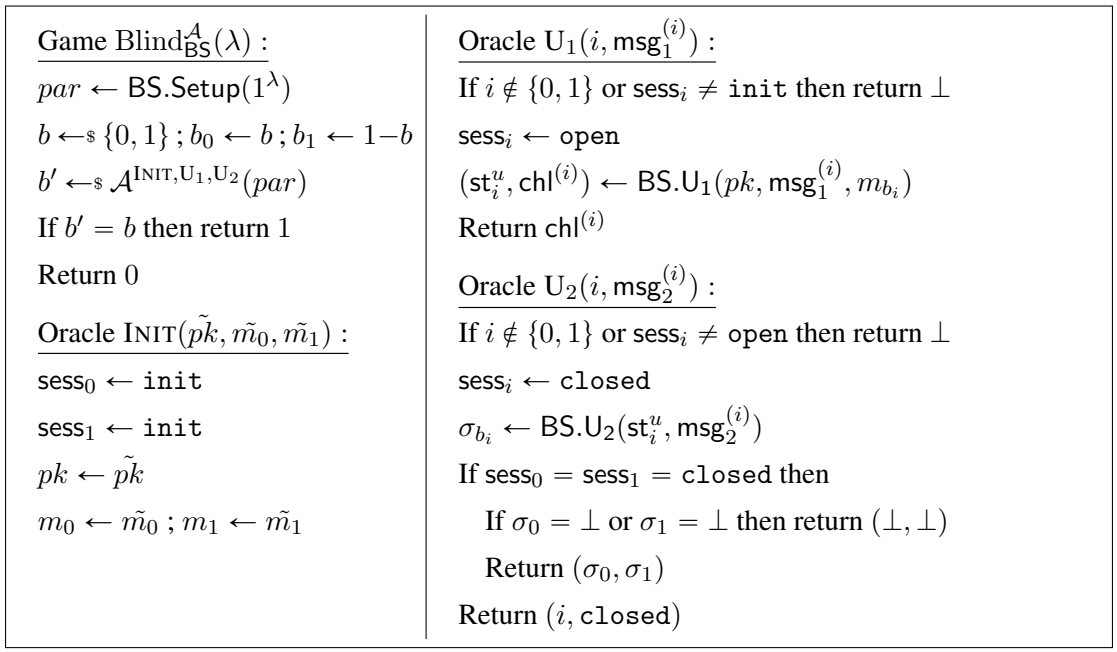

Figure 3.2: The Blind security game for a blind signature scheme BS.

Chapter 4

THE WEIGHTED FRACTIONAL ROS PROBLEM

4.1 Problem Description and Lower Bound

This section introduces and analyzes an unconditionally hard problem underlying all of our proofs, which we call the *Weighted Fractional ROS* problem (WFROS). It is a variant of the original ROS problem [117, 80, 65], which, in turn, stands for *Random inhomogeneities in a Overdetermined Solvable system of linear equations*. While ROS can be solved in polynomial time [24] and its mROS variant can be solved in sub-exponential time [65], we are going to prove an *exponential* lower bound for WFROS.

THE WFROS PROBLEM. The problem is defined via the game $\mathrm{WFROS}^{\mathcal{A}}_{\ell,p}$, described in Figure 4.1, which involves an adversary $\mathcal{A}$ and depends on two integer parameters ℓ and p, where p is a prime. The adversary here interacts with two oracles, H and S. The first oracle allows the adversary to link a vector pair $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^{2\ell+1}$ with a random inhomogeneous part $\delta \in \mathbb{Z}_p^*$ – each such query defines implicitly an equation $A/B = \delta$ in the unknowns $\mathsf{C}_1, \ldots, \mathsf{C}_\ell$ and $\mathsf{Y}_1, \ldots, \mathsf{Y}_\ell$. A call to $\mathrm{S}(i, c_i)$ lets us set the value of C_i to c_i and set Y_i to a random value y_i. The second oracle $\mathrm{S}(i, \cdot)$ must be called once for every $i \in [\ell]$. It is noteworthy to stress that the c_i's can be chosen arbitrarily, whereas the corresponding y_i's are random and independent.

In the end, the adversary wins the game if a subset of $\ell + 1$ equations defined by the H queries is satisfied by the assignment defined by querying S. In particular, we define $\mathsf{Adv}^{\mathrm{wfros}}_{\ell,p}(\mathcal{A}) = \mathsf{Pr}\left[\mathrm{WFROS}^{\mathcal{A}}_{\ell,p} = 1\right]$. Note that it would be possible to carry out some of the following security proofs using restricted versions of the WFROS game, but the above formulation lets us handle all schemes via a single notion.

A LOWER BOUND FOR WFROS. The following theorem, our main result on WFROS, shows that any adversary winning WFROS with constant probability requires $Q_H = \Omega(\min\{\sqrt{p}, p/\ell\})$ queries. (Also, note that all applications of interest assume $\ell \ll \sqrt{p}$.)

Theorem 4.1.1 (Lower bound for WFROS)**.** *For any $\ell > 0$, any prime number p, and any adver-*

Game $\mathrm{WFROS}^{\mathcal{A}}_{\ell,p}$:	Oracle $\mathrm{H}(\boldsymbol{\alpha}, \boldsymbol{\beta})$:
$\mathrm{hid} \leftarrow 0$; $\mathcal{I}_{\mathrm{fin}} \leftarrow \varnothing$	$\mathrm{hid} \leftarrow \mathrm{hid} + 1$
$\mathcal{J} \leftarrow \mathcal{A}^{\mathrm{H},\mathrm{S}}(p)$	$\boldsymbol{\alpha}_{\mathrm{hid}} \leftarrow \boldsymbol{\alpha}$; $\boldsymbol{\beta}_{\mathrm{hid}} \leftarrow \boldsymbol{\beta}$
If $\mathcal{J} \not\subseteq [\mathrm{hid}]$ or $\lvert\mathcal{J}\rvert \leqslant \ell$ or $\mathcal{I}_{\mathrm{fin}} \neq [\ell]$ then	$\delta_{\mathrm{hid}} \leftarrow\!\!\$\ \mathbb{Z}_p^*$
Return 0	Return $\delta_{\mathrm{hid}}, \mathrm{hid}$
For each $j \in \mathcal{J}$,	Oracle $\mathrm{S}(i, c_i)$:
$A_j \leftarrow \alpha_j^{(0)} + \sum_{i\in[\ell]} y_i(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)})$	If $i \notin [\ell] \backslash \mathcal{I}_{\mathrm{fin}}$ then return $\perp$
$B_j \leftarrow \beta_j^{(0)} + \sum_{i\in[\ell]} y_i(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)})$	$y_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$
If $\forall j \in \mathcal{J} : (A_j = \delta_j B_j \ \wedge\ B_j \neq 0)$ then	$\mathcal{I}_{\mathrm{fin}} \leftarrow \mathcal{I}_{\mathrm{fin}} \cup \{i\}$
Return 1	Return y_i
Return 0	

Figure 4.1: The WFROS problem. Here, $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^{2\ell+1}$, which is indexed as $\boldsymbol{\alpha} = (\alpha^{(0)}, \ldots, \alpha^{(2\ell)})$ and $\boldsymbol{\beta} = (\beta^{(0)}, \ldots, \beta^{(2\ell)})$.

---

sary $\mathcal{A}$ playing the $\mathrm{WFROS}^{\ell,p}$ *game that makes at most Q_{H} queries to* H*, we have*

$$\mathsf{Adv}^{\mathrm{wfros}}_{\ell,p}(\mathcal{A}) \leqslant \frac{Q_{\mathrm{H}}(2\ell + Q_{\mathrm{H}})}{p-1} \ .$$

The proof is given in the next section. To gain some very high-level intuition, we observe that a key contributor to the hardness of WFROS are values y_i, which are defined *after* the c_i's are fixed and hence randomize the A_j and B_j's. Therefore, to satisfy $A_j = \delta_j \cdot B_j$, the adversary is restricted in the way it plays. For example, to satisfy an equation defined by an H query $(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j)$, the adversary can pick c_i's such that $(\alpha_j^{(2i-1)} + c_i\alpha_j^{(2i)}) = \delta_j \cdot (\beta_j^{(2i-1)} + c_i\beta_j^{(2i)})$ for all $i \in [\ell]$. Then, the equation $A_j = \delta_j B_j$ is satisfied no matter what the y_i's are. Our proof shows that the adversary *has* to pick c_i's this way – and in fact, it has to follow even more restrictions. Finally, we show that under these restrictions, no set of $\ell + 1$ equations can be satisfied simultaneously.

4.2 Proof of the Lower Bound for WFROS (Theorem 4.1.1)

Let $\mathcal{A}$ be an adversary for the WFROS game that makes at most Q_{H} queries to H. Without loss of generality, we assume that $\mathcal{A}$ makes exactly one query (i, c_i) to S for each $i \in [\ell]$ and that $\mathcal{A}$ always outputs $\mathcal{J} \subseteq [Q_{\mathrm{H}}]$.

In the $\mathrm{WFROS}^{\mathcal{A}}_{\ell,p}$ game, for each $j \in [Q_{\mathrm{H}}]$, denote the event W_j as

$$\alpha_j^{(0)} + \sum_{i\in[\ell]} y_i(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)}) = \delta_j \left(\beta_j^{(0)} + \sum_{i\in[\ell]} y_i(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}) \right) \tag{W1}$$

$$\wedge\ \beta_j^{(0)} + \sum_{i\in[\ell]} y_i(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}) \neq 0\ . \tag{W2}$$

In other words, W_j is the event that the equation defined by the j-th H query is satisfied. Then, $\mathcal{A}$ wins if and only if $|\mathcal{J}| > \ell$ and W_j occur for each $j \in \mathcal{J}$. Denote $W := (|\mathcal{J}| > \ell) \ \wedge\ \left(\bigwedge_{j\in\mathcal{J}} W_j\right)$ and we have $\mathsf{Adv}^{\mathrm{wfros}}_{\ell,p}(\mathcal{A}) = \mathsf{Pr}[W]$.

To bound $\mathsf{Pr}[W]$, we need notation to refer to some values (formally, random variables) defined in the execution of the $\mathrm{WFROS}^{\mathcal{A}}_{\ell,p}$ game. First, denote as $\mathcal{I}^{(j)}_{\mathrm{fin}}$ the contents of the set $\mathcal{I}_{\mathrm{fin}}$ when the adversary makes the j-th query to H, and let $(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j)$ be the input of this query to H, which is answered with δ_j. Also, let $\mathcal{I}^{(j)}_{\mathrm{unk}} := [\ell]\backslash\mathcal{I}^{(j)}_{\mathrm{fin}}$, i.e., the set of indices $i \in [\ell]$ for which $\mathcal{A}$ has not yet made any query $(i, \cdot)$ to S when the j-th query to H is made. Further, $c_1, \ldots, c_\ell$ and $y_1, \ldots, y_\ell$ are the values defined by querying S.

Now, for each $j \in [Q_{\mathrm{H}}]$, we define the following events:

Event $E^{(1)}_j$. First, let $E^{(1)}_{1,j}$ be the event that $\beta_j^{(0)} + \sum_{i\in\mathcal{I}^{(j)}_{\mathrm{fin}}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) \neq 0$. For each $i \in \mathcal{I}^{(j)}_{\mathrm{unk}}$, also let $E^{(1)}_{2,(j,i)}$ be the event that $\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} \neq \delta_j \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right)$. Finally, let $E^{(1)}_j := E^{(1)}_{1,j} \ \vee\ \left(\bigvee_{i\in[\mathcal{I}^{(j)}_{\mathrm{unk}}]} E^{(1)}_{2,(j,i)}\right)$.

Event $E^{(2)}_j$. We denote the event $E^{(2)}_j$ as the event where

$$\forall\ i \in \mathcal{I}^{(j)}_{\mathrm{unk}} : \alpha_j^{(2i)} \cdot \beta_j^{(2i-1)} = \alpha_j^{(2i-1)} \cdot \beta_j^{(2i)}\ . \tag{4.1}$$

Note that events $E^{(1)}_j$ and $E^{(2)}_j$ are, by themselves, not necessarily unlikely – the adversary can certainly provoke them. However, we intend to show that this has implications on the ability to satisfy the j-th equation. In particular, we prove the following two lemmas in Sections 4.2.1 and 4.2.2 below, respectively.

Lemma 4.2.1. $\mathsf{Pr}[W_j \ \wedge\ E^{(1)}_j] \leqslant \frac{\ell+1}{p-1}$.

Lemma 4.2.2. $\mathsf{Pr}[W_j \ \wedge\ (\neg E^{(1)}_j) \ \wedge\ E^{(2)}_j] \leqslant \frac{\ell}{p-1}$.

Now, if we denote $E^{(1)} := \bigvee_{j\in[Q_{\mathrm{H}}]}(W_j \wedge E_j^{(1)})$ and $E^{(2)} := \bigvee_{j\in[Q_{\mathrm{H}}]}(W_j \wedge (\neg E_j^{(1)}) \wedge E_j^{(2)})$, the union bound yields $\mathsf{Pr}[E^{(1)}] \leqslant \frac{Q_{\mathrm{H}}(\ell+1)}{p-1}$ and $\mathsf{Pr}[E^{(2)}] \leqslant \frac{Q_{\mathrm{H}}\cdot\ell}{p-1}$. Our final lemma (proved in Section 4.2.3) is then the following:

Lemma 4.2.3. $\mathsf{Pr}[W \wedge (\neg E^{(1)}) \wedge (\neg E^{(2)})] \leqslant \frac{Q_{\mathrm{H}}(Q_{\mathrm{H}}-1)}{p-1}$.

The three lemmas can be combined to obtain

$$\mathsf{Pr}[W] \leqslant \mathsf{Pr}[E^{(1)}] + \mathsf{Pr}[E^{(2)}] + \mathsf{Pr}[W \wedge (\neg E^{(1)}) \wedge (\neg E^{(2)})] \leqslant \frac{Q_{\mathrm{H}}(2\ell + Q_{\mathrm{H}})}{p-1} \ .$$

which concludes the proof. In the next three sections, we prove the three preceding lemmas.

4.2.1 Proof of Lemma 4.2.1

Throughout this proof, let us fix $j \in [Q_{\mathrm{H}}]$. We first define a sequence of random variables $(D_0, D_1, \ldots, D_n, X_1, \ldots, X_n)$, where $n = \ell + 1$, such that $E_j^{(1)}$ implies one of $D_0, \ldots, D_n$ is not equal to 0 and $D_0 + \sum_{k\in[n]} D_k X_k = 0$. Further, we also ensure that X_k is uniformly distributed over $\mathbb{Z}_p^*$ independent of $(D_0, D_1, \ldots, D_k, X_1, \ldots, X_{k-1})$ for each $k \in [n]$ and use this to bound $\mathsf{Pr}[E_j^{(1)}]$. More concretely:

- Let

$$D_0 := \alpha_j^{(0)} + \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{(j)}} y_i \left(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i-1)}\right) \ ,$$

$$X_1 := -\delta_j \ , \ D_1 := \beta_j^{(0)} + \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{(j)}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) \ ,$$

  and note that $E_{1,j}^{(1)}$ is equivalent to $D_1 \neq 0$.

- Further, for $1 \leqslant k \leqslant |\mathcal{I}_{\mathrm{unk}}^{(j)}|$, denote $i_k \in \mathcal{I}_{\mathrm{unk}}^{(j)}$ as the index such that (i_k, c_{i_k}) is the k-th query made to S among the indexes in $\mathcal{I}_{\mathrm{unk}}^{(j)}$ and let

$$X_{k+1} = y_{i_k} \ , \ D_{k+1} := \alpha_j^{(2i_k-1)} + c_{i_k} \cdot \alpha_j^{(2i_k)} - \delta_j \left(\beta_j^{(2i_k-1)} + c_{i_k} \cdot \beta_j^{(2i_k)}\right) \ ,$$

  we have $E_{2,(j,i_k)}^{(1)}$ occurs is equivalent to $D_{k+1} \neq 0$.

- For $|\mathcal{I}_{\mathrm{unk}}^{(j)}| + 1 < k \leqslant n$, let $D_k = 0$ and X_k be a random variable uniformly distributed in $\mathbb{Z}_p^*$ independent of $(D_0, D_1, \ldots, D_k, X_1, \ldots, X_{k-1})$. [1]

[1] For $|\mathcal{I}_{\mathrm{unk}}^{(j)}| + 1 < k \leqslant n$, D_k, X_k act as placeholders so that we can apply Lemma 4.2.5 for an a priori fixed value n instead of a random variable $|\mathcal{I}_{\mathrm{unk}}^{(j)}| + 1$.

Note that

$$\begin{aligned}
D_0 + \sum_{k=1}^{n} D_k X_k &= \alpha_j^{(0)} + \sum_{i \in \mathcal{I}_{\text{fin}}^{(j)}} y_i \left(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} \right) \\
&\quad - \delta_j \left(\beta_j^{(0)} + \sum_{i \in \mathcal{I}_{\text{fin}}^{(j)}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} \right) \right) \\
&\quad + \sum_{i \in \mathcal{I}_{\text{unk}}^{(j)}} y_i \left(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} - \delta_j \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} \right) \right) \\
&= \alpha_j^{(0)} + \sum_{i \in [\ell]} y_i \left(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} \right) \\
&\quad - \delta_j \left(\beta_j^{(0)} + \sum_{i \in [\ell]} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} \right) \right) .
\end{aligned}$$

Therefore, by (W1), we know W_j occurs implies $D_0 + \sum_{i=1}^{n} D_i X_i = 0$. Thus, the event $W_j \wedge E_j^{(1)}$ implies, in addition, that one of $D_0, \ldots, D_n$ is not equal to 0. Also, we prove the following claim.

Claim 4.2.4. *For each $k \in [n]$, X_k is uniformly distributed over $\mathbb{Z}_p^*$ independent of $(D_0, \ldots, D_k, X_1, \ldots, X_{k-1})$.*

proof of Claim 4.2.4. For $k = 1$, we have $X_1 = -\delta_j$. Consider the step when δ_j is generated. Since $\mathcal{A}$ has made the j-th query to H, we know $\mathcal{I}_{\text{unk}}^{(j)}$, $\boldsymbol{\beta}_j$, $\boldsymbol{\alpha}_j$, and $\{y_i, c_i\}_{i \in \mathcal{I}_{\text{fin}}^{(j)}}$ are already determined, which implies D_0 and D_1 are also determined. Since δ_j is picked uniformly at random from $\mathbb{Z}_p^*$, we know $X_1 = -\delta_j$ is uniformly distributed over $\mathbb{Z}_p^*$ independent of (D_0, D_1).

For $2 \leqslant k \leqslant |\mathcal{I}_{\text{unk}}^{(j)}| + 1$, we have $X_k = y_{i_{k-1}}$. Consider the step when $y_{i_{k-1}}$ is generated. We know $\mathcal{A}$ has made the query $(i_{k-1}, c_{i_{k-1}})$ to S and the values i_{k-1}, $c_{i_{k-1}}$ are determined. Since $i_{k-1} \in \mathcal{I}_{\text{unk}}^{(j)}$, we know $\mathcal{A}$ has made the j-th query to H, and thus the values $\boldsymbol{\beta}_j$, $\boldsymbol{\alpha}_j$, δ_j, and (D_0, D_1) are determined. For $1 \leqslant k' < k - 1$, since the query $(i_{k'}, c_{i_{k'}})$ to S has returned, we know the values $i_{k'}, c_{i_{k'}}, y_{i_{k'}}$ are determined, which implies $D_{k'+1}$ and $X_{k'+1}$ are determined. Also, since i_{k-1}, $c_{i_{k-1}}$ are determined, we know D_k is determined. Therefore, since $y_{i_{k-1}}$ is picked uniformly at random from $\mathbb{Z}_p^*$, we know $X_k = y_{i_{k-1}}$ is uniformly distributed over $\mathbb{Z}_p^*$ independent of $(D_0, \ldots, D_k, X_1, \ldots, X_{k-1})$.

For $|\mathcal{I}_{\text{unk}}^{(j)}| + 1 < k \leqslant n$, by the definition of X_k, we know X_k is uniformly distributed over $\mathbb{Z}_p^*$ independent of $(D_0, \ldots, D_k, X_1, \ldots, X_{k-1})$. Therefore, the claim holds. □

Now, we can show the upper bound $\mathsf{Pr}[W_j \ \wedge \ E_j^{(1)}] \leqslant \frac{\ell+1}{p-1}$ by the following lemma.[2]

Lemma 4.2.5. *Let p be prime. Let $D_0, D_1, \ldots, D_n, X_1, \ldots, X_n \in \mathbb{Z}_p$ be random variables such that for all $k \in [n]$, X_k is uniform over $U_k \subseteq \mathbb{Z}_p$ and independent of $(D_0, \ldots, D_k, X_1, \ldots, X_{k-1})$. Then,*

$$\mathsf{Pr}\left[\exists\, i \in \{0,\ldots,n\} \ : \ D_i \neq 0 \ \wedge \ D_0 + \sum_{j=1}^{n} D_j X_j = 0\right] \leqslant \sum_{i=1}^{n} \frac{1}{|U_i|} \ .$$

Proof. For $k \in \{0,\ldots,n\}$, define E_k as

$$\exists\, i \in \{0,\ldots,k\} \text{ such that } D_i \neq 0 \ \wedge \ D_0 + \sum_{j=1}^{k} D_j X_j = 0 \ .$$

We will prove the theorem using induction. It is clear that $\mathsf{Pr}[E_0] = 0$. For $k \geqslant 1$, assume $\mathsf{Pr}[E_{k-1}] \leqslant \sum_{i=1}^{k-1} \frac{1}{|U_i|}$. It holds that

$$\begin{aligned}
\mathsf{Pr}[E_k] &= \mathsf{Pr}[E_k|E_{k-1}]\mathsf{Pr}[E_{k-1}] + \mathsf{Pr}[E_k|\neg E_{k-1}]\mathsf{Pr}[\neg E_{k-1}] \\
&\leqslant \mathsf{Pr}[E_{k-1}] + \mathsf{Pr}[E_k|\neg E_{k-1}] \\
&= \mathsf{Pr}[E_{k-1}] + \mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k \neq 0\right]\mathsf{Pr}[D_k \neq 0|\neg E_{k-1}] \\
&\quad + \mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k = 0\right]\mathsf{Pr}[D_k = 0|\neg E_{k-1}] \\
&\leqslant \mathsf{Pr}[E_{k-1}] + \mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k \neq 0\right] + \mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k = 0\right] \ .
\end{aligned} \tag{4.2}$$

It is left to bound $\mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k \neq 0\right]$ and $\mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k = 0\right]$.

Suppose E_{k-1} does not occur and then we have either $D_i = 0$ for all $0 \leqslant i < k$ or $D_0 + \sum_{j=1}^{k-1} D_j X_j = 0$.

If $D_k = 0$, we have either $D_i = 0$ for all $0 \leqslant i \leqslant k$, or $D_0 + \sum_{j=1}^{k} D_j X_j = D_0 + \sum_{j=1}^{k-1} D_j X_j \neq 0$, which means E_k does not occur. Therefore, we have

$$\mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k = 0\right] = 0. \tag{4.3}$$

Otherwise, if $D_k \neq 0$, we know E_k occurs if and only if $D_0 + \sum_{j=1}^{k} D_j X_j \neq 0$. Since X_k is uniformly distributed over U_k independent of $(D_0, \ldots, D_k, X_1, \ldots, X_{k-1})$ given $D_k \neq 0$ and E_{k-1}

[2] Note that this lemma cannot be directly derived from the Schwartz-Zippel lemma by viewing $D_0 + \sum_{j=1}^{n} D_j X_j = 0$ as a polynomial of $X_1, \ldots, X_n$, since we cover for example the case where $D_0, D_1, \ldots, D_n$ are adaptively chosen, i.e., each D_i can depend on $X_1 \ldots, X_{i-1}$.

does not occur, it holds that

$$\begin{aligned}\mathsf{Pr}\left[E_k \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k \neq 0\right] &= \mathsf{Pr}\left[D_0 + \sum_{j=1}^{k} D_j X_j = 0 \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k \neq 0\right] \\ &= \mathsf{Pr}\left[X_k = \frac{D_0 + \sum_{j=1}^{k-1} D_j X_j}{D_k} \,\middle|\, (\neg E_{k-1}) \ \wedge \ D_k \neq 0\right] \\ &\leqslant \frac{1}{|U_i|} \ .\end{aligned} \tag{4.4}$$

Therefore, from (4.2), (4.3), and (4.4), we have

$$\mathsf{Pr}[E_k] \leqslant \mathsf{Pr}[E_{k-1}] + \frac{1}{|U_i|} \leqslant \sum_{i=1}^{k} \frac{1}{|U_i|} \ .$$

Therefore, by induction, we have

$$\mathsf{Pr}\left[\exists \ i \in \{0, \ldots, n\} \ : \ D_i \neq 0 \ \wedge \ D_0 + \sum_{j=1}^{n} D_j X_j = 0\right] = \mathsf{Pr}[E_n] \leqslant \sum_{i=1}^{n} \frac{1}{|U_i|}.$$

□

4.2.2 Proof of Lemma 4.2.2

It is easier to introduce a new event F_j and show that $W_j \ \wedge \ (\neg E_j^{(1)})$ implies F_j. We will then bound $\mathsf{Pr}[F_j \ \wedge \ E_j^{(2)}]$. In particular, define the event F_j as

$$\forall \ i \in \mathcal{I}_{\text{unk}}^{(j)} \ : \ \alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} - \delta_j \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) = 0 \tag{F1}$$

$$\wedge \sum_{i \in \mathcal{I}_{\text{unk}}^{(j)}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) \neq 0 \ , \tag{F2}$$

and we have the following lemma.

Lemma 4.2.6. *If $W_j \ \wedge \ (\neg E_j^{(1)})$ occurs, then the event F_j occurs.*

of Lemma 4.2.6. By the definition of F_j, we need only show that if $W_j \ \wedge \ (\neg E_j^{(1)})$ occurs, then (F1) and (F2) hold for j.

Suppose W_j occurs but $E_j^{(1)}$ does not occur. Since $E_j^{(1)} = E_{1,j}^{(1)} \vee \left(\bigvee_{i\in[\mathcal{I}_{\text{unk}}^{(j)}]} E_{2,(j,i)}^{(1)}\right)$, we know all of $E_{1,j}^{(1)}$ and $\{E_{2,(j,i)}^{(1)}\}_{i\in[\mathcal{I}_{\text{unk}}^{(j)}]}$ do not occur. Since the event $E_{2,(j,i)}^{(1)}$ does not occur implies

$$\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} - \delta_j \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) = 0 \,,$$

we know (F1) holds for j.

Also, since the event $E_{1,j}^{(1)}$ does not occur, we have

$$\beta_j^{(0)} + \sum_{i\in\mathcal{I}_{\text{fin}}^{(j)}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) = 0.$$

Since W_j occurs, we know (W2) holds and, by the above equation, we have

$$\sum_{i\in\mathcal{I}_{\text{unk}}^{(j)}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) = \beta_j^{(0)} + \sum_{i\in[\ell]} y_i(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}) \neq 0 \,.$$

Therefore, we know (F2) holds for j. □

We also denote

$$\mathcal{D}_j := \left\{ \tfrac{\alpha_j^{(2i)}}{\beta_j^{(2i)}} \mid i \in \mathcal{I}_{\text{unk}}^{(j)}, \beta_j^{(2i)} \neq 0 \right\} \cup \left\{ \tfrac{\alpha_j^{(2i-1)}}{\beta_j^{(2i-1)}} \mid i \in \mathcal{I}_{\text{unk}}^{(j)}, \beta_j^{(2i)} = 0, \beta_j^{(2i-1)} \neq 0 \right\}.$$

We have $|\mathcal{D}_j| \leqslant |\{i \in \mathcal{I}_{\text{unk}}^{(j)} \mid \beta_j^{(2i)} \neq 0\} \cup \{i \in \mathcal{I}_{\text{unk}}^{(j)} \mid \beta_j^{(2i)} = 0\}| = |\mathcal{I}_{\text{unk}}^{(j)}|$.

Claim 4.2.7. *The event $F_j \wedge E_j^{(2)}$ implies $\delta_j \in \mathcal{D}_j$.*

Proof. Suppose $F_j \wedge E_j^{(2)}$ occurs but $\delta_j \notin \mathcal{D}_j$. We are going to show that $\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} = 0$ for each $i \in \mathcal{I}_{\text{unk}}^{(j)}$. Then, since F_j occurs, we know (F2) holds, which yields a contradiction, and thus the claim holds.

For $i \in \mathcal{I}_{\text{unk}}^{(j)}$, if $\beta_j^{(2i)} \neq 0$, since $\delta_j \notin \mathcal{D}_j$, we have $\delta_j \neq \frac{\alpha_j^{(2i)}}{\beta_j^{(2i)}}$, which implies $\alpha_j^{(2i)} - \delta_j \cdot \beta_j^{(2i)} \neq 0$. Since F_j occurs, by (F1), we have $c_i = -\frac{\alpha_j^{(2i-1)} - \delta_j \cdot \beta_j^{(2i-1)}}{\alpha_j^{(2i)} - \delta_j \cdot \beta_j^{(2i)}}$. Since $E_j^{(2)}$ occurs, by (4.1), we have $\alpha_j^{(2i)} \cdot \beta_j^{(2i-1)} = \alpha_j^{(2i-1)} \cdot \beta_j^{(2i)}$, and thus

$$\begin{aligned}
\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} &= \beta_j^{(2i-1)} - \frac{\alpha_j^{(2i-1)} \cdot \beta_j^{(2i)} - \delta_j \cdot \beta_j^{(2i-1)} \cdot \beta_j^{(2i)}}{\alpha_j^{(2i)} - \delta_j \cdot \beta_j^{(2i)}} \\
&= \beta_j^{(2i-1)} - \frac{\alpha_j^{(2i)} \cdot \beta_j^{(2i-1)} - \delta_j \cdot \beta_j^{(2i-1)} \cdot \beta_j^{(2i)}}{\alpha_j^{(2i)} - \delta_j \cdot \beta_j^{(2i)}} \\
&= \beta_j^{(2i-1)} - \beta_j^{(2i-1)} = 0 \,.
\end{aligned}$$

Otherwise, suppose $\beta_j^{(2i)} = 0$. Then, if $\beta_j^{(2i-1)} = 0$, we also have $\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} = 0$. If $\beta_j^{(2i-1)} \neq 0$, since $\alpha_j^{(2i)} \cdot \beta_j^{(2i-1)} = \alpha_j^{(2i-1)} \cdot \beta_j^{(2i)} = 0$, we have $\alpha_j^{(2i)} = 0$. Since $\beta_j^{(2i)} = 0$, $\beta_j^{(2i-1)} \neq 0$, and $\delta_j \notin \mathcal{D}_j$, we have $\delta_j \neq \frac{\alpha_j^{(2i-1)}}{\beta_j^{(2i-1)}}$ and thus we have

$$\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)} - \delta_j \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) = \alpha_j^{(2i-1)} - \delta_j \cdot \beta_j^{(2i-1)} \neq 0,$$

which contradicts (F1). Therefore, it is impossible that $\beta_j^{(2i)} = 0$ and $\beta_j^{(2i-1)} \neq 0$.

Therefore, from the above arguments, we have $\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} = 0$ for any $i \in \mathcal{I}_{\text{unk}}^{(j)}$, and thus $\sum_{i \in \mathcal{I}_{\text{unk}}^{(j)}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}\right) = 0$. However, since F_j occurs, we know (F2) holds, which yields a contradiction, and thus the claim holds. □

Note that δ_j is generated uniformly at random, independently of $\mathcal{D}_j$, since the latter is defined by the j-th H query. Therefore, Lemma 4.2.6 and Claim 4.2.7 yield

$$\begin{aligned}\mathsf{Pr}[W_j \ \wedge \ (\neg E_j^{(1)}) \ \wedge \ E_j^{(2)}] &\leqslant \mathsf{Pr}[F_j \ \wedge \ E_j^{(2)}] \\ &\leqslant \mathsf{Pr}[\delta_j \in \mathcal{D}_j] \leqslant \frac{|\mathcal{I}_{\text{unk}}^{(j)}|}{p-1} \leqslant \frac{\ell}{p-1} \,.\end{aligned}$$

4.2.3 Proof of Lemma 4.2.3

To conclude the analysis, we introduce yet another event, $E^{(3)}$. We will show below that $W \ \wedge \ (\neg E^{(1)}) \ \wedge \ (\neg E^{(2)})$ implies $E^{(3)}$, and thus it is enough to upper bound the probability of $E^{(3)}$ occurring. Concretely, $E^{(3)}$ is defined as follows (the definition of the following events $F_{j'}$ is given in Section 4.2.2).

Event $E^{(3)}$. For each $j_1, j_2 \in [Q_{\text{H}}]$ and $j_1 < j_2$, denote the event $E^{(3)}_{(j_1,j_2)}$ as

$$\exists\, i \in \mathcal{I}_{\text{unk}}^{(j_1)} \cap \mathcal{I}_{\text{unk}}^{(j_2)} : \alpha_{j_1}^{(2i)} \cdot \beta_{j_1}^{(2i-1)} \neq \alpha_{j_1}^{(2i-1)} \cdot \beta_{j_1}^{(2i)} \ \wedge \ \alpha_{j_2}^{(2i)} \cdot \beta_{j_2}^{(2i-1)} \neq \alpha_{j_2}^{(2i-1)} \cdot \beta_{j_2}^{(2i)} \,.$$

Denote $E'^{(3)}_{(j_1,j_2)} := E^{(3)}_{(j_1,j_2)} \ \wedge \ F_{j_1} \ \wedge \ F_{j_2}$ and $E^{(3)} := \bigvee_{j_1,j_2 \in [Q_{\text{H}}], j_1 < j_2} E'^{(3)}_{(j_1,j_2)}$.

To see why the above implication is true, assume that W indeed occurs, but both $E^{(1)}$ and $E^{(2)}$ do not occur. We now fix some $j \in \mathcal{J}$. We know W_j occurs, but both $E_j^{(1)}$ and $E_j^{(2)}$ do not occur. In particular, by the definition of $E_j^{(2)}$, we know there exists $i \in \mathcal{I}_{\text{unk}}^{(j)}$ such that $\alpha_j^{(2i)} \cdot \beta_j^{(2i-1)} \neq \alpha_j^{(2i-1)} \cdot \beta_j^{(2i)}$.

Let $i^{(j)}_{\min}$ be the smallest index in $\mathcal{I}^{(j)}_{\mathrm{unk}}$ such that $\alpha_j^{(2i^{(j)}_{\min})} \cdot \beta_j^{(2i^{(j)}_{\min}-1)} \neq \alpha_j^{(2i^{(j)}_{\min}-1)} \cdot \beta_j^{(2i^{(j)}_{\min})}$. Since W occurs, we know $|\mathcal{J}| > \ell$. Then, since $i^{(j)}_{\min} \in \mathcal{I}^{(j)}_{\mathrm{unk}} \subseteq [\ell]$ for each $j \in \mathcal{J}$ and $|\mathcal{J}| > \ell$, by the pigeonhole principle, we know there exists $j_1, j_2 \in \mathcal{J}$ such that $j_1 < j_2$ and $i^{(j_1)}_{\min} = i^{(j_2)}_{\min}$, which implies $E^{(3)}_{(j_1,j_2)}$ occurs. Also, since we know both $W_{j_1} \wedge (\neg E^{(1)}_{j_1})$ and $W_{j_2} \wedge (\neg E^{(1)}_{j_2})$ occur, by Lemma 4.2.6, we have F_{j_1} and F_{j_2} both occur. Therefore, we know $E'^{(3)}_{(j_1,j_2)} = E^{(3)}_{(j_1,j_2)} \wedge F_{j_1} \wedge F_{j_2}$ occurs, which implies $E^{(3)}$ occurs.

Therefore, we have

$$\mathsf{Pr}\left[W \wedge (\neg E^{(1)}) \wedge (\neg E^{(2)})\right] \leqslant \mathsf{Pr}[E^{(3)}] \leqslant \sum_{j_1,j_2 \in [Q_{\mathrm{H}}], j_1 < j_2} \mathsf{Pr}[E'^{(3)}_{(j_1,j_2)}] \,.$$

We now just need to bound $\mathsf{Pr}[E'^{(3)}_{(j_1,j_2)}]$ for any $j_1 < j_2$.

To gain insight, suppose $E'^{(3)}_{(j_1,j_2)}$ occurs. We can show that there exists $i \in \mathcal{I}^{(j_1)}_{\mathrm{unk}} \cap \mathcal{I}^{(j_2)}_{\mathrm{unk}}$ such that $\alpha_{j_1}^{(2i)} - \delta_{j_1}\beta_{j_1}^{(2i)} \neq 0$ and $\alpha_{j_2}^{(2i)} - \delta_{j_2}\beta_{j_2}^{(2i)} \neq 0$. Then, since F_{j_1} and F_{j_2} occur, by (F1), it holds that

$$\frac{\alpha_{j_1}^{(2i-1)} - \delta_{j_1} \cdot \beta_{j_1}^{(2i-1)}}{\alpha_{j_1}^{(2i)} - \delta_{j_1} \cdot \beta_{j_1}^{(2i)}} = c_i = \frac{\alpha_{j_2}^{(2i-1)} - \delta_{j_2} \cdot \beta_{j_2}^{(2i-1)}}{\alpha_{j_2}^{(2i)} - \delta_{j_2} \cdot \beta_{j_2}^{(2i)}} \,.$$

However, this can occur with only small probability since δ_{j_1} and δ_{j_2} are sampled independently. The following claim, proved in Section 4.2.4, makes this formal.

Claim 4.2.8. *For any $j_1, j_2 \in [Q_{\mathrm{H}}]$ such that $j_1 < j_2$, suppose $E'^{(3)}_{(j_1,j_2)}$ occurs. Let i_{dif} be the smallest index in $\mathcal{I}^{(j_1)}_{\mathrm{unk}} \cap \mathcal{I}^{(j_2)}_{\mathrm{unk}}$ such that $\alpha_{j_1}^{(2i_{\mathrm{dif}})} \cdot \beta_{j_1}^{(2i_{\mathrm{dif}}-1)} \neq \alpha_{j_1}^{(2i_{\mathrm{dif}}-1)} \cdot \beta_{j_1}^{(2i_{\mathrm{dif}})}$ and $\alpha_{j_2}^{(2i_{\mathrm{dif}})} \cdot \beta_{j_2}^{(2i_{\mathrm{dif}}-1)} \neq \alpha_{j_2}^{(2i_{\mathrm{dif}}-1)} \cdot \beta_{j_2}^{(2i_{\mathrm{dif}})}$. Then, we have $\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}})} \neq 0$. Moreover, let $T = \frac{\alpha_{j_1}^{(2i_{\mathrm{dif}}-1)} - \delta_{j_1} \cdot \beta_{j_1}^{(2i_{\mathrm{dif}}-1)}}{\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1} \cdot \beta_{j_1}^{(2i_{\mathrm{dif}})}}$, and we have $\beta_{j_2}^{(2i_{\mathrm{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\mathrm{dif}})} \neq 0$ and $\delta_{j_2} = \frac{\alpha_{j_2}^{(2i_{\mathrm{dif}}-1)} - T \cdot \alpha_{j_2}^{(2i_{\mathrm{dif}})}}{\beta_{j_2}^{(2i_{\mathrm{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\mathrm{dif}})}}$.*

Let T and i_{dif} be the values defined in the above claim. Consider the step when δ_{j_2} is generated. We know the j_2-th query to H has been made, and thus $\boldsymbol{\alpha}_{j_2}$ and $\boldsymbol{\beta}_{j_2}$ are determined. Also, since $j_1 < j_2$, the j_1-th query to H has returned, and thus $\boldsymbol{\alpha}_{j_1}$, $\boldsymbol{\alpha}_{j_2}$, and δ_{j_1} are determined. Therefore, we know i_{dif} and T are also determined. Thus, we know δ_{j_2} is picked uniformly at random from

$\mathbb{Z}_p^*$ independent of i_{dif}, $\boldsymbol{\alpha}_{j_1}$, $\boldsymbol{\alpha}_{j_2}$, $\boldsymbol{\beta}_{j_1}$, $\boldsymbol{\beta}_{j_2}$, δ_{j_1}, and T. Then, by the above claim,

$$\begin{aligned}
\mathsf{Pr}[E'^{(3)}_{(j_1,j_2)}] &\leqslant \mathsf{Pr}\left[\begin{array}{c} \alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}})} \neq 0 \\ \wedge\ \beta_{j_2}^{(2i_{\mathrm{dif}}-1)} - T\cdot\beta_{j_2}^{(2i_{\mathrm{dif}})} \neq 0 \end{array} \wedge\ \delta_{j_2} = \frac{\alpha_{j_2}^{(2i_{\mathrm{dif}}-1)} - T\cdot\alpha_{j_2}^{(2i_{\mathrm{dif}})}}{\beta_{j_2}^{(2i_{\mathrm{dif}}-1)} - T\cdot\beta_{j_2}^{(2i_{\mathrm{dif}})}}\right] \\
&\leqslant \mathsf{Pr}\left[\delta_{j_2} = \frac{\alpha_{j_2}^{(2i_{\mathrm{dif}}-1)} - T\cdot\alpha_{j_2}^{(2i_{\mathrm{dif}})}}{\beta_{j_2}^{(2i_{\mathrm{dif}}-1)} - T\cdot\beta_{j_2}^{(2i_{\mathrm{dif}})}} \;\middle|\; \begin{array}{c} \alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}})} \neq 0 \\ \wedge\ \beta_{j_2}^{(2i_{\mathrm{dif}}-1)} - T\cdot\beta_{j_2}^{(2i_{\mathrm{dif}})} \neq 0 \end{array}\right] \\
&\leqslant \frac{1}{p-1}\,.
\end{aligned}$$

4.2.4 *Proof of Claim 4.2.8*

This proof relies on the following simple lemma, which we first state and prove.

Lemma 4.2.9. *Let p be a prime number. Let $a, b, c, d \in \mathbb{Z}_p$ be arbitrary values such that $a\cdot d \neq c\cdot b$. Then, for any $T \in \mathbb{Z}_p$ such that $a + T\cdot b = 0$, we have $c + T\cdot d \neq 0$.*

Proof. Since $a + T\cdot b = 0$ and $a\cdot d \neq c\cdot b$, we have

$$0 = d(a + T\cdot b) = a\cdot d + T\cdot b\cdot d \neq b\cdot c + T\cdot b\cdot d = b(c + T\cdot d),$$

which implies $c + T\cdot d \neq 0$. □

Proof of Claim 4.2.8. Consider $j_1, j_2 \in [Q_{\mathrm{H}}]$ such that $j_1 < j_2$. Suppose $E'^{(3)}_{j_1,j_2}$ occurs. We know the events $E^{(3)}_{(j_1,j_2)}$, F_{j_1}, and F_{j_2} occur. Since $E^{(3)}_{j_1,j_2}$ occurs, let i_{dif} be the smallest index in $\mathcal{I}^{(j_1)}_{\mathrm{unk}} \cap \mathcal{I}^{(j_2)}_{\mathrm{unk}}$ such that $\alpha_{j_1}^{(2i_{\mathrm{dif}})}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}}-1)} \neq \alpha_{j_1}^{(2i_{\mathrm{dif}}-1)}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}})}$ and $\alpha_{j_2}^{(2i_{\mathrm{dif}})}\cdot\beta_{j_2}^{(2i_{\mathrm{dif}}-1)} \neq \alpha_{j_2}^{(2i_{\mathrm{dif}}-1)}\cdot\beta_{j_2}^{(2i_{\mathrm{dif}})}$.

We first show that $\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}})} \neq 0$. Suppose $\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}})} = 0$. Since $\alpha_{j_1}^{(2i_{\mathrm{dif}})}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}}-1)} \neq \alpha_{j_1}^{(2i_{\mathrm{dif}}-1)}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}})}$, by Lemma 4.2.9, we know

$$\alpha_{j_1}^{(2i_{\mathrm{dif}}-1)} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}}-1)} \neq 0\,.$$

Therefore, we have

$$\begin{aligned}
&\alpha_{j_1}^{(2i_{\mathrm{dif}}-1)} + c_{i_{\mathrm{dif}}}\cdot\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\left(\beta_{j_1}^{(2i_{\mathrm{dif}}-1)} + c_{i_{\mathrm{dif}}}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}})}\right) \\
&= \alpha_{j_1}^{(2i_{\mathrm{dif}}-1)} - \delta_{j_1}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}}-1)} + c_{i_{\mathrm{dif}}}\left(\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\cdot\beta_{j_1}^{(2i_{\mathrm{dif}})}\right) \\
&= \alpha_{j_1}^{(2i_{\mathrm{dif}}-1)} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}}-1)} \neq 0\,.
\end{aligned}$$

However, since F_{j_1} occurs, we know (F1) holds for $j = j_1$, which yields a contradiction. Thus, we have $\alpha_{j_1}^{(2i_{\mathrm{dif}})} - \delta_{j_1}\beta_{j_1}^{(2i_{\mathrm{dif}})} \neq 0$.

Similarly, we have $\alpha_{j_2}^{(2i_{\text{dif}})} - \delta_{j_2}\beta_{j_2}^{(2i_{\text{dif}})} \neq 0$. Then, since F_{j_1} and F_{j_2} both occur, we know (F1) holds for $j = j_1$ and $j = j_2$, and thus

$$\frac{\alpha_{j_1}^{(2i_{\text{dif}}-1)} - \delta_{j_1} \cdot \beta_{j_1}^{(2i_{\text{dif}}-1)}}{\alpha_{j_1}^{(2i_{\text{dif}})} - \delta_{j_1} \cdot \beta_{j_1}^{(2i_{\text{dif}})}} = c_{i_{\text{dif}}} = \frac{\alpha_{j_2}^{(2i_{\text{dif}}-1)} - \delta_{j_2} \cdot \beta_{j_2}^{(2i_{\text{dif}}-1)}}{\alpha_{j_2}^{(2i_{\text{dif}})} - \delta_{j_2} \cdot \beta_{j_2}^{(2i_{\text{dif}})}} \ .$$

Denote $T = \frac{\alpha_{j_1}^{(2i_{\text{dif}}-1)} - \delta_{j_1} \cdot \beta_{j_1}^{(2i_{\text{dif}}-1)}}{\alpha_{j_1}^{(2i_{\text{dif}})} - \delta_{j_1} \cdot \beta_{j_1}^{(2i_{\text{dif}})}}$ and we have

$$\alpha_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \alpha_{j_2}^{(2i_{\text{dif}})} - \delta_{j_2}(\beta_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\text{dif}})}) = 0 \ . \tag{4.5}$$

We now show that $\beta_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\text{dif}})} \neq 0$. Suppose $\beta_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\text{dif}})} = 0$. Since $\alpha_{j_2}^{(2i_{\text{dif}})} \cdot \beta_{j_2}^{(2i_{\text{dif}}-1)} \neq \alpha_{j_2}^{(2i_{\text{dif}}-1)} \cdot \beta_{j_2}^{(2i_{\text{dif}})}$, by Lemma 4.2.9, we know $\alpha_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \alpha_{j_2}^{(2i_{\text{dif}})} \neq 0$ and

$$\alpha_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \alpha_{j_2}^{(2i_{\text{dif}})} - \delta_{j_2}(\beta_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\text{dif}})}) = \alpha_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \alpha_{j_2}^{(2i_{\text{dif}})} \neq 0,$$

which contradicts (4.5). Therefore, we have

$$\beta_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\text{dif}})} \neq 0 \ ,$$

and from (4.5), it holds that

$$\delta_{j_2} = \frac{\alpha_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \alpha_{j_2}^{(2i_{\text{dif}})}}{\beta_{j_2}^{(2i_{\text{dif}}-1)} - T \cdot \beta_{j_2}^{(2i_{\text{dif}})}} \ .$$

□

Chapter 5

EFFICIENT BLIND SIGNATURES IN THE GGM

5.1 Construction and Security

This section introduces our first blind signature scheme, BS_1, which relies on a prime-order cyclic group and a hash function H. We describe this scheme formally in Figure 5.1. Roughly, it extends (blind) Schnorr Signatures by sending an additional group element $Y = X^y$ in the first round. Then, the signer's final response to challenge c reveals y along with $s = a + cxy$. We also note that we could consider a variant of the scheme where the signature consists of $\sigma = (A', s', y')$, where A' replaces c'.

SECURITY ANALYSIS. First off, we observe that the protocol is blind.

Theorem 5.1.1. *For a group generation algorithm* GGen*, the blind signature scheme* $\mathsf{BS}_1[\mathsf{GGen}]$ *is perfectly blind.*

Proof of Theorem 5.1.1. Let $\mathcal{A}$ be an adversary playing the $\mathrm{Blind}^{\mathcal{A}}_{\mathsf{BS}_1[\mathsf{GGen}]}$ game. Without loss of generality, we can assume the randomness of $\mathcal{A}$ is fixed and $\mathcal{A}$ always finishes both signing sessions and receives valid signatures (σ_0, σ_1). [1]

Define the view of $\mathcal{A}$ after its execution as $\pi = (X, m_0, m_1, T_0, T_1, \sigma_0, \sigma_1)$, where $T_i := (A_i, Y_i, c_i, s_i, y_i)$, denoting the transcripts learned from interactions with the i-th signing session and $\sigma_i = (c'_i, s'_i, y'_i)$. Since the randomness of $\mathcal{A}$ is fixed, the only randomness left is the randomness in U_1 and U_2. Denote $\eta := (r_1^{(0)}, r_2^{(0)}, \gamma^{(0)}, r_1^{(1)}, r_2^{(1)}, \gamma^{(1)})$ as the total randomness. To prove the theorem, we need only show that the distribution of π is identical in both the case $b = 0$ and $b = 1$. We prove this by showing that for any fixed view Δ such that $\mathsf{Pr}[\pi = \Delta | b = 1] > 0$, there exists a unique value of the randomness η that makes $\pi = \Delta$ for the cases $b = 0$ and $b = 1$.

[1] Since the output of each query to U_1 that does not return $\perp$ is uniformly random over $\mathbb{Z}_p$, we know the behavior of $\mathcal{A}$ is identical in both the case $b = 0$ and $b = 1$ before $\mathcal{A}$ receives the valid signature (σ_0, σ_1). Therefore, we know the probability that $\mathcal{A}$ returns before receiving (σ_0, σ_1) or receives $(\perp, \perp)$ after finishing both signing sessions is equal in both the case $b = 0$ and $b = 1$, which means we consider only the case where $\mathcal{A}$ receives valid signatures.

Algorithm $\mathsf{Setup}(1^\kappa)$:
$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$
Select $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_p$
Return $par \leftarrow (p, g, \mathrm{H})$

Algorithm $\mathsf{KeyGen}(par)$:
$(p, g, \mathrm{H}) \leftarrow par$
$x \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $X \leftarrow g^x$
$sk \leftarrow x$; $pk \leftarrow X$
Return (sk, pk)

Algorithm $\mathsf{S}_1(sk)$:
$x \leftarrow sk$; $X \leftarrow g^x$
$a \leftarrow\!\!\$\ \mathbb{Z}_p$; $y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$A \leftarrow g^a$; $Y \leftarrow X^y$
$\mathsf{st}^s \leftarrow (a, y, x)$; $\mathsf{msg}_1 \leftarrow (A, Y)$
Return $(\mathsf{st}^s, \mathsf{msg}_1)$

Algorithm $\mathsf{S}_2(\mathsf{st}^s, c)$:
$(a, y, x) \leftarrow \mathsf{st}^s$
$s \leftarrow a + c \cdot y \cdot x$
Return $\mathsf{msg}_2 \leftarrow (s, y)$

Algorithm $\mathsf{U}_1(pk, \mathsf{msg}_1, m)$:
$X \leftarrow pk$; $(A, Y) \leftarrow \mathsf{msg}_1$
$r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p$; $\gamma \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$Y' \leftarrow Y^\gamma$; $A' \leftarrow g^{r_1} \cdot A^\gamma \cdot Y'^{r_2}$
$c' \leftarrow H(A' \,\|\, Y' \,\|\, m)$
$c \leftarrow c' + r_2$
$\mathsf{st}^u \leftarrow (c, c', r_1, \gamma, X, Y, A)$
Return (st^u, c)

Algorithm $\mathsf{U}_2(\mathsf{st}^u, \mathsf{msg}_2)$:
$(c, c', r_1, \gamma, X, Y, A) \leftarrow \mathsf{st}^u$
$(s, y) \leftarrow \mathsf{msg}_2$
If $y = 0$ or $Y \neq X^y$ or $g^s \neq A \cdot Y^c$
then return $\perp$
$s' \leftarrow \gamma \cdot s + r_1$; $y' \leftarrow \gamma \cdot y$
Return $\sigma \leftarrow (c', s', y')$

Algorithm $\mathsf{Ver}(pk, \sigma, m)$:
$(c, s, y) \leftarrow \sigma$
If $y = 0$ then return 0
$Y \leftarrow X^y$; $A \leftarrow g^s \cdot Y^{-c}$
If $c \neq \mathrm{H}(A \,\|\, Y \,\|\, m)$ then return 0
Return 1

Figure 5.1: The blind signature scheme $\mathsf{BS}_1 = \mathsf{BS}_1[\mathsf{GGen}]$.

For both the cases $b = 0$ and $b = 1$, we now show that $\pi = \Delta$ if and only if for each $i \in \{0, 1\}$, it holds that

$$\begin{aligned} \gamma^{(i)} &= {y'_{b_i}}^{\Delta} / y_i^{\Delta} \ , \\ r_1^{(i)} &= {s'_{b_i}}^{\Delta} - \gamma^{(i)} \cdot s_i^{\Delta} \ , \\ r_2^{(i)} &= c_i^{\Delta} - {c'_{b_i}}^{\Delta} \ , \end{aligned} \tag{5.1}$$

where the superscript $(\cdot)^\Delta$ represents the corresponding value in Δ. From the algorithms U_1 and U_2, it is clear that the "only if" part holds. For the "if" part, suppose (5.1) holds. Since the

randomness of $\mathcal{A}$ is fixed, the view of $\mathcal{A}$ can differ only on the outputs c_0, c_1 from the oracle U_1 or the output (σ_0, σ_1) from the oracle U_2. Since both signatures in Δ are valid, we have

$$A_i^{\Delta} = g^{s_i^{\Delta}} X^{\Delta^{-c_i^{\Delta} \cdot y_i^{\Delta}}} \ , \ Y_i^{\Delta} = X^{\Delta^{y_i^{\Delta}}} \ , \tag{5.2}$$

$$c'_{b_i}{}^{\Delta} = \mathrm{H}(g^{s'_{b_i}{}^{\Delta}} X^{\Delta^{-y'_{b_i}{}^{\Delta} \cdot c'_{b_i}{}^{\Delta}}} \parallel X^{\Delta^{y'_{b_i}{}^{\Delta}}} \parallel m_{b_i}^{\Delta}) \ . \tag{5.3}$$

For c_i where $i \in \{0, 1\}$, suppose the values in the view of $\mathcal{A}$ that have already determined when c_i is generated, which must include (X, m_i, A_i, Y_i), are consistent with Δ. By (5.1), we have

$$\begin{aligned}
c_i &= r_2^{(i)} + \mathrm{H}(g^{r_1^{(i)}} A_i^{\gamma^{(i)}} Y_i^{\gamma^{(i)} \cdot r_2^{(i)}} \parallel Y_i^{\gamma^{(i)}} \parallel m_{b_i}) \\
&= r_2^{(i)} + \mathrm{H}(g^{r_1^{(i)}} A_i^{\Delta^{\gamma^{(i)}}} Y_i^{\Delta^{\gamma^{(i)} \cdot r_2^{(i)}}} \parallel Y_i^{\Delta^{\gamma^{(i)}}} \parallel m_{b_i}^{\Delta}) \\
&= r_2^{(i)} + \mathrm{H}(g^{\gamma^{(i)} \cdot s_i^{\Delta} + r_1^{(i)}} X^{\Delta^{-y_i^{\Delta} \cdot \gamma^{(i)} \cdot (c_i^{\Delta} - r_2^{(i)})}} \parallel X^{\Delta^{y_i^{\Delta} \cdot \gamma^{(i)}}} \parallel m_{b_i}^{\Delta}) \\
&= r_2^{(i)} + \mathrm{H}(g^{s'_{b_i}{}^{\Delta}} X^{\Delta^{-y'_{b_i}{}^{\Delta} \cdot c'_{b_i}{}^{\Delta}}} \parallel X^{\Delta^{y'_{b_i}{}^{\Delta}}} \parallel m_{b_i}^{\Delta}) \\
&= r_2^{(i)} + c'_{b_i}{}^{\Delta} = c_i^{\Delta} \ ,
\end{aligned}$$

where the third equality is due to (5.2), the fourth equality is due to (5.1), and the final equality is due to (5.3). Then, consider the step when (σ_0, σ_1) is output. Suppose the current view, which contains T_i, is consistent with Δ. By (5.1), we have

$$\begin{aligned}
y'_{b_i} &= \gamma^{(i)} \cdot y_i = \gamma^{(i)} \cdot y_i^{\Delta} = y'_{b_i}{}^{\Delta} \ , \\
s'_{b_i} &= r_1^{(i)} + \gamma^{(i)} \cdot s_i = r_1^{(i)} + \gamma^{(i)} \cdot s_i^{\Delta} = s'_{b_i}{}^{\Delta} \ , \\
c'_{b_i} &= c_i - r_2^{(i)} = c_i^{\Delta} - r_2^{(i)} = c'_{b_i}{}^{\Delta} \ .
\end{aligned}$$

which implies $(\sigma_0, \sigma_1) = (\sigma_0^{\Delta}, \sigma_1^{\Delta})$. Therefore, by induction, if (5.1) holds, we know $\pi = \Delta$. □

Our main result shows OMUF security of BS_1 in the *generic-group model* (GGM) following Shoup's original formalization [118], which encodes every group element with a random label. To this end, we present in Figure 5.2 a game describing a GGM-version of OMUF security for BS_1, adapting the one from Section 3.3. We also define a corresponding advantage $\mathsf{Adv}^{\text{omuf-ggm}}_{\mathsf{BS}_1[\mathsf{GGen}]}(\mathcal{A}, \kappa)$ to measure the probability that $\mathcal{A}$ wins the game. Note that to keep notation homogenous, it is convenient to allow the game to depend on $\mathbb{G}$, although the game itself only makes use of the order of the group. The game also models the hash function H as a random oracle, to which the adversary is given oracle access.

Game $\text{OMUF-GGM}^{\mathcal{A}}_{\mathsf{BS}_1[\mathsf{GGen}]}(\kappa)$:

$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$

$\text{sid} \leftarrow 0$; $\ell \leftarrow 0$; $\mathcal{I}_{\text{fin}} \leftarrow \varnothing$; $\mathsf{Cur} \leftarrow \varnothing$

$\Xi \leftarrow ()$; $T \leftarrow ()$

$\{(m_k, \sigma_k)\}_{k \in [\ell+1]} \leftarrow\!\!\$\ \mathcal{A}^{\Pi, \mathrm{S}_1, \mathrm{S}_2, \mathrm{H}}(p, \Phi(1), \Phi(x))$

If $\exists\ k_1 \neq k_2$ such that $(m_{k_1}, \sigma_{k_1}) = (m_{k_2}, \sigma_{k_2})$ then

  Return 0

If $\exists\ k \in [\ell + 1]$ such that $y^*_k = 0$

  or $c_k \neq \mathrm{H}(\Phi(s_k - c_k \cdot y_k \cdot x) \,\|\, \Phi(y_k \cdot x) \,\|\, m_i)$

where $(c_k, s_k, y_k) = \sigma_k$ then return 0

Return 1

Oracle $\Phi(v)$:

If $v \in \mathsf{Cur}$ then return $\Xi(v)$

$\Xi(v) \leftarrow\!\!\$\ \{0,1\}^{\log(p)} \backslash \Xi(\mathsf{Cur})$

$\mathsf{Cur} \leftarrow \mathsf{Cur} \cap \{v\}$

Return $\Xi(v)$

Oracle $\Pi(\xi, \xi', b)$:

If $\exists v, v' \in \mathsf{Cur}$ such that $\xi = \Xi(v)$ and $\xi' = \Xi(v')$ then

  Return $\Phi(v + (-1)^b v')$

Else return $\perp$

Oracle S_1 :

$\text{sid} \leftarrow \text{sid} + 1$

$a_{\text{sid}} \leftarrow\!\!\$\ \mathbb{Z}_p$; $y_{\text{sid}} \leftarrow\!\!\$\ \mathbb{Z}^*_p$

$\mathsf{st}^s_{\text{sid}} \leftarrow (a_{\text{sid}}, y_{\text{sid}})$

$\mathsf{msg}_1 \leftarrow (\Phi(a_{\text{sid}}), \Phi(x y_{\text{sid}}))$

Return $(\text{sid}, \mathsf{msg}_1)$

Oracle $\mathrm{S}_2(i, c_i)$:

If $i \notin [\text{sid}] \backslash \mathcal{I}_{\text{fin}}$ then

  Return $\perp$

$(a_i, y_i) \leftarrow \mathsf{st}^s_i$

$s_i \leftarrow a_i + c_i \cdot y_i \cdot x$

$\mathsf{msg}_2 \leftarrow (s_i, y_i)$

$\mathcal{I}_{\text{fin}} \leftarrow \mathcal{I}_{\text{fin}} \cup \{i\}$

$\ell \leftarrow \ell + 1$

Return msg_2

Oracle $\mathrm{H}(\text{str})$:

If $T(\text{str}) = \perp$ then

  $T(\text{str}) \leftarrow\!\!\$\ \mathbb{Z}_p$

Return $T(\text{str})$

Figure 5.2: The OMUF security game in GGM for the blind signature scheme $\mathsf{BS}_1[\mathsf{GGen}]$.

The following theorem states our main result in the form of a reduction to WFROS and is proved in Section 5.2.

Theorem 5.1.2 (OMUF Security of BS_1)**.** *Let* GGen *be a group generation algorithm. For any adversary* $\mathcal{A}$ *for the* $\text{OMUF-GGM}^{\mathsf{BS}_1[\mathsf{GGen}]}(\kappa)$ *game making at most* Q_Π *queries to* Π*,* Q_{S_1} *queries to* S_1*, and* Q_H *queries to the random oracle* H*, there exists an adversary* $\mathcal{B}$ *for the* $\text{WFROS}_{Q_{\mathrm{S}_1}, p}$ *problem, where* p *denotes the size of the group generated by* $\mathsf{GGen}(1^\kappa)$*, making at most* $Q_\mathrm{H} + Q_{\mathrm{S}_1} + 1$

queries to the random oracle H *such that*

$$\mathsf{Adv}^{\text{omuf-ggm}}_{\mathsf{BS}_1[\mathsf{GGen}]}(\mathcal{A},\kappa) \leqslant \mathsf{Adv}^{\text{wfros}}_{Q_{\mathrm{S}_1},p}(\mathcal{B}) + \frac{Q_\Phi(Q_\Phi + 2Q_{\mathrm{H}} + 2Q_{\mathrm{S}_1} + 2)}{p - (1 + Q_{\mathrm{S}_1} + Q_\Phi^2)} ,$$

where Q_Φ *is the maximum number of queries to* Φ *during the game* OMUF-GGM*, and we have* $Q_\Phi = Q_\Pi + 4Q_{\mathrm{S}_1} + 4$.

By Theorem 4.1.1, we have the following corollary.

Corollary 5.1.3. *Let* GGen *be a group generation algorithm. For any adversary* $\mathcal{A}$ *playing game* $\text{OMUF-GGM}^{\mathsf{BS}_1[\mathsf{GGen}]}(\lambda)$ *making at most* Q_Π *queries to* Π*,* Q_{S_1} *queries to* S_1*, and* Q_{H} *queries to the random oracle* H*, we have*

$$\mathsf{Adv}^{\text{omuf-ggm}}_{\mathsf{BS}_1[\mathsf{GGen}]}(\mathcal{A},\kappa) \leqslant \frac{2Q_\Phi(Q_\Phi + 2Q_{\mathrm{H}} + 2Q_{\mathrm{S}_1} + 2)}{2^\kappa - (1 + Q_{\mathrm{S}_1} + Q_\Phi^2)} ,$$

where $Q_\Phi = Q_\Pi + 4Q_{\mathrm{S}_1} + 4$.

We note in particular that the concrete security of BS_1 in the GGM is comparable to that of the discrete logarithm problem, in that $Q_\Phi = \Omega(\min\{\sqrt{p}, p/Q_{\mathrm{H}}, p/Q_{\mathrm{S}_1}\})$ is necessary to break security with constant probability.

5.2 Proof of OMUF (Theorem 5.1.2)

Let us fix an adversary $\mathcal{A}$ that makes (without loss of generality) exactly Q_Π queries to Π, Q_{S_1} queries to S_1, and Q_{H} queries to the random oracle H. Without loss of generality, assume it also makes exactly one query (i, c_i) to S_2 for each $i \in [Q_{\mathrm{S}_1}]$. Then, after $\mathcal{A}$ returns, we know $\ell = Q_{\mathrm{S}_1}$ and $\mathcal{I}_{\text{fin}} = [Q_{\mathrm{S}_1}]$. Also, it is clear that the overall number of queries to Φ in $\text{OMUF-GGM}^{\mathcal{A}}_{\mathsf{BS}_1}$ is at most $Q_\Phi := Q_\Pi + 4Q_{\mathrm{S}_1} + 4$.

We prove the theorem by going through a series of games, from Game_0 to Game_4, where Game_0 is the $\text{OMUF-GGM}^{\mathcal{A}}_{\mathsf{BS}_1}$ game and Game_4 is an intermediate game that enables an easier reduction to WFROS. Here, however, we first introduce Game_4 and Lemma 5.2.1 and then discuss the reduction to WFROS, which is the core of the proof. We leave the definition of the intermediate games between Game_0 to Game_4 to the proof of Lemma 5.2.1. The game-hopping argument is non-trivial, but it follows the same blueprint as in [12] and is hence deferred to Section 5.2.1.

Game Game_4:
$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$
$\mathrm{sid} \leftarrow 0$; $\ell \leftarrow 0$; $\mathcal{S} \leftarrow \varnothing$; $\mathsf{Cur} \leftarrow \varnothing$
$\Xi \leftarrow ()$; $T \leftarrow ()$
$\{(m_k, \sigma_k)\}_{k \in [\ell+1]} \leftarrow\!\!\$\ \mathcal{A}^{\Pi, \mathrm{S}_1, \mathrm{S}_2, \mathrm{H}}(p, \Phi(1), \Phi(\mathsf{X}))$
If $\exists\ k_1 \neq k_2$ such that $(m_{k_1}, \sigma_{k_1}) = (m_{k_2}, \sigma_{k_2})$ then
  Return 0
If $\exists\ k \in [\ell+1]$ such that $y_k^* = 0$
  or $c_k \neq \mathrm{H}(\Phi(s_k - c_k \cdot y_k \cdot \mathsf{X}) \,\|\, \Phi(y_k \cdot \mathsf{X}) \,\|\, m_i)$
where $(c_k, s_k, y_k) = \sigma_k$ then return 0
Return 1

Oracle $\Phi(P)$:
If $\exists P' \in \mathsf{Cur}$ such that $P =_L P'$ then
  Return $\Xi(P')$
$\Xi(P) \leftarrow\!\!\$\ \{0,1\}^{\lceil \log(p) \rceil} \backslash \Xi(\mathsf{Cur})$
$\mathsf{Cur} \leftarrow \mathsf{Cur} \cap \{P\}$
Return $\Xi(P)$

Oracle $\Pi(\xi, \xi', b)$:
If $\exists P, P' \in \mathsf{Cur}$ such that $\xi = \Xi(P)$
  and $\xi' = \Xi(P')$ then
  Return $\Phi(P + (-1)^b P')$
Else return $\perp$

Oracle S_1 :
$\mathrm{sid} \leftarrow \mathrm{sid} + 1$
$\mathsf{msg}_1 \leftarrow (\Phi(\mathsf{A}_{sid}), \Phi(\mathsf{Y}_{\mathrm{sid}}))$
Return $(\mathrm{sid}, \mathsf{msg}_1)$

Oracle $\mathrm{S}_2(i, c_i)$:
If $i \notin [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$ then return $\perp$
$s_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $y_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$R_1 \leftarrow \mathsf{A}_i + c_i \mathsf{Y}_i - s_i$
$R_2 \leftarrow \mathsf{Y}_i - y_i \mathsf{X}$
$L \leftarrow L \cup \{R_1, R_2\}$
$\mathsf{msg}_2 \leftarrow (s_i, y_i)$
If $\exists\ P_1, P_2 \in \mathsf{Cur}$ such that
  $P_1 \neq P_2$ and $P_1 =_L P_2$
then **abort game**
$\mathcal{I}_{\mathrm{fin}} \leftarrow \mathcal{I}_{\mathrm{fin}} \cup \{i\}$
$\ell \leftarrow \ell + 1$
Return msg_2

Oracle $\mathrm{H}(\mathrm{str})$:
If $T(\mathrm{str}) = \perp$ then
  $T(\mathrm{str}) \leftarrow\!\!\$\ \mathbb{Z}_p$
Return $T(\mathrm{str})$

Figure 5.3: The definition of Game_4. The symbols P and P' denote polynomials over variables X, $\{\mathsf{A}_i, \mathsf{Y}_i\}_{i \in [\mathrm{sid}]}$. Also, a new equality notation, "$=_L$", is used. We say $P_1 =_L P_2$ if and only if $P_1 - P_2$ can be represented as a linear combination of polynomials in L.

DEFINITION OF Game_4. The pseudocode description of Game_4 is given in Figure 5.3. The main difference from $\mathrm{OMUF\text{-}GGM}^{\mathcal{A}}_{\mathsf{BS}_1}$ is that the encoding oracle Φ takes as input a polynomial instead of an integer in $\mathbb{Z}_p$. (Note that the adversary cannot query Φ directly, and thus this difference is not directly surfaced.) This essentially captures the algebraic core of our proof.

Also, for a valid query (i, c_i) to S_2, the output values (s_i, y_i) are directly sampled uniformly from $\mathbb{Z}_p \times \mathbb{Z}_p^*$. Furthermore, when this happens, two polynomials, $R_1 = \mathsf{A}_i + c_i \cdot \mathsf{Y}_i - s_i$ and $R_2 = \mathsf{Y}_i - y_i \cdot \mathsf{X}$, are recorded in the set L. Then, in the encoding oracle Φ, two polynomials, P_1 and P_2, are considered to differ if and only if $P_1 \neq_L P_2$, where $P_1 =_L P_2$ means that $P_1 - P_2$ can be generated as a linear combination of polynomials in L. Still, $P_1 \neq_L P_2$ could occur when queries P_1 and P_2 are made to Φ, but they becomes equal (in the sense of "$=_L$") after L is updated. The game aborts when this happens. Overall, we prove the following lemma in Section 5.2.1.

Lemma 5.2.1. $\mathsf{Adv}^{\text{omuf-ggm}}_{\mathsf{BS}_1[\mathbb{G}]}(\mathcal{A}, \lambda) \leqslant \mathsf{Pr}[\mathrm{Game}_4^{\mathcal{A}} = 1] + \frac{Q_\Phi^2}{p-(1+Q_{\mathrm{S}_1}+Q_\Phi^2)}$.

REDUCTION TO WFROS. The core of the proof is to relate the probability of the adversary $\mathcal{A}$ winning Game_4 with the advantage of an adversary $\mathcal{B}$ winning the WFROS problem, as stated in the following lemma. The proof is given in Section 5.2.3.

Lemma 5.2.2. *For every λ, there exists an adversary $\mathcal{B}$ for the $\mathrm{WFROS}_{Q_{\mathrm{S}_1},p}$ problem, where $p = |\mathbb{G}_\lambda|$, making at most $Q_{\mathrm{H}} + Q_{\mathrm{S}_1} + 1$ queries to H such that*

$$\mathsf{Pr}[\mathrm{Game}_4^{\mathcal{A}} = 1] \leqslant \mathsf{Adv}^{\text{wfros}}_{Q_{\mathrm{S}_1},p}(\mathcal{B}) + \frac{(2Q_\Phi + 1)(Q_{\mathrm{H}} + Q_{\mathrm{S}_1} + 1)}{p - Q_\Phi} . \tag{5.4}$$

The statement of Theorem 5.1.2 follows by combining Lemmas 5.2.1 and 5.2.2.

5.2.1 *Proof of Lemma 5.2.1*

We prove the lemma by going through a serious of games.

$\mathrm{Game}_0^{\mathcal{A}}$: This is $\text{OMUF-GGM}^{\mathcal{A}}_{\mathsf{BS}_1}$ (Figure 5.2).

$\mathrm{Game}_1^{\mathcal{A}}$: This is defined in Figure 5.4 that only contains the dashed box. We introduce variables X, $\mathsf{A}_1, \mathsf{Y}_1, \ldots, \mathsf{A}_{Q_{\mathrm{S}_1}}, \mathsf{Y}_{Q_{\mathrm{S}_1}}$ in $\mathrm{Game}_1^{\mathcal{A}}$. Each variable is assigned a value, that is, X is assigned x, A_i is assigned a_i, and Y_i is assigned $y_i \cdot x$. The input to Φ is a polynomial P of variables $\mathsf{X}, \{\mathsf{A}_i, \mathsf{Y}_i\}_{i\in[Q_{\mathrm{S}_1}]}$ over $\mathbb{Z}_p$ instead of a single value $v \in \mathbb{Z}_p$ and the set Cur is a set of polynomials. Also, in Φ we check the equality of two polynomials by its evaluation on the assigned values, which is denoted by $=_{\text{eval}}$ (see Definition 5.2.3).

Definition 5.2.3. *For two ploynomial P and P' of the variables $\mathsf{X}_1, \ldots, \mathsf{X}_n$ over a field $\mathcal{F}$, suppose each X_i is assigned with a value $x_i \in \mathcal{F}$. We say $P =_{\text{eval}} P'$ if and only if $P(\mathsf{X}_1 = x_1, \ldots, \mathsf{X}_n = x_n) = P'(\mathsf{X}_1 = x_1, \ldots, \mathsf{X}_n = x_n)$.*

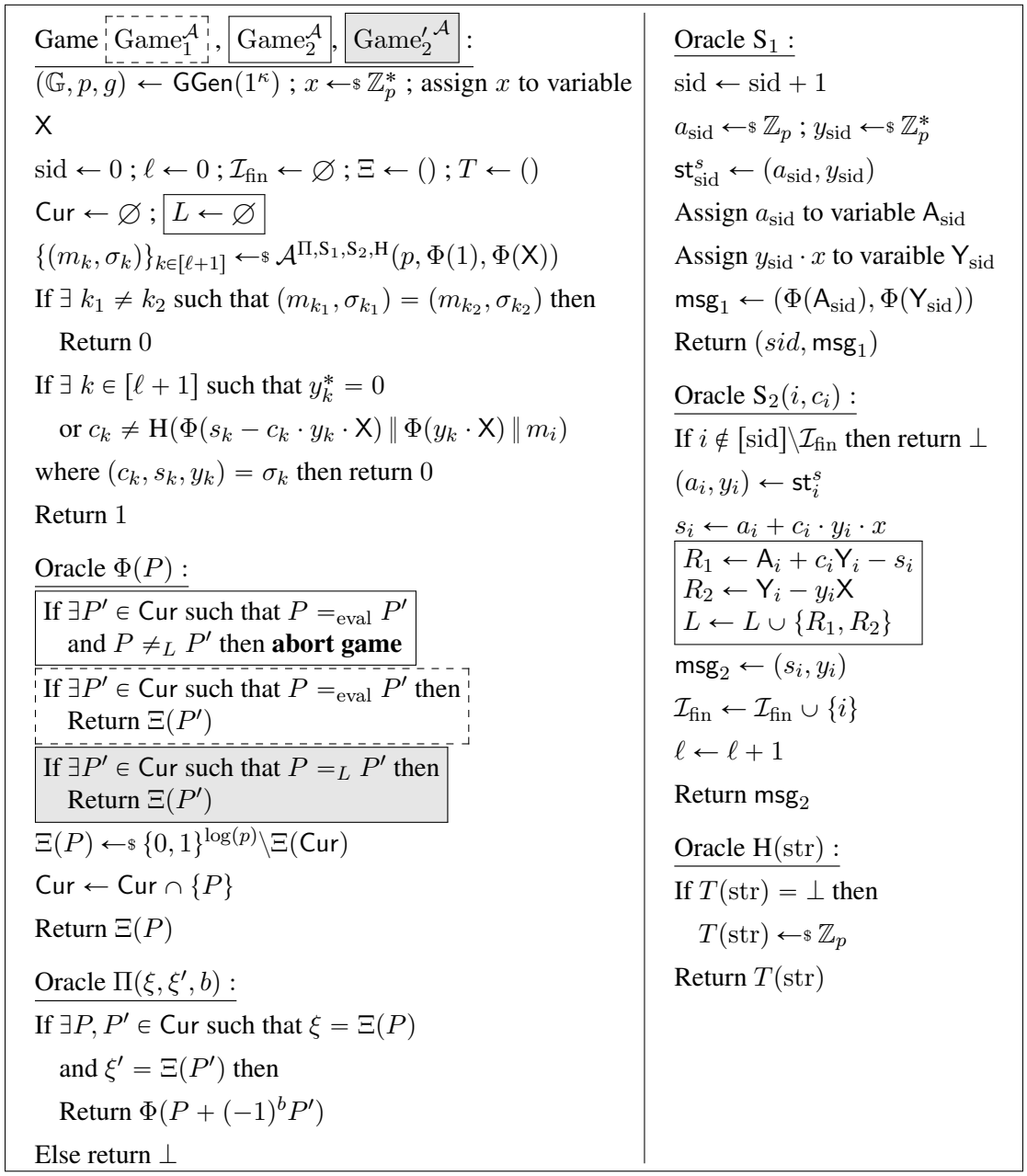

Figure 5.4: The definition for $\mathrm{Game}_1^{\mathcal{A}}$, $\mathrm{Game}_2^{\mathcal{A}}$, and $\mathrm{Game}_2'^{\mathcal{A}}$, where $\mathrm{Game}_1^{\mathcal{A}}$ only contains the dashed box, $\mathrm{Game}_2^{\mathcal{A}}$ contains all but the gray box, and $\mathrm{Game}_2'^{\mathcal{A}}$ contains all but the dashed box.

For convenience, we also have $P =_{\text{eval}} P(\mathsf{X}_1 = x_1, \ldots, \mathsf{X}_n = x_n)$.

It is easy to check that $=_{\text{eval}}$ *is an equivalence relation over the polynomials of the variables* $\mathsf{X}_1, \ldots, \mathsf{X}_n$.

We first show that the oracle Φ in $\mathrm{Game}_1^{\mathcal{A}}$ is well-defined, that is, for each query P to Φ, there exists at most one $P' \in \mathsf{Cur}$ such that $P =_{\text{eval}} P'$. Suppose there exists $P', P'' \in \mathsf{Cur}$ such that $P' \neq P''$, $P' =_{\text{eval}} P =_{\text{eval}} P''$. Suppose P'' is added to Cur after P'. Consider the query to Φ during which P'' is added to Cur. Since P' is already in Cur when P'' is added, we have $P' \neq_{\text{eval}} P''$, which yields a contradiction. Therefore, for each query to Φ, if there exists $P' \in \mathsf{Cur}$ such that $P =_{\text{eval}} P'$, then P' is the unique polynomial in Cur such that $P =_{\text{eval}} P'$.

We now show that the views of the adversary in Game_0 and Game_1 are identical. Define an intermediate game $\mathrm{Game}_1'^{\mathcal{A}}$ such that it is identical to $\mathrm{Game}_1^{\mathcal{A}}$ except each the polynomial P appear in the game is replaced by its evaluation value $P(\mathsf{X} = x, \mathsf{A}_1 = a_1, \mathsf{Y}_1 = y_1 \cdot x, \ldots, \mathsf{A}_{sid} = a_{sid}, \mathsf{Y}_{sid} = y_{sid} \cdot x)$. It is clear that $\mathrm{Game}_1'^{\mathcal{A}}$ is identical to $\mathrm{Game}_0^{\mathcal{A}}$. Also, since in the oracle Φ in $\mathrm{Game}_1^{\mathcal{A}}$, a polynomial P is considered equal or not equal to another polynomials by its evaluation value, the view of the adversary in Game_1 and Game_1' are identical. Thus, we know the views of the adversary in Game_0 and Game_1 are identical, which implies

$$\mathsf{Pr}[\mathrm{Game}_0^{\mathcal{A}} = 1] = \mathsf{Pr}[\mathrm{Game}_1^{\mathcal{A}} = 1] \,. \tag{5.5}$$

$\underline{\mathrm{Game}_2^{\mathcal{A}}}$: This is defined in Figure 5.4 by ignoring the graybox. A set L is introduced to record the information leaked to the adversary by S_2. For the query (i, c_i) to S_2, polynomials $R_1 = \mathsf{A}_i + c_i\mathsf{Y}_i - s_i$ and $R_2 = \mathsf{Y}_i - y_i\mathsf{X}$ are added to L. Suppose L is also recorded in $\mathrm{Game}_1^{\mathcal{A}}$. In $\mathrm{Game}_1^{\mathcal{A}}$, define the event E_1 as after an query P to Φ is made,

$$\exists P' \in \mathsf{Cur} \text{ such that } P =_{\text{eval}} P' \text{ and } P \neq_L P' \,.$$

Then, $\mathrm{Game}_2^{\mathcal{A}}$ is identical to $\mathrm{Game}_1^{\mathcal{A}}$ except it aborts when E_1 occurs and we have

$$\mathsf{Pr}[\mathrm{Game}_1^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_2^{\mathcal{A}} = 1] + \mathsf{Pr}[E_1] \,, \tag{5.6}$$

To bound $\mathsf{Pr}[E_1]$, for each $j \in [Q_\Phi]$, we denote the event $E_{1,j}$ in $\mathrm{Game}_1^{\mathcal{A}}$ as during the j-th query to Φ

$$\exists P' \in \mathsf{Cur} \text{ such that } P_j =_{\text{eval}} P' \text{ and } P_j \neq_L P' \,.$$

Then, we have $E_1 = \bigvee_{j \in [Q_\Phi]} E_{1,j}$. Denote $E'_{1,j} := E_{1,j} \bigwedge_{i \in [j]} (\neg E_{1,i})$. We now bound $\mathsf{Pr}[E'_{1,j}]$ for each $j \in [Q_\Phi]$.

We now fix a certain $j \in [Q_\Phi]$. Consider the step when the j-th query to Φ is made during $\mathrm{Game}_1^{\mathcal{A}}$. Denote the transcripts between the oracles and adversarys when the j-th query to Φ is made as π_j, which contains $\Phi(1), \Phi(\mathsf{X})$, and all the inputs and outputs of the queries to S_1, S_2, Π, and H made before the j-th query to Φ. For a certain transcript $\pi_j = \Delta$, for $1 \leqslant k \leqslant j$, denote the k-th query to Φ in Δ as P_k^Δ. From the transcript Δ, one can compute the set $\mathcal{I}_{\mathrm{fin}}$, Cur, and L at the step when the j-th query to Φ is been made. Denote them by $\mathcal{I}_{\mathrm{fin}}^\Delta$, Cur^Δ, and L^Δ. For each $i \in \mathcal{I}_{\mathrm{fin}}^\Delta$, denote the input and output of the query to the S_2 for the session i in the transcript Δ as c_i^Δ and (s_i^Δ, y_i^Δ). Also, from the transcript π_j, one can tell whether $E_{1,k}$ occurs or not for $k \in [j-1]$, since the event $E_{1,k}$ occurs if and only if $P_k \neq_L P'$ for all $P' \in \mathsf{Cur}$ but P_k is not added to Cur. Denote the value of sid when the j-th query to Φ is made as sid^Δ.

Denote $\mathcal{T}_j$ as the set of all transcripts Δ such that $\mathsf{Pr}[\pi_j = \Delta] > 0$ and none of $\{E_{1,k}\}_{k\in[j]}$ occurs given $\pi_j = \Delta$. We just need to bound $\mathsf{Pr}[E'_{1,j}|\pi_j = \Delta]$ for each $\Delta \in \mathcal{T}_j$.

We now fix a certain $\Delta \in \mathcal{T}_j$. For any polynomial P, denote the event F_P as $P =_{\mathrm{eval}} P_j$ and $P \neq_L P_j$. Then we know $E'_{1,j}$ implies one of $\{F_P\}_{P\in\mathsf{Cur}^\Delta}$ occurs and we have

$$\mathsf{Pr}[E'_{1,j}|\pi_j = \Delta] \leqslant \mathsf{Pr}\left[\bigvee_{P\in\mathsf{Cur}^\Delta} F_P|\pi_j = \Delta\right].$$

Therefore, it is left to bound $\mathsf{Pr}[F_P]$ for each $P \in \mathsf{Cur}^\Delta$.

We now fix a certain $\hat{P} \in \mathsf{Cur}^\Delta$. Since P_j^Δ and L^Δ are fixed in Δ, we can directly check whether $\hat{P} =_{L^\Delta} P_j^\Delta$ or not. If $\hat{P} =_{L^\Delta} P_j^\Delta$, then we have $\mathsf{Pr}[F_{\hat{P}}] = 0$. Therefore, we can assume $\hat{P} \neq_{L^\Delta} P_j^\Delta$. Then, we only need to bound the probability of $\hat{P} =_{\mathrm{eval}} P_j^\Delta$. Since we fix $\pi_j = \Delta$, the only randomness here is the values assigned to the random variables $\mathsf{X}, \{\mathsf{A}_i, \mathsf{Y}_i\}_{i\in[\mathrm{sid}^\Delta]}$. Denote the values as $\boldsymbol{\eta} := (x, a_1, y_1 \cdot x, \ldots, a_{\mathrm{sid}^\Delta}, y_{\mathrm{sid}^\Delta} \cdot x) \in \mathbb{Z}_p^{1+2\mathrm{sid}^\Delta}$, where $x, \{a_i, y_i\}_{i\in[\mathrm{sid}^\Delta]}$ are random variables sampled in the game, and we have $P =_{\mathrm{eval}} P(\mathsf{X} = \eta_1, \{\mathsf{A}_i = \eta_{2i}, \mathsf{Y}_i = \eta_{2i+1}\}_{i\in[\mathrm{sid}^\Delta]})$.

To bound $\mathsf{Pr}[\hat{P} =_{\mathrm{eval}} P_j^\Delta|\pi_j = \Delta]$, we first introduce Lemma 5.2.4 below. Then the proof structure can be described as follows. We first define a sequence of polynomials $D_0, D_1, \ldots, D_m$, $B_1, \ldots, B_{q+1}$ over variables $\mathsf{X}, \{\mathsf{A}_i, \mathsf{Y}_i\}_{i\in[\mathrm{sid}^\Delta]}$ such that $B_{q+1} := \hat{P} - P_j^\Delta$. Then, we try to apply Lemma 5.2.4 to bound the probability by showing η is uniformly distributed over $\mathcal{C}$, $\mathrm{Zero}(B_{q+1}) \cap \mathcal{C} \neq \varnothing$, and $B_{q+1} \notin \mathsf{Span}(\{1, B_1, \ldots, B_q\})$ given $\pi_j = \Delta$, where $\mathcal{C}$ is defined in Lemma 5.2.4.

Lemma 5.2.4 (Lemma 1 in [12]). *Let $D_1, \ldots, D_m, B_1, \ldots, B_{q+1}$ be polynomials in $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$*

of degree 1. Let

$$\mathcal{C} := \left(\bigcap_{i\in[q]} \mathrm{Zero}(B_i)\right) \setminus \left(\bigcup_{i\in[m]} \mathrm{Zero}(D_i)\right) ,$$

where $\mathrm{Zero}(P)$ *means the zero set of* P*. Assume* $\mathrm{Zero}(B_{q+1})\cap\mathcal{C} \neq \varnothing$ *and* $B_{q+1} \notin \mathsf{Span}(\{1, B_1, \ldots, B_q\})$*. If* $\boldsymbol{x}$ *is picked uniformly at random from* $\mathcal{C}$ *then*

$$\frac{p-m}{p^2} \leqslant \mathsf{Pr}[B_{q+1}(\boldsymbol{x}) = 0] \leqslant \frac{1}{p-m} .$$

Let $m := \mathrm{sid}^{\Delta} + 1 + |\mathsf{Cur}^{\Delta}|(|\mathsf{Cur}^{\Delta} - 1|)$. Denote $D_1 := \mathsf{X}$ and $D_{i+1} := \mathsf{Y}_i$ for $i \in [\mathrm{sid}^{\Delta}]$. For each $P, P' \in \mathsf{Cur}^{\Delta}$ such that $P \neq P'$, denote $D_{P,P'} := P - P'$. We can relable $\{D_{P,P'}\}_{P,P'\in\mathsf{Cur}^{\Delta},P\neq P'}$ to $D_{\mathrm{sid}^{\Delta}+2}, \ldots, D_m$.

Let $q := 2|\mathcal{I}^{\Delta}_{\mathrm{fin}}|$. For each $i \in \mathcal{I}^{\Delta}_{\mathrm{fin}}$, denote

$$B_{(i,1)} := \mathsf{A}_i + c_i^{\Delta}\mathsf{Y}_i - s_i^{\Delta} \ , \ B_{i,2} := \mathsf{Y}_i - y_i^{\Delta}\mathsf{X} .$$

We can relabel $\{B_{(i,1)}, B_{(i,2)}\}_{i\in\mathcal{I}^{\Delta}_{\mathrm{fin}}}$ to $B_1, \ldots, B_q$ and denote $B_{q+1} := \hat{P} - P_j^{\Delta}$. Here one thing to notice is that we have $L^{\Delta} = \{B_1, \ldots, B_q\}$.

Denote $\mathcal{C} := \left(\bigcap_{i\in[q]} \mathrm{Zero}(B_i)\right) \setminus \left(\bigcup_{i\in[m]} \mathrm{Zero}(D_i)\right)$ and we have the following claim. The proof of the claim is deferred to Section 5.2.2.

Claim 5.2.5. *In* $\mathrm{Game}_1^{\mathcal{A}}$*, for any* $\Delta \in \mathcal{T}_j$*, given* $\pi_j = \Delta$*, we have* $\boldsymbol{\eta}$ *is uniformly distributed over* $\mathcal{C}$*.*

We now continue to show that $\mathrm{Zero}(B_{q+1}) \cap \mathcal{C} \neq \varnothing$. If $\mathrm{Zero}(B_{q+1}) \cap \mathcal{C} = \varnothing$, since by the above claim η must be in $\mathcal{C}$ given $\pi_j = \Delta$, we know $B_{q+1} =_{\mathrm{eval}} B_{q+1}(\eta) \neq 0$, which implies $\mathsf{Pr}[\hat{P} =_{\mathrm{eval}} P_j | \pi_j = \Delta] = 0$. Therefore, we only need to consider the case when $\mathrm{Zero}(B_{q+1})\cap\mathcal{C} \neq \varnothing$.

We then show that $B_{q+1} \notin \mathsf{Span}(\{1, B_1, \ldots, B_q\})$. Since $\hat{P} \neq_{L^{\mathsf{Cur}}} P_j^{\Delta}$ and $L^{\Delta} = \{B_1, \ldots, B_q\}$, we know $B_{q+1} \notin \mathsf{Span}(\{B_1, \ldots, B_q\})$. If $B_{q+1} \in \mathsf{Span}(\{1, B_1, \ldots, B_q\})$, we know there exists a constant $\delta \in \mathbb{Z}_p$ such that $\delta \neq 0$ and $B_{q+1} + \delta \in \mathsf{Span}(\{B_1, \ldots, B_q\})$. Let $B' = B_{q+1} + \delta$. Then, we have for any $\boldsymbol{\eta}_0 \in \mathcal{C}$, $B'(\boldsymbol{\eta}) = 0$ and thus $B_{q+1}(\boldsymbol{\eta}) = B'(\boldsymbol{\eta}) - \delta = -\delta \neq 0$, which means $\mathrm{Zero}(B_{q+1}) \cap \mathcal{C} = \varnothing$. This contradicts with the above argument that $\mathrm{Zero}(B_{q+1}) \cap \mathcal{C} \neq \varnothing$. Therefore, we have $B_{q+1} \notin \mathsf{Span}(\{1, B_1, \ldots, B_q\})$.

Then, by the above claim, we can apply Lemma 5.2.4 here and we have

$$\mathsf{Pr}[\hat{P} =_{\mathrm{eval}} P_j^{\Delta} | \pi_j = \Delta] = \mathsf{Pr}[B_{q+1}(\boldsymbol{\eta}) = 0 \mid \pi_j = \Delta] \leqslant \frac{1}{p-m} .$$

Since $m = \mathrm{sid}^{\Delta} + 1 + |\mathsf{Cur}^{\Delta}|(|\mathsf{Cur}^{\Delta} - 1|) \leqslant 1 + Q_{\mathrm{S}_1} + Q_{\Phi}^2$, we have

$$\begin{aligned}\mathsf{Pr}[E'_{1,j}] &= \sum_{\Delta \in \mathcal{T}_j} \mathsf{Pr}[E'_{1,j} \ \wedge \ \pi_j = \Delta] \\ &= \sum_{\Delta \in \mathcal{T}_j} \mathsf{Pr}[\pi_j = \Delta] \sum_{\hat{P} \in \mathsf{Cur}^{\Delta}} \mathsf{Pr}[F_{\hat{P}} \mid \pi_j = \Delta] \leqslant \frac{Q_{\Phi}}{p - (1 + Q_{\mathrm{S}_1} + Q_{\Phi}^2)} \ .\end{aligned}$$

Therefore, we have $\mathsf{Pr}[E_1] = \sum_{j \in [Q_{\Phi}]} \mathsf{Pr}[E'_{1,j}] \leqslant \frac{Q_{\Phi}^2}{p - (1 + Q_{\mathrm{S}_1} + Q_{\Phi}^2)}$ and by (5.6)

$$\mathsf{Pr}[\mathrm{Game}_1^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_2^{\mathcal{A}} = 1] + \frac{Q_{\Phi}^2}{p - (1 + Q_{\mathrm{S}_1} + Q_{\Phi}^2)} \ . \tag{5.7}$$

$\underline{\mathrm{Game}_2'^{\mathcal{A}}}$: This is defined in Figure 5.4 by ignoring the dashed box. The only difference between $\mathrm{Game}_2^{\mathcal{A}}$ and $\mathrm{Game}_2'^{\mathcal{A}}$ is that in the oracle Φ the condition "$\exists P' \in \mathsf{Cur}$ such that $P =_{\mathrm{eval}} P'$" is changed to "$\exists P' \in \mathsf{Cur}$ such that $P =_L P'$". We will show that $P =_{\mathrm{eval}} P'$ is equivalent to $P =_L P'$ here in Game_2, and thus we know the view of adversary are identical in these two games.

In $\mathrm{Game}_2^{\mathcal{A}}$, consider an query P to the oracle Φ. Let P' be an arbitrary polynomial in Cur. Consider the step when the condition "$\exists P' \in \mathsf{Cur}$ such that $P =_{\mathrm{eval}} P'$" is checked. We now show that $P =_{\mathrm{eval}} P'$ is if and only if $P =_L P'$. Suppose $P =_{\mathrm{eval}} P'$. Since the game does not abort, it must hold that $P =_L P'$. Therefore, we know $P =_{\mathrm{eval}} P'$ implies $P =_L P'$.

On the other hand, we show the following lemma.

Lemma 5.2.6. *In* $\mathrm{Game}_2^{\mathcal{A}}$*, at any step of the execution, we have*

$$\forall \ P \in \mathsf{Span}(L) \ : \ P =_{\mathrm{eval}} 0 \ , \tag{5.8}$$

which implies for any two polynomials P, P' *of variables* X *and* $\{\mathsf{A}_i, \mathsf{Y}_i\}_{i \in [sid]}$

$$P =_L P' \textit{ implies } P =_{\mathrm{eval}} P' \ . \tag{5.9}$$

Proof. We just need show that for each $R \in L$, we have $R =_{\mathrm{eval}} 0$. From the description of S_2, we know

$$L = \{\mathsf{A}_i + c_i \mathsf{Y}_i - s_i, \mathsf{Y}_i - y_i \mathsf{X}\}_{i \in \mathcal{I}_{\mathrm{fin}}} \ .$$

For $R = \mathsf{A}_i + c_i \mathsf{Y}_i - s_i$, we have $R =_{\mathrm{eval}} a_i + c_i \cdot y_i \cdot x_i - s_i = 0$, since $s_i = a_i + c_i \cdot y_i \cdot x_i$. For $R = \mathsf{Y}_i - y_i \mathsf{X}$, we have $R =_{\mathrm{eval}} y_i \cdot x - y_i \cdot x = 0$. Therefore, we know the lemma holds. □ □

From the above lemma, we know $P =_L P'$ is equivalent to $P =_{\text{eval}} P'$ at the step in Φ when the condition "$\exists P' \in \mathsf{Cur}$ such that $P =_{\text{eval}} P'$". Therefore, we know the view of the adversary is identical in these two games, which implies

$$\mathsf{Pr}[\mathrm{Game}_2^{\mathcal{A}} = 1] = \mathsf{Pr}[\mathrm{Game}_2'^{\mathcal{A}} = 1] \ . \tag{5.10}$$

$\underline{\mathrm{Game}_3^{\mathcal{A}}}$: $\mathrm{Game}_3^{\mathcal{A}}$ is defined in Figure 5.5 by ignoring the dashed box, where the only difference from $\mathrm{Game}_2'^{\mathcal{A}}$ is the original abort condition is removed from Φ and a new abort condition is added to S_2. Also, in $\mathrm{Game}_3^{\mathcal{A}}$, since the new abort condition only uses the information L, we do not need to assign values to the variables anymore.

We first show that the oracle Φ in $\mathrm{Game}_3^{\mathcal{A}}$ is well-defined, that is, for each query P to Φ, there exists at most one $P' \in \mathsf{Cur}$ such that $P =_L P'$. Suppose during a query P to Φ in $\mathrm{Game}_2^{\mathcal{A}}$, the game does not abort and there exists $P', P''The \in \mathsf{Cur}$ such that $P' =_L P =_L P''$. Without loss of generality assume P'' is added to Cur after P'. If L is not updated after P'' is added to Cur, then by the description of Φ, we know $P' \neq_L P''$, which yields a contradiction. Otherwise, L is updated after P'' is added to Cur. Consider the last time L is updated in S_2. Since $P' =_L P''$ and $P', P'' \in \mathsf{Cur}$, we know $\mathrm{Game}_3^{\mathcal{A}}$ must abort in S_2, which yields a contradiction. Therefore, we know the oracle Φ in $\mathrm{Game}_3^{\mathcal{A}}$ is well-defined.

To show that the probability $\mathcal{A}$ wins $\mathrm{Game}_2'^{\mathcal{A}}$ is bounded by the probability $\mathcal{A}$ wins $\mathrm{Game}_3^{\mathcal{A}}$, we introduce an itermidiate game $\mathrm{Game}_3'^{\mathcal{A}}$ which is defined in Figure 5.5 containing everything. We first show that the probability $\mathcal{A}$ wins $\mathrm{Game}_2'^{\mathcal{A}}$ is bounded by the probability $\mathcal{A}$ wins $\mathrm{Game}_3'^{\mathcal{A}}$. Denote the event E_2 in $\mathrm{Game}_2'^{\mathcal{A}}$ as during a query to S_2 after L is updated,

$$\exists P_1, P_2 \in \mathsf{Cur} \text{ such that } P_1 \neq P_2 \text{ and } P_1 =_L P_2 \ .$$

Then, we have $\mathrm{Game}_3'^{\mathcal{A}}$ is identical to $\mathrm{Game}_2'^{\mathcal{A}}$ except it aborts when E_2 occurs, which implies

$$\mathsf{Pr}[\mathrm{Game}_2'^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_3'^{\mathcal{A}} = 1] + \mathsf{Pr}[E_2] \ , \tag{5.11}$$

We now show that $\mathsf{Pr}[E_2] = 0$. Suppose E_2 occurs. Then, we know at some timestep in $\mathrm{Game}_2'^{\mathcal{A}}$ there exists $P_1, P_2 \in \mathsf{Cur}$ such that $P_1 \neq P_2$ and $P_1 =_L P_2$. We first show that $P_1 \neq_{\text{eval}} P_2$. Suppose $P_1 =_{\text{eval}} P_2$. Without loss of generality, assume P_1 is added to Cur before P_2. Consider the step when P_2 is added to Cur. Since P_1 is already in Cur, we know $P_1 \neq_L P_2$. However, since $P_2 \neq_L P_1$ but $P_2 =_{\text{eval}} P_1$, the game aborts, which yields a contradiction. Thus, we know

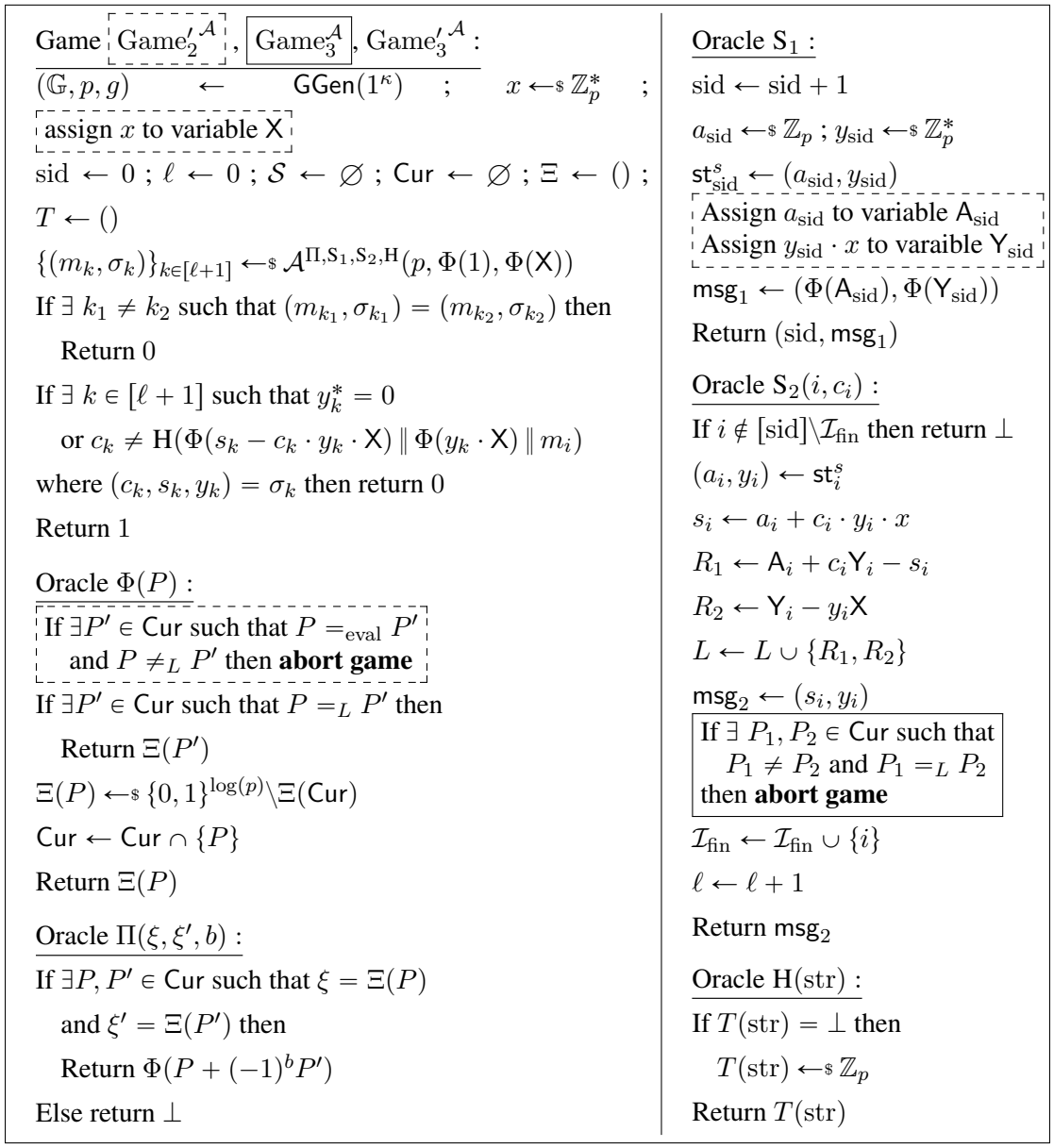

Figure 5.5: The definition for $\mathrm{Game}_3^{\mathcal{A}}$ and its difference from $\mathrm{Game}_2^{\mathcal{A}}$. $\mathrm{Game}_2^{\mathcal{A}}$ contains all but the solid boxes and $\mathrm{Game}_3^{\mathcal{A}}$ contains all but the dashed boxes. We also define an intermediate game $\mathrm{Game}_3'^{\mathcal{A}}$ which contains both dashed and solid boxes.

$P_1 \neq_{\mathrm{eval}} P_2$. Then, by Lemma 5.2.6, we know $P_1 \neq_L P_2$ at any timestep in $\mathrm{Game}_2'^{\mathcal{A}}$, which yields

a contradiction. Therefore, we know E_2 never occurs in $\mathrm{Game}_2'^{\mathcal{A}}$, which implies

$$\mathsf{Pr}[\mathrm{Game}_2'^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_3'^{\mathcal{A}} = 1] \ .$$

Also, since the only difference between $\mathrm{Game}_3'^{\mathcal{A}}$ and $\mathrm{Game}_3^{\mathcal{A}}$ is that $\mathrm{Game}_3'^{\mathcal{A}}$ might abort in Φ while $\mathrm{Game}_3^{\mathcal{A}}$ never abort in Φ, we have $\mathsf{Pr}[\mathrm{Game}_3'^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_3^{\mathcal{A}} = 1]$. Therefore, we have

$$\mathsf{Pr}[\mathrm{Game}_2'^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_3'^{\mathcal{A}} = 1] \leqslant \mathsf{Pr}[\mathrm{Game}_3^{\mathcal{A}} = 1] \ . \tag{5.12}$$

$\underline{\mathrm{Game}_4^{\mathcal{A}}}$: This is defined in Figure 5.6 by ignoring the dashed box. $\mathrm{Game}_4^{\mathcal{A}}$ is identical to $\mathrm{Game}_3^{\mathcal{A}}$, except the generation of $x, \{a_i, y_i, s_i\}_{i\in[sid]}$ are changed. More precisely, the sampling of x is removed from the main procedure, the sampling of a_{sid}, y_{sid} is removed from S_1, and in S_2, y_i is sampled from $\mathbb{Z}_p^*$ and s_i is sampled from $\mathbb{Z}_p$ instead of computing from a_i and y_i. The oracle Φ in $\mathrm{Game}_4^{\mathcal{A}}$ is well-defined, which can be shown using the same way as in $\mathrm{Game}_3^{\mathcal{A}}$.

We now show that the views of the adversary in Game_3 and Game_4 are identical. Since the value x and a_i are not used in $\mathrm{Game}_3^{\mathcal{A}}$ except the dashed box, we just need to show that the distribution of (s_i, y_i) are identical in $\mathrm{Game}_3^{\mathcal{A}}$ and $\mathrm{Game}_4^{\mathcal{A}}$ for each query (i, c_i) to S_2. Consider the step when the adversary makes a query (i, c_i) to S_2 in $\mathrm{Game}_3^{\mathcal{A}}$ and assume $i \in [sid]\backslash\mathcal{I}_{\text{fin}}$. The value y_i and a_i are not used anywhere in the game yet. Therefore, given the current transcript, the distribution of (s_i, y_i) is uniformly random in $\mathbb{Z}_p \times \mathbb{Z}_p^*$. Since $a_i \leftarrow s_i + c_i \cdot y_i \cdot x$ and s_i is uniformly in $\mathbb{Z}_p$ given y_i, we know the distribution of a_i is uniformly random in $\mathbb{Z}_p$ even given y_i. Therefore, the distribution of (a_i, y_i) is uniformly random in $\mathbb{Z}_p \times \mathbb{Z}_p^*$. Thus, we know the view of the adversary in $\mathrm{Game}_3^{\mathcal{A}}$ and $\mathrm{Game}_4^{\mathcal{A}}$ are identical, which implies

$$\mathsf{Pr}[\mathrm{Game}_3^{\mathcal{A}} = 1] = \mathsf{Pr}[\mathrm{Game}_4^{\mathcal{A}} = 1] \ . \tag{5.13}$$

5.2.2 *Proof of Claim 5.2.5*

Proof. Without loss of generality, assume the randomness used in Φ and the randomness of $\mathcal{A}$ are fixed, and assume $\mathsf{Pr}[\pi_j = \Delta] > 0$ given the fixed randomness.

The claim is equivalent to show that

$$\forall\ \boldsymbol{\eta}_0 \in \mathcal{C}\ :\ \mathsf{Pr}_{x,\boldsymbol{a},\boldsymbol{y}}[\boldsymbol{\eta} = \boldsymbol{\eta}_0 \mid \pi_j = \Delta] = \frac{1}{|\mathcal{C}|} \ .$$

The probability here is taken over the randomness $x, \boldsymbol{a}, \boldsymbol{y}$, where $\boldsymbol{a} = (a_1, \ldots, a_{\mathrm{sid}^\Delta})$, $\boldsymbol{y} = (y_1, \ldots, y_{\mathrm{sid}^\Delta})$. Also, x, $y_1, \ldots, y_{\mathrm{sid}^\Delta}$ are picked uniformly at random from $\mathbb{Z}_p^*$ and $a_1, \ldots, a_{\mathrm{sid}^\Delta}$ are picked uniformly at random from $\mathbb{Z}_p$.

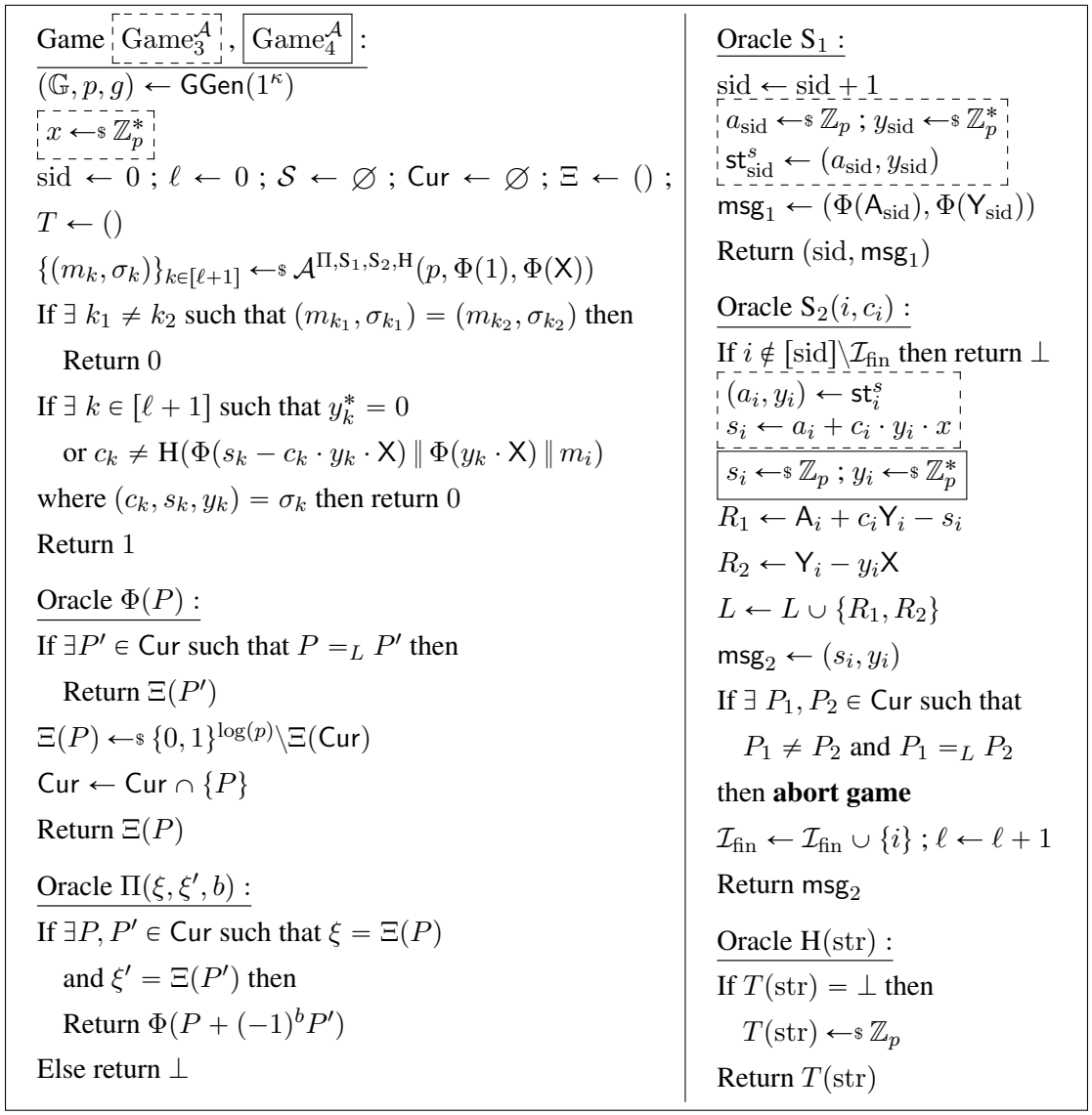

Figure 5.6: The definition for $\mathrm{Game}_4^{\mathcal{A}}$ and its difference from $\mathrm{Game}_3^{\mathcal{A}}$. $\mathrm{Game}_3^{\mathcal{A}}$ contains all but the solid box and $\mathrm{Game}_4^{\mathcal{A}}$ contains all but the dashed box.

We first show that

$$\pi_j = \Delta \quad \text{implies} \quad \boldsymbol{\eta} \in \mathcal{C} \ .$$

Suppose $\pi_j = \Delta$ occurs. We just need to show $D_i(\eta) \neq 0$ for each $i \in [m]$ and $B_i(\eta) = 0$ for each

$i \in [q]$. For $D_1, \ldots, D_{\mathrm{sid}^\Delta+1}$, since $x \neq 0$ and $y_i \neq 0$ for each $i \in [\mathrm{sid}^\Delta]$, we know $D_1(\eta) = x \neq 0$ and $D_{i+1}(\eta) = y_i \cdot x \neq 0$ for each $i \in [\mathrm{sid}^\Delta]$. For $D_{\mathrm{sid}^\Delta+1}, \ldots, D_m$, we make the argument using the original label $\{D_{P,P'}\}_{P,P' \in \mathsf{Cur}^\Delta, P \neq P'}$. For each $P, P' \in \mathsf{Cur}^\Delta$ such that $P \neq P'$, assume without loss of generality P is added to Cur before P'. When P' is added to Cur, since P is already in Cur, we know $P' \neq_{\mathrm{eval}} P$, which implies $D_{P,P'}(\boldsymbol{\eta}) = P'(\boldsymbol{\eta}) - P(\boldsymbol{\eta}) \neq 0$.

Also, for each $i \in \mathcal{I}^\Delta_{\mathrm{fin}}$, consider the query (i, c_i^Δ) made to S_2. Since $\pi_j = \Delta$, we have $s_i^\Delta = a_i + c_i^\Delta \cdot y_i \cdot x$ and $y_i^\Delta = y_i$. Therefore, we have $B_{(i,1)}(\boldsymbol{\eta}) = a_i + c_i^\Delta \cdot y_i \cdot x - s_i^\Delta = 0$ and $B_{(i,2)}(\boldsymbol{\eta}) = y_i \cdot x - y_i^\Delta \cdot x = 0$.[2] Therefore, we have $\boldsymbol{\eta} \in \mathcal{C}$.

We then show that

$$\boldsymbol{\eta} \in \mathcal{C} \quad \text{implies} \quad \pi_j = \Delta \ .$$

Since $\mathsf{Pr}[\pi_j = \Delta] > 0$, we know there exists $(x_0, \boldsymbol{a}_0, \boldsymbol{y}_0) \in \mathbb{Z}_p^{1+2\mathrm{sid}^\Delta}$ such that $\pi_j = \Delta$ when $(x, \boldsymbol{a}, \boldsymbol{y}) = (x_0, \boldsymbol{a}_0, \boldsymbol{y}_0)$. We now show that for any $(x_1, \boldsymbol{a}_1, \boldsymbol{y}_1) \in \mathbb{Z}_p^{1+2\mathrm{sid}^\Delta}$, given $(x, \boldsymbol{a}, \boldsymbol{y}) = (x_1, \boldsymbol{a}_1, \boldsymbol{y}_1)$ and $\boldsymbol{\eta} \in \mathcal{C}$, it must have $\pi_j = \Delta$.

Denote the case when $(x, \boldsymbol{a}, \boldsymbol{y}) = (x_0, \boldsymbol{a}_0, \boldsymbol{y}_0)$ as case 0 and the case when $(x, \boldsymbol{a}, \boldsymbol{y}) = (x_1, \boldsymbol{a}_1, \boldsymbol{y}_1)$ as case 1. We will show that the transcripts between the adversary and the oracles are exactly the same in these two cases, which implies $\pi_j = \Delta$ in case 1. We show this by induction. It is clear that the transcripts are the same at the beginning. For a time step T, suppose the transcripts are the same prior to this step and we have the following situations:

- Query to Φ, S_1, Π: Suppose the adversary receives $(\Phi(1), \Phi(\mathsf{X}))$ or makes query to S_1 or Π at step T. For the case that the adversary makes a query to S_1 or Φ, the transcripts can only differ on the invocation of Φ in S_1 or Π. Therefore, we only need to consider the queries and outputs of each Φ.

  For the k-th query to Φ where $k < j$, since the prior transcripts are the same in these two cases and the adversary is deterministic, we know the query P_k and the set Cur are the same in the two cases. If $P_k \neq_{\mathrm{eval}} P'$ for any $P' \in \mathsf{Cur}$ in case 0, then we know P_k is added to Cur in case 0. Since $\pi_j = \Delta$ occurs in case 0, we know $\{P_k\} \cup \mathsf{Cur} \subseteq \mathsf{Cur}^\Delta$. Since $\boldsymbol{\eta}_1 \in \mathcal{C}$, we know $P_k(\boldsymbol{\eta}_1) \neq P'(\boldsymbol{\eta}_1)$ for any $P' \in \mathsf{Cur}^\Delta$. Therefore, we have $P_k \neq_{\mathrm{eval}} P'$ for any $P' \in \mathsf{Cur}$ in case 1 too. Then, the outputs of Φ are the same in the two cases.

  Otherwise, if $P_k =_{\mathrm{eval}} P'$ for some $P' \in \mathsf{Cur}$, we know such P' must be unique. Since $E_{1,k}$

[2] Note here the value $y_i \cdot x$ is assigned to Y_i

does not occur in case 0, we have $P_k =_L P'$ in case 0. Since the current L is the same in the two cases, we know $P_k =_L P'$ in case 1 too. Since $P_k =_L P'$ implies $P_k =_{\text{eval}} P'$, we have $P_k =_{\text{eval}} P'$ in case 1 too. Thus, the output of Φ must be the same in the two cases. Therefore, we know the transcripts in these two cases must be the same after the k-th query to Φ is finished.

- Query to S_2: Suppose the adversary makes query (i, c_i) to S_2 at step T. Since $\pi_j = \Delta$ occurs in case 0, we know $i \in \mathcal{I}^{\Delta}_{\text{fin}}$, $c_i = c_i^{\Delta}$, $y_i = y_{0,i} = y_i^{\Delta}$, and $s_i = a_{0,i} + c_i \cdot y_{0,i} \cdot x_0 = s_i^{\Delta}$ in case 0. Since the transcripts are the same in the two cases prior to T and the adversary is deterministic, we know c_i is the same in both cases. Therefore, we know $c_i = c_i^{\Delta}$ in case 1. Since $\boldsymbol{\eta}_1 \in \mathcal{C}$, we have

$$B_{(i,1)}(\boldsymbol{\eta}_1) = a_{1,i} + c_i^{\Delta} \cdot y_{1,i} \cdot x_1 - s_i^{\Delta} = 0 \ ,$$

$$B_{(i,2)}(\boldsymbol{\eta}_1) = y_{1,i} \cdot x_1 - y_i^{\Delta} \cdot x_1 = 0 \ .$$

  Therefore, we have $y_i = y_{i,1} = y_i^{\Delta}$ and $s_i = a_{1,i} + c_i \cdot y_{1,i} \cdot x_1 = a_{1,i} + c_i^{\Delta} \cdot y_{1,i} \cdot x_1 = s_i^{\Delta}$ in case 1. Since the output (y_i, s_i) is the same in the two cases, we know the transcripts must be the same in these two cases after the query to S_2 is finished.

- Query to H : Since H does not envolve the randomness $x, \boldsymbol{a}, \boldsymbol{y}$, we know the transcripts are the same in the two cases after the query.

By induction, we know the transcript is the same by the step when the j-the query is made to Φ in the two cases. Therefore, we know $\pi_j = \Delta$ in case 1. Since it holds for any $(x_1, \boldsymbol{a}_1, \boldsymbol{y}_1) \in \mathbb{Z}_p^{\text{sid}^{\Delta}}$, we know $\boldsymbol{\eta} \in \mathcal{C}$ implies $\pi_j = \Delta$. Therefore, $\pi_j = \Delta$ is equivalent to $\boldsymbol{\eta} \in \mathcal{C}$, which implies for any $\boldsymbol{\eta}_0 \in \mathcal{C}$

$$\mathsf{Pr}_{x,\boldsymbol{a},\boldsymbol{y}}[\boldsymbol{\eta} = \boldsymbol{\eta}_0 | \pi_j = \Delta] = \mathsf{Pr}_{x,\boldsymbol{a},\boldsymbol{y}}[\boldsymbol{\eta} = \boldsymbol{\eta}_0 | \boldsymbol{\eta} \in \mathcal{C}] \ .$$

It is left to show $\mathsf{Pr}_{x,\boldsymbol{a},\boldsymbol{y}}[\boldsymbol{\eta} = \boldsymbol{\eta}_0 | \boldsymbol{\eta} \in \mathcal{C}] = \frac{1}{|\mathcal{C}|}$ for any $\boldsymbol{\eta}_0 \in \mathcal{C}$. Denote $\mathcal{E} := \mathbb{Z}_p^* \times (\mathbb{Z}_p \times \mathbb{Z}_p^*)^{\text{sid}^{\Delta}}$ and we know $(x, a_1, y_1, \ldots, a_{\text{sid}^{\Delta}}, y_{\text{sid}^{\Delta}})$ is uniformly distributed over $\mathcal{E}$. Therefore, $\boldsymbol{\eta} = (x, a_1, y_1 \cdot x, \ldots, a_{\text{sid}^{\Delta}} isalsody_{\text{sid}^{\Delta}} . x)$ is also uniformly distributed over $\mathcal{E}$, which implies for any $\boldsymbol{\eta}_0 \in \mathcal{E}$,

$$\mathsf{Pr}_{x,\boldsymbol{a},\boldsymbol{y}}[\boldsymbol{\eta} = \boldsymbol{\eta}_0] = \frac{1}{|\mathcal{E}|} \ .$$

Since $\mathcal{C} \subseteq \mathcal{E}$, we have for any $\boldsymbol{\eta}_0 \in \mathcal{C}$

$$\mathsf{Pr}_{x,\boldsymbol{a},\boldsymbol{y}}[\boldsymbol{\eta} = \boldsymbol{\eta}_0 | \boldsymbol{\eta} \in \mathcal{C}] = \frac{1/|\mathcal{E}|}{|\mathcal{C}|/|\mathcal{E}|} = \frac{1}{|\mathcal{C}|} \ .$$

□

5.2.3 Proof of Lemma 5.2.2

We construct $\mathcal{B}$ that interacts with $\mathcal{A}$ by simulating the oracles from Game_4 using the two oracles S and H in WFROS. In particular, we extract suitable vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to query to H in WFROS, i.e., each RO query str is decomposed as $\mathrm{str} = \xi^A \,\|\, \xi^Y \,\|\, m$, where ξ^A and ξ^Y are encodings of group elements. If both encodings are valid, there must exist P^A, P^Y such that $\Xi(P^A) = \xi^A$ and $\Xi(P^Y) = \xi^Y$; then, $\mathcal{B}$ defines two vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to make a corresponding query to H in WFROS. The oracle S is also used to simulate the signer's second stage. Finally, when $\mathcal{A}$ outputs $Q_{\mathrm{S}_1} + 1$ different valid message-signature pairs in Game_4, $\mathcal{B}$ tries to map each valid message-signature pair to a query to H in WFROS. We show that this strategy succeeds with probability close to that of $\mathcal{A}$ succeeding.

THE ADVERSARY $\mathcal{B}$. Specifically, $\mathcal{B}$ initializes the variables sid, Cur, $\mathcal{I}_{\mathrm{fin}}$, Ξ, and T as in Game_4. In addition, $\mathcal{B}$ initializes an empty table Hid, used later in the simulation of $\widetilde{\mathrm{H}}$.

Then, $\mathcal{B}$ runs $\mathcal{A}$ on input $(p, \widetilde{\Phi}(1), \widetilde{\Phi}(\mathsf{X}))$ and with access to the oracles $\widetilde{\Pi}$, $\widetilde{\mathrm{S}}_1$, $\widetilde{\mathrm{S}}_2$, and $\widetilde{\mathrm{H}}$. These oracles, along with $\widetilde{\Phi}$, operate as follows:

Oracles $\widetilde{\Phi}, \widetilde{\Pi}$: Same as in Game_4. In particular, L is updated by calls to $\widetilde{\mathrm{S}}_2$.

Oracle $\widetilde{\mathbf{S}}_1$: Same as in Game_4.

Oracle $\widetilde{\mathbf{S}}_2$: Same as Game_4 except that instead of sampling y_i randomly, if $i \in [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$, $\mathcal{B}$ makes a query (i, c_i) to S and uses its output as the value y_i.

Oracle $\widetilde{\mathbf{H}}$: After receiving a query str, if $T(\mathrm{str}) \neq \perp$, the value $T(\mathrm{str})$ is returned. Otherwise, str is decomposed as $\mathrm{str} = \xi^A \,\|\, \xi^Y \,\|\, m$ such that the length of ξ^A and ξ^Y is $\lceil \log(p) \rceil$.

- If there exist $P^A, P^Y \in \mathsf{Cur}$ such that $\Xi(P^A) = \xi^A$ and $\Xi(P^Y) = \xi^Y$, denote the coefficients of P^A, P^Y as

$$P^A = \hat{\alpha}^g + \hat{\alpha}^{\mathsf{X}}\mathsf{X} + \sum_{i \in [\mathrm{sid}]} \hat{\alpha}^{\mathsf{A}_i}\mathsf{A}_i + \sum_{i \in [\mathrm{sid}]} \hat{\alpha}^{\mathsf{Y}_i}\mathsf{Y}_i \ , \tag{5.14}$$

$$P^Y = \hat{\beta}^g + \hat{\beta}^{\mathsf{X}}\mathsf{X} + \sum_{i\in[\mathrm{sid}]} \hat{\beta}^{\mathsf{A}_i}\mathsf{A}_i + \sum_{i\in[\mathrm{sid}]} \hat{\beta}^{\mathsf{Y}_i}\mathsf{Y}_i \ . \tag{5.15}$$

Then, $\mathcal{B}$ issues the query $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ to H, where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^{2Q_{\mathrm{S}_1}+1}$ are such that

$$\alpha^{(i')} = \begin{cases} \hat{\alpha}^{\mathsf{X}} \ , & i' = 0 \\ \hat{\alpha}^{\mathsf{Y}_i} \ , & i' = 2i-1 \ , \ i \in [\mathrm{sid}] \\ -\hat{\alpha}^{\mathsf{A}_i} \ , & i' = 2i \ , \ i \in [\mathrm{sid}] \\ 0 \ , & o.w. \end{cases} ,$$

$$\beta^{(i')} = \begin{cases} -\hat{\beta}^{\mathsf{X}} \ , & i' = 0 \\ -\hat{\beta}^{\mathsf{Y}_i} \ , & i' = 2i-1 \ , \ i \in [\mathrm{sid}] \\ \hat{\beta}^{\mathsf{A}_i} \ , & i' = 2i \ , \ i \in [\mathrm{sid}] \\ 0 \ , & o.w. \end{cases} . \tag{5.16}$$

After receiving the output $(\delta_{\mathrm{hid}}, \mathrm{hid})$, $\mathcal{B}$ sets $T(\mathrm{str}) \leftarrow \delta_{\mathrm{hid}}$ and $\mathrm{Hid}(\mathrm{str}) \leftarrow \mathrm{hid}$.

– Otherwise, if $\xi^A \notin T(\mathsf{Cur})$ or $\xi^Y \notin T(\mathsf{Cur})$ (or if the decomposition of str is not possible), $\mathcal{B}$ samples $T(\mathrm{str})$ uniformly from $\mathbb{Z}_p$ and sets $\mathrm{Hid}(\mathrm{str}) = \perp$.

Finally, $\mathcal{B}$ returns $T(\mathrm{str})$.

After $\mathcal{A}$ outputs $\{(m_k^*, \sigma_k^*)\}_{k\in[Q_{\mathrm{S}_1}+1]}$, $\mathcal{B}$ aborts if the signatures are not valid, i.e., one of the following conditions is not satisfied:

$$\forall \ k_1, k_2 \in [Q_{\mathrm{S}_1} + 1] \text{ and } k_1 \neq k_2 \ : \ (m_{k_1}^*, \sigma_{k_1}^*) \neq (m_{k_2}^*, \sigma_{k_2}^*) \ , \tag{5.17}$$

$$\forall \ k \in [Q_{\mathrm{S}_1} + 1] \ : \ y_k^* \neq 0 \ \wedge \ c_k^* = \widetilde{\mathrm{H}}(\mathrm{str}_k^*) \ , \tag{5.18}$$

where $(c_k^*, s_k^*, y_k^*) = \sigma_k^*$ and $\mathrm{str}_k^* = \widetilde{\Phi}(s_k^* - c_k^* \cdot y_k^* \cdot \mathsf{X}) \,\|\, \widetilde{\Phi}(y_k^* \cdot \mathsf{X}) \,\|\, m_k^*$. (Here, $\widetilde{\mathrm{H}}$ and $\widetilde{\Phi}$ are the oracles described previously.) Further, $\mathcal{B}$ aborts if the following condition does not hold:

$$\forall \ k \in [Q_{\mathrm{S}_1} + 1] \ : \ \mathrm{Hid}(\mathrm{str}_k^*) \neq \perp \ . \tag{5.19}$$

Otherwise, $\mathcal{B}$ outputs $\mathcal{J} := \{\mathrm{Hid}(\mathrm{str}_k^*)\}_{k\in[Q_{\mathrm{S}_1}+1]}$.

ANALYSIS OF $\mathcal{B}$. Note that $\mathcal{B}$ queries to H at most once when it receives a query to $\widetilde{\mathrm{H}}$ and makes $Q_{\mathrm{S}_1}+1$ more queries to $\widetilde{\mathrm{H}}$ when checking the validity of the output. Therefore, $\mathcal{B}$ makes at most $Q_{\mathrm{H}}+Q_{\mathrm{S}_1}+1$ queries to H. Also, it is clear that $\mathcal{B}$ simulates oracles S_1, S_2 in Game_4 perfectly. For the simulation of $\widetilde{\mathrm{H}}$, the only difference is that the distribution of δ_{hid} outputting from H in WFROS is uniformly over $\mathbb{Z}_p^*$, where in Game_4 it is always uniformly from $\mathbb{Z}_p$. However, the statistical distance between the two distributions is $1/p$. Since $\mathcal{B}$ makes at most $Q_{\mathrm{H}}+Q_{\mathrm{S}_1}+1$ queries to H, the statistical difference between the view of $\mathcal{A}$ in Game_4 and that in the one simulated by $\mathcal{B}$ is bounded by $(Q_{\mathrm{H}}+Q_{\mathrm{S}_1}+1)/p$.

Denote the event E_1 such that when $\mathcal{B}$ checks the output from $\mathcal{A}$, both (5.17) and (5.18) hold. As these are exactly the winning conditions of Game_4, which is simulated statistically close to perfect, we have $\mathsf{Pr}[E_1]+\frac{Q_{\mathrm{H}}+Q_{\mathrm{S}_1}+1}{p} \geqslant \mathsf{Pr}[\mathrm{Game}_4^{\mathcal{A}}=1]$. Also, let E_2 be the event for which the condition (5.19) holds immediately afterward. If E_2 does not happen, but E_1 does, then we know $\mathcal{A}$ outputs a valid message-signature pair (m_k^*,σ_k^*) such that $\mathrm{Hid}(\mathrm{str}_k^*)=\perp$, which is unlikely to happen. The following formalizes this, and the proof is in Section 5.2.4.

Claim 5.2.7. $\mathsf{Pr}[E_1 \wedge (\neg E_2)] \leqslant \frac{2Q_\Phi(Q_{\mathrm{H}}+Q_{\mathrm{S}_1}+1)}{p-Q_\Phi}$.

Then, we can conclude the proof with the following claim.

Claim 5.2.8. *If both E_1 and E_2 happen, then $\mathcal{B}$ outputs a valid WFROS solution $\mathcal{J}$, which in turn implies that* $\mathsf{Pr}[E_1 \wedge E_2] \leqslant \mathsf{Adv}^{\mathrm{wfros}}_{Q_{\mathrm{S}_1},p}(\mathcal{B})$.

Before we proceed with a proof, we state a simple lemma for Game_4 that is used in the proof of Claim 5.2.8. The proof is immediate and follows from the uniqueness of values returned by the oracle.

Lemma 5.2.9. *At any step of Game_4, for any two polynomials P and P', suppose we make queries P and P' to Φ. If $\Phi(P)=\Phi(P')$, then $P=_L P'$. Equivalently, if $P\neq_L P'$, then we have $\Phi(P)\neq\Phi(P')$.*

Proof of Claim 5.2.8. Suppose both E_1 and E_2 happen. We first show that for any $k_1,k_2\in[Q_{\mathrm{S}_1}+1]$ and $k_1\neq k_2$, it holds that $\mathrm{str}_{k_1}^*\neq\mathrm{str}_{k_2}^*$, which implies $|\mathcal{J}|=Q_{\mathrm{S}_1}+1$. We then show that $\mathcal{J}$ is valid for the WFROS game.

For $k_1,k_2\in[Q_{\mathrm{S}_1}+1]$ and $k_1\neq k_2$, suppose $\mathrm{str}_{k_1}^*=\mathrm{str}_{k_2}^*$. Then, we have

$$c_{k_1}^*=\widetilde{\mathrm{H}}(\mathrm{str}_{k_1}^*)=\widetilde{\mathrm{H}}(\mathrm{str}_{k_2}^*)=c_{k_2}^*\ ,\ m_{k_1}^*=m_{k_2}^*\ ,$$

$$\widetilde{\Phi}(s^*_{k_1} - c^*_{k_1} \cdot y^*_{k_1} \cdot \mathsf{X}) = \widetilde{\Phi}(s^*_{k_2} - c^*_{k_2} \cdot y^*_{k_2} \cdot \mathsf{X}) \ , \ \widetilde{\Phi}(y^*_{k_1} \cdot \mathsf{X}) = \widetilde{\Phi}(y^*_{k_2} \cdot \mathsf{X}) \ .$$

By Lemma 5.2.9, it holds that $(m^*_{k_1}, (c^*_{k_1}, s^*_{k_1}, y^*_{k_1})) = (m^*_{k_2}, (c^*_{k_2}, s^*_{k_2}, y^*_{k_2}))$. However, since E_1 happens, by (5.17), we have $(m^*_{k_1}, \sigma^*_{k_1}) \neq (m^*_{k_2}, \sigma^*_{k_2})$, which yields a contradiction. Therefore, we know $\mathrm{str}^*_{k_1} \neq \mathrm{str}^*_{k_2}$. From the simulation of $\widetilde{\mathsf{H}}$, we have $\mathrm{Hid}(\mathrm{str}^*_{k_1}) \neq \mathrm{Hid}(\mathrm{str}^*_{k_2})$, and thus we have $|\mathcal{J}| = \ell + 1$.

We now show that for each $j \in \mathcal{J}$, it holds that

$$\alpha^{(0)}_j + \sum_{i \in \mathcal{I}_{\mathrm{fin}}} y_i \left(\alpha^{(2i-1)}_j + c_i \cdot \alpha^{(2i)}_j \right) = \delta_j \left(\beta^{(0)}_j + \sum_{i \in \mathcal{I}_{\mathrm{fin}}} y_i \left(\beta^{(2i-1)}_j + c_i \cdot \beta^{(2i)}_j \right) \right) \ , \tag{C1}$$

$$\beta^{(0)}_j + \sum_{i \in \mathcal{I}_{\mathrm{fin}}} y_i \left(\beta^{(2i-1)}_j + c_i \cdot \beta^{(2i)}_j \right) \neq 0 \ , \tag{C2}$$

which implies $\mathcal{J}$ is valid for the WFROS game.

Let us fix a $j \in \mathcal{J}$. Since $j \in \mathcal{J}$, there exists $k \in [Q_{\mathsf{S}_1} + 1]$ such that $\mathrm{Hid}(\mathrm{str}^*_k) = j$, and there exists $P^A_j, P^Y_j \in \mathsf{Cur}$ and m_j such that $\mathrm{str}^*_k = \Xi(P^A_j) \,\|\, \Xi(P^Y_j) \,\|\, m_j$. Let $\hat{\alpha}_j$ and $\hat{\beta}_j$ denote the coefficients of P^A_j and P^Y_j. Since E_1 happens, by (5.18) and Lemma 5.2.9, we have $P^A_j =_L s^*_k - \delta_j \cdot y^*_k \cdot \mathsf{X}$, $P^Y_j =_L y^*_k \cdot \mathsf{X}$, which implies there exists $\{r^{P^A_j}_{1,i}, r^{P^A_j}_{2,i}, r^{P^Y_j}_{1,i}, r^{P^Y_j}_{2,i}\}_{i \in \mathcal{I}_{\mathrm{fin}}}$ such that

$$P^A_j = s^*_k - \delta_j \cdot y^*_k \cdot \mathsf{X} + \sum_{i \in [Q_{\mathsf{S}_1}]} r^{P^A_j}_{1,i} \left(\mathsf{A}_i + c_i \mathsf{Y}_i - s_i \right) + \sum_{i \in [Q_{\mathsf{S}_1}]} r^{P^A_j}_{2,i} \left(\mathsf{Y}_i - y_i \mathsf{X} \right) \ ,$$

$$P^Y_j = y^*_k \cdot \mathsf{X} + \sum_{i \in [Q_{\mathsf{S}_1}]} r^{P^Y_j}_{1,i} \left(\mathsf{A}_i + c_i \mathsf{Y}_i - s_i \right) + \sum_{i \in [Q_{\mathsf{S}_1}]} r^{P^Y_j}_{2,i} \left(\mathsf{Y}_i - y_i \mathsf{X} \right) \ . \tag{5.20}$$

By looking into the coefficients of X, $\{\mathsf{A}_i, \mathsf{Y}_i\}_{i \in [Q_{\mathsf{S}_1}]}$ on both sides of (5.20), we have $\hat{\alpha}^{\mathsf{A}_i}_j = r^{P^A_j}_{1,i}$, $\hat{\alpha}^{\mathsf{Y}_i}_j = r^{P^A_j}_{1,i} \cdot c_i + r^{P^A_j}_{2,i}$, $\hat{\beta}^{\mathsf{A}_i}_j = r^{P^Y_j}_{1,i}$, $\hat{\beta}^{\mathsf{Y}_i}_j = r^{P^Y_j}_{1,i}$ and $c_i + r^{P^Y_j}_{2,i}$ for each $i \in [Q_{\mathsf{S}_1}]$, $\hat{\beta}^{\mathsf{X}}_j = y^*_k - \sum_{i \in [Q_{\mathsf{S}_1}]} r^{P^Y_j}_{2,i} \cdot y_i$, and $\hat{\alpha}^{\mathsf{X}}_j = -\delta_j \cdot y^*_k - \sum_{i \in [Q_{\mathsf{S}_1}]} r^{P^A_j}_{2,i} \cdot y_i$. By sorting out the equations, we have

$$y^*_k = \beta^{\mathsf{X}}_j + \sum_{i \in \mathcal{I}_{\mathrm{fin}}} y_i \left(\hat{\beta}^{\mathsf{Y}_i}_j - c_i \cdot \hat{\beta}^{\mathsf{A}_i}_j \right) \ ,$$

$$\hat{\alpha}^{\mathsf{X}}_j + \sum_{i \in [Q_{\mathsf{S}_1}]} y_i \left(\hat{\alpha}^{\mathsf{Y}_i}_j - c_i \cdot \hat{\alpha}^{\mathsf{A}_i}_j \right) = -\delta_j \cdot \left(\hat{\beta}^{\mathsf{X}}_j + \sum_{i \in [Q_{\mathsf{S}_1}]} y_i \left(\hat{\beta}^{\mathsf{Y}_i}_j - c_i \cdot \hat{\beta}^{\mathsf{A}_i}_j \right) \right) .$$

By the definition of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (5.16), we know (C1) holds and

$$y_k^* = \beta_j^{(0)} + \sum_{i \in \mathcal{I}_{\text{fin}}} y_i \left(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)} \right) .$$

Since E_1 happens, by (5.18), we know $y_k^* \neq 0$, which implies (C2) happens. □

5.2.4 *Proof of Claim 5.2.7*

Suppose $E_1 \ \wedge \ (\neg E_2)$ occurs. Denote str_j as the input of the j-th query to $\widetilde{\text{H}}$. Denote the total number of queries to $\widetilde{\text{H}}$ as $\text{num}_{\widetilde{\text{H}}}^{\text{tot}}$. Denote the decompositin of str_j as $\text{str}_j = \xi_j^A \| \xi_j^Y \| m_j$. Denote Cur_j as the set Cur by the step when the j-th query to $\widetilde{\text{H}}$ is made and denote $\mathsf{Cur}^{\text{tot}}$ as the set Cur after $\mathcal{B}$ finishes the check of the condition (5.17) and (5.18). Since $\mathcal{B}$ makes a query str_k^* to $\widetilde{\text{H}}$ to check the condition (5.18), there exists $j \in [\text{num}_{\widetilde{\text{H}}}^{\text{tot}}]$ such that $\text{str}_j = \text{str}_k^*$. Let $j_{\min}$ be the smallest index such that $\text{str}_{j_{\min}} = \text{str}_k^*$. Since $\text{Hid}(\text{str}_k^*) = \perp$, from the simulation of $\widetilde{\text{H}}$, we know $\xi_{j_{\min}}^A \notin \Xi(\mathsf{Cur}_{j_{\min}})$ or $\xi_{j_{\min}}^Y \notin \Xi(\mathsf{Cur}_{j_{\min}})$. However, since $\xi_{j_{\min}}^A = \widetilde{\Phi}(s_k^* - c_k^* \cdot y_k^* \cdot \mathsf{X})$ and $\xi_{j_{\min}}^Y = \widetilde{\Phi}(y_k^* \cdot \mathsf{X})$, we know $\xi_{j_{\min}}^A, \xi_{j_{\min}}^Y \in \mathsf{Cur}^{\text{tot}}$. Therefore, denote the set of all ξ_j^Y and ξ_j^A that do not correspond to any encoding of polynomials when the j-th query to $\widetilde{\text{H}}$ is made as

$$D^{\text{tot}} := \{\xi_j^A | j \in [\text{num}_{\widetilde{\text{H}}}^{\text{tot}}], \xi_j^A \notin \Xi(\mathsf{Cur}_j)\} \cup \{\xi_j^Y | j \in [\text{num}_{\widetilde{\text{H}}}^{\text{tot}}], \xi_j^Y \notin \Xi(\mathsf{Cur}_j)\} ,$$

and then we have at least one of $\xi_{j_{\min}}^A$ and $\xi_{j_{\min}}^Y$ is in $D \cap \Xi(\mathsf{Cur}^{\text{tot}})$, which implies $D \cap \Xi(\mathsf{Cur}^{\text{tot}}) \neq \varnothing$. Therefore, we have the event E occurs implies $D \cap \Xi(\mathsf{Cur}^{\text{tot}}) \neq \varnothing$, which means

$$\mathsf{Pr}[E_1 \ \wedge \ (\neg E_2)] \leqslant \mathsf{Pr}[D^{\text{tot}} \cap \Xi(\mathsf{Cur}^{\text{tot}}) \neq \varnothing] . \tag{5.21}$$

It is left to bound $\mathsf{Pr}[D \cap \Xi(\mathsf{Cur}^{\text{tot}}) \neq \varnothing]$.

Denote

$$D_j := \{\xi_{j'}^A | j' \in [j], \xi_{j'}^A \notin \Xi(\mathsf{Cur}_{j'})\} \cup \{\xi_{j'}^Y | j' \in [j], \xi_{j'}^Y \notin \Xi(\mathsf{Cur}_{j'})\} .$$

Denote $\mathsf{Cur}^{(i)}$ as the set Cur after the i-th query to $\widetilde{\Phi}$ is finished and $\mathsf{Cur}^{(0)} = \varnothing$. Consider the step when the i-th query to $\widetilde{\Phi}$ is made. Denote the number of queries to $\widetilde{\text{H}}$ before the i-th query to $\widetilde{\Phi}$ is made as $\text{num}_{\widetilde{\text{H}}}^{(i)}$. Denote the event E_i' as $D_{\text{num}_{\widetilde{\text{H}}}^{(i)}} \cap \Xi(\mathsf{Cur}^{(i-1)}) = \varnothing$ and $D_{\text{num}_{\widetilde{\text{H}}}^{(i)}} \cap \Xi(\mathsf{Cur}^{(i)}) \neq \varnothing$. We first show that if $D^{\text{tot}} \cap \Xi(\mathsf{Cur}^{\text{tot}}) \neq \varnothing$, then there exists i such that E_i' occurs, and then bound $\mathsf{Pr}[E_i']$ for each i.

Denote the total number of queries to $\widetilde{\Phi}$ as $\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}}$. Suppose none of $E'_1, \ldots, E'_{\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}}}$ occurs. We show that at any time step, supposing the number of queries to $\widetilde{\Phi}$ made so far is i and the number of queries to $\widetilde{\mathrm{H}}$ made so far is j, we have $D_j \cap T(\mathsf{Cur}^{(i)}) = \varnothing$, which implies $D^{\mathrm{tot}} \cap \Xi(\mathsf{Cur}^{\mathrm{tot}}) = \varnothing$. We show the statement by induction. At the begining, we know $i = 0$, $j = 0$, $\mathsf{Cur}^{(0)} = \varnothing$, and $D_0 = \varnothing$. Thus, the statement holds trivially. For any time step with $i > 0$ or $j > 0$, suppose the latest query is made to $\widetilde{\mathrm{H}}$ and we have $D_{j-1} \cap T(\mathsf{Cur}^{(i)}) = \varnothing$. Consider the step when the j-th query to $\widetilde{\mathrm{H}}$ is made. If $T(\mathrm{str}_j) \neq \perp$, we have $D_j = D_{j-1}$ and $D_j \cap T(\mathsf{Cur}^{(i)}) = \varnothing$. Otherwise, if $T(\mathrm{str}_j) = \perp$, we have $D_j = D_{j-1} \cup (\{\xi_j^A, \xi_j^Y\} \backslash T(\mathsf{Cur}_j))$. Since $\mathsf{Cur}_j = \mathsf{Cur}^{(i)}$, we have $D_j \cap T(\mathsf{Cur}^{(i)}) = D_{j-1} \cup T(\mathsf{Cur}^{(i)}) = \varnothing$. Therefore, we have $D_j \cap T(\mathsf{Cur}^{(i)}) = \varnothing$. Otherwise, suppose the latest query is made to $\widetilde{\Phi}$ and we have $D_j \cap T(\mathsf{Cur}^{(i-1)}) = \varnothing$. Since we have $j = \mathrm{num}_{\widetilde{\mathrm{H}}}^{(i)}$ and E'_i does not occur, we have $D_j \cap T(\mathsf{Cur}^{(i-1)}) = D_{\mathrm{num}_{\widetilde{\mathrm{H}}}^{(i)}} \cap \Xi(\mathsf{Cur}^{(i)}) = \varnothing$. Therefore, by induction, the statement holds. Then, considering the step when $\mathcal{B}$ finishes the check of the condition (5.17) and (5.18), we have $D^{\mathrm{tot}} \cap \Xi(\mathsf{Cur}^{\mathrm{tot}}) = D_{\mathrm{num}_{\widetilde{\mathrm{H}}}^{\mathrm{tot}}} \cap \Xi(\mathsf{Cur}^{(\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}})}) = \varnothing$. Therefore, if $D^{\mathrm{tot}} \cap \Xi(\mathsf{Cur}^{\mathrm{tot}}) \neq \varnothing$, then at least one of $\{E'_i\}_{i \in [\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}}]}$ occurs.

Finally, to bound $\mathsf{Pr}[E'_i]$, consider the i-th query to $\widetilde{\Phi}$. Denote the input of the i-th query to $\widetilde{\Phi}$ as P_i. Denote $j = \mathrm{num}_{\widetilde{\mathrm{H}}}^{(i)}$ for simplicity. Suppose E'_i occurs. We know $\mathsf{Cur}^{(i)} \neq \mathsf{Cur}^{(i-1)}$, which implies $\mathsf{Cur}^{(i)} = \mathsf{Cur}^{(i-1)} \cup \{P_i\}$ and $\Xi(\mathsf{Cur}^{(i)}) = \Xi(\mathsf{Cur}^{(i-1)}) \cup \{\Xi(P_i)\}$. Since $D_j \cap \Xi(\mathsf{Cur}^{(i-1)}) = \varnothing$ and $D_j \cap \Xi(\mathsf{Cur}^{(i)}) \neq \varnothing$, we know $\Xi(P_i) \in D_j$. Therefore, we have

$$\mathsf{Pr}[E'_i] \leqslant \mathsf{p}[\mathsf{Cur}^{(i)} = \mathsf{Cur}^{(i-1)} \cap \{P_i\} \ \wedge \ \Xi(P_i) \in D_j] \ .$$

Consider the step when $\Xi(P_i)$ is generated. We know D_j is already determined. Therefore, we know $\Xi(P_i)$ is sampled from $\{0,1\}^{\log(p)} \backslash \Xi(\mathsf{Cur}^{(i-1)})$ uniformly at random independent of D_j, which implies

$$\begin{aligned}
\mathsf{Pr}[E'_i] &\leqslant \mathsf{Pr}[\mathsf{Cur}^{(i)} = \mathsf{Cur}^{(i-1)} \cup \{P_i\} \ \wedge \ \Xi(P_i) \in D_j] \\
&\leqslant \mathsf{Pr}[\Xi(P_i) \in D_j | \mathsf{Cur}^{(i)} = \mathsf{Cur}^{(i-1)} \cup \{P_i\}] \\
&\leqslant \frac{|D_j|}{p - |\mathsf{Cur}^{(i-1)}|} \leqslant \frac{|D^{\mathrm{tot}}|}{p - |\mathsf{Cur}^{\mathrm{tot}}|} \ .
\end{aligned}$$

Therefore, we have

$$\mathsf{Pr}[D^{\mathrm{tot}} \cap \Xi(\mathsf{Cur}^{\mathrm{tot}}) \neq \varnothing] \leqslant \mathsf{Pr}\left[\bigvee_{i \in [\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}}]} E'_i\right] \leqslant \sum_{i \in [\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}}]} \mathsf{Pr}[E'_i] \leqslant \frac{\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}} \cdot |D^{\mathrm{tot}}|}{p - |\mathsf{Cur}^{\mathrm{tot}}|}.$$

Since $|D^{\mathrm{tot}}| \leqslant 2\mathrm{num}_{\widetilde{\mathsf{H}}}^{\mathrm{tot}} \leqslant 2(Q_{\mathsf{H}} + Q_{\mathsf{S}_1} + 1)$ and $|\mathsf{Cur}^{\mathrm{tot}}| \leqslant \mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}} \leqslant Q_{\Phi}$, by (5.21), we have

$$\mathsf{Pr}[E_1 \ \wedge \ (\neg E_2)] \leqslant \frac{2\mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}} \cdot \mathrm{num}_{\widetilde{\mathsf{H}}}^{\mathrm{tot}}}{p - \mathrm{num}_{\widetilde{\Phi}}^{\mathrm{tot}}} = \frac{2Q_{\Phi}(Q_{\mathsf{H}} + Q_{\mathsf{S}_1} + 1)}{p - Q_{\Phi}} \ .$$

Chapter 6

EFFICIENT BLIND SIGNATURES IN THE AGM

We now present schemes that are secure in the *algebraic group model* (AGM) [64]. This model considers security for *algebraic adversaries* - these are adversaries that, when used within a reduction, provide a representation of any group element they output in terms of all prior group elements input to the adversary. (We dispense with a more formal definition since the use of the AGM is self-evident in our proofs.)

6.1 Construction and Security

In this section, we introduce a scheme, which we refer to as BS_2, that relies on the hardness of the (plain) discrete logarithm (DL) problem, which is formalized in Figure 2.1. In contrast to BS_1, our new scheme (described in Figure 6.1) requires an extra group element Z in the public key, and the commitment X^y in is replaced by $g^t Z^y$. (This will necessary result in an additional scalar in the signature.) However, one could generate Z as an output of a hash function (assuming the hash function is a random oracle, which we assume anyways), although, interestingly, our proof for BS_2 will show that blindness holds even when Z is chosen maliciously by the signer (who may consequently also know its discrete logarithm).

The additional group element Z will in fact allow us to develop a *partially blind* version of BS_2, which we refer to as PBS, which we discuss in Section 6.3 below. We note that in fact *all* results about BS_2 can be obtained as a corollary of our analysis of PBS, because a blind signature scheme is of course a special case of a partially blind one. However, we are opting for a separate presentation, as the main ideas behind the reduction are much simpler to understand in (plain) BS_2, and the proof of PBS adds some extra complexity (in particular, in order to obtain a tighter bound), which obfuscates the main ideas.

SECURITY ANALYSIS. The following theorem establishes the blindness of BS_2. (Its proof is very similar to the blindness proof of $\mathsf{BS}_1[\mathsf{GGen}]$.)

Algorithm $\mathsf{Setup}(1^\kappa)$:
$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$
Select $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_p^*$
Return $par \leftarrow (p, \mathbb{G}, g, \mathrm{H})$

Algorithm $\mathsf{KeyGen}(par)$:
$(p, \mathbb{G}, g, \mathrm{H}) \leftarrow par$
$x \leftarrow\!\!\$\ \mathbb{Z}_p$; $X \leftarrow g^x$; $Z \leftarrow\!\!\$\ \mathbb{G}$
$sk \leftarrow x$; $pk \leftarrow (X, Z)$
Return (sk, pk)

Algorithm $\mathsf{S}_1(sk)$:
$x \leftarrow sk$; $X \leftarrow g^x$
$a, t \leftarrow\!\!\$\ \mathbb{Z}_p$; $y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$A \leftarrow g^a$; $C \leftarrow g^t Z^y$
$\mathsf{st}^s \leftarrow (a, y, t, x)$; $\mathsf{msg}_1 \leftarrow (A, C)$
Return $(\mathsf{st}^s, \mathsf{msg}_1)$

Algorithm $\mathsf{S}_2(\mathsf{st}^s, c)$:
If $c = 0$ then return $\perp$
$(a, y, t, x) \leftarrow \mathsf{st}^s$
$s \leftarrow a + c \cdot y \cdot x$
Return $\mathsf{msg}_2 \leftarrow (s, y, t)$

Algorithm $\mathsf{U}_1(pk, \mathsf{msg}_1, m)$:
$X \leftarrow pk$; $(A, C) \leftarrow \mathsf{msg}_1$
$r_1 \leftarrow\!\!\$\ \mathbb{Z}_p$; $\gamma_1, \gamma_2 \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$A' \leftarrow g^{r_1} \cdot A^{\gamma_1/\gamma_2} \cdot C^{\gamma_1}$
$c' \leftarrow H(A' \,\|\, m)$
$c \leftarrow c' \cdot \gamma_2$
$\mathsf{st}^u \leftarrow (c, c', r_1, \gamma_1, \gamma_2, X, Z, A, C)$
Return (st^u, c)

Algorithm $\mathsf{U}_2(\mathsf{st}^u, \mathsf{msg}_2)$:
$(c, c', r_1, \gamma_1, \gamma_2, X, Z, A, C) \leftarrow \mathsf{st}^u$
$(s, y, t) \leftarrow \mathsf{msg}_2$
If $y = 0$ or $C \neq g^t Z^y$ or $g^s \neq A \cdot X^{c \cdot y}$
then return $\perp$
$s' \leftarrow r_1 + (\gamma_1/\gamma_2) \cdot s + \gamma_1 \cdot t$
$y' \leftarrow \gamma_1 \cdot y$
Return $\sigma \leftarrow (c', s', y')$

Algorithm $\mathsf{Ver}(pk, \sigma, m)$:
$(c, s, y) \leftarrow \sigma$
If $y = 0$ then return 0
$A \leftarrow g^s \cdot X^{-c \cdot y} \cdot Z^y$
If $c \neq \mathrm{H}(A \,\|\, m)$ then return 0
Return 1

Figure 6.1: The blind signature scheme $\mathsf{BS}_2 = \mathsf{BS}_2[\mathsf{GGen}]$.

Theorem 6.1.1. *For a group generation algorithm* GGen*, the blind signature scheme* $\mathsf{BS}_2[\mathsf{GGen}]$ *is perfectly blind.*

Proof. Let $\mathcal{A}$ be an adversary playing the $\mathrm{Blind}^{\mathcal{A}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game. Similar to the blindness proof of $\mathsf{BS}_1[\mathsf{GGen}]$, we can assume the randomness of $\mathcal{A}$ is fixed and $\mathcal{A}$ always finishes both signing sessions and receives valid signatures (σ_0, σ_1) without loss of generality.

Define the view of $\mathcal{A}$ after its execution as $\pi = (X, Z, m_0, m_1, T_0, T_1, \sigma_0, \sigma_1)$, where $T_i := (A_i,$

$C_i, c_i, s_i, y_i, t_i)$, denoting the transcripts learned from interactions with the i-th signing session and $\sigma_i = (c'_i, s'_i, y'_i, t'_i)$. Since the randomness of $\mathcal{A}$ is fixed, the only randomness left is the randomness in U_1 and U_2. Denote $\eta := (r_1^{(0)}, \gamma_1^{(0)}, \gamma_2^{(0)}, r_1^{(1)}, \gamma_1^{(1)}, \gamma_2^{(1)})$ as the total randomness. To prove the theorem, we need only show that the distribution of π is identical in both the cases $b = 0$ and $b = 1$. We prove this by showing that for any fixed view Δ such that $\mathsf{Pr}[\pi = \Delta | b = 1] > 0$, there exists a unique value of the randomness η that makes $\pi = \Delta$ for the cases $b = 0$ and $b = 1$.

For both the cases $b = 0$ and $b = 1$, we now show that $\pi = \Delta$ if and only if for each $i \in \{0, 1\}$, it holds that

$$\begin{aligned} \gamma_1^{(i)} &= {y'_{b_i}}^{\Delta} / y_i^{\Delta} \ , \\ \gamma_2^{(i)} &= c_i^{\Delta} / {c'_{b_i}}^{\Delta} \ , \\ r_1^{(i)} &= {s'_{b_i}}^{\Delta} - s_i^{\Delta} \cdot (\gamma_1^{(i)} / \gamma_2^{(i)}) - \gamma_1^{(i)} \cdot t_i^{\Delta} \ , \end{aligned} \tag{6.1}$$

where the superscript $(\cdot)^{\Delta}$ represents the corresponding value in Δ. From the algorithms $\mathsf{BS}_2.\mathsf{U}_1$ and $\mathsf{BS}_2.\mathsf{U}_2$, it is clear that the "only if" part holds. For the "if" part, suppose (6.1) holds. Since the randomness of $\mathcal{A}$ is fixed, the view of $\mathcal{A}$ can differ only on the outputs c_0, c_1 from the oracle U_1 or the output (σ_0, σ_1) from the oracle U_2. Since both signatures in Δ are valid, we have

$$A_i^{\Delta} = g^{s_i^{\Delta}} X^{\Delta - c_i^{\Delta} \cdot y_i^{\Delta}} \ , \ C_i^{\Delta} = g^{t_i^{\Delta}} Z^{\Delta y_i^{\Delta}} \ . \tag{6.2}$$

$${c'_{b_i}}^{\Delta} = \mathrm{H}(g^{{s'_{b_i}}^{\Delta}} X^{\Delta - {y'_{b_i}}^{\Delta} \cdot {c'_{b_i}}^{\Delta}} Z^{\Delta {y'_{b_i}}^{\Delta}} \parallel m_{b_i}^{\Delta}) \ . \tag{6.3}$$

For c_i where $i \in \{0, 1\}$, suppose the values in the view of $\mathcal{A}$ that have already determined when c_i is generated, which must include (X, m_i, A_i, C_i), are consistent with Δ. By (6.1), we have

$$\begin{aligned} c_i &= \gamma_2^{(i)} \cdot \mathrm{H}(g^{r_1^{(i)}} A_i^{\gamma_1^{(i)} / \gamma_2(i)} C_i^{\gamma_1^{(i)}} \parallel m_{b_i}) \\ &= \gamma_2^{(i)} \cdot \mathrm{H}(g^{r_1^{(i)}} A_i^{\Delta \gamma_1^{(i)} / \gamma_2(i)} C_i^{\Delta \gamma_1^{(i)}} \parallel m_{b_i}^{\Delta}) \\ &= \gamma_2^{(i)} \cdot \mathrm{H}(g^{r_1^{(i)} + s_i^{\Delta} \cdot (\gamma_1^{(i)} / \gamma_2^{(i)}) + \gamma_1^{(i)} \cdot t_i^{\Delta}} X^{\Delta - y_i^{\Delta} \cdot c_i^{\Delta} \cdot (\gamma_1^{(i)} / \gamma_2^{(i)})} Z^{\Delta y_i^{\Delta} \cdot \gamma_1^{(i)}} \parallel m_{b_i}^{\Delta}) \\ &= \gamma_2^{(i)} \cdot \mathrm{H}(g^{{s'_{b_i}}^{\Delta}} X^{\Delta - {y'_{b_i}}^{\Delta} \cdot {c'_{b_i}}^{\Delta}} Z^{\Delta {y'_{b_i}}^{\Delta}} \parallel m_{b_i}^{\Delta}) \\ &= \gamma_2^{(i)} \cdot {c'_{b_i}}^{\Delta} = c_i^{\Delta} \ . \end{aligned}$$

where the third equality is due to (6.2), the fourth equality is due to (6.1), and the final equality is due to (6.3). Then, consider the step when (σ_0, σ_1) are output. Suppose the current view, which

contains T_i, is consistent with Δ. By (6.1), we have

$$\begin{aligned}
y'_{b_i} &= \gamma_1^{(i)} \cdot y_i = \gamma_1^{(i)} \cdot y_i^{\Delta} = {y'_{b_i}}^{\Delta} , \\
s'_{b_i} &= r_1^{(i)} + s_i(\gamma_1^{(i)}/\gamma_2^{(i)}) + \gamma_1^{(i)} \cdot t_i = r_1^{(i)} + s_i^{\Delta}(\gamma_1^{(i)}/\gamma_2^{(i)}) + \gamma_1^{(i)} \cdot t_i^{\Delta} = {s'_{b_i}}^{\Delta} , \\
c'_{b_i} &= c_i/\gamma_2^{(i)} = c_i^{\Delta}/\gamma_2^{(i)} = {c'_{b_i}}^{\Delta} ,
\end{aligned}$$

which implies $(\sigma_0, \sigma_1) = (\sigma_0^{\Delta}, \sigma_1^{\Delta})$. Therefore, by induction, if (6.1) holds, we know $\pi = \Delta$. □

The core of the analysis is once again a proof that the scheme is one-more unforgeable in the AGM, i.e., we only prove security against algebraic adversaries. In particular, we model the selected hash function as a random oracle H, to which the adversary is given explicit access.

Theorem 6.1.2. *Let $\mathbb{G}$ be a group generation algorithm. For any algebraic adversary $\mathcal{A}_{\mathrm{alg}}$ for the game $\mathrm{OMUF}^{\mathsf{BS}_2[\mathsf{GGen}]}(\kappa)$ making at most Q_{S_1} queries to S_1 and Q_{H} queries to the random oracle H, there exists an adversary $\mathcal{B}_{\mathrm{dlog}}$ for the DLog problem running in a similar running time as $\mathcal{A}_{\mathrm{alg}}$ such that*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{BS}_2[\mathsf{GGen}]}(\mathcal{A}_{\mathrm{alg}}, \kappa) \leqslant 2\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \kappa) + \frac{(Q_{\mathrm{H}} + Q_{\mathrm{S}_1} + 1)(Q_{\mathrm{H}} + 3Q_{\mathrm{S}_1} + 1)}{p - 1} .$$

6.2 Proof of OMUF (Theorem 6.1.2)

Let us fix an adversary $\mathcal{A}_{\mathrm{alg}}$ that makes at most Q_{S_1} queries to S_1 and Q_{H} queries to the random oracle H. Without loss of generality, assume $\mathcal{A}_{\mathrm{alg}}$ makes exactly Q_{S_1} queries to S_1 and exactly one query (i, c_i) to S_2 for each $i \in [Q_{\mathrm{S}_1}]$. Then, after $\mathcal{A}_{\mathrm{alg}}$ returns, we know $\ell = Q_{\mathrm{S}_1}$ and $\mathcal{I}_{\mathrm{fin}} = [Q_{\mathrm{S}_1}]$.

The $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game is formally defined in Figure 6.2. In addition to the original OMUF game (defined in Figure 3.1), for each query $(A \,\|\, m)$ to H, its corresponding hid is recorded in $\mathrm{Hid}(A \,\|\, m)$, and the output of the query is recorded as δ_{hid}. Also, since $\mathcal{A}_{\mathrm{alg}}$ is algebraic, it also provides the representations of A, and the corresponding coefficient $\hat{\boldsymbol{\alpha}}$ are recorded as $\hat{\boldsymbol{\alpha}}_{\mathrm{hid}}$.

Denote the event WIN as $\mathcal{A}_{\mathrm{alg}}$ wins the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game, i.e., all output message-signature pairs $\{m_k^*, \sigma_k^*\}_{k \in [Q_{\mathrm{S}_1}+1]}$ are distinct and valid. Furthermore, denote $\mathrm{str}_k^* := g^{s_k^*} X^{-c_k^* \cdot y_k^*} Z^{y_k^*} \,\|\, m_k^*$. We let E be the event in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game for which, after the validity of the output is

Game $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}(\lambda)$:

$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$; $x \leftarrow\!\!\$\ \mathbb{Z}_p$; $X \leftarrow g^x$; $Z \leftarrow \mathbb{G}$

$\mathrm{sid} \leftarrow 0$; $\ell \leftarrow 0$; $\mathcal{I}_{\mathrm{fin}} \leftarrow \varnothing$; $T \leftarrow ()$; $\mathrm{hid} \leftarrow 0$;

$\mathrm{Hid} \leftarrow ()$

$\{(m^*_k, \sigma^*_k)\}_{k\in[\ell+1]} \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{S}_1,\mathrm{S}_2,\mathrm{H}}_{\mathrm{alg}}(p, g, \mathbb{G}, X, Z)$

If $\exists\ k_1 \neq k_2$ such that $(m^*_{k_1}, \sigma^*_{k_1}) = (m^*_{k_2}, \sigma^*_{k_2})$ then

  Return 0

If $\exists\ k \in [\ell+1]$ such that $y^*_k = 0$

  or $c^*_k \neq \mathrm{H}(g^{s^*_k} X^{-c^*_k \cdot y^*_k} Z^{y^*_k} \,\|\, m^*_k)$

where $(c^*_k, s^*_k, y^*_k) = \sigma^*_k$ then return 0

Return 1

Oracle $\mathrm{H}(A \,\|\, m)$:

If $T(A \,\|\, m) = \perp$ then

  $T(A \,\|\, m) \leftarrow\!\!\$\ \mathbb{Z}_p$

  $\mathrm{hid} \leftarrow \mathrm{hid} + 1$; $\mathrm{Hid}(A \,\|\, m) \leftarrow \mathrm{hid}$

  // $A = g^{\hat\alpha^g} X^{\hat\alpha^{\mathsf{X}}} Z^{\hat\alpha^{\mathsf{Z}}} \prod_{i\in[\mathrm{sid}]} A_i^{\hat\alpha^{\mathsf{A}_i}} C_i^{\hat\alpha^{\mathsf{C}_i}}$

  $\delta_{\mathrm{hid}} \leftarrow T(\mathcal{A} \,\|\, m)$; $\hat{\boldsymbol{\alpha}}_{\mathrm{hid}} \leftarrow \hat{\boldsymbol{\alpha}}$; $\hat{\boldsymbol{\beta}}_{\mathrm{hid}} \leftarrow \hat{\boldsymbol{\beta}}$

Return $T(A \,\|\, m)$

Oracle S_1 :

$\mathrm{sid} \leftarrow \mathrm{sid} + 1$

$a_{\mathrm{sid}}, t_{\mathrm{sid}} \leftarrow\!\!\$\ \mathbb{Z}_p$

$y_{\mathrm{sid}} \leftarrow\!\!\$\ \mathbb{Z}^*_p$

$\mathsf{st}^s_{\mathrm{sid}} \leftarrow (a_{\mathrm{sid}}, y_{\mathrm{sid}}, t_{\mathrm{sid}})$

$A_{\mathrm{sid}} \leftarrow g^{a_{\mathrm{sid}}}$

$C_{\mathrm{sid}} \leftarrow g^{t_{\mathrm{sid}}} Z^{y_{\mathrm{sid}}}$

$\mathsf{msg}_1 \leftarrow (A_{\mathrm{sid}}, C_{\mathrm{sid}})$

Return $(\mathrm{sid}, \mathsf{msg}_1)$

Oracle $\mathrm{S}_2(i, c_i)$:

If $i \notin [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$

  or $c_i = 0$ then

  Return $\perp$

$(a_i, y_i, t_i) \leftarrow \mathsf{st}^s_i$

$s_i \leftarrow a_i + c_i \cdot y_i \cdot x$

$\mathsf{msg}_2 \leftarrow (s_i, y_i, t_i)$

$\mathcal{I}_{\mathrm{fin}} \leftarrow \mathcal{I}_{\mathrm{fin}} \cup \{i\}$

$\ell \leftarrow \ell + 1$

Return msg_2

Figure 6.2: The OMUF security game for the blind signature scheme $\mathsf{BS}_2[\mathsf{GGen}]$.

checked, for each $k \in [Q_{\mathrm{S}_1} + 1]$ and $j = \mathrm{Hid}(\mathrm{str}^*_k)$,[1] the following conditions hold:

$$\hat\alpha^{\mathsf{X}}_j - \sum_{i\in[Q_{\mathrm{S}_1}]} y_i \cdot c_i \cdot \hat\alpha^{\mathsf{A}_i}_j = -\delta_j \cdot y^*_k \ , \tag{6.4}$$

$$\hat\alpha^{\mathsf{Z}}_j + \sum_{i\in[Q_{\mathrm{S}_1}]} y_i \cdot \hat\alpha^{\mathsf{C}_i}_j = y^*_k \ . \tag{6.5}$$

Since $\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{BS}_2[\mathsf{GGen}]}(\mathcal{A}_{\mathrm{alg}}, \lambda) = \mathsf{Pr}[\mathrm{WIN}] = \mathsf{Pr}[\mathrm{WIN} \ \wedge\ E] + \mathsf{Pr}[\mathrm{WIN} \ \wedge\ (\neg E)]$, the theorem follows by combining the following two lemmas with Theorem 4.1.1.

[1] Here, $\mathrm{Hid}(\mathrm{str}^*_k)$ must be defined since a query str^*_k is made to H when checking the validity of the output (m^*_k, σ^*_k).

Lemma 6.2.1. *There exists an adversary* $\mathcal{B}_{\text{wfros}}$ *for the* $\text{WFROS}_{Q_{\text{S}_1},p}$ *problem making at most* $Q_{\text{H}} + Q_{\text{S}_1} + 1$ *queries to the random oracle* H *such that*

$$\mathsf{Adv}^{\text{wfros}}_{Q_{\text{S}_1},p}(\mathcal{B}_{\text{wfros}}) \geqslant \mathsf{Pr}[\text{WIN} \ \wedge \ E] \ . \tag{6.6}$$

Lemma 6.2.2. *There exists an adversary* $\mathcal{B}_{\text{dlog}}$ *for the* DLog *problem running in a similar running time as* $\mathcal{A}_{\text{alg}}$ *such that*

$$\mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\text{dlog}},\lambda) \geqslant \frac{1}{2}\mathsf{Pr}[\text{WIN} \ \wedge \ (\neg E)] \ . \tag{6.7}$$

6.2.1 Proof of Lemma 6.2.1

We first give a detailed description of $\mathcal{B}_{\text{wfros}}$ playing the game $\text{WFROS}_{Q_{\text{S}_1},p}$. To start with, $\mathcal{B}_{\text{wfros}}$ initializes sid, $\mathcal{I}_{\text{fin}}$, ℓ, T, hid, and Hid as described in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game. In addition, $\mathcal{B}_{\text{wfros}}$ samples x, z uniformly from $\mathbb{Z}_p$, sets X to g^x and Z to g^z.

Then, $\mathcal{B}_{\text{wfros}}$ runs $\mathcal{A}_{\text{alg}}$ on input $(p, g, \mathbb{G}, X, Z)$, and with access to the oracles $\widetilde{\text{S}}_1$, $\widetilde{\text{S}}_2$, and $\widetilde{\text{H}}$. These oracles operate as follows:

Oracle $\widetilde{\text{S}}_1$: Same as the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game except that instead of sampling y_{sid}, t_{sid} randomly and setting $C_{\text{sid}} \leftarrow g^{t_{\text{sid}}} Z^{y_{\text{sid}}}$, $\mathcal{B}_{\text{wfros}}$ samples a new variable t'_{sid} uniformly from $\mathbb{Z}_p$ and sets $C_{\text{sid}} = g^{t'_{\text{sid}}}$.

Oracle $\widetilde{\text{S}}_2$: After receiving a query (i, c_i) to $\widetilde{\text{S}}_2$ from $\mathcal{A}_{\text{alg}}$, if $i \notin [\text{sid}] \backslash \mathcal{I}_{\text{fin}}$ or $c_i = 0$, $\mathcal{B}_{\text{wfros}}$ returns $\perp$. Otherwise, $\mathcal{B}_{\text{wfros}}$ makes a query (i, c_i) to S and uses its output as the value y_i. Also, $\mathcal{B}_{\text{wfros}}$ sets $t_i = t'_i - y_i \cdot z$. With the value (a_i, y_i, t_i), the rest of $\widetilde{\text{S}}_2$ is the same as S_2 in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game.

Oracle $\widetilde{\text{H}}$: After receiving a query $(A \,\|\, m)$ to $\widetilde{\text{H}}$ from $\mathcal{A}_{\text{alg}}$, if $T(A \,\|\, m) \neq \perp$, the value $T(A \,\|\, m)$ is returned. Otherwise, since $\mathcal{A}_{\text{alg}}$ is algebraic, $\mathcal{B}_{\text{wfros}}$ also knows the coefficient $\hat{\boldsymbol{\alpha}}$ such that

$$A = g^{\hat{\alpha}^g} X^{\hat{\alpha}^{\mathsf{X}}} Z^{\hat{\alpha}^{\mathsf{Z}}} \prod_{i \in [\text{sid}]} A_i^{\hat{\alpha}^{\mathsf{A}_i}} C_i^{\hat{\alpha}^{\mathsf{C}_i}} \ .$$

Then, $\mathcal{B}_{\text{wfros}}$ issues the query $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ to H, where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^{2Q_{\mathrm{S}_1}+1}$ are such that

$$\alpha^{(i')} = \begin{cases} \hat{\alpha}^{\mathsf{X}} , & i' = 0 \\ -\hat{\alpha}^{\mathsf{A}_i} , & i' = 2i \ , \ i \in [\text{sid}] \\ 0 , & o.w. \end{cases} , \qquad \beta^{(i')} = \begin{cases} -\hat{\alpha}^{\mathsf{Z}} , & i' = 0 \\ -\hat{\alpha}^{\mathsf{C}_i} , & i' = 2i-1 \ , \ i \in [\text{sid}] \\ 0 , & o.w. \end{cases} . \tag{6.8}$$

After receiving the output $(\delta_{\text{hid}}, \text{hid})$, $\mathcal{B}_{\text{wfros}}$ sets $T(A \,\|\, m) \leftarrow \delta_{\text{hid}}$ and $\text{Hid}(A \,\|\, m) \leftarrow \text{hid}$. Finally, $\mathcal{B}_{\text{wfros}}$ returns $T(A \,\|\, m)$.

After $\mathcal{A}_{\text{alg}}$ outputs $\{(m_k^*, \sigma_k^*)\}_{k \in [Q_{\mathrm{S}_1}+1]}$, $\mathcal{B}_{\text{wfros}}$ aborts if the conditions from the event $\text{WIN} \ \wedge \ E$ do not occur. Otherwise, $\mathcal{B}_{\text{wfros}}$ outputs $\mathcal{J} := \{\text{Hid}(\text{str}_k^*) \mid k \in [Q_{\mathrm{S}_1}+1]\}$. Since $\mathcal{B}_{\text{wfros}}$ simulates the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game perfectly, the probability that $\text{WIN} \ \wedge \ E$ occurs when running $\mathcal{B}_{\text{wfros}}$ is the same as in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game with $\mathcal{A}_{\text{alg}}$.

Following the similar analysis of $\mathcal{B}$ in the GGM proof (Section 5.2.3), we know $\mathcal{B}_{\text{wfros}}$ makes at most $Q_{\mathrm{H}} + Q_{\mathrm{S}_1} + 1$ queries to H.

It is left to show that if $\text{WIN} \ \wedge \ E$ occurs within the simulation, then $\mathcal{B}_{\text{wfros}}$ wins the WFROS game. We first show that $|\mathcal{J}| = Q_{\mathrm{S}_1} + 1$. Suppose $|\mathcal{J}| \leqslant Q_{\mathrm{S}_1}$. Since $\mathcal{A}_{\text{alg}}$ wins the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game, we know there exists $k_1, k_2 \in [Q_{\mathrm{S}_1}+1]$ such that $k_1 \neq k_2$ and $\text{Hid}(\text{str}_{k_1}^*) = \text{Hid}(\text{str}_{k_2}^*)$, which implies $\text{str}_{k_1}^* = \text{str}_{k_2}^*$. Therefore, we have

$$g^{s_{k_1}^*} X^{-c_{k_1}^* \cdot y_{k_1}^*} Z^{y_{k_1}^*} = g^{s_{k_2}^*} X^{-c_{k_2}^* \cdot y_{k_2}^*} Z^{y_{k_2}^*} \ , \ m_{k_1}^* = m_{k_2}^* \ . \tag{6.9}$$

Also, let $j = \text{Hid}(\text{str}_{k_1}^*) = \text{Hid}(\text{str}_{k_2}^*)$. Since E occurs in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game simulated by $\mathcal{B}_{\text{wfros}}$, by (6.5), we have

$$y_{k_1}^* = \hat{\alpha}_j^{\mathsf{X}} + \sum_{i \in [Q_{\mathrm{S}_1}]} y_i \cdot \hat{\alpha}_j^{\mathsf{C}_i} = y_{k_2}^* \ .$$

Since $y_{k_1}^* = y_{k_2}^*$ and $c_{k_1}^* = c_{k_2}^*$, by (6.9), we have $s_{k_1}^* = s_{k_2}^*$. However, since $(m_{k_1}^*, \sigma_{k_1}^*)$ and $(m_{k_2}^*, \sigma_{k_2}^*)$ are different message-signature pairs, we have

$$(m_{k_1}^*, c_{k_1}^*, s_{k_1}^*, y_{k_1}^*) \neq (m_{k_2}^*, c_{k_2}^*, s_{k_2}^*, y_{k_2}^*),$$

which yields a contradiction. Therefore, we have $|\mathcal{J}| = Q_{\mathrm{S}_1} + 1$.

Then, since in particular E occurs, by (6.4) and (6.5), it holds that for any $j \in \mathcal{J}$

$$\hat{\alpha}_j^{\mathsf{X}} - \sum_{i\in[Q_{\mathrm{S}_1}]} y_i \cdot c_i \cdot \hat{\alpha}_j^{\mathsf{A}_i} = -\delta_j \left(\hat{\alpha}_j^{\mathsf{Z}} + \sum_{i\in[Q_{\mathrm{S}_1}]} y_i \cdot \hat{\alpha}_j^{\mathsf{C}_i} \right).$$

From the simulation of $\widetilde{\mathrm{H}}$, by (6.8), we have for any $j \in \mathcal{J}$

$$\alpha_j^{(0)} + \sum_{i\in[Q_{\mathrm{S}_1}]} y_i(\alpha_j^{(2i-1)} + c_i \cdot \alpha_j^{(2i)}) = \delta_j \left(\beta_j^{(0)} + \sum_{i\in[Q_{\mathrm{S}_1}]} y_i(\beta_j^{(2i-1)} + c_i \cdot \beta_j^{(2i)}) \right).$$

Therefore, $\mathcal{B}_{\mathrm{wfros}}$ wins the $\mathrm{WFROS}_{Q_{\mathrm{S}_1},p}$ game, which concludes the proof.

6.2.2 *Proof of Lemma 6.2.2*

Proof. We first partition the event $\mathrm{WIN} \ \wedge \ (\neg E)$ into two cases. Denote F_1 as the event in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game that there exists $k \in [Q_{\mathrm{S}_1}+1]$ such that (6.4) does not hold, and denote F_2 as the event that there exists $k \in [Q_{\mathrm{S}_1}+1]$ such that (6.5) does not hold. Then, if E does not occur, we know either F_1 or F_2 occurs. Therefore, we have $\mathrm{WIN} \ \wedge \ (\neg E) = (\mathrm{WIN} \ \wedge \ F_1) \ \vee \ (\mathrm{WIN} \ \wedge \ F_2)$. We then prove the following two claims.

Claim 6.2.3. *There exists* $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ *for the* DLog *problem running in a similar running time as* $\mathcal{A}_{\mathrm{alg}}$ *such that* $\mathsf{Pr}[\mathrm{WIN} \ \wedge \ F_1] \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{B}^{(0)}_{\mathrm{dlog}}, \lambda)$.

Claim 6.2.4. *There exists* $\mathcal{B}^{(1)}_{\mathrm{dlog}}$ *for the* DLog *problem running in a similar running time as* $\mathcal{A}_{\mathrm{alg}}$ *such that* $\mathsf{Pr}[\mathrm{WIN} \ \wedge \ F_2] \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{B}^{(1)}_{\mathrm{dlog}}, \lambda)$.

By the above two claims, we can construct an adversary $\mathcal{B}_{\mathrm{dlog}}$ for the DLog problem that runs either $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ or $\mathcal{B}^{(1)}_{\mathrm{dlog}}$ with 1/2 probability, and we can conclude the lemma since

$$\begin{aligned}\mathsf{Pr}[\mathrm{WIN} \ \wedge \ (\neg E)] &\leqslant \mathsf{Pr}[\mathrm{WIN} \ \wedge \ F_1] + \mathsf{Pr}[\mathrm{WIN} \ \wedge \ F_2] \\ &\leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{B}^{(0)}_{\mathrm{dlog}}, \lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{B}^{(1)}_{\mathrm{dlog}}, \lambda) = 2\mathsf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathcal{B}_{\mathrm{dlog}}, \lambda).\end{aligned}$$

□

Proof of Claim 6.2.3. We first give a detailed description of $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ playing the $\mathrm{DLog}_{\mathbb{G}}$ game.

THE ADVERSARY $\mathcal{B}^{(0)}_{\mathrm{dlog}}$. Initially, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ initializes sid, $\mathcal{I}_{\mathrm{fin}}$, ℓ, T, hid, and Hid as described in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game. After $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ receives $(p, g, \mathbb{G}, W)$ from the $\mathrm{DLog}_{\mathbb{G}}$ game, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ samples z uniformly from $\mathbb{Z}_p$ and sets $X \leftarrow W, Z \leftarrow g^z$. Then, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ runs $\mathcal{A}_{\mathrm{alg}}$ on input $(p, g, \mathbb{G}, X)$ and with access to the oracles $\widetilde{\mathrm{S}}_1$, $\widetilde{\mathrm{S}}_2$, and $\widetilde{\mathrm{H}}$. These oracles operate as follows:

Oracle $\widetilde{\mathbf{S}}_1$: $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ samples $s_{\mathrm{sid}}, t'_{\mathrm{sid}}$ uniformly from $\mathbb{Z}_p$ and y'_{sid} unifomly from $\mathbb{Z}_p^*$ and sets $A_{\mathrm{sid}} = g^{s_{\mathrm{sid}}} X^{-y'_{\mathrm{sid}}}$ and $C_{\mathrm{sid}} = g^{t'_{\mathrm{sid}}}$. Then, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ returns $(\mathrm{sid}, A_{\mathrm{sid}}, C_{\mathrm{sid}})$.

Oracle $\widetilde{\mathbf{S}}_2$: Same as in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game if $i \notin [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$ or $c_i = 0$. Otherwise, after receving a query (i, c_i) to $\widetilde{\mathrm{S}}_2$ from $\mathcal{A}_{\mathrm{alg}}$, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ sets $y_i \leftarrow y'_i / c_i$ and $t_i \leftarrow t'_i - y_i \cdot z$. Then, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ returns (s_i, y_i, t_i) to $\mathcal{A}_{\mathrm{alg}}$.

Oracle $\widetilde{\mathbf{H}}$: Same as in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game.

After receiving the output $\{(m^*_k, \sigma^*_k)\}_{k \in [Q_{\mathrm{S}_1}+1]}$, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ aborts if the event $\mathrm{WIN} \wedge F_1$ does not occur.

It is clear that $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ simulates the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game perfectly. Therefore, it is left to show that if the event $\mathrm{WIN} \wedge F_1$ occurs within the simulation, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ can compute the discrete log of X, which equals to W.

Suppose $\mathrm{WIN} \wedge F_1$ occurs. There exists $k \in [Q_{\mathrm{S}_1} + 1]$ and $j = \mathrm{Hid}(\mathrm{str}^*_k)$ such that (6.4) does not hold. Since $\mathrm{Hid}(\mathrm{str}^*_k) = j$ and $\delta_j = c^*_k$, we have

$$g^{s^*_k} X^{-\delta_j \cdot y^*_k} Z^{y^*_k} = g^{\hat{\alpha}^{\mathsf{g}}_j} X^{\hat{\alpha}^{\mathsf{X}}_j} Z^{\hat{\alpha}^{\mathsf{Z}}_j} \prod_{i \in [\mathrm{sid}]} A_i^{\hat{\alpha}^{\mathsf{A}_i}_j} C_i^{\hat{\alpha}^{\mathsf{C}_i}_j} \,. \tag{6.10}$$

Similar to the preceding case, since $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ knows the discrete log of Z as z, by substituting $A_i = g^{s_i} X^{-c_i \cdot y_i}$, $C_i = g^{t_i} Z^{y_i}$, and $Z = g^z$ into (6.10), we have

$$g^{s^*_k + y^*_k \cdot z} X^{-\delta_j \cdot y^*_k} = g^{\hat{\alpha}^{\mathsf{g}}_j + \hat{\alpha}^{\mathsf{Z}}_j \cdot z + \sum_{i \in [Q_{\mathrm{S}_1}]} (\hat{\alpha}^{\mathsf{A}_i}_j \cdot s_i + \hat{\alpha}^{\mathsf{C}_i}_j \cdot (t_i + y_i \cdot z))} X^{\hat{\alpha}^{\mathsf{X}}_j - \sum_{i \in [Q_{\mathrm{S}_1}]} y_i \cdot c_i \cdot \hat{\alpha}^{\mathsf{A}_i}_j} \,.$$

Since (6.4) does not hold, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ can compute the discrete log of X as

$$x := \frac{s^*_k + y^*_k \cdot z - \hat{\alpha}^{\mathsf{g}}_j - \hat{\alpha}^{\mathsf{Z}}_j \cdot z - \sum_{i \in [Q_{\mathrm{S}_1}]} (\hat{\alpha}^{\mathsf{A}_i}_j \cdot s_i + \hat{\alpha}^{\mathsf{C}_i}_j \cdot (t_i + y_i \cdot z))}{\hat{\alpha}^{\mathsf{X}}_j - \sum_{i \in [Q_{\mathrm{S}_1}]} y_i \cdot c_i \cdot \hat{\alpha}^{\mathsf{A}_i}_j + \delta_j \cdot y^*_k} \,.$$

□

Proof of Claim 6.2.4. We first give a detailed description of $\mathcal{B}^{(1)}_{\mathrm{dlog}}$ playing the $\mathrm{DLog}_{\mathbb{G}}$ game.

THE ADVERSARY $\mathcal{B}^{(1)}_{\text{dlog}}$. Initially, $\mathcal{B}^{(1)}_{\text{dlog}}$ initializes sid, $\mathcal{I}_{\text{fin}}$, ℓ, T, hid, and Hid as described in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game. After $\mathcal{B}^{(1)}_{\text{dlog}}$ receives $(p, g, \mathbb{G}, W)$ from the $\text{DLog}_{\mathbb{G}}$ game, $\mathcal{B}^{(1)}_{\text{dlog}}$ samples x uniformly from $\mathbb{Z}_p$ and sets $X \leftarrow g^x, Z \leftarrow W$. Then, $\mathcal{B}^{(1)}_{\text{dlog}}$ runs $\mathcal{A}_{\text{alg}}$ on input $(p, g, \mathbb{G}, X)$ and with access to the oracles $\widetilde{\text{S}}_1$, $\widetilde{\text{S}}_2$, and $\widetilde{\text{H}}$. Since $\mathcal{B}^{(1)}_{\text{dlog}}$ knows $X = g^x$, $\mathcal{B}^{(1)}_{\text{dlog}}$ can simulate all the oracles $\widetilde{\text{S}}_1$, $\widetilde{\text{S}}_2$, and $\widetilde{\text{H}}$ the same as in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game. After receiving the output $\{(m^*_k, \sigma^*_k)\}_{k\in[Q_{\text{S}_1}+1]}$, $\mathcal{B}^{(1)}_{\text{dlog}}$ aborts if the event $\text{WIN} \wedge F_2$ does not occur.

It is clear that $\mathcal{B}^{(1)}_{\text{dlog}}$ simulates the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{BS}_2[\mathsf{GGen}]}$ game perfectly. Therefore, it is left to show that if the event $\text{WIN} \wedge F_2$ occurs within the simulation, $\mathcal{B}^{(1)}_{\text{dlog}}$ can compute the discrete log of Z, which equals to W.

Suppose $\text{WIN} \wedge F_2$ occurs. There exists $k \in [Q_{\text{S}_1} + 1]$ and $j = \text{Hid}(\text{str}^*_k)$ such that (6.5) does not hold. Since $\text{Hid}(\text{str}^*_k) = j$, we have

$$g^{s^*_k} X^{-c^*_k \cdot y^*_k} Z^{y^*_k} = g^{\hat{\alpha}^g_j} X^{\hat{\alpha}^{\mathsf{X}}_j} Z^{\hat{\alpha}^{\mathsf{Z}}_j} \prod_{i\in[\text{sid}]} A_i^{\hat{\alpha}^{\mathsf{A}_i}_j} C_i^{\hat{\alpha}^{\mathsf{C}_i}_j} \ . \tag{6.11}$$

From the simulation of $\widetilde{\text{S}}_2$, for each $i \in [Q_{\text{S}_1}]$, we have

$$g^{s_i} = A_i X^{c_i \cdot y_i} \ , \ g^{t_i} = C_i Z^{-y_i} \ .$$

Also, $\mathcal{B}^{(1)}_{\text{dlog}}$ knows the discrete log of X as x. By substituting $A_i = g^{s_i} X^{-c_i \cdot y_i}$, $C_i = g^{t_i} Z^{y_i}$, and $X = g^x$ into (6.11), we have

$$g^{s^*_k - x \cdot c^*_k \cdot y^*_k} Z^{y^*_k} = g^{\hat{\beta}^g_j + \hat{\beta}^{\mathsf{X}}_j \cdot x + \sum_{i\in[Q_{\text{S}_1}]} (\hat{\beta}^{\mathsf{A}_i}_j \cdot (s_i - c_i \cdot y_i \cdot x) + \hat{\beta}^{\mathsf{C}_i}_j \cdot t_i)} Z^{\hat{\beta}^{\mathsf{Z}}_j + \sum_{i\in[Q_{\text{S}_1}]} y_i \cdot \hat{\beta}^{\mathsf{C}_i}_j} \ .$$

Since (6.5) does not hold, $\mathcal{B}^{(1)}_{\text{dlog}}$ can compute the discrete log of Z as

$$z := \frac{s^*_k - x \cdot c^*_k \cdot y^*_k - \hat{\beta}^g_j - \hat{\beta}^{\mathsf{X}}_j \cdot x - \sum_{i\in[Q_{\text{S}_1}]} (\hat{\beta}^{\mathsf{A}_i}_j \cdot (s_i - c_i \cdot y_i \cdot x) + \hat{\beta}^{\mathsf{C}_i}_j \cdot t_i)}{\hat{\beta}^{\mathsf{Z}}_j + \sum_{i\in[Q_{\text{S}_1}]} y_i \cdot \hat{\beta}^{\mathsf{C}_i}_j - y^*_k} \ .$$

□

6.3 Partially Blind Signatures

This section presents our partially blind signature scheme, PBS, which is detailed in Figure 6.3. The scheme builds on top of the BS_2 scheme by replacing the extra generator Z contained in the public key with the output of a hash function F (also modeled as a random oracle in the OMUF

Algorithm $\mathsf{Setup}(1^\lambda)$:
$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$
Select $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_p^*$
Select $\mathrm{F} : \{0,1\}^* \to \mathbb{G}$
Return $par \leftarrow (p, \mathbb{G}, g, \mathrm{H}, \mathrm{F})$

Algorithm $\mathsf{KeyGen}(par)$:
$(p, \mathbb{G}, g, \mathrm{H}, \mathrm{F}) \leftarrow par$
$x \leftarrow\!\!\$\ \mathbb{Z}_p$; $X \leftarrow g^x$
$sk \leftarrow x$; $pk \leftarrow X$
Return (sk, pk)

Algorithm $\mathsf{S}_1(sk, \mathsf{info})$:
$x \leftarrow sk$; $X \leftarrow g^x$; $Z \leftarrow \mathrm{F}(\mathsf{info})$
$a, t \leftarrow\!\!\$\ \mathbb{Z}_p$; $y \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$A \leftarrow g^a$; $C \leftarrow g^t Z^y$
$\mathsf{st}^s \leftarrow (a, y, t, x)$; $\mathsf{msg}_1 \leftarrow (A, C)$
Return $(\mathsf{st}^s, \mathsf{msg}_1)$

Algorithm $\mathsf{S}_2(\mathsf{st}^s, c)$:
If $c = 0$ then return $\perp$
$(a, y, t, x) \leftarrow \mathsf{st}^s$
$s \leftarrow a + c \cdot y \cdot x$
Return $\mathsf{msg}_2 \leftarrow (s, y, t)$

Algorithm $\mathsf{U}_1(pk, \mathsf{msg}_1, \mathsf{info}, m)$:
$X \leftarrow pk$; $(A, C) \leftarrow \mathsf{msg}_1$; $Z \leftarrow \mathrm{F}(\mathsf{info})$
$r_1 \leftarrow\!\!\$\ \mathbb{Z}_p$; $\gamma_1, \gamma_2 \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$A' \leftarrow g^{r_1} \cdot A^{\gamma_1/\gamma_2} C^{\gamma_1}$
$c' \leftarrow \mathrm{H}(\mathsf{info} \,\|\, A' \,\|\, m)$
$c \leftarrow c' \cdot \gamma_2$
$\mathsf{st}^u \leftarrow (c, c', r_1, \gamma_1, \gamma_2, X, Z, A, C)$
Return (st^u, c)

Algorithm $\mathsf{U}_2(\mathsf{st}^u, \mathsf{msg}_2)$:
$(c, c', r_1, \gamma_1, \gamma_2, X, Z, A, C) \leftarrow \mathsf{st}^u$
$(s, y, t) \leftarrow \mathsf{msg}_2$
If $y = 0$ or $C \neq g^t Z^y$ or $g^s \neq A \cdot X^{c \cdot y}$
then return $\perp$
$s' \leftarrow r_1 + (\gamma_1/\gamma_2) \cdot s + \gamma_1 \cdot t$
$y' \leftarrow \gamma_1 \cdot y$
Return $\sigma \leftarrow (c', s', y')$

Algorithm $\mathsf{Ver}(pk, \mathsf{info}, \sigma, m)$:
$X \leftarrow pk$; $Z \leftarrow \mathrm{F}(\mathsf{info})$; $(c, s, y) \leftarrow \sigma$
If $y = 0$ then return 0
$A \leftarrow g^s \cdot X^{-c \cdot y} \cdot Z^y$
If $c \neq \mathrm{H}(\mathsf{info} \,\|\, A \,\|\, m)$ then return 0
Return 1

Figure 6.3: The partially blind signature scheme $\mathsf{PBS} = \mathsf{PBS}[\mathsf{GGen}]$.

proof) applied to the public input info. We do not formally redefine the syntax of partially blind signatures, but we note that it simply extends that of blind signatures by adding the extra input $\mathsf{info} \in \{0,1\}^*$ to the signer, the user, and the verification algorithm.

BLINDNESS. We first study the blindness of PBS. The $\mathrm{PBlind}^{\mathcal{A}}_{\mathsf{PBS}}$ game is defined in Figure 6.4. The only difference between PBlind and Blind is that initially, the adversary $\mathcal{A}$ also picks a public information info and interacts with $\mathsf{PBS}.\mathsf{U}_1$ and $\mathsf{PBS}.\mathsf{U}_2$ for signing (info, m_0) and (info, m_1).

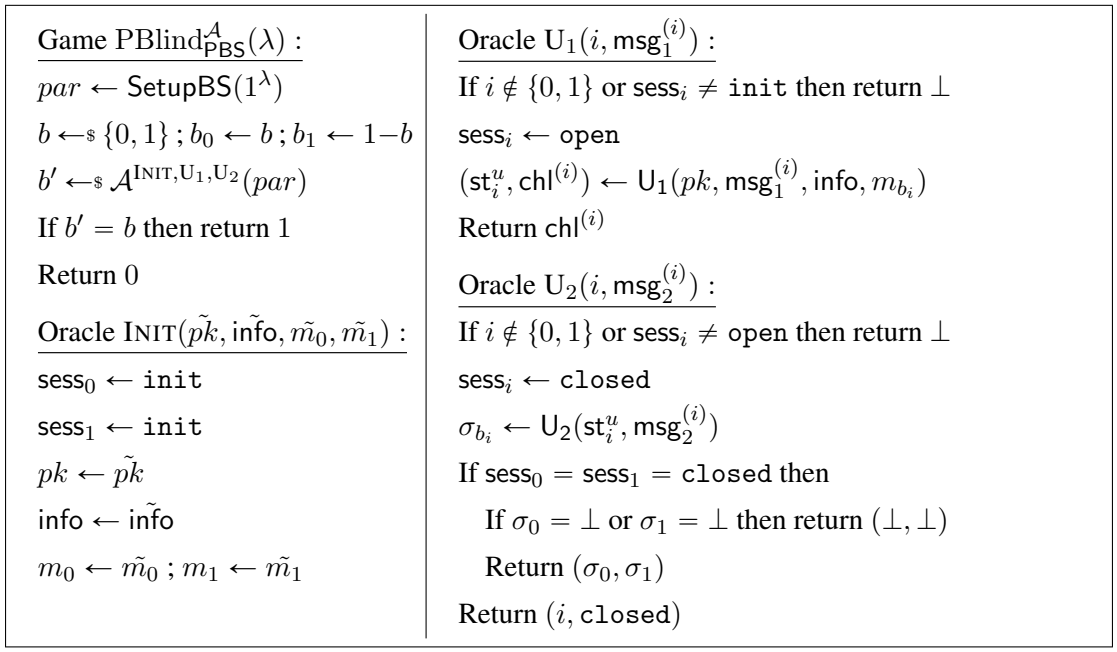

Figure 6.4: The PBlind security game for a partially blind signature scheme PBS.

Denote the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{pblind}}_{\mathsf{PBS}}(\mathcal{A}, \lambda) := \left| \mathsf{Pr}[\mathrm{PBlind}^{\mathcal{A}}_{\mathsf{PBS}}(\lambda) = 1] - \frac{1}{2} \right| .$$

We say a partially blind signature scheme PBS is perfectly blind if and only if $\mathsf{Adv}^{\mathrm{pblind}}_{\mathsf{PBS}}(\mathcal{A}) = 0$ for any $\mathcal{A}$.

Theorem 6.3.1. *For a group generation algorithm $\mathbb{G}$, the partially blind signature scheme* $\mathsf{PBS}[\mathsf{GGen}]$ *is perfectly blind.*

Since the algorithm $\mathsf{PBS}.\mathsf{U}_1$ and $\mathsf{PBS}.\mathsf{U}_2$ are almost the same as $\mathsf{U}_1\mathsf{BS}_2$ and $\mathsf{U}_2\mathsf{BS}_2$, we can use a proof similar to the one for BS_2 (Section 6.1) to show $\mathsf{PBS}[\mathsf{GGen}]$ is perfectly blind. The only difference is that in BS_2, Z is given in the public key, while in $\mathsf{PBS}[\mathsf{GGen}]$, Z is given by $\mathrm{F}(\mathsf{info})$.

OMUF SECURITY. We next study the OMUF security of PBS. Note that the definition must also be adjusted: The main difference is that the adversary wins as long as it can produce $\ell + 1$ valid message-signature pairs for some info for which it has run only ℓ signing sessions, regardless of

how many signing sessions are run with $\mathsf{info}' \neq \mathsf{info}$ (i.e., their number could be higher than ℓ). The corresponding game is defined in Figure 6.5, for the specific case of the scheme PBS. We prove the following theorem.

Game $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}(\lambda)$:

$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\kappa)$; $x \leftarrow\!\!\$\ \mathbb{Z}_p$; $X \leftarrow g^x$

$\mathrm{sid} \leftarrow 0$; $\mathcal{I}_{\mathrm{fin}} \leftarrow \varnothing$; $T_1 \leftarrow ()$; $T_2 \leftarrow ()$

$\ell \leftarrow$ a table where all entry are initially set to 0

$\mathrm{fid} \leftarrow 0$; $\mathrm{Fid} \leftarrow ()$; $\mathrm{Hid} \leftarrow ()$

$(\mathsf{info}^*, \{(m^*_k, \sigma^*_k)\}_{k \in [\ell(\mathsf{info}^*)+1]})$
$\quad \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{S}_1,\mathrm{S}_2,\mathrm{H},\mathrm{F}}_{\mathrm{alg}}(p, g, \mathbb{G}, X)$

If $\exists\ k_1 \neq k_2\ :\ (m^*_{k_1}, \sigma^*_{k_1}) = (m^*_{k_2}, \sigma^*_{k_2})$ then
  Return 0

If $\exists\ k \in [\ell(\mathsf{info}^*) + 1]$ such that $y^*_k = 0$
  or $c^*_k \neq \mathrm{H}(\mathsf{info}^* \,\|\, g^{s^*_k} X^{-c^*_k \cdot y^*_k} Z^{y^*_k} \,\|\, m^*_k)$
  where $(c^*_k, s^*_k, y^*_k) = \sigma^*_k$ and $Z = \mathrm{F}(\mathsf{info}^*)$
  then return 0

Return 1

Oracle $\mathrm{H}(\mathsf{info} \,\|\, A \,\|\, m)$:

If $T_1(\mathsf{info} \,\|\, A \,\|\, m) = \perp$ then
  $T_1(\mathsf{info} \,\|\, A \,\|\, m) \leftarrow\!\!\$\ \mathbb{Z}_p$
  $\mathrm{hid} \leftarrow \mathrm{hid} + 1$
  $\mathrm{Hid}(\mathsf{info} \,\|\, A \,\|\, m) \leftarrow \mathrm{hid}$
  $/\!/\ A = g^{\hat{\alpha}_g} X^{\hat{\alpha}_{\mathsf{X}}} \prod_{i \in [\mathrm{fid}]} Z_i^{\hat{\alpha}_{\mathsf{Z}_i}} \prod_{i \in [\mathrm{sid}]} A_i^{\hat{\alpha}_{\mathsf{A}_i}} C_i^{\hat{\alpha}_{\mathsf{C}_i}}$
  $\delta_{\mathrm{hid}} \leftarrow T_1(\mathsf{info} \,\|\, A \,\|\, m)$
  $\hat{\boldsymbol{\alpha}}_{\mathrm{hid}} \leftarrow \hat{\boldsymbol{\alpha}}$; $\hat{\boldsymbol{\beta}}_{\mathrm{hid}} \leftarrow \hat{\boldsymbol{\beta}}$
Return $T_1(\mathsf{info} \,\|\, A \,\|\, m)$

Oracle $\mathrm{S}_1(\mathsf{info})$:

$Z \leftarrow \mathrm{F}(\mathsf{info})$
$\mathrm{sid} \leftarrow \mathrm{sid} + 1$; $\mathsf{info}_{\mathrm{sid}} \leftarrow \mathsf{info}$
$a_{\mathrm{sid}}, t_{\mathrm{sid}} \leftarrow\!\!\$\ \mathbb{Z}_p$; $y_{\mathrm{sid}} \leftarrow\!\!\$\ \mathbb{Z}^*_p$
$\mathsf{st}^s_{\mathrm{sid}} \leftarrow (a_{\mathrm{sid}}, y_{\mathrm{sid}}, t_{\mathrm{sid}})$
$A_{\mathrm{sid}} \leftarrow g^{a_{\mathrm{sid}}}$; $C_{\mathrm{sid}} \leftarrow g^{t_{\mathrm{sid}}} Z^{y_{\mathrm{sid}}}$
$\mathsf{msg}_1 \leftarrow (A_{\mathrm{sid}}, C_{\mathrm{sid}})$
Return $(\mathrm{sid}, \mathsf{msg}_1)$

Oracle $\mathrm{S}_2(i, c_i)$:

If $i \notin [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$ or $c_i = 0$ then
  Return $\perp$
$(a_i, y_i, t_i) \leftarrow \mathsf{st}^s_i$
$s_i \leftarrow a_i + c_i \cdot y_i \cdot x$
$\mathsf{msg}_2 \leftarrow (s_i, y_i, t_i)$
$\mathcal{I}_{\mathrm{fin}} \leftarrow \mathcal{I}_{\mathrm{fin}} \cup \{i\}$
$\mathcal{I}^{(\mathsf{info}_i)}_{\mathrm{fin}} \leftarrow \mathcal{I}^{(\mathsf{info}_i)}_{\mathrm{fin}} \cup \{i\}$
$\ell(\mathsf{info}) \leftarrow \ell(\mathsf{info}) + 1$
Return msg_2

Oracle $\mathrm{F}(\mathsf{info})$:

If $T_2(\mathsf{info}) = \perp$ then
  $T_2(\mathsf{info}) \leftarrow\!\!\$\ \mathbb{G}$
  $\mathrm{fid} \leftarrow \mathrm{fid} + 1$; $\mathrm{Fid}(\mathsf{info}) \leftarrow \mathrm{fid}$
  $Z_{\mathrm{fid}} = T_2(\mathsf{info})$
  $\mathcal{I}^{(\mathsf{info})}_{\mathrm{fin}} \leftarrow \varnothing$
Return $T_2(\mathsf{info})$

Figure 6.5: The OMUF security game for the partially blind signature scheme PBS[GGen].

Theorem 6.3.2. *Let* GGen *be a group generation algorithm. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary for the game* $\mathrm{OMUF}^{\mathsf{PBS[GGen]}}(\lambda)$ *such that for each public information* info*, makes at most* Q_{S_1} *queries to* S_1 *and* Q_{H} *queries to the random oracle* H *that start with* info*. Also, let the total number of distinct public information* info*'s queried by* $\mathcal{A}_{\mathrm{alg}}$ *to* S_1 *be bounded by* Q_{info}*. Then, there exists an adversary* $\mathcal{B}_{\mathrm{dlog}}$ *for the* DLog *problem running in similar running time as* $\mathcal{A}_{\mathrm{alg}}$ *such that*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PBS[GGen]}}(\mathcal{A}_{\mathrm{alg}}, \lambda) \leqslant 2\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\mathrm{dlog}}, \lambda) + \frac{Q_{\mathsf{info}}(Q_{\mathrm{H}} + 3Q_{\mathrm{S}_1} + 1)^2 + 2}{2^{\kappa} - 1} .$$

The proof is very similar to that for BS_2 except we need to additionally perform a hybrid argument over queries to F, guessing which info will be the one leading to a one-more forgery. However, we need to work harder here to ensure the discrete logarithm advantage does not scale with Q_{info}.

We also note that we have no argument supporting the fact that the information-theoretic term in Theorem 6.3.2 is tight and the inclusion of info in H is necessary. However, a tighter analysis appears to require studying a more general version of WFROS. We leave this to future work.

Proof of Theorem 6.3.2. Let $\mathcal{A}_{\mathrm{alg}}$ be an algebraic adversary described in the theorem. The game $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ is formally defined in Figure 6.5. Without loss of generality, we assume that if $\mathcal{A}_{\mathrm{alg}}$ outputs the public information info^*, then $\mathcal{A}_{\mathrm{alg}}$ makes exactly Q_{S_1} queries to S_1 and Q_{S_1} queries to S_2 that do not return $\perp$ for info^*. Then, when $\mathcal{A}_{\mathrm{alg}}$ returns, we know $\ell(\mathsf{info}^*) = Q_{\mathrm{S}_1}$.

In the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game, the corresponding hid for each query $(\mathsf{info} \,\|\, A \,\|\, m)$ to H is recorded in $\mathrm{Hid}(\mathsf{info} \,\|\, A \,\|\, m)$, and the output of the query is recorded as δ_{hid}. Also, since $\mathcal{A}_{\mathrm{alg}}$ is algebraic, $\mathcal{A}_{\mathrm{alg}}$ also provides the representation of A, and the corresponding coefficients $\boldsymbol{\alpha}$ are recorded as $\boldsymbol{\alpha}_{\mathrm{hid}}$. The corresponding fid for each new query info to S_1 is recorded in $\mathrm{Fid}(\mathsf{info})$. Also, $\mathcal{I}^{(\mathsf{info})}_{\mathrm{fin}}$ records the subset of $\mathcal{I}_{\mathrm{fin}}$ corresponding to signing sessions with public information info.

Denote the event WIN as $\mathcal{A}_{\mathrm{alg}}$ wins the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game, i.e., all the output message-signature pairs $\{m^*_k, \sigma^*_k\}_{k \in [Q_{\mathrm{S}_1}+1]}$ are distinct and valid for info^*. Furthermore, we denote $\mathrm{str}^*_k := \mathsf{info}^* \,\|\, g^{s^*_k} X^{-c^*_k \cdot y^*_k} Z^{y^*_k}_{\mathrm{Fid}(\mathsf{info}^*)} \,\|\, m^*_k$. We let E be the event in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game that after the validity of the output is checked, for each $k \in [Q_{\mathrm{S}_1} + 1]$, $j = \mathrm{Hid}(\mathrm{str}^*_k)$, and $i^* = \mathrm{Fid}(\mathsf{info}^*)$ [2], the

[2]Here, $\mathrm{Hid}(\mathrm{str}^*_k)$ must be defined since a query str^*_k is made to H when checking the validity of the output (m^*_k, σ^*_k).

following conditions hold:

$$\hat{\alpha}^{\mathsf{Z}_{i^*}} + \sum_{i \in \mathcal{I}_{\text{fin}}^{(\mathsf{info}^*)}} y_i \cdot \hat{\alpha}_j^{\mathsf{C}_i} = y_k^* \ , \tag{6.12}$$

$$\hat{\alpha}_j^{\mathsf{X}} - \sum_{i \in \mathcal{I}_{\text{fin}}} y_i \cdot c_i \cdot \hat{\alpha}_j^{\mathsf{A}_i} = -\delta_j \cdot y_k^* \ , \tag{6.13}$$

$$\forall i \in [\text{sid}] \backslash \mathcal{I}_{\text{fin}} \ : \ \hat{\alpha}_j^{\mathsf{A}_i} = 0 \ . \tag{6.14}$$

Since $\mathsf{Adv}^{\text{omuf}}_{\mathsf{PBS[GGen]}}(\mathcal{A}_{\text{alg}}, \lambda) = \mathsf{Pr}[\text{WIN}] = \mathsf{Pr}[\text{WIN} \ \wedge \ E] + \mathsf{Pr}[\text{WIN} \ \wedge \ (\neg E)]$, the theorem follows by combining the following two lemmas with Theorem 4.1.1.

Lemma 6.3.3. *There exists an adversary $\mathcal{B}_{\text{wfros}}$ for the $\text{WFROS}_{Q_{\text{S}_1},p}$ problem making at most $Q_{\text{H}} + Q_{\text{S}_1} + 1$ queries to the random oracle H such that*

$$\mathsf{Adv}^{\text{wfros}}_{Q_{\text{S}_1},p}(\mathcal{B}_{\text{wfros}}) \geqslant \frac{1}{Q_{\mathsf{info}}} \mathsf{Pr}[\text{WIN} \ \wedge \ E] \ . \tag{6.15}$$

Lemma 6.3.4. *There exists an adversary $\mathcal{B}_{\text{dlog}}$ for the DLog problem running in a similar running time as $\mathcal{A}_{\text{alg}}$ such that*

$$\mathsf{Pr}[\text{WIN} \ \wedge \ (\neg E)] \leqslant 2\mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\text{dlog}}, \lambda) + \frac{2}{p-1} \ . \tag{6.16}$$

□

6.3.1 Proof of Lemma 6.3.3

We first give a detailed description of $\mathcal{B}_{\text{wfros}}$ playing the WFROS game.

THE ADVERSARY $\mathcal{B}_{\text{wfros}}$. To start with, $\mathcal{B}_{\text{wfros}}$ first samples a label $\hat{\imath}^*$ uniformly from $[Q_{\mathsf{info}}]$. Also, $\mathcal{B}_{\text{wfros}}$ samples x uniformly from $\mathbb{Z}_p$, sets X to g^x, and initializes sid, hid, fid, Hid, Fid, $\mathcal{I}_{\text{fin}}$, T_1, and T_2 as described in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game. In addition, $\mathcal{B}_{\text{wfros}}$ initializes tfid to 0 and tFid to an empty table, which are used to record the labels of info queries to S_1, and initializes tsid to 0 and tSid to an empty table, which are used to record the labels of session IDs for info such that $\text{tFid}(\mathsf{info}) = \hat{\imath}^*$.

Then, $\mathcal{B}_{\text{wfros}}$ runs $\mathcal{A}_{\text{alg}}$ on input $(p, g, \mathbb{G}, X)$ and with access to the oracles $\widetilde{\text{F}}$, $\widetilde{\text{S}}_1$, $\widetilde{\text{S}}_2$, and $\widetilde{\text{H}}$. These oracles, operate as follows:

Oracles $\widetilde{\text{F}}$: Same as in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game except instead of sampling $T_2(\mathsf{info})$ uniformly from $\mathbb{G}$, if $T_2(\mathsf{info}) = \perp$, $\mathcal{B}_{\text{wfros}}$ samples z_{fid} uniformly from $\mathbb{Z}_p$ and sets $T_2(\mathsf{info}) \leftarrow g^{z_{\text{fid}}}$.

Oracles $\widetilde{\mathrm{S}}_1$: After receiving a query info to $\widetilde{\mathrm{S}}_1$ from $\mathcal{A}_{\mathrm{alg}}$, if $\mathrm{tFid}(\mathsf{info}) = \perp$, $\mathcal{B}_{\mathrm{wfros}}$ increases tfid by 1 and sets $\mathrm{tFid}(\mathsf{info}) = \mathrm{tfid}$. Then, there are two cases:

- If $\mathrm{tFid}(\mathsf{info}) \neq \hat{i}^*$, $\mathcal{B}_{\mathrm{wfros}}$ samples $s_{\mathrm{sid}}, t'_{\mathrm{sid}}$ uniformly from $\mathbb{Z}_p$ and samples y'_{sid} uniformly from $\mathbb{Z}_p^*$. Then, $\mathcal{B}_{\mathrm{wfros}}$ sets $A_{\mathrm{sid}} = g^{s_{\mathrm{sid}}} X^{-y'_{\mathrm{sid}}}$ and $C_{\mathrm{sid}} = g^{t'_{\mathrm{sid}}}$.
- If $\mathrm{tFid}(\mathsf{info}) = \hat{i}^*$, $\mathcal{B}_{\mathrm{wfros}}$ samples $a_{\mathrm{sid}}, t'_{\mathrm{sid}}$ uniformly from $\mathbb{Z}_p$ and sets $A_{\mathrm{sid}} = g^{a_{\mathrm{sid}}}$ and $C_{\mathrm{sid}} = g^{t'_{\mathrm{sid}}}$. Also, $\mathcal{B}_{\mathrm{wfros}}$ increases tsid by 1 and sets $\mathrm{tSid}(\mathrm{tsid}) \leftarrow \mathrm{sid}$.

Finally, $\mathcal{B}_{\mathrm{wfros}}$ returns $(\mathrm{sid}, A_{\mathrm{sid}}, C_{\mathrm{sid}})$.

Oracles $\widetilde{\mathrm{S}}_2$: After receiving a query (i, c_i) to $\widetilde{\mathrm{S}}_2$ from $\mathcal{A}_{\mathrm{alg}}$, if $i \notin [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$ or $c_i = 0$, $\mathcal{B}_{\mathrm{wfros}}$ returns $\perp$. Otherwise, there are two cases:

- If $\mathrm{tFid}(\mathsf{info}_i) \neq \hat{i}^*$, $\mathcal{B}_{\mathrm{wfros}}$ computes $y_i \leftarrow y'_i / c_i$ and $t_i \leftarrow t'_i - y_i \cdot z_{\mathrm{Fid}(\mathsf{info}_i)}$.
- If $\mathrm{tFid}(\mathsf{info}_i) = \hat{i}^*$, let i' be the index in $[\mathrm{sid}]$ such that $\mathrm{tSid}(i') = i$ and $\mathcal{B}_{\mathrm{wfros}}$ sets $\tilde{c}_{i'} \leftarrow c_i$. Then, $\mathcal{B}_{\mathrm{wfros}}$ makes a query $(i', \tilde{c}_{i'})$ to S. After $\mathcal{B}_{\mathrm{wfros}}$ receives $\tilde{y}_{i'}$ from S, $\mathcal{B}$ sets $y_i \leftarrow \tilde{y}_{i'}$ and $t_i \leftarrow t'_i - y_i \cdot z_{\mathrm{Fid}(\mathsf{info}_i)}$.

Finally, $\mathcal{B}_{\mathrm{wfros}}$ returns (s_i, y_i, t_i).

Oracles $\widetilde{\mathrm{H}}$: After receiving a query $(\mathsf{info} \,\|\, A \,\|\, m)$ to $\widetilde{\mathrm{H}}$ from $\mathcal{A}_{\mathrm{alg}}$, if $T_1(\mathsf{info} \,\|\, A \,\|\, m) \neq \perp$ or $\mathrm{tFid}(\mathsf{info}) \neq \hat{i}^*$, then $\widetilde{\mathrm{H}}$ is the same as H in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game. Otherwise, since $\mathcal{A}_{\mathrm{alg}}$ is algebraic, $\mathcal{B}_{\mathrm{wfros}}$ also knows $\hat{\boldsymbol{\alpha}}$ such that

$$A = g^{\hat{\alpha}^g} X^{\hat{\alpha}^{\mathsf{X}}} \prod_{i \in [\mathrm{fid}]} Z_i^{\hat{\alpha}^{\mathsf{Z}_i}} \prod_{i \in [\mathrm{sid}]} A_i^{\hat{\alpha}^{\mathsf{A}_i}} C_i^{\hat{\alpha}^{\mathsf{C}_i}} \ .$$

Then, $\mathcal{B}_{\mathrm{wfros}}$ issues the query $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ to H, where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^{2Q_{\mathrm{S}_1}+1}$ such that

$$\alpha^{(i')} = \begin{cases} \hat{\alpha}^{\mathsf{X}} - \sum_{i \in [\mathrm{sid}], \mathrm{tFid}(\mathsf{info}_i) \neq \hat{i}^*} \hat{\alpha}^{\mathsf{A}_i} \cdot y'_i \ , & i' = 0 \\ -\hat{\alpha}^{\mathsf{A}_{\mathrm{tSid}(i)}} \ , & i' = 2i \ , \ i \in [\mathrm{tsid}] \\ 0 \ , & o.w. \end{cases} \ , \tag{6.17}$$

$$\beta^{(i')} = \begin{cases} -\hat{\alpha}^{\mathsf{Z}_{\hat{i}^*}} \ , & i' = 0 \\ -\hat{\alpha}^{\mathsf{C}_{\mathrm{tSid}(i)}} \ , & i' = 2i - 1 \ , \ i \in [\mathrm{tsid}] \\ 0 \ , & o.w. \end{cases} \ . \tag{6.18}$$

After receiving the output $(\delta_{\mathrm{hid}}, \mathrm{hid})$, $\mathcal{B}_{\mathrm{wfros}}$ sets $T_1(\mathsf{info} \parallel A \parallel m) \leftarrow \delta_{\mathrm{hid}}$ and $\mathrm{Hid}(\mathsf{info} \parallel A \parallel m) \leftarrow \mathrm{hid}$. Finally, $\mathcal{B}_{\mathrm{wfros}}$ returns $T_1(\mathsf{info} \parallel A \parallel m)$.

After receiving the output $\{\mathsf{info}^*, (m_k^*, \sigma_k^*)\}_{k\in[Q_{\mathrm{S}_1}+1]}$ from $\mathcal{A}_{\mathrm{alg}}$, $\mathcal{B}_{\mathrm{wfros}}$ aborts if the conditions from the event $\mathrm{WIN} \ \wedge \ E$ do not occur. Otherwise, $\mathcal{B}_{\mathrm{wfros}}$ outputs $\mathcal{J} := \{\mathrm{Hid}(\mathrm{str}_k^*) \mid k \in [Q_{\mathrm{S}_1}+1]\}$.

ANALYSIS OF $\mathcal{B}_{\mathrm{wfros}}$. Note that $\mathcal{B}_{\mathrm{wfros}}$ makes a query to H at most once when it receives a query to $\widetilde{\mathrm{H}}$ for $\mathsf{info}_{\mathrm{tfid}}$ and at most $Q_{\mathrm{S}_1}+1$ more queries to $\widetilde{\mathrm{H}}$ when checking the validity of the output. Therefore, $\mathcal{B}$ makes at most $Q_{\mathrm{H}}+Q_{\mathrm{S}_1}+1$ queries to H. Also, it is clear that $\mathcal{B}$ simulates oracles F, S_1, S_2, H in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game perfectly no matter what label is assigned to tfid. Therefore, the probability that $\mathrm{tFid}(\mathsf{info}^*) = \hat{i}^*$ and $\mathrm{WIN} \ \wedge \ E$ occurs when running $\mathcal{B}_{\mathrm{wfros}}$ is equal to $\frac{1}{Q_{\mathsf{info}}}\mathsf{Pr}[\mathrm{WIN} \ \wedge \ E]$.

It is left to show that if $\mathrm{tFid}(\mathsf{info}^*) = \hat{i}^*$ and $\mathrm{WIN} \ \wedge \ E$ occurs within the simulation, then $\mathcal{B}_{\mathrm{wfros}}$ wins the WFROS game. Suppose $\mathrm{WIN} \ \wedge \ E$ occurs and $\mathrm{tFid}(\mathsf{info}^*) = \hat{i}^*$. Following the similar analysis of $\mathcal{B}_{\mathrm{wfros}}$ in the proof of Lemma 6.2.1, we have $|\mathcal{J}| = Q_{\mathrm{S}_1}+1$.

Denote $\mathcal{I}_{\mathrm{fin}}^{\mathrm{tot}}$ and $\mathrm{sid}^{\mathrm{tot}}$ as the values of $\mathcal{I}_{\mathrm{fin}}$ and sid when $\mathcal{A}_{\mathrm{alg}}$ returns. Then, since E occurs, by (6.12) and (6.13), for any $j \in \mathcal{J}$ it holds that

$$\hat{\alpha}_j^{\mathsf{X}} - \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{\mathrm{tot}}} y_i \cdot c_i \cdot \hat{\alpha}_j^{\mathsf{A}_i} = -\delta_j \left(\hat{\alpha}_j^{\mathsf{Z}_{\hat{i}^*}} + \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{(\mathsf{info}^*)}} y_i \cdot \hat{\alpha}_j^{\mathsf{C}_i} \right). \tag{6.19}$$

Then, by (6.14), we have

$$\begin{aligned} -\delta_j \left(\hat{\alpha}_j^{\mathsf{Z}_{\hat{i}^*}} + \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{(\mathsf{info}^*)}} y_i \cdot \hat{\alpha}_j^{\mathsf{C}_i} \right) x &= \hat{\alpha}_j^{\mathsf{X}} - \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{\mathrm{tot}}} y_i \cdot c_i \cdot \hat{\alpha}_j^{\mathsf{A}_i} \\ &= \hat{\alpha}_j^{\mathsf{X}} - \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{\mathrm{tot}}} y_i \cdot c_i \cdot \hat{\alpha}_j^{\mathsf{A}_i} - \sum_{i\in[\mathrm{sid}^{\mathrm{tot}}]\setminus\mathcal{I}_{\mathrm{fin}}^{\mathrm{tot}}} y_i' \cdot \hat{\alpha}_j^{\mathsf{A}_i} \\ &= \hat{\alpha}_j^{\mathsf{X}} - \sum_{i\in[\mathrm{sid}^{\mathrm{tot}}],\mathrm{tFid}(\mathsf{info}_i)\neq\hat{i}^*} y_i' \cdot \hat{\alpha}_j^{\mathsf{A}_i} - \sum_{i\in\mathcal{I}_{\mathrm{fin}}^{(\mathsf{info}^*)}} y_i \cdot c_i \cdot \hat{\alpha}_j^{\mathsf{A}_i}. \end{aligned}$$

Then, from the simulation, by (6.17), we have for any $j \in \mathcal{J}$

$$\alpha_j^{(0)} + \sum_{i\in[Q_{\mathrm{S}_1}]} \tilde{y}_i(\alpha_j^{(2i-1)} + \tilde{c}_i \cdot \alpha_j^{(2i)}) = \delta_j \left(\beta_j^{(0)} + \sum_{i\in[Q_{\mathrm{S}_1}]} \tilde{y}_i(\beta_j^{(2i-1)} + \tilde{c}_i \cdot \beta_j^{(2i)}) \right).$$

Therefore, $\mathcal{B}_{\mathrm{wfros}}$ wins the $\mathrm{WFROS}_{Q_{\mathrm{S}_1},p}$ game.

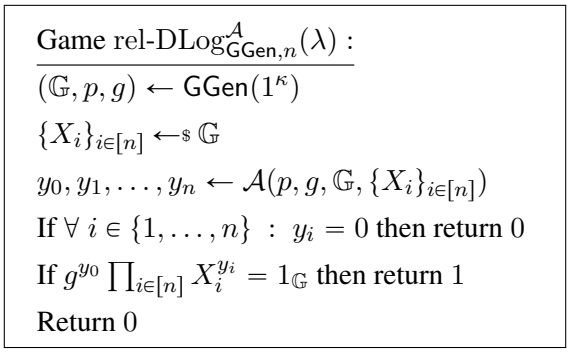

Figure 6.6: The rel-DLog game.

6.3.2 *Proof of Lemma 6.3.4*

We first partition the event WIN $\wedge$ $(\neg E)$ into two cases. Denote F_1 as the event in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game that there exists $k \in [Q_{\mathrm{S}_1} + 1]$ such that either (6.13) or (6.14) does not hold, and denote F_2 as the event that there exists $k \in [Q_{\mathrm{S}_1} + 1]$ such that (6.12) does not hold. Then, if E does not occur, we know either F_1 or F_2 occurs. Therefore, we have WIN $\wedge$ $(\neg E) =$ (WIN $\wedge$ F_1) $\vee$ (WIN $\wedge$ F_2). For the case that WIN $\wedge$ F_1 occurs, we show the following claim.

Claim 6.3.5. *There exists* $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ *for the* DLog *problem running in a similar running time as* $\mathcal{A}_{\mathrm{alg}}$ *such that*

$$\mathsf{Pr}[\mathrm{WIN} \ \wedge \ F_1] \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}^{(0)}_{\mathrm{dlog}}, \lambda) + \frac{1}{p-1}\ . \tag{6.20}$$

For the case that WIN $\wedge$ F_2 occurs, we construct an adversary $\mathcal{B}_{\text{rel-dlog}}$ for the $\text{rel-DLog}_{\mathbb{G},Q_{\mathrm{F}}}$ game (defined in 6.6) with advantage equals to the probability that WIN $\wedge$ F_2 occurs, where Q_{F} denotes the maximum number of queries to F issued in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game, and we summarize it into the following claim.

Claim 6.3.6. *There exists* $\mathcal{B}_{\text{rel-dlog}}$ *for the* rel-DLog *problem running in a similar running time as* $\mathcal{A}_{\mathrm{alg}}$ *such that*

$$\mathsf{Pr}[\mathrm{WIN} \ \wedge \ F_2] \leqslant \mathsf{Adv}^{\text{rel-dlog}}_{\mathsf{GGen},Q_{\mathrm{F}}}(\mathcal{B}_{\text{rel-dlog}}, \lambda)\ . \tag{6.21}$$

The rel-DLog problem is equivalent to the DLog problem, as shown in the following lemma from [84].

Lemma 6.3.7 (Lemma 3 in [84][3]). *Let* GGen *be a group generation algorithm. For any* $n > 0$ *and any adversary* $\mathcal{B}_{\text{rel-dlog}}$ *for the* $\text{rel-DLog}_{\mathsf{GGen},n}$ *game, there exists an adversary* $\mathcal{B}_{\text{dlog}}$ *for the* $\text{DLog}_{\mathsf{GGen}}$ *game such that*

$$\mathsf{Adv}^{\text{rel-dlog}}_{\mathsf{GGen},n}(\mathcal{B}_{\text{rel-dlog}}, \lambda) \leqslant \mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\text{dlog}}, \lambda) + 1/p\ .$$

By the lemma and Claim 6.3.6, there exists an adversary $\mathcal{B}^{(1)}_{\text{dlog}}$ for the $\text{DLog}_{\mathsf{GGen}}$ problem such that $\mathsf{Pr}[\text{WIN} \ \wedge \ F_1] \leqslant \mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}^{(1)}_{\text{dlog}}, \lambda) + \frac{1}{p}$. Therefore, together with Claim 6.3.5, we can construct an adversary $\mathcal{B}_{\text{dlog}}$ for the $\text{DLog}_{\mathsf{GGen}}$ problem that runs either $\mathcal{B}^{(0)}_{\text{dlog}}$ or $\mathcal{B}^{(1)}_{\text{dlog}}$ with $1/2$ probability, and we can conclude the lemma since

$$\begin{aligned}\mathsf{Pr}[\text{WIN} \ \wedge \ (\neg E)] &\leqslant \mathsf{Pr}[\text{WIN} \ \wedge \ F_1] + \mathsf{Pr}[\text{WIN} \ \wedge \ F_2] \\ &\leqslant \mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}^{(0)}_{\text{dlog}}, \lambda) + \mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}^{(1)}_{\text{dlog}}, \lambda) + \frac{2}{p-1} = 2\mathsf{Adv}^{\text{dlog}}_{\mathsf{GGen}}(\mathcal{B}_{\text{dlog}}, \lambda) + \frac{2}{p-1}.\end{aligned}$$

Proof of Claim 6.3.5. We first give a detailed description of $\mathcal{B}^{(0)}_{\text{dlog}}$ playing the $\text{DLog}_{\mathsf{GGen}}$ game.

THE ADVERSARY $\mathcal{B}^{(0)}_{\text{dlog}}$. To start with, $\mathcal{B}^{(0)}_{\text{dlog}}$ initializes sid, hid, fid, Hid, Fid, $\mathcal{I}_{\text{fin}}$, T_1, and T_2 as described in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game. After $\mathcal{B}^{(0)}_{\text{dlog}}$ receives $(p, g, \mathbb{G}, W)$ from the $\text{DLog}_{\mathsf{GGen}}$ game, sets $X \leftarrow W$. Then, $\mathcal{B}^{(0)}_{\text{dlog}}$ runs $\mathcal{A}_{\text{alg}}$ on input $(p, g, \mathbb{G}, X)$, and with access to the oracles $\widetilde{\text{F}}$, $\widetilde{\text{S}}_1$, $\widetilde{\text{S}}_2$, and $\widetilde{\text{H}}$. These oracles operate as follows:

Oracle $\widetilde{\text{F}}$: Same as in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game except instead of sampling $T_2(\mathsf{info})$ uniformly from $\mathbb{G}$, if $T_2(\mathsf{info}) = \perp$, $\mathcal{B}^{(0)}_{\text{dlog}}$ samples z uniformly from $\mathbb{Z}_p$ and sets $T_2(\mathsf{info}) \leftarrow g^{z_{\text{fid}}}$.

Oracle $\widetilde{\text{S}}_1$: After receiving a query info from $\mathcal{A}_{\text{alg}}$, $\mathcal{B}_{\text{wfros}}$ samples $s_{\text{sid}}, t'_{\text{sid}}$ uniformly from $\mathbb{Z}_p$ and samples y'_{sid} uniformly from $\mathbb{Z}^*_p$. Then, $\mathcal{B}_{\text{wfros}}$ sets $A_{\text{sid}} \leftarrow g^{s_{\text{sid}}} X^{-y'_{\text{sid}}}$ and $C_{\text{sid}} \leftarrow g^{t'_{\text{sid}}}$ and returns $(\text{sid}, A_{\text{sid}}, C_{\text{sid}})$.

Oracle $\widetilde{\text{S}}_2$: After receiving a query (i, c_i) to $\widetilde{\text{S}}_2$ from $\mathcal{A}_{\text{alg}}$, if $i \notin [\text{sid}] \backslash \mathcal{I}_{\text{fin}}$ or $c_i = 0$, $\mathcal{B}^{(0)}_{\text{dlog}}$ returns $\perp$. Otherwise, let $\hat{i} := \text{Fid}(\mathsf{info}_i)$, and $\mathcal{B}^{(0)}_{\text{dlog}}$ computes $y_i \leftarrow y'_i / c_i$ and $t_i \leftarrow t'_i - y_i \cdot z_{\hat{i}}$. Then, $\mathcal{B}^{(0)}_{\text{dlog}}$ returns (s_i, y_i, t_i).

[3]The DLog and rel-DLog games defined in [84] differ slightly from our descriptions, but the lemma follows by a similar proof.

Oracle $\widetilde{\mathbf{H}}$: Same as in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game.

After receiving the output $(\mathsf{info}^*, \{(m_k^*, \sigma_k^*)\}_{k\in[Q_{\mathrm{S}_1}+1]})$, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ aborts if the event $\mathrm{WIN} \ \wedge \ F_1$ does not occur.

It is clear that $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ simulates the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game perfectly, and thus it is left to show that if $\mathrm{WIN} \ \wedge \ F_1$ occurs, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ can compute the discrete log of W except for probability $1/p$.

Suppose $\mathrm{WIN} \ \wedge \ F_1$ occurs in the $\mathrm{OMUF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{PBS[GGen]}}$ game simulated by $\mathcal{B}^{(0)}_{\mathrm{dlog}}$. There exists $k \in [Q_{\mathrm{S}_1}+1]$ and $j = \mathrm{Hid}(\mathrm{str}_k^*)$ such that either (6.13) or (6.14) does not hold. Since $j = \mathrm{Hid}(\mathrm{str}_k^*)$ and $\delta_j = c_k^*$, we have

$$g^{s_k^*} X^{-\delta_j \cdot y_k^*} Z_{i^*}^{y_k^*} = g^{\hat{\alpha}_j^g} X^{\hat{\alpha}_j^{\mathsf{X}}} \prod_{i\in[\mathrm{fid}]} Z_i^{\hat{\alpha}_j^{\mathsf{Z}_i}} \prod_{i\in[\mathrm{sid}]} A_i^{\hat{\alpha}_j^{\mathsf{A}_i}} C_i^{\hat{\alpha}_j^{\mathsf{C}_i}} \ . \tag{6.22}$$

From the simulation of $\widetilde{\mathrm{S}}_1$, for each $i \in [\mathrm{sid}]$, we have

$$A_i = g^{s_i} X^{-y'_i} \ , \ C_i = g^{t'_i} \ .$$

Also, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ knows the discrete log of Z_i as z_i for each $i \in [\mathrm{fid}]$. By substituting $A_i = g^{s_i} X^{-y'_i}$, $C_i = g^{t'_i}$, and $Z_i = g^{z_i}$ into (6.22), we have

$$g^{s_k^* + y_k^* \cdot z_{i^*}} X^{-\delta_j \cdot y_k^*} = g^{\eta_j^g} X^{\eta_j^Z} \ ,$$

where

$$\eta_j^g := \hat{\alpha}_j^g + \sum_{i\in[\mathrm{fid}]} \hat{\alpha}_j^{\mathsf{Z}_i} \cdot z_i + \sum_{i\in[\mathrm{sid}]} (\hat{\alpha}_j^{\mathsf{A}_i} \cdot s_i + \hat{\alpha}_j^{\mathsf{C}_i} \cdot t'_i) \ ,$$

$$\eta_j^{\mathsf{X}} := \hat{\alpha}_j^{\mathsf{X}} - \sum_{i\in[\mathrm{sid}]} y'_i \cdot \hat{\alpha}_j^{\mathsf{A}_i} \ .$$

If $\eta_j^{\mathsf{X}} \neq -\delta_j \cdot y_k^*$, $\mathcal{B}^{(0)}_{\mathrm{dlog}}$ can compute the discrete log of X, which is also W, as

$$x := \frac{s_k^* + y_k^* \cdot z_{i^*} - \eta_j^g}{\eta_j^{\mathsf{X}} + \delta_j \cdot y_k^*} \ .$$

Therefore, it is left to bound the probability that $\eta_j^{\mathsf{X}} = -\delta_j \cdot y_k^*$, and there are the following two cases.

(6.14) does not hold for k, j. Consider the transcript π^{tot} that the adversary sees before it returns. Given the transcript π^{tot}, since for each $i \in [\mathrm{sid}] \backslash \mathcal{I}_{\mathrm{fin}}$, the adversary sees only A_i but does not know

either s_i or y'_i, the value y'_i is uniformly distributed over $\mathbb{Z}_p^*$ independent of all other $y'_{i'}$ for $i' \neq i$. Therefore, the probability that $\eta_j^{\mathsf{X}} = -\delta_j \cdot y_k^*$ is $\frac{1}{p-1}$.

(6.14) holds but (6.13) does not hold for k, j. Since (6.14) holds and for each $i \in \mathcal{I}_{\text{fin}}$ it holds that $y'_i = y_i \cdot c_i$, we have

$$\eta_{\hat{j}}^{\mathsf{X}} = \hat{\alpha}_{\hat{j}}^{\mathsf{X}} - \sum_{i \in \mathcal{I}_{\text{fin}}} y_i \cdot c_i \cdot \hat{\alpha}_{\hat{j}}^{\mathsf{A}_i} \ .$$

Then, since (6.13) does not hold, we have

$$\eta_{\hat{j}}^{\mathsf{X}} \neq -\delta_{\hat{j}} \cdot y_{\hat{k}}^* \ ,$$

which means the probability that $\eta_j^{\mathsf{X}} = -\delta_j \cdot y_k^*$ is 0. Therefore, for both cases, the probability that $\eta_j^{\mathsf{X}} = -\delta_j \cdot y_k^*$ is bounded by $\frac{1}{p-1}$. □

Proof of Claim 6.3.6. We first give a detailed description of $\mathcal{B}_{\text{rel-dlog}}$ playing the $\text{rel-DLog}_{\mathbb{G},Q_{\text{F}}}$ game.

THE ADVERSARY $\mathcal{B}_{\text{rel-dlog}}$. To start with, $\mathcal{B}_{\text{rel-dlog}}$ initializes sid, hid, fid, Hid, Fid, $\mathcal{I}_{\text{fin}}$, T_1, and T_2 as described in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game. Also, $\mathcal{B}_{\text{rel-dlog}}$ samples x uniformly from $\mathbb{Z}_p$ and sets $X \leftarrow g^x$. After $\mathcal{B}_{\text{rel-dlog}}$ receives $(p, g, \mathbb{G}, Z_1, \ldots, Z_{Q_{\text{F}}})$ from the $\text{rel-DLog}_{\mathbb{G},Q_{\text{F}}}$ game, $\mathcal{B}_{\text{rel-dlog}}$ runs $\mathcal{A}_{\text{alg}}$ on input $(p, g, \mathbb{G}, X)$ and with access to the oracles $\widetilde{\text{F}}$, $\widetilde{\text{S}}_1$, $\widetilde{\text{S}}_2$, and $\widetilde{\text{H}}$. These oracles operate as follows:

Oracle $\widetilde{\mathbf{F}}$: Same as in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game except instead of sampling $T_2(\mathsf{info})$ uniformly from $\mathbb{G}$, if $T_2(\mathsf{info}) = \perp$, $\mathcal{B}_{\text{rel-dlog}}$ sets $T_2(\mathsf{info}) \leftarrow Z_{\text{fid}}$.

Oracle $\widetilde{\mathbf{S}}_1$, $\widetilde{\mathbf{S}}_2$, $\widetilde{\mathbf{H}}$: The same as in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game.

After receiving the output $(\mathsf{info}^*, \{(m_k^*, \sigma_k^*)\}_{k \in [Q_{\text{S}_1}+1]})$, $\mathcal{B}_{\text{rel-dlog}}$ aborts if $\text{WIN} \ \wedge \ F_2$ does not occur.

It is clear that $\mathcal{B}_{\text{rel-dlog}}$ simulates the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game perfectly, and thus it is left to show that if $\text{WIN} \ \wedge \ F_2$ occurs, $\mathcal{B}_{\text{rel-dlog}}$ can win the $\text{rel-DLog}_{\mathbb{G},Q_{\text{F}}}$ game.

Suppose $\text{WIN} \ \wedge \ F_2$ occurs in the $\text{OMUF}^{\mathcal{A}_{\text{alg}}}_{\mathsf{PBS[GGen]}}$ game simulated by $\mathcal{B}_{\text{rel-dlog}}$. There exists $k \in [Q_{\text{S}_1} + 1]$ and $j = \text{Hid}(\text{str}_k^*)$ such that (6.12) does not hold. Since $j = \text{Hid}(\text{str}_k^*)$, we have

$$g^{s_k^*} X^{-\delta_j \cdot y_k^*} Z_{i^*}^{y_k^*} = g^{\hat{\alpha}_j^g} X^{\hat{\alpha}_j^{\mathsf{X}}} \prod_{i \in [\text{fid}]} Z_i^{\hat{\alpha}_j^{\mathsf{Z}_i}} \prod_{i \in [\text{sid}]} A_i^{\hat{\alpha}_j^{\mathsf{A}_i}} C_i^{\hat{\alpha}_j^{\mathsf{C}_i}} \ . \tag{6.23}$$

From the simulation of $\widetilde{\mathrm{S}}_1$, for each $i \in [\mathrm{sid}]$, we have

$$A_i = g^{a_i} \ , \ g^{t_i} = C_i Z_i^{-y_i} \ .$$

Also, $\mathcal{B}_{\text{rel-dlog}}$ knows the discrete log of X as x. By substituting $A_i = g^{a_i}$, $C_i = g^{t_i} Z_i^{y_i}$, and $X = g^x$ into (6.23), we have

$$g^{s_k^*} X^{-\delta_j \cdot y_k^*} Z_{i^*}^{y_k^*} = g^{\hat{\alpha}_j^g + \hat{\alpha}_j^{\mathsf{X}} \cdot x + \sum_{i \in [\mathrm{sid}]} (\hat{\alpha}_j^{\mathsf{A}_i} \cdot a_i + \hat{\alpha}_j^{\mathsf{C}_i} \cdot t_i)} \prod_{i \in [\mathrm{fid}]} Z_i^{\hat{\alpha}_j^{\mathsf{Z}_i} + \sum_{i' \in [\mathrm{sid}], \mathrm{tSid}(i') = i} y_{i'} \cdot \hat{\alpha}^{\mathsf{C}_{i'}}} \ .$$

Therefore, $\mathcal{B}_{\text{rel-dlog}}$ can compute $(w_0, \ldots, w_{Q_{\mathrm{F}}})$ such that $g^{w_0} \prod_{i \in [Q_{\mathrm{F}}]} W_i^{w_i} = g^{w_0} \prod_{i \in [\mathrm{fid}]} Z_i^{w_i} = 1_{\mathbb{G}_\lambda}$ as

$$w_i := \begin{cases} \hat{\alpha}_j^g + \hat{\alpha}_j^{\mathsf{X}} \cdot x + \sum_{i \in [\mathrm{sid}]} (\hat{\alpha}_j^{\mathsf{A}_i} \cdot a_i + \hat{\alpha}_j^{\mathsf{C}_i} \cdot t_i) - s_k^* + x \cdot \delta_j \cdot y_k^* \ , & i = 0 \\ \hat{\alpha}_j^{\mathsf{Z}_i} + \sum_{i' \in [\mathrm{sid}], \mathrm{tSid}(i') = i} y_{i'} \cdot \hat{\alpha}^{\mathsf{C}_{i'}} \ , & i \in [\mathrm{fid}], i \neq i^* \\ -y_k^* + \hat{\alpha}_j^{\mathsf{Z}_i} + \sum_{i' \in [\mathrm{sid}], \mathrm{tSid}(i') = i} y_{i'} \cdot \hat{\alpha}_j^{\mathsf{C}_{i'}} \ , & i = i^* \\ 0 \ , & o.w. \end{cases}$$

Since (6.12) does not hold, we have

$$w_{i^*} = -y_k^* + \hat{\alpha}_j^{\mathsf{Z}_{i^*}} + \sum_{i \in \mathcal{I}_{\mathrm{fin}}^{(\mathsf{info}^*)}} y_{i'} \cdot \hat{\alpha}^{\mathsf{C}_{i'}} \neq 0 \ .$$

Therefore, $\mathcal{B}_{\text{rel-dlog}}$ wins the $\text{rel-DLog}_{\mathbb{G}, Q_{\mathrm{F}}}$ game by outputting $(w_0, \ldots, w_{Q_{\mathrm{F}}})$ defined above. □

Part II

THRESHOLD SIGNATURES IN PAIRING-FREE GROUPS

Chapter 7

THRESHOLD SIGNATURES: INTRODUCTION

Threshold signatures, which originated in the late 1980s [58, 59], are seeing renewed attention, driven in particular by an interest in using them to secure digital wallets in the cryptocurrencies ecosystem [70]. Parallel IETF [92] and NIST [102] standardization efforts are evidence as to the speed at which the area is moving into practice.

Whether securing a user's digital wallet, or being used by a CA to create a certificate, forgery of a digital signature is costly. The rising tide of system breaches and phishing attacks makes exposure of a signing key too likely to ignore. The idea of a threshold signature scheme is to distribute the secret signing key across multiple parties who then interact to produce a signature, the intent being to retain security even in the face of compromise of up to a threshold number of these parties. Over the years, threshold versions of many schemes have been presented, including RSA [53, 74, 119], DSA/ECDSA [71, 73, 70, 28, 69, 96, 38], Schnorr signatures [120, 72, 91] and BLS signatures [27].

In this thesis, we focus on FROST [91], the state-of-the-art construction of threshold Schnorr signatures, and its variants, referred to as FROST2 [50] and FROST3 [113]. (We also refer to the original version of FROST as FROST1.) FROST is a pairing-free scheme that is *partially* non-interactive, meaning signing additionally involves a message-independent pre-processing round.

Our first main contribution is to show that FROST and its variants satisfy stronger security definitions than those previously considered in the literature or proven for these schemes. These refined definitions capture natural and practically relevant strengths that may be desirable in real-world applications. Moreover, our fine-grained perspective reveals meaningful differences between existing schemes. In particular, we demonstrate that FROST2 is strictly less secure than FROST1, and FROST3 is strictly less secure than FROST2. We elaborate on this in Section 7.1.

Our second main contribution is a new approach that can transform FROST into threshold signature schemes whose security relies on weaker assumptions. Specifically, while all three FROST variants rely on the Algebraic One-More Discrete Logarithm (AOMDL) assumption [105] in the Random Oracle Model (ROM), our new schemes are provably secure under the plain DL assump-

tion in the ROM. The resulting schemes produce Okamoto signatures [107], which are just one scalar larger than Schnorr signatures. Further details are provided in Section 7.2.

7.1 Stronger Security for Non-Interactive Threshold Signatures

OUR PATH. The classical development paradigm in theoretical cryptography is to ask what security we would like, define it, and then seek schemes that meet it. Yet if we look back, there has been another path alongside; canonical, reference schemes guided a choice of definitions that model them, and, once made, these definitions went on to be influential targets for future schemes. (The formal definition of trapdoor permutations [78], for example, was crafted to model RSA.) We are inspired by the latter path. Within the space of Schnorr threshold schemes, FROST appears to be the simplest candidate. Examining it, we see strengths not captured by current definitions or results. We step back to create corresponding abstractions, including a syntax and a hierarchy of definitions of security for non-interactive threshold signature schemes. We then return to ask where, in this hierarchy, we can fit the starting schemes, giving proofs that FROST and its variants as high as possible. In terms of the proofs this needs, and that we give, this turns out to be challenging, so that we offer also some content of technical interest.

Although inspired by specific schemes, our definitional development, once started, unfolds in a logical way, and yields definitions that even go beyond what FROST achieves. These make intriguing new targets. We show how to achieve them, with minimal modifications to the existing schemes.

NON-INTERACTIVE THRESHOLD SCHEMES. We consider schemes where the signing operations involve a leader and a set of n nodes, which we refer to as servers, with server i holding a secret share sk_i of the secret signing key sk. Signing is done via an interactive protocol that begins with a leader request to some set of at least t number of servers and culminates with the leader holding the signature, where $t \leqslant n$, the threshold, is a protocol parameter.

In a *fully non-interactive* threshold signature scheme, this protocol is a simple, one-round one. The leader sends a leader request lr, which specifies a message M and possibly other things, to any server i, and obtains in response a *partial signature*, $psig_i$, that i computes as a function of sk_i and M. The leader can request partial signatures asynchronously, at any time, and independently for each server, and there is no server-to-server communication. Once it has enough partial signatures, the leader aggregates them into a signature sig of M under the verification key pk corresponding to

sk. The canonical example is the threshold BLS scheme [27, 32], where $\mathsf{sk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n \in \mathbb{Z}_p$ for a public prime p, and $psig_i \leftarrow \mathrm{H}(\mu)^{\mathsf{sk}_i}$ where $\mathrm{H} : \{0,1\}^* \to \mathbb{G}$ is a public hash function with range a group $\mathbb{G}$ of order p. Aggregation produces sig as a weighted product of the partial signatures.

A *partially non-interactive* threshold signature scheme adds to the above a message-independent pre-processing round in which, pinged by the leader at any point, a server i returns a pre-processing token pp_i. The leader's request for a partial signatures will now depend on tokens it has received. The canonical example is FROST [91].

This understanding of a non-interactive scheme encompasses what FROST calls flexibility; obtaining $psig_i$ from *any* $\geqslant t$ servers will allow us to reconstruct a signature.

WHICH FORGERIES ARE NON-TRIVIAL? For a regular (non-threshold) signature scheme, the first and most basic notion of security is un-forgeability (UF) [78]. The adversary (given access to a signing oracle) outputs a forgery consisting of a message μ and a valid signature for it. To win, the forgery must be *non-trivial*, meaning not legitimately obtained. This is naturally captured, in this context, as meaning that μ was not a signing query.

Turning to define un-forgeability for a non-interactive threshold signature scheme, we assume the adversary has corrupted the leader, and up to $t-1$ servers, where $1 \leqslant t \leqslant n$ is the threshold. Furthermore it has access to the honest servers. Again, it outputs a forgery consisting of a message μ and valid signature for it, and, to win, the forgery must be *non-trivial*, meaning not legitimately obtained. Deciding what "non-trivial" means, however, is now a good deal more delicate, and interesting, than it was for regular signatures.

In this regard, we suggest that many of the prior works have set a low bar, being more generous than necessary in declaring a forgery trivial, leading to definitions that are weaker than one can desire, and weaker even than what their own schemes seem to meet. The definitions we formulate rectify this by giving a hierarchy of five non-triviality conditions of increasing stringency, yielding a corresponding hierarchy TS-UF-0 $\leftarrow$ TS-UF-1 $\leftarrow$ TS-UF-2 $\leftarrow$ TS-UF-3 $\leftarrow$ TS-UF-4 of five notions of un-forgeability of increasing strength. (Here an arrow B $\leftarrow$ A means A implies B: any scheme that is A-secure is also B-secure.) TS-UF-0, the lowest in the hierarchy, is the notion used in [74, 71, 73, 27]. TS-UF-1 was introduced by Shoup [119].

Returning to regular (non-threshold) signature schemes, strong un-forgeability (SUF) has the same template as UF, but makes the non-triviality condition more stringent, asking that there have been no signing query μ that returned sig. We ask if SUF has any analogue in the threshold setting. For non-interactive schemes, we suggest it does and give a hierarchy of three definitions

of strong unforgeability TS-SUF-2 $\leftarrow$ TS-SUF-3 $\leftarrow$ TS-SUF-4. The numbering reflects that TS-UF-i $\leftarrow$ TS-SUF-i for $i = 2, 3, 4$.

TS-UF-0 AND TS-UF-1. For the basic security notion TS-UF-0, the non-triviality condition is that *no* server was asked to issue a partial signature on the forgery message μ. However, allowing asynchronous requests is a feature of non-interactive schemes. A corrupted leader could ask one honest server i for a partial signature. No other server would even be aware of this request, but the adversary would now have $psig_i$. Under TS-UF-0, the forgery is now trivial, and the adversary does not win. Yet (assuming a threshold $t \geqslant 2$), there is no reason possession of just $psig_i$ should allow creation of a signature, and indeed for FROST there is no attack that seems able to create such a signature, indicating the scheme is achieving more than TS-UF-0. This leads to the next level of the hierarchy, TS-UF-1, where, following [119], the non-triviality condition is that a partial signature of μ was requested from at most $t - 1 - c$ honest servers, where c is the number of corrupted servers.

The distinction between TS-UF-1 and TS-UF-0 is not just academic. Implicit in applications of threshold signing in wallets is the fact that servers also perform well-formedness checks of what is being signed (typically, as part of a transaction). TS-UF-1 guarantees that every issued signature has been inspected by sufficiently many servers, but TS-UF-0 does not.

BEYOND TS-UF-1. Yet the hierarchy needs to go higher, and this becomes apparent when looking at partially non-interactive schemes like FROST [91]. Here, the discussion becomes more subtle, and interesting.

In more detail, a FROST pre-processing token takes the form of a pair $pp_i = (g^{r_i}, g^{s_i})$ of group elements for one-time use. (A server will ensure that the pre-processing token in its name in the leader request is one it has previously sent, and will never use it again.) An honest request lr includes, along with the message μ to be signed, a sufficiently large server set $lr.\mathsf{SS} \subseteq [1..n]$, and, for each i in this set, a pre-processing token pp_i that i previously sent. Each server $i \in lr.\mathsf{SS}$ will then generate a signature share $psig_i = (R, z_i)$, where R is a value which can be computed (publicly) from the tokens included in lr, whereas z_i depends on the discrete logarithms of the server's token and its own key share sk_i. The z_i's can then be aggregated into a value z such that (R, z) is a valid Schnorr signature for μ. Three variants of FROST are known, and differ in the way in which R is computed from lr. The original version, which we refer to as FROST1 [91], requires $|lr.\mathsf{SS}|$ exponentiations, whereas a more recent optimization, which we call FROST2 [50], only requires a single exponentiation. A further optimized version, FROST3 [113], improves the communica-

tion complexity by only sending an aggregated nonce. The current security analyses [50, 113] do not surface any security difference between the three variants, but only because they consider a non-triviality notion as in our TS-UF-0 notion.

In terms of our framework, we show that FROST1 achieves TS-SUF-3 security. The proof is under the AOMDL assumption of [105] in the ROM of [18]. This considers a signature trivial even if some of the honest servers in lr.SS do not respond to a (malicious) leader request, as long as the tokens associated with these servers are not honestly generated. In particular, the honest servers may not respond because they recognize these tokens as invalid, or because the malicious leader did not submit the request to them. We show that, while FROST2 fails to achieve TS-SUF-3, it achieves the next step down in our hierarchy, TS-SUF-2. (Again the proof is under AOMDL in the ROM.) This is still stronger than the notions lower in the hierarchy. Finally, we show FROST3 achieves TS-UF-1 under AOMDL in ROM but fails to achieve TS-UF-2.

STRONGER GOALS. A stronger security goal (TS-UF-4 in our hierarchy) is to expect that the *only* way to obtain a signature for a message μ is to follow the above blueprint, i.e., to issue the same honest leader request lr to all servers in lr.SS. In fact, we may even ask for more, in terms of *strong* unforgeability — the value R is uniquely defined by lr, and, along with the message μ, it defines a *unique* signature (although not efficiently computable given the verification key alone). An ideal goal, which corresponds to our strongest security goal, is to ensure that the *only* way to generate the signature associated with lr is to obtain a signature share for lr from every honest server whose tokens are included in lr. This is a notion we refer to as TS-SUF-4.

We will however show that neither FROST1 nor FROST2 meet TS-SUF-4. To overcome this, we will show a general transformation which can boost the security of a TS-SUF-3-secure scheme like FROST1 to achieve TS-SUF-4. Our framework allows schemes more general than the FROST ones, and also leaves the question open of better and more efficient designs achieving the stronger notions. Moreover, we provide simple reference schemes for all of our notions, which, while inefficient, guide us in understanding the subtle differences among notions and baseline requirements. In particular, these schemes will enable us to separate the proposed notions.

A SUMMARY FOR OUR NOTIONS. In summary, our unforgeabilty notions declare a signature for a message μ trivial in the following cases:

- TS-UF-0: A partial signature for the message μ was generated by at least one honest server.

- TS-UF-1: A partial signature for the message μ was generated by at least $t-c$ honest servers,

where c is the number of corrupted servers.

- TS-UF-2: There exists a leader request lr for the message μ which was answered by at least $t - c$ honest servers.

- TS-UF-3: There exists a leader request lr for the message μ such that every honest server $i \in lr.\mathsf{SS}$ either answered lr or the token pp_i associated with i in lr is maliciously generated.

- TS-UF-4: There exists a leader request lr for the message μ such that every honest server $i \in lr.\mathsf{SS}$ answered lr.

Analogous notions of strong unforgeability are obtained by further associating a request lr to a (unique) signature, in addition to a message μ.

We stress that it is not clear which scenarios demand which notions in our hierarchy. This is especially true because we are still lacking formal analyses of full-fledged systems using threshold signatures, but it is not hard to envision a potential mismatch between natural expectations from such schemes and what they actually achieve. In both FROST variants, for example, it is natural to expect that a signature can only be generated by a sufficient number of honest servers answering the *same* request, a property which we show is actually achieved. Further, one may also expect that all honest servers that generated these honest tokens need to be involved in the generation of a valid signature, but this stronger property is actually not achieved by either of the FROST variants.

WHAT WE DO NOT DO. Some schemes like FROST come with a concrete *distributed key-generation* (DKG) protocol. Security proofs frequently (but not always) consider, monolithically, the composition of DKG and threshold signing. This lack of modular treatment is due to the fact that efficient DKG protocols like Pedersen's [109] are not secure [73] in the strongest possible sense *in isolation*, but it may still be possible to show security when they are used with a particular threshold signature scheme. Here, instead, we idealize DKG protocols, as the points we are trying to express are orthogonal to the concrete choice of a DKG. Our result would still guarantee security of the schemes when used with truly secure DKGs (such as the DKG from [73]), but further investigation is needed to extend our proofs to consider more efficient DKGs.

Our framework does not handle adaptive corruptions, i.e., we demand instead that the adversary declares its corruption set initially. We could extend our definitions to adaptive corruptions rather

easily, but our concrete bounds would be impacted. In particular, we would resort to a generic reduction guessing the corrupted set beforehand, with a multiplicative loss of 2^n, which is acceptable for the smaller values of the number n of parties that we consider common in practice.

Our framework cannot cover recent protocols, like that of Canetti et al. [38], which combine a *multi-round* message-independent pre-processing phase with a final, message-dependent, round. (Conversely, their UC security analysis does not give definitions which help our fine-grained framework.)

Finally, many prior works [73, 113, 23] also consider *robustness*, i.e., the guarantee that a signature is always produced. Here, we follow the same viewpoint as in FROST, and do not focus on robustness explicitly. This allows us to prevent imposing a small t (relative to n) just for the sake of ensuring it. However, our schemes all implicitly give verification keys pk_i for each server, and it is not hard to verify individual partial signatures $psig_i$. Any t valid partial signatures will always aggregate into a valid signature.

7.2 Partially Non-Interactive Threshold Signatures from Weaker Assumptions

As our second main contribution, we develop new two-round protocols that are secure under the discrete logarithm assumption in the ROM. The signatures produced by both schemes resemble those proposed by Okamoto [107]. Furthermore, our signing protocols are partially non-interactive, i.e., the first round messages do not depend on the message being signed, which is a desirable property in practice.

SIGNIFICANCE. Our DL-based schemes are the first partially non-interactive 2-round schemes based solely on the hardness of the DL assumption. For threshold signatures, in particular, no two-round scheme is known from only the DL assumption.

OUR APPROACH. Our schemes are the outcome of the same paradigm applied to FROST [91, 14] and its variants. It is not known how to prove the security of either scheme under the plain discrete logarithm assumption, and they are instead proved secure under the (stronger) *AOMDL [105]* assumption, which is a variant of the OMDL assumption [16], an assumption that has been the subject of criticism [90, 89]. As we explain next, our paradigm can be seen as a general recipe to remove the AOMDL assumption from these schemes.

The main ingredient of our approach are *linear hash functions*, which have also been used in recent works [10, 80, 81] to abstract identification schemes from which signature variants are de-

rived. Here, we observe that FROST can naturally be generalized by replacing the exponentiation map $x \mapsto g^x$ with a linear hash function $\mathsf{F} : \mathcal{D} \to \mathcal{R}$, where $\mathcal{D}, \mathcal{R}$ are $\mathcal{S}$-modules for a field $\mathcal{S}$. We generically refer to these instantiations as FROST-H. (In fact, we present two variants for FROST-H but make no distinction in the introduction.) In particular, we require that:

- F is an epimorphism of $\mathcal{S}$-modules from $\mathcal{D}$ to $\mathcal{R}$, i.e., F is a surjection from $\mathcal{D}$ to $\mathcal{R}$ such that for any $r \in \mathcal{S}$ and $x, y \in \mathcal{D}$, $\mathsf{F}(x + r \cdot y) = \mathsf{F}(x) + r \cdot \mathsf{F}(y)$.

- F is *not a monomorphism*, which is equivalent to postulating that there exists $z^* \in \mathcal{D}$ such that $z^* \neq 0$ and $\mathsf{F}(z^*) = 0$.

We then define a natural analogue of the AOMDL assumption, which we refer to as the *Algebraic One-More Preimage Resistance* (AOMPR). Roughly speaking, the corresponding security game allows the attacker to obtain multiple *challenges* $X_i = \mathsf{F}(x_i)$ for a random element $x_i \leftarrow\!\!\$\, \mathcal{D}$, and the attacker also gets access to an *inversion oracle* which, on input $X \in \mathcal{R}$, returns a element in the preimage set of X under F. The restriction here, and hence the term *algebraic*, is that X must be an affine combination of previously obtained X_i's, and this affine combination is given to the inversion oracle, along with X. (This makes the assumption falsifiable since the oracle can efficiently answer such inversion queries.) To win the game, the attacker is then asked to invert $q + 1$ challenges after querying the inversion oracle at most q times.

Our results then follow from the combination of the following two theorems, which we state here informally:

> **Theorem (informal).** The security of FROST-H follows from the AOMPR assumption on the underlying linear hash function.

> **Theorem (informal).** If F is collision-resistant, then the AOMPR assumption holds with respect to F.

The proof of the first theorem is, on its own, not particularly surprising and mostly generalizes the prior proofs in the literature, in particular those of [105] and [14]. Our main contribution here is to notice that these proofs, and the resulting schemes, can be abstracted in terms of linear hash functions. In particular, for threshold signatures, as in [14], we consider an abstract setting with an ideal distributed key generation, and we target the security notions of TS-SUF-2 and TS-SUF-3, which

were shown to be achieved by two variants of FROST, both of which we model here abstractly. Since we are targeting feasibility, we are less concerned with the concrete round complexity of distributed key generation and could use any secure multi-party computation protocol for this task.

In contrast, the rough intuition behind a proof of the latter theorem is that for any execution of a (wlog deterministic) adversary $\mathcal{A}$ playing the AOMPR game with challenges $\boldsymbol{X} = \mathsf{F}(\boldsymbol{x})$, since F is not a monomorphism, there exists another execution with challenges $\boldsymbol{X} = \mathsf{F}(\boldsymbol{x}')$ such that $\boldsymbol{x} \neq \boldsymbol{x}'$, but the views of $\mathcal{A}$ are identical in the two executions. Then, if $\mathcal{A}$ wins the game given $\boldsymbol{x}$ by outputting $\boldsymbol{y}$ such that $\mathsf{F}(\boldsymbol{y}) = \boldsymbol{X}$, $\mathcal{A}$ also wins the game given $\boldsymbol{x}'$ by outputting $\boldsymbol{y}$. Therefore, we have $\mathsf{F}(\boldsymbol{x}) = \mathsf{F}(\boldsymbol{y}) = \mathsf{F}(\boldsymbol{x}') \ \wedge \ (\boldsymbol{x} \neq \boldsymbol{y} \ \vee \ \boldsymbol{x}' \neq \boldsymbol{y})$, which implies that we can find a collision in at least one of the executions. Indeed, special cases of this technique already underlie several works, including Okamoto's [107], but our main challenges are to prove the concrete mapping of $\boldsymbol{x}'$ from $\boldsymbol{x}$ and to package this in terms of the AOMPR abstraction.

7.2.1 DL-based Instantiations

To obtain an instantiation of FROST-H based on the hardness of the discrete logarithm (DL) problem, we can use the Pedersen linear hash function [109]

$$\mathsf{F}(x_1, x_2) = g^{x_1} Z^{x_2} \ ,$$

which is well known to be collision-resistant under the hardness of DL whenever g, Z are generators of a group with prime size p. While FROST produces valid Schnorr signatures [114], the signatures produced by our DL-based instantiations of FROST-H are slightly less efficient, and effectively compatible with Okamoto's signatures [107]. Here, as in Schnorr signatures, the secret signing key is $x \in \mathbb{Z}_p$, and the public verification key $\mathsf{pk} = g^x$, and a signature for a message $m \in \{0,1\}^*$ has format

$$\sigma = (R = g^a Z^b, a + \mathrm{H}(\mathsf{pk}, m, R) \cdot x, b) \ ,$$

where H is a hash function that is modeled as a random oracle in our proofs. To verify a signature (R, a, b), we check that $g^a Z^b = R \cdot \mathsf{pk}^{\mathrm{H}(\mathsf{pk}, M, R)}$. The only difference from Okamoto's scheme [107] is that the latter uses a secret key $(x_1, x_2) \in \mathbb{Z}_p^2$, and a signature has form $(R = g^a Z^b, a + c \cdot x_1, b + c \cdot x_2)$, where $c = \mathrm{H}(\mathsf{pk}, m, R)$, i.e., here, we restrict the scheme to the case where $(x_1, x_2) = (x, 0)$. This optimization is generic and could have been applied to Okamoto's scheme directly; however, it is particularly advantageous for threshold signatures since it lets us leverage any distributed

key generation protocol for Schnorr signatures. Here, we need a trusted setup to generate Z as a random group element independent of g, but we note that this is a minimal setup since it can be made transparent, e.g., g, Z can be generated as outputs of a hash function.

RELATED WORK. Our DL-based threshold signatures are the first two-round scheme with security proved based solely on the discrete logarithm assumption in the ROM. The most efficient protocol is FROST [91, 14], which is slightly more efficient than our scheme since it generates plain Schnorr signatures; however, FROST relies on the stronger AOMDL assumption. Though schemes based solely on the discrete logarithm assumption exist [120, 73, 95], they use more rounds. We stress that not all schemes achieve the same security goals, and here we target our security notions, whereas Lindell [95] targets UC security.

Chapter 8

A FRAMEWORK FOR NON-INTERACTIVE THRESHOLD SIGNATURES

We present our hierarchy of definitions of security for non-interactive threshold schemes, formalizing both unforgeability (UF) and strong unforgeability (SUF) in several ways. We provide relations between all notions considered.

8.1 Syntax and Correctness

MAINTAINING STATE. Parties as implemented in protocols would maintain state. When activated with some inputs (which include messages from other parties), they would apply some algorithm Alg to these and their current state to get outputs (including outgoing messages) and an updated state. To model this, we do not change our definition of algorithms, but make the state an explicit input and output that will, in definitions, be maintained by the overlying game. Thus, we would write something like $(\cdots, \mathsf{st}) \leftarrow\!\!\$\ \mathsf{Alg}(\cdots, \mathsf{st})$.

SYNTAX. A non-interactive threshold signature scheme TS specifies a number $n \geqslant 1$ of signers, a reconstruction threshold t, a set HF of functions from which the random oracle is drawn, a key-generation algorithm KeyGen, a server pre-processing algorithm SPP, a leader pre-processing algorithm LPP, a leader signing-request algorithm LR, a server partial-signature algorithm PS, a leader partial-signature aggregation algorithm Agg and a verification algorithm Vf. If disambiguation is needed, we write $\mathsf{TS}.n, \mathsf{TS}.t, \mathsf{TS.HF}, \mathsf{TS.Setup}, \mathsf{TS.KeyGen}, \mathsf{TS.SPP}, \mathsf{TS.LPP}, \mathsf{TS.LR}, \mathsf{TS.PS}, \mathsf{TS.Agg}, \mathsf{TS.Vf}$, respectively. We now explain the operation and use of these components, the understanding of which may be aided by already looking at the correctness game $\text{TS-COR}_{\mathsf{TS}}$ of Figure 8.1.

Parties involved are a leader (numbered 0, implicit in some prior works, but made explicit here) and signers numbered $1, \ldots, n$, for a total of $n + 1$ parties. The setup algorithm $\mathsf{Setup}(1^\kappa)$ initializes the state st_i for each signer $i \in [n]$, as well as the state st_0 for the leader, and additionally a public parameter par, which is given as input to all other algorithms. Algorithms have oracle

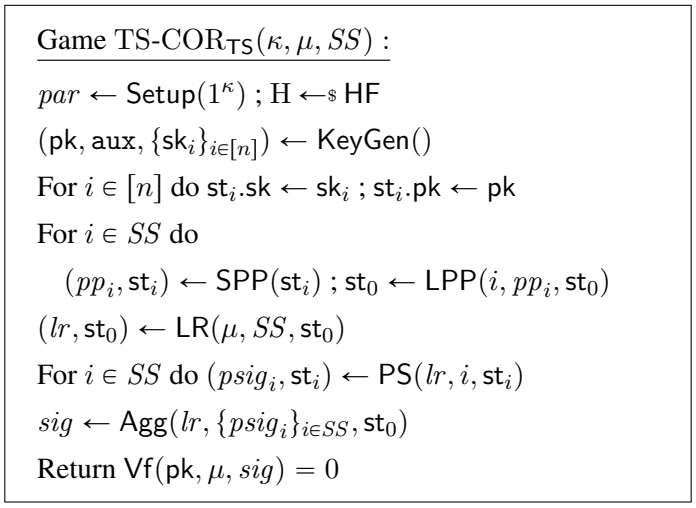

Figure 8.1: The TS-COR game for a threshold signature scheme TS for n signers and threshold t.

---

access to a function H that is drawn at random from HF in games and plays the role of the random oracle. Specifying HF as part of the scheme allows the domain and range of the random oracle to be scheme-dependent. We omit par and H from the input of all algorithms for simplicity.

The key-generation algorithm KeyGen, run once at the beginning, creates a public signature-verification key pk, associated public auxiliary information aux, and an individual secret signing key sk_i for each server $i \in [1..n]$. (Usually, $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ will be shares of a global secret key sk, but the definitions do not need to make sk explicit. The leader does not hold any secrets associated to pk.) While key-generation may in practice be performed by a distributed key-generation protocol, our syntax assumes it done by a trusted algorithm to allow a modular treatment. Keys are held by parties in their state, encoded into dedicated fields of the latter as shown in Figure 8.1. For specific schemes, we will typically use aux to model additional information that can be leaked by key generation step without violating security (e.g., the values g^{sk_i} in most cases).

The signing protocol can be seen as having two rounds, which we think as a pre-processing and online stage. In a pre-processing round, any server i can run $(pp, \mathsf{st}_i) \leftarrow\!\!\$\ \mathsf{SPP}(\mathsf{st}_i)$ to get a *pre-processing token* pp which it sends to the leader. (Here st_i is the state of i.) Via $\mathsf{st}_0 \leftarrow \mathsf{LPP}(pp, \mathsf{st}_0)$, the leader updates its state st_0 to incorporate token pp.

In a signing round, the leader begins with a message and a choice of a signer set $SS \subseteq [1..n]$

of size at least t. Via $(lr, \mathsf{st}_0) \leftarrow\!\!\$\ \mathsf{LR}(M, SS, \mathsf{st}_0)$ it generates a leader request lr that, through st_0, implicitly depends on a choice of pre-processing tokens. The leader request is sent to each $i \in SS$, who, via $(psig_i, \mathsf{st}_i) \leftarrow\!\!\$\ \mathsf{PS}(lr, i, \mathsf{st}_i)$, computes a partial signature $psig_i$ and returns it to the leader. Via $(sig, \mathsf{st}_0) \leftarrow\!\!\$\ \mathsf{Agg}(lr, \{psig_i\}_{i \in SS}, \mathsf{st}_0)$, the leader aggregates the partial signatures into a signature sig of M, the desired output of the protocol. Also, we require that for $i \notin SS$, $\mathsf{PS}(lr, i, \mathsf{st}_i)$ always outputs $(\perp, \mathsf{st}_i)$.

The verification algorithm, like in a standard signature scheme, takes pk, a message μ and a candidate signature, and returns a boolean validity decision.

ECHO SCHEMES. We define a sub-class of non-interactive threshold schemes that we call *echo schemes*. Recall that a leader request lr is mandated to specify a message $lr.\mathsf{msg}$ and a set $lr.\mathsf{SS} \subseteq [1..n]$ of signers from whom partial signatures are being requested. In an echo scheme, lr additionally specifies a function $lr.\mathsf{PP} : lr.\mathsf{SS} \to \{0,1\}^*$. If the leader is honest, $lr.\mathsf{PP}(i)$ is a token pp that i had previously sent to the leader. That is, the leader is echoing tokens back to the signers, whence the name. In considering security, of course, $lr.\mathsf{PP}(i)$ is picked by the adversary and may not be a prior token. For an echo scheme, we impose the additional requirement that the state st_i of party i includes a set $\mathsf{st}_i.\mathsf{PP}$ in which SPP stores prior tokens: more formally, the execution $(pp, \mathsf{st}_i) \leftarrow \mathsf{SPP}(\mathsf{st}_i)$ is required to include the update $\mathsf{st}_i.\mathsf{PP} \leftarrow \mathsf{st}_i.\mathsf{PP} \cup \{pp\}$. In this way, st_i is recording the tokens that party i has previously sent to the leader. Finally, we require that $\mathsf{PS}(lr, i, \mathsf{st}_i)$ always outputs $(\perp, \mathsf{st}_i)$ if $lr.\mathsf{PP}(i) \notin \mathsf{st}_i.\mathsf{PP}$. This is a natural requirement since an honest signer should not answer a leader request that is clearly malicious. As we will discuss in Section 9.1, FROST1 and FROST2 are examples of an echo scheme.

CORRECTNESS OF A TS SCHEME. We say that TS is *correct* with correctness error $\varepsilon_{\mathsf{cor}}$ if for $\mu \in \{0,1\}^*$ and $SS \subseteq [n]$ with $|SS| \geqslant t$, we have $\mathsf{Pr}[\text{TS-COR}_{\mathsf{TS}}(\kappa, \mu, SS) = 1] \leqslant \varepsilon_{\mathsf{cor}}$. We say that TS is *perfectly correct* if $\varepsilon_{\mathsf{cor}} = 0$. By default, when we say that TS is correct, we mean it is perfectly correct.

The way in which we are supposed to interpret the correctness definition is that a request lr is associated with a set SS and a message μ, and if such a request is issued successfully by the leader (i.e., $lr \neq \perp$), then the signers in SS would all accept lr producing partial signatures which aggregate into a valid signature for μ.

Game TS-UF-$i^{\mathcal{A}}_{\mathsf{TS}}(\kappa)$ $(i = 0, 1, 2, 3, 4)$ and TS-SUF-$i^{\mathcal{A}}_{\mathsf{TS}}(\kappa)$ $(i = 2, 3, 4)$:

$par \leftarrow \mathsf{Setup}(1^\kappa)$; $\mathrm{H} \leftarrow\!\!\$\ \mathsf{TS.HF}$; $\mathrm{TF}_\mathsf{M} \leftarrow \varnothing$; $\mathrm{TF}_\mathsf{L} \leftarrow \varnothing$; $\mathrm{curSS}_\mathsf{M}, \mathrm{curSS}_\mathsf{L} \leftarrow ()$

$(\mu, sig) \leftarrow \mathcal{A}^{\text{INIT,PPO,PSIGNO,RO}}(par)$

If $\mathsf{Vf}(\mathsf{pk}, \mu, sig) = 0$ then return 0

For each lr do

  If $\mathtt{tf}_i(lr.\mathsf{msg})$ then $\mathrm{TF}_\mathsf{M} \leftarrow \mathrm{TF}_\mathsf{M} \cup \{lr.\mathsf{msg}\}$ // For TS-UF-$i^{\mathcal{A}}_{\mathsf{TS}}$ $(i = 0, 1)$

  If $\mathtt{tf}_i(lr)$ then $\mathrm{TF}_\mathsf{M} \leftarrow \mathrm{TF}_\mathsf{M} \cup \{lr.\mathsf{msg}\}$ // For TS-UF-$i^{\mathcal{A}}_{\mathsf{TS}}$ $(i = 2, 3, 4)$

  If $\mathtt{tf}_i(lr)$ then $\mathrm{TF}_\mathsf{L} \leftarrow \mathrm{TF}_\mathsf{L} \cup \{lr\}$ // For TS-SUF-$i^{\mathcal{A}}_{\mathsf{TS}}$

Return $\mu \notin \mathrm{TF}_\mathsf{M}$ // For TS-UF-$i^{\mathcal{A}}_{\mathsf{TS}}$

Return $(\nexists\ lr \in \mathrm{TF}_\mathsf{L} : lr.\mathsf{msg} = \mu\ \wedge\ \mathsf{SVf}(\mathsf{pk}, lr, sig) = 1)$ // For TS-SUF-$i^{\mathcal{A}}_{\mathsf{TS}}$

Oracle $\text{INIT}(CS)$:

Require: $CS \subseteq [n]$ and $|CS| < t$

$HS \leftarrow [n] \backslash CS$; $(\mathsf{pk}, \mathtt{aux}, \{\mathsf{sk}_i\}_{i \in [n]}) \leftarrow \mathsf{KeyGen}()$

For $i \in HS$ do $\mathsf{st}_i.\mathsf{sk} \leftarrow \mathsf{sk}_i$; $\mathsf{st}_i.\mathsf{pk} \leftarrow \mathsf{pk}$

Return $(\mathsf{pk}, \mathtt{aux}, \{\mathsf{sk}_i\}_{i \in CS})$

Oracle $\text{PPO}(i)$:

Require: $i \in HS$

$(pp, \mathsf{st}_i) \leftarrow\!\!\$\ \mathsf{SPP}(\mathsf{st}_i)$

$\mathrm{PP}_i \leftarrow \mathrm{PP}_i \cup \{pp\}$

Return pp

Oracle $\text{PSIGNO}(i, lr)$:

Require: $lr.\mathsf{SS} \subseteq [n]$ and $i \in HS \cap lr.\mathsf{SS}$

Require: $lr.\mathsf{PP}(i) \in \mathrm{PP}_i$ // Only for the case where TS is an echo scheme

$(psig, \mathsf{st}'_i) \leftarrow\!\!\$\ \mathsf{PS}(lr, i, \mathsf{st}_i)$; If $psig = \perp$ then return $\perp$

$\mathrm{curSS}_\mathsf{M}(lr.\mathsf{msg}) \leftarrow \mathrm{curSS}_\mathsf{M}(lr.\mathsf{msg}) \cup \{i\}$; $\mathrm{curSS}_\mathsf{L}(lr) \leftarrow \mathrm{curSS}_\mathsf{M}(lr) \cup \{i\}$

Return $psig$

Oracle $\text{RO}(x)$:

Return $\mathrm{H}(x)$

Figure 8.2: The games TS-UF-i and TS-SUF-i of a threshold signature scheme TS. In the game TS-UF-i (resp. TS-SUF-i), the set TF_M (resp. TF_L) records the set of messages (resp. leader requests) that are considered trivial under the trivial forgery predicate $\mathtt{tf}_i$ (Figure 8.3). The oracle INIT must be invoked exactly once, and only before any other oracle is queried. In particular, the predicate $\mathtt{tf}_3$ and, thus, TS-UF-3 and TS-SUF-3 are defined only if TS is an echo scheme.

8.2 Unforgeability and Strong Unfogeability

UNFORGEABILITY. Unforgeability as usual asks that the adversary be unable to produce a valid signature sig on some message M of its choice except in a trivial way. The question is what "trivial" means. For regular signatures, it means that the adversary did not obtain a signature of M from the signing oracle [78]. For threshold signatures, it is more subtle. We will give several definitions.

Figure 8.2 simultaneously describes several games, TS-UF-i_{TS} for $i = 0, 1, 2, 3, 4$, where TS-UF-3_{TS} is only defined if TS is an echo scheme. (We will get to the second set of games later.) They are almost the same, differing only in the winning conditions. The corresponding advantages of an adversary $\mathcal{A}$ are $\mathsf{Adv}^{\text{ts-uf-}i}_{\mathsf{TS}}(\mathcal{A}, \kappa) = \mathsf{Pr}[\text{TS-UF-}i^{\mathcal{A}}_{\mathsf{TS}}(\kappa)]$. The adversary calls INIT with a choice of a set of signers to corrupt. The oracle INIT must be invoked exactly once, and only before any other oracle is queried. It is also viewed as having corrupted the leader. Playing the leader role, it can request pre-processing tokens via oracle PPO. It can provide a server with a leader-request lr of its choice to obtain a partial signature PSIGNO. At the end, it outputs its forgery message M and signature sig. The adversary does not win if the signature is not valid. Now, to win, the signature must be non-trivial. It is in how this is defined that the games differ. Associated to i is a *trivial forgery* predicate $\mathtt{tf}_i$ that is invoked in the PSIGNO oracle to decide whether a message is added to the trivial forgery set TF_{M}. The choices for these predicates are shown in the table in Figure 8.3. When $i = 0$ we have the usual notion from the literature, used in particular in [27, 71, 73]. As i increases, we get more stringent (less generous) in declaring a forgery trivial, and the notion gets stronger.

Concretely, TS-UF-0 considers a signature for a message M trivial if a request lr with $lr.\mathsf{msg}$ was answered by server with a partial signature. Moving on, TS-UF-1 strengthens this by declaring a signature trivial only if at least $t - |CS|$ signers have responded to some request for message M, where these requests could have been different. In turn, TS-UF-2 strengthens this even further by requiring that there was a single prior request lr for M which was answered by $t - |CS|$ signers.

The notions TS-UF-3 only deals with echo schemes. Recall that for these schemes, a request lr contains a map $lr.\mathsf{PP} : lr.\mathsf{SS} \to \{0,1\}^*$, where $lr.\mathsf{PP}(i)$ is meant to be a token issued by server i. Here, we consider a signature for message M trivial if there exists a request lr for M which is answered by all honest signers i for which $lr.\mathsf{PP}(i)$ is a valid token previously output by i, and this set consists of at least $t - |CS|$ signers. Finally, our strongest notion, TS-UF-4 simply considers

$\mathtt{tf}_0(\mu)$	:	$\mathrm{curSS}_{\mathsf{M}}(\mu) \neq \varnothing$
$\mathtt{tf}_1(\mu)$	:	$\lvert\mathrm{curSS}_{\mathsf{M}}(\mu)\rvert \geqslant t - \lvert CS\rvert$
$\mathtt{tf}_2(lr)$	:	$\lvert\mathrm{curSS}_{\mathsf{L}}(lr)\rvert \geqslant t - \lvert CS\rvert$
$\mathtt{tf}_3(lr)$	:	$\mathtt{tf}_2(lr)$ and $\mathrm{curSS}_{\mathsf{L}}(lr) = \{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i\}$
$\mathtt{tf}_4(lr)$	:	$\mathtt{tf}_2(lr)$ and $\mathrm{curSS}_{\mathsf{L}}(lr) = HS \cap lr.\mathsf{SS}$

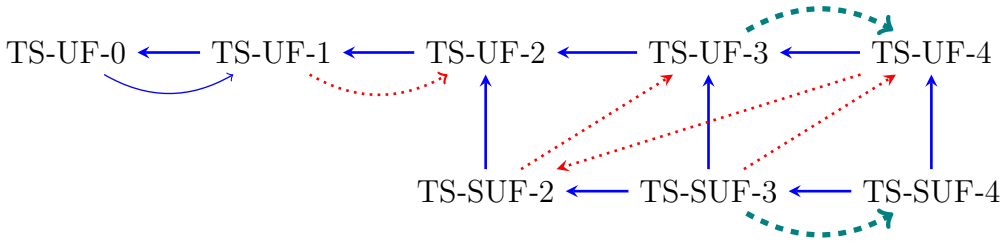

Figure 8.3: **Top:** Trivial-forgery conditions $\mathtt{tf}_i(lr)$ ($i = 0, 1, 2, 3, 4$) used to define games TS-UF-i and TS-SUF-i (Figure 8.2). **Bottom:** Relations between notions of security.

---

a signature trivial if there exists a request lr for M which is answered by all honest signers in $i \in lr.\mathsf{SS}$.

It is natural to expect TS-UF-3 and TS-UF-4 to be similar, but as we will see below, they are actually not equivalent. (Although we will give a transformation that boosts an TS-UF-3-secure scheme into an TS-UF-4-secure one.)

STRONG UNFORGEABILITY. For standard signatures, strong unforgeability asks, in addition to unforgeability, that the adversary be unable to produce a new signature on any message, where new means different from any obtained legitimately for that message. We ask, does this have any counterpart in threshold signatures? In fact, FROST seems to have such a property. We now provide formalisms to capture such properties.

It turns out that giving a general definition of strong unforgeability is rather complex, and we will restrict ourselves to a natural subclass of schemes (which includes FROST). Concretely, we ask that there is an algorithm SVf, called a *strong verification algorithm*, that takes a public key pk, a leader request lr, and a signature sig as inputs and outputs 1 or 0. We require that for any

pk, lr there exists at most one signature sig such that $\mathsf{SVf}(\mathsf{pk}, lr, sig) = 1$. Also, if there exists sig such that $\mathsf{SVf}(\mathsf{pk}, lr, sig) = 1$, it must hold that $\mathsf{Vf}(\mathsf{pk}, lr.\mathsf{msg}, sig) = 1$, and we say that lr is *intended to produce* the signature sig for the message $lr.\mathsf{msg}$. Moreover, TS is asked to satisfy a strong correctness property which is defined using the same game as $\text{TS-COR}_{\mathsf{TS}}$ except the winning condition $\mathsf{Vf}(\mathsf{pk}, M, sig) = 0$ is replaced with $\mathsf{SVf}(\mathsf{pk}, lr, sig) = 0$. As we will discuss in Section 9.1, FROST schemes exactly have this property and we can define SVf easily.

For a scheme TS with a strong verification algorithm, we consider the $\text{TS-SUF-}i_{\mathsf{TS}}$ ($i = 2, 3, 4$) games in Figure 8.2, where $\text{TS-SUF-3}_{\mathsf{TS}}$ is only defined if TS additionaly is an echo scheme. The differences (across the different values of i) are only in the trivial forgery predicates $\mathtt{tf}_i$ (defined in Figure 8.3). The difference between TS-SUF-i and TS-SUF-i is that we record the set of trivial leader requests TF_{L} instead of the set of trivial messages TF_{M}. More precisely, in TS-SUF-i, we record all leader requests lr that satisfy $\mathtt{tf}_i(lr)$ in the set TF_{L}. A forgery (μ, sig) is considered non-trivial only if no leader request in TF_{L} is intended to produce the signature sig for μ. The corresponding advantage of an adversary $\mathcal{A}$ is $\mathsf{Adv}^{\text{ts-suf-}i}_{\mathsf{TS}}(\mathcal{A}, \kappa) = \mathsf{Pr}[\text{TS-SUF-}i_{\mathsf{TS}}(\mathcal{A})]$. The ensuing notion is called TS-SUF-i.

8.3 Relations and Transformations

RELATIONS BETWEEN NOTIONS. Figure 8.3 shows relations between the notions of unforgeability and strong unforgeabilty that we have defined. A (blue, non-dotted) arrow $\text{A} \rightarrow \text{B}$ is an implication, saying that A implies B: any scheme that is A-secure is also B-secure. Now see the nodes as forming a graph with edges the blue, non-dotted arrows. The thin arrow from TS-UF-0 to TS-UF-1 indicates us that the implication only holds under a quantatively loose reduction. (We prove this in Theorem 8.3.2.) We claim that in this graph, if there is no path from a notion B to a notion A, they are separate or distinct: there exists a scheme that is B-secure but not A-secure. The dotted arrows are separations that we explicitly prove. These, together with the full arrows, prove the claim just made. The thick dotted arrows indicate the existence of a generic transformation lifting security of a scheme to achieve a stronger notion. (We establish this below as part of Theorem 8.3.3.)

8.3.1 Reference schemes and proofs of relations.

In this section, we give a set of (fully) non-interactive threshold schemes that we call reference schemes. They represent simple, canonical ways to achieve the different notions. They may not

be of practical interest, because they have key and signature sizes proportional to n, but the point is to embody notions in a representative way. A few things emanate from these schemes. One is that we use them, in the same Appendix, to establish the separations given by the dotted lines in Figure 8.3, thereby showing that any notions between which there is no path, in the graph given by the full arrows, are indeed separate. Second, we get a scheme that achieves our strongest notion, TS-SUF-4, which neither FROST schemes achieve. (Although we can get such a scheme by applying our transformation from Theorem 8.3.3 to FROST1.) Finally, reference schemes, as canonical examples, are ways to understand the notions.

PROOFS OF IMPLICATION RELATIONS. To show an unforgeability notion A implies B, since the unforgeability games defined in Figure 8.2 only differ on the winning conditions, we only need to show if the winning condition of the game of A does not hold when the adversary outputs (M, sig), the winning condition of the game of B does not hold either.

- TS-UF-1 $\Rightarrow$ TS-UF-0: Since $|CS| < t$, $|\mathrm{curSS}_{\mathsf{M}}(M)| \geqslant t - |CS|$ implies $\mathrm{curSS}_{\mathsf{M}}(M) \neq \varnothing$.

- TS-UF-2 $\Rightarrow$ TS-UF-1: If there exists lr such that $lr.\mathsf{msg} = M$ and $|\mathrm{curSS}_{\mathsf{L}}(lr)| \geqslant t - |CS|$, from the execution of the games, $|\mathrm{curSS}_{\mathsf{M}}(M)| \geqslant |\mathrm{curSS}_{\mathsf{L}}(lr)| \geqslant t - |CS|$.

- TS-(S)UF-3 $\Rightarrow$ TS-(S)UF-2: It follows from the trivial fact that $\mathtt{tf}_3(lr)$ is stronger than $\mathtt{tf}_2(lr)$.

- TS-(S)UF-4 $\Rightarrow$ TS-(S)UF-3: Due to the requirement that $lr.\mathsf{PP}(i) \in \mathrm{PP}_i$ in the oracle PSIGNO, $\mathrm{curSS}_{\mathsf{L}}(lr) \subseteq \{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i\} \subseteq HS \cap lr.\mathsf{SS}$ for any lr. Therefore, $\mathrm{curSS}_{\mathsf{L}}(lr) = HS \cap lr.\mathsf{SS}$ implies $\mathrm{curSS}_{\mathsf{L}}(lr) = \{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i\}$, which implies $\mathtt{tf}_4(lr)$ is stronger than $\mathtt{tf}_3(lr)$.

- TS-SUF-i $\Rightarrow$ TS-UF-i, for $i \in \{2, 3, 4\}$: If the adversary outputs a valid forgery (μ, sig) but does not win TS-SUF-i, there exists lr such that $\mathtt{tf}_i(lr) = 1$, $lr.\mathsf{msg} = \mu$ and $\mathsf{SVf}(\mathsf{pk}, lr, sig) = 1$. Therefore, $lr.\mathsf{msg}$ is added to TF_{M}, which means the adversary does not win TS-UF-i either.

REFERENCE SCHEMES. Fix t, n such that $2 \leqslant t < n$. The reference schemes are shown in Figure 8.4.

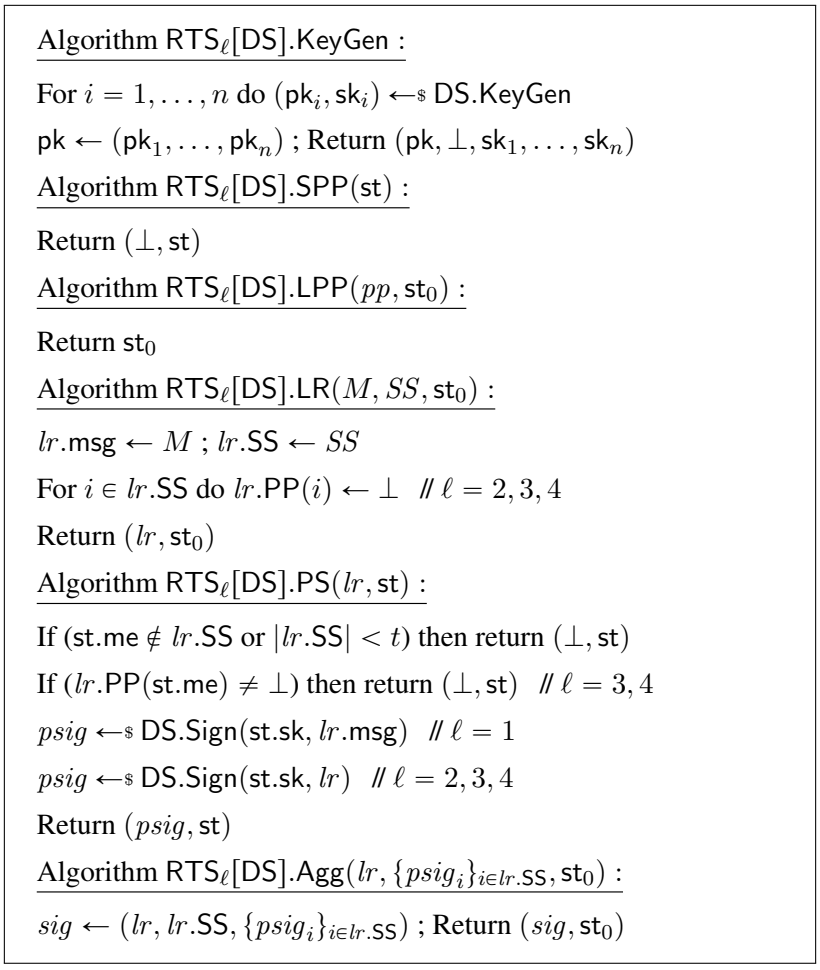

Figure 8.4: Reference threshold signature schemes $\mathsf{RTS}_\ell[\mathsf{DS}]$ associated to signature scheme DS for $\ell = 1, 2, 3, 4$. The verification and stronger verification algorithms are provided in Figure 8.5.

Proposition 8.3.1. *Suppose* DS *is a signature scheme and* $2 \leqslant t < n$. *Let* $\mathsf{RTS}_\ell[\mathsf{DS}]$ *(*$\ell = 1, 2, 3, 4$*) be the reference threshold schemes defined in Figure 8.4. Then:* ***(1)*** *For all* $\ell \in \{1, 2, 3, 4\}$*, if* DS *is UF-CMA-secure then* $\mathsf{RTS}_\ell[\mathsf{DS}]$ *is* TS-UF-ℓ*-secure, and* ***(2)*** *For all* $\ell \in \{2, 3, 4\}$*, if* DS *is unique and SUF-CMA-secure then* $\mathsf{RTS}_\ell[\mathsf{DS}]$ *is* TS-SUF-ℓ*-secure.*

The proof of the above proposition is rather straightforward. We now use these schemes to prove the separations claimed by the dotted arrows in Figure 8.3.

TS-UF-1 $\not\Rightarrow$ TS-UF-2. We need to exhibit a scheme TS that is TS-UF-1-secure but not TS-UF-2-secure. Let $\mathsf{TS} = \mathsf{RTS}_1[\mathsf{DS}]$ (Figure 8.4) where DS is a UF-CMA-secure standard signature

Algorithm $\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{Vf}(\mathsf{pk}, M, sig)$:

$(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) \leftarrow \mathsf{pk}$; $(lr, F, \{psig_i\}_{i \in F}) \leftarrow sig$

If ($lr.\mathsf{msg} \neq M$ or $F \not\subseteq lr.\mathsf{SS}$) then return 0

$T \leftarrow \{\, i \in F : \mathsf{DS}.\mathsf{Vf}(\mathsf{pk}_i, M, psig_i) \,\}$ // $\ell = 1$

$T \leftarrow \{\, i \in F : \mathsf{DS}.\mathsf{Vf}(\mathsf{pk}_i, lr, psig_i) \,\}$ // $\ell = 2, 3, 4$

Return ($T = F$ and $|T| \geqslant t$) // $\ell = 1, 2$

$E \leftarrow \{\, i \in lr.\mathsf{SS} : lr.\mathsf{PP}(i) = \perp \,\}$; Return ($T = E = F$ and $|T| \geqslant t$) // $\ell = 3$

Return ($T = lr.\mathsf{SS} = F$ and $|T| \geqslant t$) // $\ell = 4$

Algorithm $\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{SVf}(\mathsf{pk}, lr, sig)$:

// Only for $\ell = 2, 3, 4$

$(lr', F, \{psig_i\}_{i \in F}) \leftarrow sig$

Return ($\mathsf{RTS}_\ell[\mathsf{DS}].\mathsf{Vf}(\mathsf{pk}, lr.\mathsf{msg}, sig)$ and $lr = lr'$)

Figure 8.5: The verification and strong verification algorithms of the reference threshold signature schemes $\mathsf{RTS}_\ell[\mathsf{DS}]$ associated to signature scheme DS for $\ell = 1, 2, 3, 4$.

---

scheme. Proposition 8.3.1 says this achieves TS-UF-1. We now give an attack showing it fails TS-UF-2. The idea is that the adversary can make partial-signing requests to t servers, all with the same message but with $lr.\mathsf{SS}$, and thus lr itself, varying across the requests, so that $\mathrm{curSS}_{\mathsf{L}}(lr)$ stays small for any particular lr, and non-triviality under $\mathtt{tf}_2$ is maintained. Proceeding to the details of the attack, fix a message M, and consider the following adversary $\mathcal{A}$ for game TS-UF-2_{TS}:

Adversary $\mathcal{A}$

1. $CS \leftarrow \varnothing$; $(\mathsf{pk}, \varnothing, \varnothing) \leftarrow\!\!\$\ \textsc{Init}(CS)$
2. $lr_1.\mathsf{msg} \leftarrow \mu$; $lr_1.\mathsf{SS} \leftarrow [1..t]$; $lr_2.\mathsf{msg} \leftarrow M$; $lr_2.\mathsf{SS} \leftarrow [2..t+1]$
3. For $i \in [1..t-1]$ do $psig_i \leftarrow\!\!\$\ \textsc{PSignO}(i, lr_1)$
4. $psig_t \leftarrow\!\!\$\ \textsc{PSignO}(t, lr_2)$
5. $sig \leftarrow (lr_1, [1..t], \{psig_i\}_{i \in [1..t]}$; Return (μ, sig)

We claim that $\mathsf{Adv}^{\text{ts-uf-2}}_{\mathsf{TS}}(\mathcal{A}, \kappa) = 1$. The adversary has gathered t valid signatures, so the condition at line 18 of Figure 8.4 is met, and $\mathsf{TS}.\mathsf{Vf}(\mathsf{pk}, \mu, sig) = 1$. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tf}_2(lr_1) = \mathtt{tf}_2(lr_2) = 0$. Indeed, we have $|\mathrm{curSS}_{\mathsf{L}}(lr_1)| = t-1 < t-|CS| = t$ and $|\mathrm{curSS}_{\mathsf{L}}(lr_1)| = 1 < t - |CS| = t$.

TS-SUF-2 $\not\Rightarrow$ TS-UF-3. We need to exhibit a scheme TS that is TS-SUF-2-secure but not TS-UF-3-secure. Let $\mathsf{TS} = \mathsf{RTS}_2[\mathsf{DS}]$ where DS is a unique SUF-CMA-secure standard signature scheme. Proposition 8.3.1 says this achieves TS-SUF-2. We now give an attack showing it fails TS-UF-3. The idea is to allow the adversary to corrupt $t-1$ signers, while setting $lr.\mathsf{SS}$ to include all corrupted signers along with at least two honest signers. In this case, the adversary can produce a valid signature for lr by querying only a single honest signer in $lr.\mathsf{SS}$. However, $\mathrm{curSS}_{\mathsf{L}}(lr)$ does not match the set $\{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i\}$, which contains at least two honest signers. Therefore, the adversary wins TS-UF-3. Proceeding to the details of the attack, fix a message μ, and consider the following adversary $\mathcal{A}$ for game TS-UF-3_{TS}:

Adversary $\mathcal{A}$

1. $CS \leftarrow [t-1]$; $(\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_i\}_{i\in[t-1]}) \leftarrow\!\!\$\ \textsc{Init}(CS)$
2. $lr.\mathsf{msg} \leftarrow \mu$; $lr.\mathsf{SS} \leftarrow [t+1]$; For $i \in [t+1]$ do $lr.\mathsf{PP}(i) \leftarrow \bot$
3. For $i \in [t-1]$ do $psig_i \leftarrow\!\!\$\ \mathsf{DS.Sign}(\mathsf{sk}_i, lr)$
4. $psig_t \leftarrow\!\!\$\ \textsc{PSignO}(t, lr)$
5. $sig \leftarrow (lr, [t], \{psig_i\}_{i\in[t]})$; Return (μ, sig)

We claim that $\mathsf{Adv}^{\text{ts-uf-3}}_{\mathsf{TS},t}\mathcal{A} = 1$. The adversary has gathered t valid signatures, so the verification condition in $\mathsf{RTS}_2[\mathsf{DS}].\mathsf{Vf}$ (Figure 8.4) is met. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tf}_3(lr) = 0$. Indeed, we have $\mathrm{curSS}_{\mathsf{L}}(lr) = \{t\}$ but $\{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i]\} = \{t, t+1\}$.

TS-SUF-3 $\not\Rightarrow$ TS-UF-4. We need to exhibit a scheme TS that is TS-SUF-3-secure but not TS-UF-4-secure. Let $\mathsf{TS} = \mathsf{RTS}_3[\mathsf{DS}]$ where DS is a unique SUF-CMA-secure standard signature scheme. Proposition 8.3.1 says this achieves TS-SUF-3. We now give an attack showing it fails TS-UF-4. Letting E be the set of all $i \in lr.\mathsf{SS}$ such that $lr.\mathsf{PP}(i)$ is correct, meaning equals $\bot$, the idea is that the adversary can let E be a strict subset of $lr.\mathsf{SS}$ and then pass verification using only signatures from E. Proceeding to the details of the attack, fix a message μ, and consider the following adversary $\mathcal{A}$ for game TS-UF-4_{TS}:

Adversary $\mathcal{A}$

1. $CS \leftarrow \varnothing$; $(\mathsf{pk}, \mathsf{aux}, \varnothing) \leftarrow\!\!\$\ \textsc{Init}(CS)$
2. $lr.\mathsf{msg} \leftarrow \mu$; $lr.\mathsf{SS} \leftarrow [1..t+1]$
3. $lr.\mathsf{PP}(t+1) \leftarrow 0$; For $i \in [1..t]$ do $lr.\mathsf{PP}(i) \leftarrow \bot$

4. For $i \in [1..t]$ do $psig_i \leftarrow\!\!\$\ \text{PSIGNO}(i, lr)$
5. $sig \leftarrow (lr, [1..t], \{psig_i\}_{i\in[1..t]}$; Return (μ, sig)

We claim that $\mathsf{Adv}^{\text{ts-uf-4}}_{\mathsf{TS},t}(\mathcal{A}, \kappa) = 1$. From Figure 8.5, we have $E = [1..t]$ and thus $\mathsf{TS.Vf}(\mathsf{pk}, \mu, sig) = 1$. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tf}_4(lr) = 0$. Indeed, we have $\text{curSS}_{\mathsf{L}}(lr) = [1..t]$ but $HS \cap lr.\mathsf{SS} = [1..t+1]$.

TS-UF-4 $\not\Rightarrow$ TS-SUF-2. We need to exhibit a scheme TS that is TS-UF-4-secure but not TS-SUF-2-secure. First, if $x \in \{0,1\}^*$, it is convenient to let $\text{pre}(x)$ be the first bit of x and $\text{suff}(x)$ the rest. Now, let DS^* be a unique SUF-CMA-secure standard signature scheme, and modify it to a scheme DS as follows: Let $\mathsf{DS.Sign}(\cdot,\cdot) \leftarrow 0 \| \mathsf{DS}^*.\mathsf{Sign}(\cdot,\cdot)$ and let $\mathsf{DS.Vf}(\cdot,\cdot,psig) \leftarrow \mathsf{DS}^*.\mathsf{Vf}(\cdot,\cdot,\text{suff}(psig))$. Then DS is UF-CMA-secure, but, since flipping the first bit of $psig$ does not affect its validity, not SUF-CMA-secure. Now, consider $\mathsf{RTS}_4[\mathsf{DS}]$, and, for it, an alternative verification algorithm $\mathsf{RTS}_4[\mathsf{DS}].\mathsf{Vf}'$ that is as in Figure 8.4 except that line 17 is changed to $T \leftarrow \{\ i \in F\ :\ \mathsf{DS.Vf}(\mathsf{pk}_i, lr, psig_i)$ and $\text{pre}(psig_i) = 0\ \}$. Let TS be the same as $\mathsf{RTS}_4[\mathsf{DS}]$ except that we change SVf as follows: at line 22 of Figure 8.4, invoke $\mathsf{RTS}_4[\mathsf{DS}].\mathsf{Vf}'$ rather than $\mathsf{RTS}_4[\mathsf{DS}].\mathsf{Vf}$. (Note that TS.Vf stays as in $\mathsf{RTS}_4[\mathsf{DS}]$ as shown in Figure 8.4. The modified verification algorithm is only used by TS.SVf. Also note the latter meets the required condition of accepting at most one signature per key and message, due to the uniqueness of DS^*.) Proposition 8.3.1 says $\mathsf{RTS}_4[\mathsf{DS}]$ is TS-UF-4-secure, and hence so is TS, because the only difference between these two is in SVf and TS-UF-4 does not depend on this. We now give an attack showing TS fails TS-SUF-2. The idea is to exploit lack of SUF-CMA-security of the base scheme DS. Proceeding to the details of the attack, fix a message μ, and consider the following adversary $\mathcal{A}$ for game $\text{TS-SUF-2}_{\mathsf{TS}}$, where $\text{flip1}(x)$ returns string x with its first bit flipped:

Adversary $\mathcal{A}$

1. $CS \leftarrow \varnothing$; $(\mathsf{pk}, \mathtt{aux}, \varnothing) \leftarrow\!\!\$\ \text{INIT}(CS)$
2. $lr.\mathsf{msg} \leftarrow \mu$; $lr.\mathsf{SS} \leftarrow [1..t]$; For $i \in [1..t]$ do $lr.\mathsf{PP}(i) \leftarrow \perp$
3. For $i \in [1..t]$ do $psig^*_i \leftarrow\!\!\$\ \text{PSIGNO}(i, lr)$; $psig_i \leftarrow \text{flip1}(psig^*_i)$
4. $sig \leftarrow (lr, [1..t], \{psig_i\}_{i\in[1..t]}$; Return (M, sig)

It is not hard to see that $\mathsf{Adv}^{\text{ts-suf-2}}_{\mathsf{TS},t}(\mathcal{A}, \kappa) = 1$ from Figure 8.5. Now we need to show that the forgery is non-trivial, meaning $\mathtt{tsf}_2(lr, \mathsf{pk}, sig) = 0$. Indeed, $\mathsf{TS.SVf}(\mathsf{pk}, lr, sig) = 0$ because the first bit of $psig_i$ is 1 for all $i \in [1..t]$.

8.3.2 *From* TS-UF-0 *to* TS-UF-1*, loosely*

The following theorem shows TS-UF-1 security is implied by TS-UF-0 security, although with an exponential loss in t, which is acceptable in settings where t is expected to be constant.

Theorem 8.3.2. *Let* TS *be a threshold signature scheme. For any* TS-UF-1 *adversary* $\mathcal{A}$ *there exists a* TS-UF-0 *adversary* $\mathcal{B}$ *such that* $\mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{TS}}(\mathcal{A}, \kappa) \leqslant \binom{n}{t-1} \cdot \mathsf{Adv}^{\text{ts-uf-0}}_{\mathsf{TS}}(\mathcal{B}, \kappa)$ *. Moreover,* $\mathcal{B}$ *runs in time roughly equal that of* $\mathcal{A}$*, and the number of* $\mathcal{B}$*'s queries to each oracle is at most that of* $\mathcal{A}$*.*

If the adversary always corrupts $t-1$ parties, it is clear that TS-UF-0 and TS-UF-1 are equivalent. Otherwise, in general, for an adversary that breaks TS-UF-1 security and corrupts a subset CS of signers with size less than $t-1$, if the adversary wins the game TS-UF-1_{TS} by outputting (μ^*, sig^*), we know $|\mathrm{curSS}_{\mathsf{M}}(\mu^*)| < t - |CS|$. Therefore, we can modify the adversary to initially guess a subset $\overline{CS} \subseteq [1..n] \backslash CS$ with size $t - |CS| - 1$ and corrupt all parties in $\overline{CS}$. If $\overline{CS}$ happens to contain $\mathrm{curSS}_{\mathsf{M}}(\mu^*)$, the adversary actually wins. It is not hard to see that the probability that this is true is $1/\binom{n-|CS|}{t-|CS|-1} \geqslant 1/\binom{n}{t-1}$.

Proof of Theorem 8.3.2. We describe a construction of the adversary $\mathcal{B}$ as follows. $\mathcal{B}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$, $\widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: $\mathcal{B}$ randomly samples a set $\overline{CS} \in [1..n] \backslash CS$ of size $t - |CS| - 1$ and makes an oracle query $\text{INIT}(CS \cup \overline{CS})$ in the game TS-UF-0_{TS}. After receiving $\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_i\}_{i \in CS \cup \overline{CS}}$, $\mathcal{B}$ sets $\mathsf{st}_i.\mathsf{sk} \leftarrow \mathsf{sk}_i$, $\mathsf{st}_i.\mathsf{vk} \leftarrow \mathsf{pk}$, and $\mathsf{st}_i.\mathsf{aux} \leftarrow \mathsf{aux}$ for all $i \in \overline{CS}$. Finally, $\mathcal{B}$ returns $\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_i\}_{i \in CS}$.

$\widetilde{\text{PPO}}(i)$ query: Same as in the game TS-UF-1_{TS}, except when $i \in [1..n] \backslash (CS \cup \overline{CS})$, $\mathcal{B}$ directly relays the query to oracle PPO in the game TS-UF-0_{TS}.

$\widetilde{\text{PSIGNO}}(i, lr)$ query: Same as in the game TS-UF-1_{TS}, except when $i \in [1..n] \backslash (CS \cup \overline{CS})$ directly relays the query to oracle PSIGNO in the game TS-UF-0_{TS}. In addition, denote $\mathrm{curSS}'_{\mathsf{M}}$ as $\mathrm{curSS}_{\mathsf{M}}$ defined in the game TS-UF-1_{TS}. $\mathcal{B}$ also updates the set $\mathrm{curSS}'_{\mathsf{M}}(lr.\mathsf{msg})$ the same as in the game TS-UF-1_{TS}.

$\widetilde{\text{RO}}(x)$ query: $\mathcal{B}$ directly relays the query to oracle RO in the game TS-UF-0_{TS}.

After receiving the output (M^*, sig^*) from $\mathcal{A}$, denote the event $\mathsf{GoodECS}$ as $\mathrm{curSS}'_{\mathsf{M}}(M^*) \subseteq \overline{CS}$. If $\mathcal{A}$ wins the game $\text{TS-UF-1}_{\mathsf{TS}}$ and $\mathsf{GoodECS}$ occurs, $\mathcal{B}$ returns (M^*, sig^*). Otherwise, $\mathcal{B}$ aborts.

Denote the event WIN as $\mathcal{A}$ wins the game $\text{TS-UF-1}_{\mathsf{TS}}$ simulated by $\mathcal{B}$. We first show $\mathcal{B}$ wins the game $\text{TS-UF-0}_{\mathsf{TS}}$ if $\mathsf{WIN} \wedge \mathsf{GoodECS}$ occurs. From the simulation, we know $\mathrm{curSS}_{\mathsf{M}}(M^*) = \mathrm{curSS}'_{\mathsf{M}}(M^*) \backslash \overline{CS}$, where $\mathrm{curSS}_{\mathsf{M}}(M^*)$ is defined in the game $\text{TS-UF-0}_{\mathsf{TS}}$. Since $\mathsf{WIN} \wedge \mathsf{GoodECS}$ occurs, we know (M^*, sig^*) is valid for the public key pk and $\mathrm{curSS}'_{\mathsf{M}}(M^*) = \varnothing$, which implies $\mathcal{B}$ wins the game $\text{TS-UF-0}_{\mathsf{TS}}$.

Therefore, it is left to show that $\mathsf{Pr}[\mathsf{WIN} \wedge \mathsf{GoodECS}] \geqslant \frac{1}{\binom{n}{t-1}} \mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{TS}}(\mathcal{A}, \kappa)$. We first fix a set $S \in [1..n]$ with size less than t and consider the case when $\overline{CS} = S$. If WIN occurs, we know $|\mathrm{curSS}'_{\mathsf{M}}(M^*)| < t - |CS|$. Since $\mathcal{B}$ perfectly simulates the game $\text{TS-UF-0}_{\mathsf{TS}}$ no matter which $\overline{CS}$ is picked, we know the set $\mathrm{curSS}'_{\mathsf{M}}(M^*)$ is independent of the choice of $\overline{CS}$, which implies

$$\begin{aligned}\mathsf{Pr}\left[\,\mathsf{GoodECS} \,\middle|\, \mathsf{WIN} \wedge CS = S\,\right] &= \mathsf{Pr}\left[\,\overline{CS} \in \mathrm{curSS}'_{\mathsf{M}}(M^*) \,\middle|\, \mathsf{WIN} \wedge CS = S\,\right] \\ &\geqslant \frac{1}{\binom{n-|S|}{t-1-|S|}} \geqslant \frac{1}{\binom{n}{t-1}} .\end{aligned}$$

Therefore,

$$\begin{aligned}\mathsf{Pr}[\mathsf{GoodECS}] &= \sum_{\substack{S \subseteq [1..n], \\ |S|<t}} \mathsf{Pr}\left[\,\mathsf{GoodECS} \,\middle|\, \mathsf{WIN} \wedge CS = S\,\right] \cdot \mathsf{Pr}\left[\,\mathsf{WIN} \,\middle|\, CS = S\,\right] \\ &\geqslant \sum_{\substack{S \subseteq [1..n], \\ |S|<t}} \frac{1}{\binom{n}{t-1}} \mathsf{Pr}\left[\,\mathsf{WIN} \,\middle|\, CS = S\,\right] \\ &= \frac{1}{\binom{n}{t-1}} \mathsf{Pr}[\mathsf{WIN}] = \frac{1}{\binom{n}{t-1}} \mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{TS}}(\mathcal{A}, \kappa) .\end{aligned}$$

□

8.3.3 From TS-(S)UF-3 *to* TS-(S)UF-4.

Figure 8.6 gives a general transformation from TS-(S)UF-3 security to TS-(S)UF-4 security. Concretely, we give a construction ATS from any TS-(S)UF-3-secure echo scheme TS and a digital signature scheme DS. The size of signatures produced by ATS and the verification algorithm Vf are exactly the same as TS. The main idea is to use signatures to authenticate each token con-

Algorithm $\mathsf{Setup}(1^\kappa)$:

$par_1 \leftarrow \mathsf{TS.Setup}(1^\kappa)$

$par_2 \leftarrow \mathsf{DS.Setup}(1^\kappa)$

Return (par_1, par_2)

Algorithm KeyGen :

$\mathsf{pk}, \mathtt{taux}, \{\mathsf{tsk}_i\}_{i\in[1..n]} \leftarrow \mathsf{TS.KeyGen}$

For $i \in [1..n]$ do

  $(\mathsf{svk}_i, \mathsf{ssk}_i) \leftarrow\!\!\$\ \mathsf{DS.KeyGen}$

  $\mathsf{sk}_i \leftarrow (\mathsf{tsk}_i, \mathsf{ssk}_i)$

$\mathtt{aux} \leftarrow (\mathtt{taux}, \mathsf{svk}_1, \ldots, \mathsf{svk}_n)$

Return $\mathsf{pk}, \mathtt{aux}, \{\mathsf{sk}_i\}_{i\in[1..n]}$

Algorithm $\mathsf{SPP}(\mathsf{st}_i)$:

$(tpp, \mathsf{st}_i) \leftarrow\!\!\$\ \mathsf{SPP}(\mathsf{st}_i)$

$(\mathsf{tsk}_i, \mathsf{ssk}_i) \leftarrow \mathsf{st}_i.\mathsf{sk}$

$tsig \leftarrow\!\!\$\ \mathsf{DS.Sign}(\mathsf{ssk}_i, tpp)$

Return $((tpp, tsig), \mathsf{st}_i)$

Algorithm $\mathsf{LPP}(i, pp, \mathsf{st}_0)$:

$(tpp, tsig) \leftarrow pp$

$\mathsf{st}_0.\mathsf{SigMap}(i, tpp) \leftarrow tsig$

Return $\mathsf{TS.LPP}(i, tpp, \mathsf{st}_0)$

Algorithm $\mathsf{Vf}(\mathsf{pk}, \mu, sig)$:

Return $\mathsf{TS.Vf}(\mathsf{pk}, \mu, sig)$

Algorithm $\mathsf{OriginLR}(lr)$:

For $i \in lr.\mathsf{SS}$ do

  $(tpp, tsig) \leftarrow lr.\mathsf{PP}(i)$

  $lr.\mathsf{PP}(i) \leftarrow tpp$

Return lr

Algorithm $\mathsf{LR}(\mu, SS, \mathsf{st}_0)$:

$(lr, \mathsf{st}_0) \leftarrow \mathsf{TS.LR}(\mu, SS, \mathsf{st}_0)$

For $i \in SS$ do

  $tpp_i \leftarrow lr.\mathsf{PP}(i)$

  $lr.\mathsf{PP}(i) \leftarrow (tpp_i, \mathsf{st}_0.\mathsf{SigMap}(i, tpp_i))$

Return (lr, st_0)

Algorithm $\mathsf{PS}(lr, i, \mathsf{st}_i)$:

$(\mathtt{taux}, \mathsf{svk}_1, \ldots, \mathsf{svk}_n) \leftarrow \mathsf{st}_i.\mathtt{aux}$

For $i \in lr.\mathsf{SS}$ do

  $(tpp_i, tsig_i) \leftarrow lr.\mathsf{PP}(i)$

  If $\mathsf{DS.Vf}(\mathsf{svk}_i, tpp_i, tsig_i) = 0$ then

    Return $\perp$

Return $\mathsf{TS.PS}(\mathsf{OriginLR}(lr), i, \mathsf{st}_i)$

Algorithm $\mathsf{Agg}(\mathrm{PS}, \mathsf{st}_0)$:

Return $\mathsf{TS.Agg}(\mathrm{PS}, \mathsf{st}_0)$

Algorithm $\mathsf{SVf}(\mathsf{pk}, lr, sig)$:

Return $\mathsf{TS.SVf}(\mathsf{pk}, \mathsf{OriginLR}(lr), sig)$

Figure 8.6: The threshold signature $\mathsf{ATS}[\mathsf{TS}, \mathsf{DS}]$ constructed from an echo scheme TS and a digital signature scheme DS such that $\mathsf{ATS}.n = \mathsf{TS}.n$ and $\mathsf{ATS}.t = \mathsf{TS}.t$. The algorithm OriginLR transforms a well-formed leader request lr for ATS to a well-formed leader request in TS. $\mathsf{st}_0.\mathsf{SigMap}$ is a table that stores the signature corresponding to each token generated by honest signers, which is initially set to empty. PS denotes a set of partial signatures.

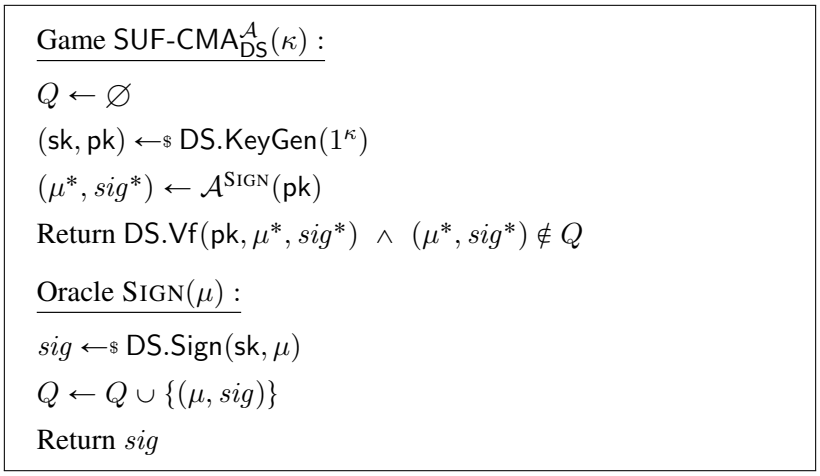

Figure 8.7: The game $\mathsf{SUF\text{-}CMA}_{\mathsf{DS}}$, where DS is a diginal signature scheme.

---

tained in a leader request lr from TS, so that an honest server only answers the request if all the authentications are valid. The rest of the protocol remains the same.

In the game $\text{TS-UF-4}_{\mathsf{ATS}}$, we can show that as long as the adversary does not break the strong unforgeability of DS, for any leader request lr such that $\text{curSS}_{\mathsf{L}}(lr) > 0$, it holds that $\{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \text{PP}_i\} = HS \cap lr.\mathsf{SS}$, which implies the conditions $\mathtt{tf}_3$ and $\mathtt{tf}_4$ are equivalent. Therefore, we can reduce TS-(S)UF-4 security of ATS to TS-(S)UF-3 security of TS and SUF-CMA security of DS. (The latter notion is formally defined via the game in Figure 8.7.) This is captured by the the following theorem.

Theorem 8.3.3. *Let* $XX \in \{\mathsf{SUF}, \mathsf{UF}\}$. *Let* TS *be an echo scheme and* DS *be a digital signature scheme. For any* TS-XX-4 *adversary* $\mathcal{A}$ *there exists a* TS-XX-3 *adversary* $\mathcal{B}$ *and a SUF-CMA adversary* $\mathcal{C}$ *such that*

$$\mathsf{Adv}^{\mathtt{ts\text{-}xx\text{-}4}}_{\mathsf{ATS}[\mathsf{TS},\mathsf{DS}]}(\mathcal{A}, \kappa) \leqslant \mathsf{Adv}^{\mathtt{ts\text{-}xx\text{-}3}}_{\mathsf{TS}}(\mathcal{B}, \kappa) + n \cdot \mathsf{Adv}^{\mathsf{suf}\text{-}cma}_{\mathsf{DS}}(\mathcal{C}, \kappa) \ .$$

Moreover, $\mathcal{B}$ *and* $\mathcal{C}$ *run in time roughly equal that of* $\mathcal{A}$. *The number of* $\mathcal{B}$*'s queries to each oracle is at most that of* $\mathcal{A}$. *The number of* $\mathcal{C}$*'s* SIGN *queries is at most the number of* PPO *queries made by* $\mathcal{A}$.

Proof. This proof only deals with TS-SUF-4 security, but a similar proof also works for TS-UF-4 security.

Let $\mathcal{A}$ be the adversary described in the theorem. After $\mathcal{A}$ returns, denote the event BadLR as there exists lr such that $\mathrm{curSS}_{\mathsf{L}}(lr) > 0$ and $\{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i\} \neq HS \cap lr.\mathsf{SS}$. Denote the event WIN as $\mathcal{A}$ wins the game $\text{TS-SUF-4}_{\mathsf{ATS}}$. Then, we have

$$\mathsf{Adv}^{\text{ts-suf-4}}_{\mathsf{ATS}}(\mathcal{A}, \kappa) \leqslant \mathsf{Pr}[\mathsf{WIN} \ \wedge \ (\neg \mathsf{BadLR})] + \mathsf{Pr}[\mathsf{BadLR}] \ .$$

Thus, we can conclude the theorem with the following two lemmas.

Lemma 8.3.4. *There exists a* TS-SUF-3 *adversary* $\mathcal{B}$ *making at most* q_{s1} *queries to* PPO*, at most* q_{s2} *queries to* PSIGNO*, and at most* q_h *queries to* RO *such that*

$$\mathsf{Pr}[\mathsf{WIN} \ \wedge \ (\neg \mathsf{BadLR})] \leqslant \mathsf{Adv}^{\text{ts-suf-3}}_{\mathsf{TS}}(\mathcal{B}, \kappa) \ .$$

Moreover, $\mathcal{B}$ *runs in time roughly equal that of* $\mathcal{A}$

Lemma 8.3.5. *There exists a SUF-CMA adversary* $\mathcal{C}$ *making at most* q_{s1} *queries to* SIGN *such that*

$$\mathsf{Pr}[\mathsf{BadLR}] \leqslant n \cdot \mathsf{Adv}^{\mathsf{suf}\text{-}cma}_{\mathsf{DS}}(\mathcal{C}, \kappa) \ ,$$

Moreover, $\mathcal{C}$ *runs in time roughly equal that of* $\mathcal{A}$

□

Proof of Lemma 8.3.4. We give a construction of the adversary $\mathcal{B}$ in Figure 8.8, where $\mathcal{B}$ runs $\mathcal{A}$ by simulating the game $\text{TS-SUF-4}_{\mathsf{ATS}}$. The simulation is done simply by relaying all queries from $\mathcal{A}$ to the oracles in the game $\text{TS-SUF-3}_{\mathsf{TS}}$ and doing the extra authentication parts by $\mathcal{B}$ itself. It is clear that $\mathcal{B}$ simulates the game $\text{TS-SUF-4}_{\mathsf{ATS}}$ perfectly, which implies the probability that $\mathcal{B}$ does not abort is equal to $\mathsf{Pr}[\mathsf{WIN} \ \wedge \ (\neg \mathsf{BadLR})]$.

Therefore, it is left to show that if $\mathcal{B}$ does not abort, then $\mathcal{B}$ wins the game $\text{TS-SUF-3}_{\mathsf{TS}}$. Suppose $\mathsf{WIN} \ \wedge \ (\neg \mathsf{BadLR})$ occurs in the game $\text{TS-SUF-4}_{\mathsf{ATS}}$ simulated by $\mathcal{B}$. We use PP' and $\mathrm{curSS}'_{\mathsf{L}}$ to denote the variables PP and $\mathrm{curSS}_{\mathsf{L}}$ in the game $\text{TS-SUF-3}_{\mathsf{TS}}$. For any PSIGNO query lr, denote $\tilde{lr} = \mathsf{OriginLR}(lr)$, and we have $lr.\mathsf{msg} = \tilde{lr}.\mathsf{msg}$ and $\mathsf{TS.SVf}(\mathsf{pk}, \tilde{lr}, sig) = 1$ if and only if $\mathsf{ATS.SVf}(\mathsf{pk}, lr, sig) = 1$. Since the output (μ, sig) must be valid for the public key pk due to WIN occurs, to show $\mathcal{B}$ wins the game $\text{TS-SUF-3}_{\mathsf{TS}}$, we just need to show for any lr such that $lr.\mathsf{msg} = \mu$ and $\mathsf{ATS.SVf}(\mathsf{pk}, lr, sig) = 1$, it holds that

$$(|\mathrm{curSS}'_{\mathsf{L}}(\tilde{lr})| < t - |CS|) \ \vee \ (\mathrm{curSS}'_{\mathsf{L}}(\tilde{lr}) \neq \{i \in HS \cap \tilde{lr}.\mathsf{SS} : \tilde{lr}.\mathsf{PP}(i) \in \mathrm{PP}'_i\}) \ . \tag{8.1}$$

Adversary $\mathcal{B}^{\text{INIT},\text{PPO},\text{PSIGNO},\text{RO}}()$:

For $i \in [1..n]$ do
  $(\mathsf{svk}_i, \mathsf{ssk}_i) \leftarrow \mathsf{DS.KeyGen}()$
$(\mu, sig) \leftarrow \mathcal{A}^{\widetilde{\text{INIT}},\widetilde{\text{PPO}},\widetilde{\text{PSIGNO}},\widetilde{\text{RO}}}()$
If $\mathsf{WIN} \wedge (\neg\mathsf{BadLR})$ occurs then
  Return (μ, sig)
Else abort

Oracle $\widetilde{\text{INIT}}(CS)$:

$\mathsf{pk}, \mathtt{taux}, \{\mathsf{tsk}_i\}_{i\in CS} \leftarrow\!\!\$\ \text{INIT}(CS)$
For $i \in CS$ do
  $\mathsf{sk}_i \leftarrow (\mathsf{tsk}_i, \mathsf{ssk}_i)$
$\mathtt{aux} \leftarrow (\mathtt{taux}, \mathsf{svk}_1, \ldots, \mathsf{svk}_n)$
Return $\mathsf{pk}, \mathtt{aux}, \{\mathsf{sk}_i\}_{i\in CS}$

Oracle $\widetilde{\text{PPO}}(i)$:

$tpp \leftarrow \text{PPO}(i)$
$tsig \leftarrow\!\!\$\ \mathsf{DS.Sign}(\mathsf{ssk}_i, pp)$
Return $(tpp, tsig)$

Oracle $\widetilde{\text{PSIGNO}}(i, lr)$:

For $i \in lr.\mathsf{SS}$ do
  $(pp_i, tsig_i) \leftarrow lr.\mathsf{PP}(i)$
  If $\mathsf{DS.Vf}(\mathsf{svk}_i, pp_i, tsig_i) = 0$ then
    Return $\perp$
Return $\text{PSIGNO}(i, \mathsf{OriginLR}(lr))$

Oracle $\widetilde{\text{RO}}(x)$:

Return $\text{RO}(x)$

Figure 8.8: Adversary $\mathcal{B}$ for the proof of Lemma 8.3.4. $\mathcal{B}$ also compute the sets L, PP, and $\mathrm{curSS}_{\mathsf{L}}(lr)$ for each $lr \in \mathrm{L}$ following the same logic as in the game $\text{TS-SUF-4}_{\mathsf{ATS}}$ and thus can check whether the event $\mathsf{WIN} \wedge (\neg\mathsf{BadLR})$ occurs.

Since WIN occurs, we know either $|\mathrm{curSS}_{\mathsf{L}}(lr)| < t - |CS|$ or $\mathrm{curSS}_{\mathsf{L}}(lr) \neq HS \cap lr.\mathsf{SS}$. If $|\mathrm{curSS}_{\mathsf{L}}(lr)| < t - |CS|$, from the simulation, we know $|\mathrm{curSS}'_{\mathsf{L}}(\tilde{lr})| = |\mathrm{curSS}_{\mathsf{L}}(lr)| < t - |CS|$, which implies (8.1) holds. Otherwise, we have $|\mathrm{curSS}_{\mathsf{L}}(lr)| \geqslant t - |CS| > 0$ and $\mathrm{curSS}_{\mathsf{L}}(lr) \neq HS \cap lr.\mathsf{SS}$. Since BadLR does not occur, we have $\{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \mathrm{PP}_i\} = HS \cap lr.\mathsf{SS}$. Therefore, for any $i \in HS \cap lr.\mathsf{SS}$, it holds that $lr.\mathsf{PP}(i) \in \mathrm{PP}_i$, which implies $\tilde{lr}.\mathsf{PP}(i) \in \mathrm{PP}'_i$. Since $\tilde{lr}.\mathsf{SS} = lr.\mathsf{SS}$, we have $\{i \in HS \cap \tilde{lr}.\mathsf{SS} : \tilde{lr}.\mathsf{PP}(i) \in \mathrm{PP}'_i\} = HS \cap lr.\mathsf{SS}$. Therefore, we have $\mathrm{curSS}'_{\mathsf{L}}(\tilde{lr}) = \mathrm{curSS}_{\mathsf{L}}(lr) \neq HS \cap lr.\mathsf{SS} = \{i \in HS \cap \tilde{lr}.\mathsf{SS} : \tilde{lr}.\mathsf{PP}(i) \in \mathrm{PP}'_i\}$, which implies (8.1) holds. □

Proof of Lemma 8.3.5. We describe a construction of the adversary $\mathcal{C}$ as follows. To start with, $\mathcal{C}$ queries $\text{INIT}()$ oracle and receives svk^*. Also, $\mathcal{C}$ initializes all the states $\mathsf{st}_0, \ldots, \mathsf{st}_n$. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$, $\widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: Same as in the game $\text{TS-SUF-4}_{\mathsf{ATS}}$, except $\mathcal{C}$ additionally randomly picks an index

$i^* \in HS$ and sets $(\mathsf{svk}_{i*}, \mathsf{ssk}_{i*}) \leftarrow_{\$} (\mathsf{svk}^*, \perp)$ instead of generating them by $\mathsf{DS.KeyGen}$. Also, $\mathcal{C}$ initializes H to an empty table.

$\widetilde{\mathbf{PPO}}(i)$ query: Same as in the game $\text{TS-SUF-4}_{\mathsf{ATS}}$, except when $i = i^*$, in the execution of $\mathsf{SPP}[\mathrm{H}](\mathsf{st}_{i*})$, $\mathcal{C}$ computes $tsig \leftarrow_{\$} \textsc{Sign}(tpp)$ instead of generating it by $\mathsf{DS.Sign}$.

$\widetilde{\mathbf{PSIGNO}}(i, lr)$ query: Same as in the game $\text{TS-SUF-4}_{\mathsf{ATS}}$.

$\widetilde{\mathbf{RO}}(x)$ query: If $\mathrm{H}(x) \neq \perp$, $\mathcal{C}$ returns $\mathrm{H}(x)$. Otherwise, $\mathcal{C}$ sets $\mathrm{H}(x) \leftarrow_{\$} \mathbb{Z}_p$ and returns $\mathrm{H}(x)$.

After receiving the output from $\mathcal{A}$, denote the event $\mathsf{GoodInd}$ as there exists $lr^* \in \mathrm{L}$ such that $\mathrm{curSS}_{\mathsf{L}}(lr^*) > 0$, $i^* \in HS \cap lr^*.\mathsf{SS}$ and $lr^*.\mathsf{PP}(i^*) \notin \mathrm{PP}_{i^*}$. If $\mathsf{GoodInd}$ does not occur, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $(tpp_{i^*}, tsig_{i^*})$, where $(tpp_{i^*}, tsig_{i^*}) \leftarrow lr^*.\mathsf{PP}(i^*)$.

We first show that $\mathcal{B}$ wins the game $\text{SUF-CMA}_{\mathsf{DS}}$ if $\mathsf{GoodInd}$ occurs. Since $\mathrm{curSS}_{\mathsf{L}}(lr^*) > 0$, we know for all $i \in lr^*.\mathsf{SS}$, $\mathsf{DS.Vf}(\mathsf{svk}_i, tpp_i, tsig_i) = 1$ where $(tpp_i, tsig_i) \leftarrow lr^*.\mathsf{PP}(i)$. From the simulation, we know $\mathrm{PP}_{i^*} = Q$, where Q is defined in the game $\text{SUF-CMA}_{\mathsf{DS}}$. Since $(tpp_{i^*}, tsig_{i^*}) = lr^*.\mathsf{PP}(i^*) \notin lr^*.\mathsf{SS}$, we know $(tpp_{i^*}, tsig_{i^*})$ is valid for $\mathsf{svk}^* = \mathsf{svk}_{i^*}$ and $(tpp_{i^*}, tsig_{i^*}) \notin Q$, which implies $\mathcal{B}$ wins the game $\text{SUF-CMA}_{\mathsf{DS}}$.

It is left show that $\mathsf{Pr}[\mathsf{GoodInd}] \geqslant \frac{1}{n}\mathsf{Pr}[\mathsf{BadLR}]$. We first fix a set $S \in [1..n]$ with size less than t and consider the case when $CS = S$. If BadLR occurs, then there exists $\tilde{lr} \in \mathrm{L}$ such that $\mathrm{curSS}_{\mathsf{L}}(\tilde{lr}) > 0$ and $\{i \in HS \cap \tilde{lr}.\mathsf{SS} : \tilde{lr}.\mathsf{PP}(i) \notin \mathrm{PP}_i\} \neq \varnothing$. Since $\mathcal{C}$ perfectly simulates the game $\text{TS-SUF-4}_{\mathsf{ATS}}$ no matter which i^* is picked, we know the set $\{i \in HS \cap \tilde{lr}.\mathsf{SS} : \tilde{lr}.\mathsf{PP}(i) \notin \mathrm{PP}_i\}$ is independent of the choice of i^*, which implies

$$\begin{aligned}
&\mathsf{Pr}\left[\,\mathsf{GoodInd} \,\middle|\, \mathsf{BadLR} \ \wedge\ CS = S\,\right] \\
&\quad = \mathsf{Pr}\left[\, i^* \in \{i \in HS \cap \tilde{lr}.\mathsf{SS} : \tilde{lr}.\mathsf{PP}(i) \notin \mathrm{PP}_i\} \,\middle|\, \mathsf{BadLR} \ \wedge\ CS = S \,\right] \\
&\quad \geqslant \frac{1}{n - |S|} \geqslant \frac{1}{n}\,.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathsf{Pr}[\mathsf{GoodInd}] &= \sum_{\substack{S \subseteq [1..n], \\ |S| < t}} \mathsf{Pr}\left[\,\mathsf{GoodInd} \,\middle|\, \mathsf{BadLR} \ \wedge\ CS = S\,\right] \cdot \mathsf{Pr}\left[\,\mathsf{BadLR} \,\middle|\, CS = S\,\right] \\
&\geqslant \sum_{\substack{S \subseteq [1..n], \\ |S| < t}} \frac{1}{n}\mathsf{Pr}\left[\,\mathsf{BadLR} \,\middle|\, CS = S\,\right] = \frac{1}{n}\mathsf{Pr}[\mathsf{BadLR}]\,.
\end{aligned}$$

□

Chapter 9

SECURITY OF THE FROST SCHEMES

9.1 The FROST Schemes

SCHEME DESCRIPTIONS. This section revisits the security of FROST, first proposed in [91] by Komlo and Goldberg, as a (partially) non-interactive threshold signature scheme. We consider both the original scheme, which we refer to as FROST1, as well as optimized versions, FROST2 and FROST3, from its follow-up work [50, 113]. We give a detailed description of all three schemes in Figures 9.1 and 9.2. The leader state st_0 contains a set curPP_i for each server i representing the set of tokens generated by server i that has not yet been used in a signing request. The state st_i for server i contains a function mapPP that maps each token pp to the randomness that is used to generate pp and $\mathsf{st}_i.\mathrm{mapPP}(pp) = \bot$ if pp is not generated by server i yet or has already been used in a signing request. All three schemes are echo schemes as defined in Section 8.1. We note that in the descrption of the protocols, we do not explicitly include the set $\mathsf{st}_i.\mathsf{PP}$ as required in Section 8.1 since its functionality is already achieved by $\mathsf{st}_i.\mathrm{mapPP}$. In particular, for pp_i defined in line 9.1, the condition $pp_i \notin \mathsf{st}_i.\mathsf{PP}$ implies $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \bot$ and thus PS returns $(\bot, \mathsf{st}_i)$ if $pp_i \notin \mathsf{st}_i.\mathsf{PP}$.

The coefficient $\lambda_i^{lr.\mathsf{SS}}$ is the Lagrange coefficient for the set $lr.\mathsf{SS}$, which is defined (for any set $\subseteq [1..n]$) as

$$\lambda_i^S := \prod_{j \in S, i \neq j} \frac{j}{j-i} \,. \tag{9.1}$$

The algorithm CompPar is a helper algorithm that computes the parameters $R, c, \{d_i\}_{i \in lr.\mathsf{SS}}$ used during signing. We stress that the only difference between FROST1 and FROST2 is the way d_i is computed in CompPar. The differences between FROST_3 and FROST_1/FROST_2 are in both the generation of lr and the computation of d_i. In FROST1, each d_i is a different hash value for each server i, while in FROST2, d_i's are the same hash value for all servers. In FROST3, when generating lr, an additional aggregated token $(\tilde{R}, \tilde{S})$ is computed and put in lr. Then, d_i's are computed from parts of lr instead of the entire lr. In particular, the hash function takes as input

Protocol FROST1,FROST2,FROST3 :

Algorithm $\mathsf{Setup}(1^\kappa)$:

$(\mathbb{G}, p, g) \leftarrow\!\!\$\ \mathsf{GGen}(1^\kappa)$

Return $(\mathbb{G}, p, g)$

Algorithm KeyGen() :

For $i \in [0..t-1]$ do

  $a_i \leftarrow\!\!\$\ \mathbb{Z}_p$

For $i \in [1..n]$ do

  $\mathsf{sk}_i \leftarrow\!\!\$\ \sum_{j=0}^{t-1} i^j \cdot a_j$; $\mathsf{pk}_i \leftarrow g^{\mathsf{sk}_i}$

$\mathsf{pk} \leftarrow g^{a_0}$

$\mathsf{aux} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

Return $\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_i\}_{i\in[1..n]}$

Algorithm $\mathsf{SPP}(\mathsf{st}_i)$:

$r \leftarrow \mathbb{Z}_p$; $s \leftarrow \mathbb{Z}_p$

$pp \leftarrow (g^r, g^s)$

$\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (r, s)$

Return (pp, st_i)

Algorithm $\mathsf{LPP}(i, pp, \mathsf{st}_0)$:

$\mathsf{st}_0.\mathrm{curPP}_i \leftarrow \mathsf{st}_0.\mathrm{curPP}_i \cup \{pp\}$

Return st_0

Algorithm $\mathsf{Vf}(\mathsf{pk}, M, sig)$:

$(R, z) \leftarrow sig$

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, M, R)$

Return $(g^z = R \cdot \mathsf{pk}^c)$

Algorithm $\mathsf{PS}(lr, i, \mathsf{st}_i)$:

$pp_i \leftarrow lr.\mathsf{PP}(i)$

If $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \bot$ then

  Return $(\bot, \mathsf{st}_i)$

$(r_i, s_i) \leftarrow \mathsf{st}_i.\mathrm{mapPP}(pp_i)$

$\mathsf{st}_i.\mathrm{mapPP}(pp_i) \leftarrow \bot$

$(R, c, \{d_j\}_{j\in lr.\mathsf{SS}})$

  $\leftarrow \mathsf{CompPar}(\mathsf{st}_i.\mathsf{vk}, lr)$

$z_i \leftarrow r_i + d_i \cdot s_i + c \cdot \lambda_i^{lr.\mathsf{SS}} \cdot \mathsf{st}_i.\mathsf{sk}$

Return $((R, z_i), \mathsf{st}_i)$

Algorithm $\mathsf{Agg}(\mathrm{PS}, \mathsf{st}_0)$:

$R \leftarrow \bot$; $z \leftarrow 0$

For $(R', z') \in \mathrm{PS}$ do

  If $R = \bot$ then $R \leftarrow R'$

  If $R \neq R'$ then return $(\bot, \mathsf{st}_0)$

  $z \leftarrow z + z'$

Return $((R, z), \mathsf{st}_0)$

Algorithm $\mathsf{SVf}(\mathsf{pk}, lr, sig)$:

$(R^*, z^*) \leftarrow sig$

$(R, c, \{d_j\}_{j\in lr.\mathsf{SS}})$

  $\leftarrow \mathsf{CompPar}(\mathsf{pk}, lr)$

Return $(R = R^*) \ \wedge \ (g^{z^*} = R \cdot \mathsf{pk}^c)$

Figure 9.1: The common components of $\mathsf{FROST1}[\mathsf{GGen}]$, $\mathsf{FROST2}[\mathsf{GGen}]$ and $\mathsf{FROST3}[\mathsf{GGen}]$, where GGen is a group generation algorithm. The algorithms LR and $\mathsf{CompPar}$ are defined in Figure 9.2 Further, n is the number of parties, and t is the threshold of the schemes. PS denotes a set of partial signatures. $\lambda_i^{lr.\mathsf{SS}}$ denotes the Lagrange coefficients, which is defined in Equation (9.1).

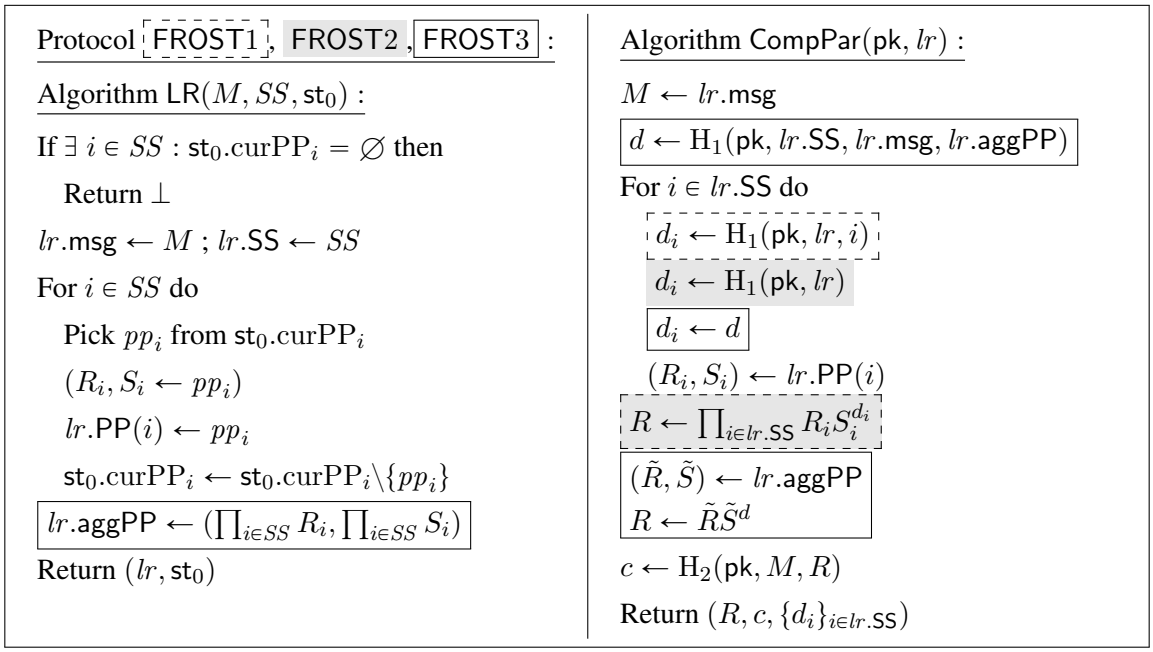

Figure 9.2: The algorithms LR and CompPar of FROST1[GGen], FROST2[GGen] and FROST3[GGen], where GGen is a group generation algorithm. In particular, FROST1 contains all the dashed boxes, FROST2 contains all the highlighted boxes, and FROST3 contains all the solid boxes. The function $\mathrm{H}_i(\cdot)$ is computed as $\mathrm{H}(i, \cdot)$ for $i = 1, 2$. PS denotes a set of partial signatures.

---

only the aggregated token, rather than all tokens in $lr.\mathsf{PP}$. We also note that upon receiving lr, each signer $i \in lr.\mathsf{SS}$ uses only $lr.\mathsf{PP}(i)$ rather than the entire $lr.\mathsf{PP}$. Therefore, in practice, the leader can reduce communication complexity by sending just $lr.\mathsf{PP}(i)$ along with the other parts of lr.

It is not hard to verify that both schemes satisfy perfect correctness.

OVERVIEW OF OUR RESULTS. Crites, Komlo, and Maller [50] argue that FROST2 improves the signing efficiency of FROST1 as the number of exponentiations for computing the nonce R is reduced from at least t to one, but they only consider TS-UF-0 security of FROST2. Ruffing et al. [113] also only show TS-UF-0 security of FROST3. In this section, we strengthen the prior results (however, in a setting without distributed key generation) by showing that FROST2 is actually TS-SUF-2-secure but not TS-UF-3 secure, and FROST3 is TS-UF-1-secure but not TS-UF-2-

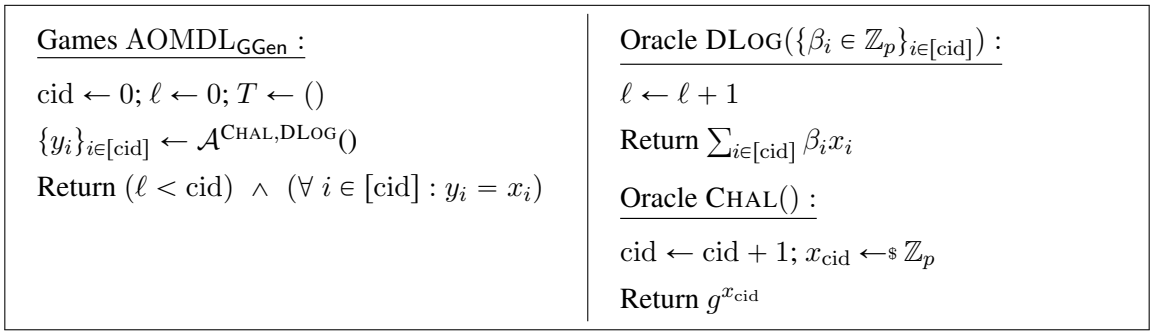

Figure 9.3: The AOMDL game, where $\mathbb{G}$ is a cyclic group with prime order p and generator g.

---

secure. In contrast, we show FROST1 is TS-SUF-3-secure but not TS-UF-4-secure. Theoretically, our results imply the separations between TS-(S)UF-2 and TS-(S)UF-3 and between TS-(S)UF-3 and TS-(S)UF-4. Practically speaking, our results indicate a separation between the security of FROST1 and FROST2. To complete the picture, a TS-SUF-4 secure variant of FROST1 can be obtained via the general transformation from Theorem 8.3.3, although it is an interesting open question whether a more efficient variant exists.

9.2 Security of FROST2

We first show that FROST2 is TS-SUF-2-secure in the ROM under the AOMDL assumption. The AOMDL assumption, introduced in [105], is formally defined in Figure 9.3. Formally, we show the following theorem.

Theorem 9.2.1. *For any* TS-SUF-2 *adversary* $\mathcal{A}$ *making at most* q_s *queries to* PPO *and at most* q_h *queries to* RO*, there exists an AOMDL adversary* $\mathcal{B}$ *making at most* $2\mathsf{q}_s + n$ *queries to* CHAL *such that*

$$\mathsf{Adv}^{\text{ts-suf-2}}_{\mathsf{FROST2}[\mathsf{GGen}]}(\mathcal{A}, \kappa) \leqslant \sqrt{\mathsf{q} \cdot (\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) + 3\mathsf{q}^2 \cdot 2^{-\kappa})} \ ,$$

where $\mathsf{q} = \mathsf{q}_s + \mathsf{q}_h + 1$*. Moreover,* $\mathcal{B}$ *runs in time roughly equal two times that of* $\mathcal{A}$*, plus the time to perform at most* $(4n + 2) \cdot \mathsf{q} + 2\mathsf{q}_s + 2n^2$ *exponentiations and group operations.*

The previous analysis of FROST2 [50] can be seen as implying TS-SUF-0 security, either in the AGM or under non-standard assumptions (which are, in turn, validated in the AGM). Our result here proves stronger security, without relying on the AGM, but also without considering

FROST's DKG. (We believe our analysis should extend, at least in the AGM, but we omit the added complexity of the DKG in this paper.) The core of the proof is a reduction from OMDL, which will need to use rewinding (via a variant of the Forking Lemma). The main challenge is to ensure that the reduction can simulate properly with a number of queries to DLOG which is smaller than the number of DL challenges. Further below, we are going to show that FROST2 is not TS-UF-3-secure, thus showing the above result is optimal with respect to our hierarchy.

Proof of Theorem 9.2.1. Let $\mathcal{A}$ be an adversary as described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(M^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\mathrm{H}_1(\mathsf{pk}, lr)$ prior to the query $\textsc{PSign}\mathrm{O}(i, lr)$ for some i and lr. (This adds up to q_s additional RO queries, and we let $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$.) Denote lr^* as the leader query such that $\mathrm{H}_1(\mathsf{pk}, lr^*)$ is the first query prior to the query $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ satisfying $\mathsf{SVf}(\mathsf{pk}, lr^*, sig^*) = \mathtt{true}$. If such lr^* does not exists, lr^* is set to $\perp$. Denote the event E_1 as

$$\mathsf{Vf}(\mathsf{pk}, M^*, sig^*) \ \wedge \ (lr^* = \perp \ \vee \ \mathrm{curSS}_{\mathsf{L}}(lr^*) < t - |CS|) \ .$$

It is clear that if $\mathcal{A}$ wins the game $\text{TS-SUF-2}_{\mathsf{FROST2}}$, then E_1 must occur, which implies $\mathsf{Pr}[E_1] \geqslant \mathsf{Adv}^{\text{ts-suf-2}}_{\mathsf{FROST2[GGen]}}(\mathcal{A}, \kappa)$. Therefore, the theorem will follow from the following lemma. (We isolate this statement as its own lemma also because it will be helpful in the proof of Theorem 9.3.1 below.) □

Lemma 9.2.2. *There exists an AOMDL adversary $\mathcal{B}$ making at most $2\mathsf{q}_s + t$ queries to* CHAL *such that*

$$\mathsf{Pr}[E_1] \leqslant \sqrt{q \cdot (\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) + 3\mathsf{q}^2 \cdot 2^{-\kappa})} \ .$$

Moreover, $\mathcal{B}$ runs in time roughly twice that of $\mathcal{A}$, plus the time to perform at most $(4n + 2) \cdot \mathsf{q} + 2\mathsf{q}_s + 2n^2$ exponentiations and group operations.

Before turning to the proof of Lemma 9.2.2, we first introduce the following variant of the forking lemma that will be used within its proof.

Lemma 9.2.3. *Let $q \geqslant 1$ be an integer, $S \subseteq [1..q]$ be a set, and H be a set. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \ldots, h_q$ outputs a pair (I, Out), where $I \in \{\perp\} \cup S$ and Out is a side output. Let IG be a randomized algorithm that generates x. The accepting probability of $\mathcal{A}$ is*

Algorithm $\mathsf{Fork}^{\mathcal{A}}(x)$:

Pick the random coin ρ of $\mathcal{A}$ at random
$h_1, h'_1, \ldots, h_q, h'_q \leftarrow H$
$(I, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$
If $I = \perp$ then return $\perp$
$(I', \mathrm{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$
If $I \neq I'$ then return $\perp$
Return $(I, \mathrm{Out}, \mathrm{Out}')$

Figure 9.4: The forking algorithm build from $\mathcal{A}$ used in Lemma 9.2.3..

---

defined as

$$\mathrm{acc}(\mathcal{A}) = \mathsf{Pr}_{x \leftarrow\$ \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[(I, \mathrm{Out}) \leftarrow\$ \mathcal{A}(x, h_1, \ldots, h_q) \ : \ I \neq \perp] \ .$$

Consider algorithm $\mathsf{Fork}^{\mathcal{A}}$ *described in Figure 9.4. The accepting probability of* $\mathsf{Fork}^{\mathcal{A}}$ *is defined as*

$$\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) = \mathsf{Pr}_{x \leftarrow\$ \mathsf{IG}}[\alpha \leftarrow\$ \mathsf{Fork}^{\mathcal{A}}(x) \ : \ \alpha \neq \perp] \ .$$

Then, $\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) \geqslant \mathrm{acc}(\mathcal{A})^2/|S|$.

The above lemma slightly extends the generalized Forking Lemma of Bellare and Neven [17] in the sense that if $\mathcal{A}$ can only output index I within a given set $S \subseteq [1..q]$, then the final bound on $\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}})$ depends only on $|S|$ instead of q. The property allows us to get better bounds in our analysis.

Proof of Lemma 9.2.3. For any $i \in S$, $h_1, \ldots, h_{i-1} \in H$, and input x, define

$$Y_i(x, h_1, \ldots, h_{i-1}) := \mathsf{Pr}_{h_i, \ldots, h_q \leftarrow\$ H}[I = i : (I, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \ .$$

Then, we have

$$\begin{aligned} \mathrm{acc}(\mathcal{A}) &= \sum_{i \in S} \mathsf{Pr}_{x \leftarrow \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[I = i : (I, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \\ &= \sum_{i \in S} \mathsf{E}_{x \leftarrow \mathsf{IG}, h_1, \ldots, h_{i-1} \leftarrow\$ H}[Y_i(x, h_1, \ldots, h_{i-1})] \ . \end{aligned}$$

Thus, we have

$$\begin{aligned}
\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) &= \sum_{i\in S} \mathsf{Pr}_{x\leftarrow \mathsf{IG},h_1,\ldots,h_n,h'_i,\ldots,h'_n \leftarrow\$ H}[I = I' = i : && (9.2)\\
&\quad (I,\mathrm{Out}) \leftarrow \mathcal{A}(x,h_1,\ldots,h_q),\\
&\quad (I',\mathrm{Out}') \leftarrow \mathcal{A}(x,h_1,\ldots,h_{i-1},h'_i,\ldots,h'_n)]\\
&= \sum_{i\in S} \mathsf{E}_{x\leftarrow \mathsf{IG},h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_i(x,h_1,\ldots,h_{i-1})^2] && (9.3)\\
&\geqslant \sum_{i\in S} \left(\mathsf{E}_{x\leftarrow \mathsf{IG},h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_{i,j}(x,h_1,\ldots,h_{i-1})]\right)^2 && (9.4)\\
&\geqslant \frac{1}{|S|}\cdot\left(\sum_{i\in S} \mathsf{E}_{x\leftarrow \mathsf{IG},h_1,\ldots,h_{i-1} \leftarrow\$ H}[Y_i(x,h_1,\ldots,h_{i-1})]\right)^2 && (9.5)\\
&= \frac{\mathrm{acc}(\mathcal{A})^2}{|S|}\ , && (9.6)
\end{aligned}$$

where Equation (9.4) is due to the fact that $\mathsf{E}[X^2] \geqslant (\mathsf{E}[X])^2$ and Equation (9.5) is due to the fact that $\sum_{i=1}^n a_i^2 \geqslant \frac{1}{n}\left(\sum_{i=1}^n a_i\right)^2$. □

Proof of Lemma 9.2.2. We first construct an algorithm $\mathcal{C}$ compatible with the syntax in Lemma 9.2.3. The input of $\mathcal{C}$ consists of $(2\mathsf{q}_s + t)$ uniformly random group elements $A_0,\ldots,A_{t-1}$, U_1, V_1, $\ldots,U_{\mathsf{q}_s},V_{\mathsf{q}_s} \in \mathbb{G}$ and uniformly random integers $h_1,\ldots,h_{2q} \in \mathbb{Z}_p$. Also, $\mathcal{C}$ can access an oracle DLOG, which on input $\boldsymbol{\alpha} \in \mathbb{Z}_p^{2\mathsf{q}_s+t}$ outputs $\mathrm{DLog}_{\mathbb{G},g}(A_0^{\alpha_1}\cdots A_{t-1}^{\alpha_t} U_1^{\alpha_{t+1}} V_1^{\alpha_{t+2}} \cdots U_{\mathsf{q}_s}^{t+2\mathsf{q}_s-1} V_{\mathsf{q}_s}^{t+2\mathsf{q}_s})$. (We can think of this oracle as part of $\mathcal{C}$ in the context of the Forking Lemma, as $\mathcal{C}$ does not need to be efficient.) To start with, $\mathcal{C}$ initializes all the states $\mathsf{st}_0,\ldots,\mathsf{st}_n$. In addition, it initializes counters ctr_s,ctr_h to 0 and a function dt to an empty table, which are used to record the DLOG query related to each (U_j,V_j). $\mathcal{C}$ also initializes $\mathrm{curLR} \leftarrow \varnothing$ to record all leader requests that appears during the game and initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. We also use a flag BadPPO to denote whether a bad event occurs, which are initially set to `false`. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$, $\widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: $\mathcal{C}$ initializes H to an empty table and sets $\mathsf{pk} \leftarrow A_0$, $\mathsf{pk}_i = \prod_{j=0}^{t-1} A_j^{ij}$ for $i \in [1..n]$, and $\mathsf{sk}_i = \mathrm{DLOG}(\mathsf{pk}_i)$ for $i \in CS$. Finally, $\mathcal{C}$ returns pk,$\mathsf{aux} = (\mathsf{pk}_1,\ldots,\mathsf{pk}_n)$, $\{\mathsf{sk}_i\}_{i\in CS}$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_1(x)$**:** If $\mathrm{H}_1(x) \neq \perp$, $\mathcal{C}$ returns $\mathrm{H}_1(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, lr)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_1(x) \leftarrow\$ \mathbb{Z}_p$ and returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1,

sets $\mathrm{H}_1(x) \leftarrow h_{2\mathrm{ctr}_h-1}$, and adds lr to curLR. Also, $\mathcal{C}$ computes $R \leftarrow \prod_{i \in lr.\mathsf{SS}} R_i S_i^{h_{2\mathrm{ctr}_h-1}}$, where $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$. If $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = \bot$, $\mathcal{C}$ sets $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = h_{2\mathrm{ctr}_h}$. In addition, define $\mathrm{mapLR}(\mathrm{ctr}_h) := lr$ and set $\mathrm{curLR} \leftarrow \mathrm{curLR} \cup \{lr\}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(x)$.

$\widetilde{\mathbf{RO}}$ query $\mathrm{H}_2(x)$**:** If $\mathrm{H}_2(x) \neq \bot$, $\mathcal{C}$ returns $\mathrm{H}_2(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, M, R)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_2(x) \leftarrow\!\!\$\ \mathbb{Z}_p$ and returns $\mathrm{H}_2(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathrm{H}_2(x) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_2(x)$.

$\widetilde{\mathbf{PPO}}(i)$ query: Same as in the game TS-SUF-2$_{\mathsf{FROST2}}$, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$, $\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (0,0)$, and $\mathrm{ctrPP}(i, pp) \leftarrow \mathrm{ctr}_s$. In addition, BadPPO is set to `true` if there exists $lr \in \mathrm{curLR}$ such that $lr.\mathsf{PP}(i) = (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$.

$\widetilde{\mathbf{PSIGNO}}(i, lr)$ query: Same as in the game TS-SUF-2$_{\mathsf{FROST2}}$, except in the simulation of algorithm PS, if $\mathsf{st}_i.\mathrm{mapPP}(pp) \neq \bot$, let $j \leftarrow \mathrm{ctrPP}(i, lr.\mathsf{PP}(i))$, and $\mathcal{C}$ computes $\boldsymbol{\alpha} \in \mathbb{Z}_p^{t+2\mathsf{q}_s}$ as

$$\alpha_k := \begin{cases} c \cdot \lambda_k^{lr.\mathsf{SS}} \cdot i^{k-1} \ , & \text{for } k \in [t] \ , \\ 1 \ , & \text{for } k = t + 2j - 1 \ , \\ d_i \ , & \text{for } k = t + 2j \ , \\ 0 \ , & o.w. \end{cases}$$

and sets $z_i \leftarrow \mathrm{DLOG}(\boldsymbol{\alpha})$, which equals to $\mathrm{DLog}_{\mathbb{G},g}\left(U_j V_j^{d_i} \mathsf{pk}_i^{c\lambda_i^{lr.\mathsf{SS}}}\right)$. In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (i, k, d_i, c\lambda_i^{lr.\mathsf{SS}}, z_i)$, where k denotes the index such that $\mathrm{H}_1(\mathsf{pk}, lr)$ is set to h_{2k-1} during the simulation.

After receiving the output $(M^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $\bot$ if BadPPO $=$ `true` or E_1 does not occur. Otherwise, $\mathcal{C}$ finds the index I such that $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ is set to h_I during the simulation. By our assumption of $\mathcal{A}$, we know such I must exist. Then, $\mathcal{C}$ returns (I, Out), where Out consists of all variables received or generated by $\mathcal{C}$.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.2.3, we define $S := \{2k\}_{k \in [1..q]}$ and IG as the algorithm that samples $2\mathsf{q}_s + t$ group elements uniformly from $\mathbb{G}$ and outputs them. From the simulation,

we know the output index I of $\mathcal{C}$ is always in S. Also, it is clear that $\mathcal{C}$ simulates the game TS-SUF-2_{FROST2} perfectly when all the inputs of $\mathcal{C}$ are uniformly sampled from their domain, which implies $\mathrm{acc}(\mathcal{C}) \geqslant \mathsf{Pr}[E_1] - \mathsf{Pr}[\mathsf{BadPPO}]$, where $\mathsf{Pr}[E_1]$ refers to the probability in the original TS-SUF-2_{FROST2} game with $\mathcal{A}$ (as in the lemma statement), whereas $\mathsf{Pr}[\mathsf{BadPPO}]$ is the probability that $\mathsf{BadPPO} = \texttt{true}$ at the end of $\mathcal{C}$'s execution. Since every pair U_j, V_j is sampled uniformly from $\mathbb{G}$, for each $\mathrm{PPO}(i)$ query, the probability BadPPO is set to $\texttt{true}$ is less than $|\mathrm{curLR}|/|G| \leqslant \mathsf{q}_h/p$. Therefore, we have $\mathsf{Pr}[\mathsf{BadPPO}] \leqslant \mathsf{q}_s\mathsf{q}_h/p$. By Lemma 9.2.3,

$$\begin{aligned}\mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) &\geqslant (\mathsf{Pr}[E_1] - \mathsf{q}_s\mathsf{q}_h/p)^2/\mathsf{q} \geqslant \mathsf{Pr}[E_1]^2/q - 2\mathsf{Pr}[E_1]\mathsf{q}_s\mathsf{q}_h/(p\cdot\mathsf{q}) \\ &\geqslant \mathsf{Pr}[E_1]^2/\mathsf{q} - 2\mathsf{q}_s/p\ .\end{aligned}$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now give a construct of the OMDL adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$, and the available DLOG oracle. To start with, $\mathcal{B}$ queries CHAL oracle $2\mathsf{q}_s + t$ times to generate $A_0, \ldots, A_{t-1}, U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$ as the input of $\mathsf{Fork}^{\mathcal{C}}$ and runs $\mathsf{Fork}^{\mathcal{C}}$. Without loss of generality, we can assume all the AOMDL challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. All DLOG queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$ to DLOG oracle of the game $\mathrm{AOMDL}_{\mathsf{GGen}}$. Denote the event $\mathsf{BadHash}$ as any two of the scalars $h_1, h'_1, \ldots, h_{\mathsf{q}}, h'_{\mathsf{q}}$ generated in the execution of $\mathsf{Fork}^{\mathcal{C}}$ are same. Since $h_1, h'_1, \ldots, h_{\mathsf{q}}, h'_{\mathsf{q}}$ are sampled uniformly from $\mathbb{Z}_p$, we know $\mathsf{Pr}[\mathsf{BadHash}] \leqslant 2\mathsf{q}^2/p$.

It is left to show that if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, \mathrm{Out}, \mathrm{Out}')$ and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ can win the game $\mathrm{AOMDL}_{\mathsf{GGen}}$, which implies

$$\mathsf{Adv}^{\mathrm{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) \geqslant \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) - \mathsf{Pr}[\mathsf{BadHash}] \geqslant \mathsf{Pr}[E_1]^2/\mathsf{q} - 3\mathsf{q}^2/p\ .$$

We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out'. By the execution of $\mathsf{Fork}^{\mathcal{C}}$, we know $(\mathsf{pk}, M^*, R^*) = (\mathsf{pk}', M^{*\prime}, R^{*\prime})$ and $\mathsf{pk} = A_0$. Since $I \in S$, let $k^* = I/2$. It is not hard to see that $\mathrm{mapLR}(k^*) = lr^*$. (If $\mathrm{mapLR}(k^*) = \perp$, lr^* is also $\perp$.)

We first show how to compute the discrete log of $A_0, \ldots, A_{t-1}$. Denote the discrete log of $A_0, \ldots, A_{t-1}$ as $a_0, \ldots, a_{t-1}$ and define a polynomial $f(x) := \sum_{i=0}^{t-1} a_i x^i$. Since $\mathsf{BadHash}$ does not occur, we have $\mathrm{H}_2(\mathsf{pk}, M^*, R^*) = h_I \neq h'_I = \mathrm{H}'_2(\mathsf{pk}, M^*, R^*)$. Since $g^{z^*} = R^x A_0^{h_I}$, $g^{z^{*\prime}} = R^x A_0^{h'_I}$, $\mathcal{B}$ computes $f(0) = a_0 = \frac{z^* - z^{*\prime}}{h_I - h'_I}$. Define $T_{\mathsf{dt}} := \{\, j : (i,k,d,c,z) \leftarrow \mathsf{dt}(j), k = k^* \,\}$. For each $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, let $(i,k,d,c,z) \leftarrow \mathsf{dt}(j)$ and $(i',k',d',c',z') \leftarrow \mathsf{dt}'(j)$, and we have $g^z = U_j V_j^d \mathsf{pk}_i^c$, $g^{z'} = U_j V_j^{d'} \mathsf{pk}_{i'}^{c'}$. Since $\mathsf{BadPPO} = \texttt{false}$ during both execution of

$\mathcal{C}$, we know (U_j, V_j) is returned by a query $\mathrm{PPO}(i)$ prior to the query $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ during the first execution of $\mathcal{C}$. Since the two executions of $\mathcal{C}$ are exactly the same prior to the query $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$, we know $i' = i$. Also, we know $d = h_k = h_{k^*} = h_{k'} = d'$. Therefore, $\mathcal{B}$ can compute $f(i) = \mathrm{DLog}_{\mathbb{G},g}(\mathsf{pk}_i) = \frac{z-z'}{c-c'}$. Denote $D := \{i\}_{j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}, (i,k,d,c,z) \leftarrow \mathsf{dt}(j)}$. Since E_1 occurs in the first execution of $\mathcal{C}$, we know $|T_{\mathsf{dt}}| = |\mathrm{curSS}_{\mathsf{L}}(lr^*)| < t - |CS|$. Therefore, we know $|D| = |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| < t - |CS|$. Therefore, $\mathcal{B}$ can pick an arbitrary set $D' \in HS \backslash D$ with size $(t - |CS| - |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| - 1)$ and for each $i \in D'$, $\mathcal{B}$ queries DLOG oracle on pk_i. Therefore, $\mathcal{B}$ knows the value of $f(i)$ for $i \in CS \cup D \cup D' \cup \{0\}$. Since $|CS \cup D \cup D' \cup \{0\}| = t$, $\mathcal{B}$ can compute the value of $a_0, \ldots, a_{t-1}$ using Lagrange interpolation.

We now show how to compute the discrete log of $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$. Denote their discrete log as $u_1, v_1, \ldots, u_{\mathsf{q}_s}, v_{\mathsf{q}_s}$. From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (i, k, d, c, z) \neq \perp$ if and only if $\mathcal{C}$ queries DLOG on $U_j V_j^d \mathsf{pk}_i^c$. Therefore, denote $\mathrm{DLOG}(U_j V_j^d \mathsf{pk}_i^c)$ as the DLOG query associated with $\mathsf{dt}(j)$. For each $j \in \mathsf{q}_s$, there are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\perp$. In this case, $\mathcal{B}$ computes u_j, v_j by directly querying oracle DLOG.

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\perp$. Without loss of generality, assume $\mathsf{dt}(j) = (i, k, d, c, z)$, which implies $g^z = U_j V_j^d \mathsf{pk}_i^c$. $\mathcal{B}$ computes v_j by directly querying oracle DLOG and computes $u_j = z - d \cdot v_j - c \cdot f(i)$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\perp$ and we denote $(i, k, d, c, z) \leftarrow \mathsf{dt}(j)$ and $(i', k', d', c', z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $k \neq k'$ or $k = k' > k^*$. In this case, we know $d = h_k \neq h'_{k'} = d'$ and $g^z = U_j V_j^d \mathsf{pk}_i^c$, $g^{z'} = U_j V_j^{d'} \mathsf{pk}_{i'}^{c'}$. Therefore, $\mathcal{B}$ computes $v_j = \frac{z - c \cdot f(i) - z' + c' \cdot f(i')}{d - d'}$, $u_j = z - d \cdot v_j - c \cdot f(i)$.

Case 3: $k = k' = k^*$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1.

Case 4: $k = k' < k^*$. $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, in this case, we have $d = d'$ and $c = c'$. Therefore, $\mathcal{B}$ queries DLOG oracle once in order to simulate the DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$.

We now count the number of DLOG queries made by $\mathcal{B}$.

- $\mathcal{B}$ queries DLOG oracle $|CS|$ times queries for simulating query $\text{DLOG}(\mathsf{pk}_i)$ made by $\mathcal{C}$ for each $i \in CS$.

- $\mathcal{B}$ queries DLOG oracle $|D'|$ times queries for computing $a_0, \ldots, a_{t-1}$.

- For each $j \in \mathsf{q}_s$, $\mathcal{B}$ queries DLOG twice for simulating query associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j, v_j in case 0, 1, 2, 4 and queries 3 times in case 3.

Since the condition of case 3 is equivalent to $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, the total number of DLOG queries made by $\mathcal{B}$ is equal to $2\mathsf{q}_s + |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| + |CS| + |D'| = 2\mathsf{q}_s + t - 1$. Therefore, $\mathcal{B}$ wins the game $\text{AOMDL}_{\mathsf{GGen}}$. □

9.3 Security of FROST1

In this section, we show that FROST1 is TS-SUF-3-secure in the ROM under the OMDL assumption. Formally, we show the following theorem.

Theorem 9.3.1. *Let* GGen *be a group generation algorithm. For any* TS-SUF-3 *adversary* $\mathcal{A}$ *making at most* q_s *queries to* PPO *and at most* q_h *queries to* RO*, there exists an AOMDL adversary* $\mathcal{B}$ *making at most* $2\mathsf{q}_s + t$ *queries to* CHAL *such that*

$$\mathsf{Adv}^{\text{ts-suf-3}}_{\mathsf{FROST1[GGen]}}(\mathcal{A}, \kappa) \leqslant 4n \cdot q \cdot \sqrt{\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) + 6\mathsf{q} \cdot 2^{-\kappa}}\ ,$$

where $\mathsf{q} = \mathsf{q}_s + \mathsf{q}_h + 1$*. Moreover,* $\mathcal{B}$ *runs in time roughly equal two times that of* $\mathcal{A}$*, plus the time to perform at most* $6n \cdot \mathsf{q} + 4\mathsf{q}_s + 2n^2$ *exponentiations and group operations.*

The proof here follows a similar pattern than that of Theorem 9.2.1, but will be more complex. In particular, the lesser tight bound is due to the fact that we need to consider an additional bad event, which we upper bound via a different reduction from OMDL. As we explain in detail below, this reduction will make use of a looser Forking Lemma, which is a variant of the "Local Forking Lemma" [15], which only resamples a single random oracle output when rewinding. The extra looseness is due to needing to ensure an extra condition when rewinding.

Proof of Theorem 9.3.1. Let $\mathcal{A}$ be the adversary described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(M^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\text{H}_2(\mathsf{pk}, M^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\text{H}_1(\mathsf{pk}, lr, i)$

prior to the query $\text{PSIGNO}(i, lr)$ for some i and lr. (This adds up to q_s additional RO queries, and we let $q = \mathsf{q}_h + \mathsf{q}_s + 1$.) Denote lr^* as the leader query such that $\mathrm{H}_1(\mathsf{pk}, lr^*, i)$ is the first RO query prior to the $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ query for some i satisfying $\mathsf{SVf}(\mathsf{pk}, lr^*, sig^*) = \texttt{true}$. If such lr^* does not exist, lr^* is set to $\perp$. Denote the event E_1 as

$$\mathsf{Vf}(\mathsf{pk}, M^*, sig^*) \ \wedge \ (lr^* = \perp \ \vee \ \mathrm{curSS}_{\mathsf{L}}(lr^*) < t - |CS|) \ .$$

Denote the event E_2 as

$$\mathsf{Vf}(\mathsf{pk}, M^*, sig^*) \ \wedge \ lr^* \neq \perp \ \wedge \ \mathrm{curSS}_{\mathsf{L}}(lr^*) \neq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\} \ .$$

If $\mathcal{A}$ wins the game $\text{TS-SUF-3}_{\mathsf{FROST1}}$ and $lr^* \neq \perp$, we know either $\mathrm{curSS}_{\mathsf{L}}(lr^*) < t - |CS|$ or $\mathrm{curSS}_{\mathsf{L}}(lr^*) \neq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\}$. Therefore, if $\mathcal{A}$ wins the game $\text{TS-SUF-3}_{\mathsf{FROST1}}$, then either E_1 or E_2 occurs, which implies

$$\mathsf{Adv}^{\text{ts-suf-3}}_{\mathsf{FROST1[GGen]}}(\mathcal{A}, \kappa) \leqslant \mathsf{Pr}[E_1] + \mathsf{Pr}[E_2] \leqslant 2\max\{\mathsf{Pr}[E_1], \mathsf{Pr}[E_2]\} \ .$$

Thus, we conclude the theorem with the following two lemmas.

Lemma 9.3.2. *There exists an OMDL adversary $\mathcal{B}$ making at most $2\mathsf{q}_s + t$ queries to* CHAL *such that*

$$\mathsf{Pr}[E_1] \leqslant \sqrt{q \cdot (\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) + 3\mathsf{q}^2(n+1)^2 \cdot 2^{-\kappa})} \ ,$$

Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$, plus the time to perform at most $6n \cdot \mathsf{q} + 4\mathsf{q}_s + 2n^2$ exponentiations and group operations.

Lemma 9.3.3. *There exists an OMDL adversary $\mathcal{B}$ making at most $2\mathsf{q}_s$ queries to* CHAL *such that*

$$\mathsf{Pr}[E_2] \leqslant n \cdot \mathsf{q}\sqrt{2(\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) + 2^{-\kappa})} \ .$$

Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$, plus the time to perform at most $6n \cdot \mathsf{q} + 4\mathsf{q}_s + 2n^2$ exponentiations and group operations.

This completes the proof of the theorem, subject to proofs of the lemmas that we discuss next. □

The proof of Lemma 9.3.2 is almost the same as Lemma 9.2.2, so we omit the full proof. The only difference is that $\mathcal{C}$ takes as input $h_1, \ldots, h_{(n+1)q}$ in order to simulate all RO queries. For a RO query $\mathrm{H}_1(\mathsf{pk}, lr, i)$, $\mathcal{C}$ first enumerates all $i' \in [n]$ and assigns $h_{(\mathrm{ctr}_h - 1)(n+1)+i'}$ to $\mathrm{H}_1(\mathsf{pk}, lr, i')$.

Then, $\mathcal{C}$ computes the nonce R for lr and assigns $h_{\mathrm{ctr}_h(n+1)}$ to $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R)$ if it is not assigned any value yet. Similarly, for a new RO query $\mathrm{H}_1(\mathsf{pk}, M, R)$, its value is set to $h_{\mathrm{ctr}_h(n+1)}$. The rest follows by similar analysis.

To prove Lemma 9.3.3, we need the following variant of the forking lemma, which extends the local forking lemma of [15]. The difference is that forking is happening on two indices I, J (leading to I', J' in the forking) while in [15] there is a single I (and corresponding I' in the forking). The difference between a local forking ([15] and Lemma 9) versus classical ([17] and Lemma 5) is that in the former only one point is resampled in forking while in the latter it is all points following the fork.

Lemma 9.3.4. *Let $q \geqslant 1$ be an integer and H and Q be two sets. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \ldots, h_q$ outputs a tuple (I, J, Out), where $I \in \{\perp\} \cup [1..q]$, $J \in Q$, and Out is a side output. Let IG be a randomized algorithm that generates x. The accepting probability of $\mathcal{A}$ is defined as*

$$\mathrm{acc}(\mathcal{A}) := \mathsf{Pr}_{x \leftarrow\$ \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[(I, \mathrm{Out}) \leftarrow\$ \mathcal{A}(x, h_1, \ldots, h_q) \ : \ I \neq \perp] \ .$$

Consider algorithm $\mathsf{Fork}_2^{\mathcal{A}}$ described in Figure 9.5. The accepting probability of $\mathsf{Fork}_2^{\mathcal{A}}$ is defined as

$$\mathrm{acc}(\mathsf{Fork}_2^{\mathcal{A}}) := \mathsf{Pr}_{x \leftarrow\$ \mathsf{IG}}[\alpha \leftarrow\$ \mathsf{Fork}^{\mathcal{A}}(x) \ : \ \alpha \neq \perp] \ .$$

Then, $\mathrm{acc}(\mathsf{Fork}_2^{\mathcal{A}}) \geqslant \mathrm{acc}(\mathcal{A})^2/(q \cdot |Q|)$.

Proof. Denote $\hat{h}_i = (h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_q) \in H^{q-1}$. For any $i \in [1..q]$, $j \in Q$, $\hat{h}_i \in H^{q-1}$, and input x, define

$$Y_{i,j}(x, \hat{h}_i) := \mathsf{Pr}_{h_i \leftarrow\$ H}[I = i, J = j : (I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \ .$$

Then, we have

$$\begin{aligned}
\mathrm{acc}(\mathcal{A}) &= \sum_{i=1}^{q} \sum_{j \in Q} \mathsf{Pr}_{x \leftarrow \mathsf{IG}, h_1, \ldots, h_q \leftarrow\$ H}[I = i, J = j : (I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)] \\
&= \sum_{i=1}^{q} \sum_{j \in Q} \mathsf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \leftarrow\$ H^{q-1}}[Y_{i,j}(x, \hat{h}_i)] \ .
\end{aligned}$$

Algorithm $\mathsf{Fork}_2^{\mathcal{A}}(x)$:

Pick the random coin ρ of $\mathcal{A}$ at random

$h_1, \ldots, h_q \leftarrow H$

$(I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q; \rho)$

If $I = \perp$ then return $\perp$

$h'_I \leftarrow H$

$(I', J', \mathrm{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{I-1}, h'_I, h_{I+1}, \ldots, h_q; \rho)$

If $I \neq I'$ or $J \neq J'$ then return $\perp$

Return $(I, J, \mathrm{Out}, \mathrm{Out}')$

Figure 9.5: The forking algorithm build from $\mathcal{A}$ used in Lemma 9.3.4.

Thus, we have

$$\begin{aligned}
\mathrm{acc}(\mathsf{Fork}^{\mathcal{A}}) &= \sum_{i=1}^{q} \sum_{j \in Q} \mathsf{Pr}_{x \leftarrow \mathsf{IG}, h_1, \ldots, h_n, h'_i \xleftarrow{\$} H}[I = I' = i, J = J' = j : && (9.7)\\
&\qquad (I, J, \mathrm{Out}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q),\\
&\qquad (I', J', \mathrm{Out}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{i-1}, h'_i, h_{i+1}, h_q)]\\
&= \sum_{i=1}^{q} \sum_{j \in Q} \mathsf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \xleftarrow{\$} H^{q-1}}[Y_{i,j}(x, \hat{h}_i)^2] && (9.8)\\
&\geqslant \sum_{i=1}^{q} \sum_{j \in Q} \left(\mathsf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \xleftarrow{\$} H^{q-1}}[Y_{i,j}(x, \hat{h}_i)] \right)^2 && (9.9)\\
&\geqslant \frac{1}{q \cdot |Q|} \cdot \left(\sum_{i=1}^{q} \sum_{j \in Q} \mathsf{E}_{x \leftarrow \mathsf{IG}, \hat{h}_i \xleftarrow{\$} H^{q-1}}[Y_{i,j}(x, \hat{h}_i)] \right)^2 && (9.10)\\
&= \frac{\mathrm{acc}(\mathcal{A})^2}{q \cdot |Q|} , && (9.11)
\end{aligned}$$

where (9.9) is due to the fact that $\mathsf{E}[X^2] \geqslant (\mathsf{E}[X])^2$ and (9.10) is due to the fact that $\sum_{i=1}^{n} a_i^2 \geqslant \frac{1}{n} \left(\sum_{i=1}^{n} a_i\right)^2$. □

Proof of Lemma 9.3.3. We first construct an algorithm $\mathcal{C}$ following the syntax of the algorithm described in Lemma 9.3.4. The input of $\mathcal{C}$ consists of $2\mathsf{q}_s$ uniformly random group elements $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s} \in \mathbb{G}$ and uniformly random vectors $h_1, \ldots, h_{n \cdot \mathsf{q}} \in (\mathbb{Z}_p)$. Similarly to the

proof of Lemma 9.2.2, $\mathcal{C}$ can access DLOG oracle and at the beginning, initializes all the states $\mathsf{st}_0, \ldots, \mathsf{st}_n$ as in the game TS-SUF-3_{FROST1}, and initializes the counters ctr_s,ctr_h to 0 and the function dt to an empty table. $\mathcal{C}$ also initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$, $\widetilde{\text{RO}}$, which are simulated as follows. In the following description, we use i to denote the index of parties, j to denote the index of $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$, and k to denote the index of $h_1, \ldots, h_{n \cdot q}$.

$\widetilde{\text{INIT}}(CS)$: $\mathcal{C}$ initializes H to an empty table and samples $a_0, \ldots, a_{t-1}$ uniformly from $\mathbb{Z}_p$. Define $f(x) := \sum_{i=0}^{t-1} a_i x^i$. Then, $\mathcal{C}$ sets $\mathsf{pk} \leftarrow g^{f(0)}$, $\mathsf{pk}_i = g^{f(i)}$ for $i \in [1..n]$, and $\mathsf{sk}_i \leftarrow f(i)$ for $i \in CS$. Finally, $\mathcal{C}$ returns pk,$\mathsf{aux} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, $\{\mathsf{sk}_i\}_{i \in CS}$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_1(x)$**:** If $\mathrm{H}_1(x) \neq \bot$, $\mathcal{C}$ returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ parses x as $(\widetilde{\mathsf{pk}}, lr, \tilde{i})$ for some $\tilde{i} \in [1..n]$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_1(x) \leftarrow_{\$} \mathbb{Z}_p$ and returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathrm{H}_1(\mathsf{pk}, lr, i) \leftarrow h_{n(\mathrm{ctr}_h - 1)+i}$ for each $i \in [1..n]$. In addition, define $\mathrm{mapLR}(\mathrm{ctr}_h) := lr$. Then, $\mathcal{C}$ computes $R \leftarrow \prod_{i \in lr.\mathsf{SS}} R_i S_i^{d_i}$, where $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$ and $d_i = \mathrm{H}_1(\mathsf{pk}, lr, i)$. If $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = \bot$, $\mathcal{C}$ sets $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) \leftarrow_{\$} \mathbb{Z}_p$. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(x)$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_2(x)$**:** If $\mathrm{H}_2(x) \neq \bot$, $\mathcal{C}$ returns $\mathrm{H}_2(x)$. Otherwise, $\mathcal{C}$ sets $\mathrm{H}_2(x) \leftarrow_{\$} \mathbb{Z}_p$ and returns $\mathrm{H}_2(x)$.

$\widetilde{\text{PPO}}(i)$ query: Same as in the game TS-SUF-3_{FROST1}, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$, $\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (0, 0)$, and $\mathrm{ctrPP}(i, pp) \leftarrow \mathrm{ctr}_s$.

$\widetilde{\text{PSIGNO}}(i, lr)$ query: Same as in the game TS-SUF-3_{FROST1}, except in the simulation of algorithm PS, if $\mathsf{st}_i.\mathrm{mapPP}(pp) \neq \bot$, let $j \leftarrow \mathrm{ctrPP}(i, lr.\mathsf{PP}(i))$, and $\mathcal{C}$ computes $\boldsymbol{\alpha} \in \mathbb{Z}_p^{t+2\mathsf{q}_s}$ as

$$\alpha_k := \begin{cases} 1\ , & \text{for } k = 2j - 1\ , \\ d_i\ , & \text{for } k = 2j\ , \\ 0\ , & o.w. \end{cases}$$

and sets $z_i \leftarrow \text{DLOG}(\boldsymbol{\alpha}) + c\lambda_i^{lr.SS} \cdot f(i)$, where $\text{DLOG}(\boldsymbol{\alpha}) = \text{DLog}_{\mathbb{G},g}\left(U_j V_j^{d_i}\right)$. In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (k, d_i, z_i - c\lambda_i^{lr.\mathsf{SS}} \cdot f(i))$, where k denotes the index such that $\text{H}_1(\mathsf{pk}, lr, i)$ is set to h_k during the simulation.

After receiving the output $(M^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $(\perp, \perp, \perp)$ if E_2 does not occur. Otherwise, we know $\text{curSS}_{\mathsf{L}}(lr^*) > 0$ and $\text{curSS}_{\mathsf{L}}(lr^*) \neq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \text{PP}_i\}$. Therefore, there exists k^* and i^* such that $\text{mapLR}(k^*) = lr^*$ and $i^* \in \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \text{PP}_i\} \backslash \text{curSS}_{\mathsf{L}}(lr^*)$. (Since $\text{curSS}_{\mathsf{L}}(lr^*) \subseteq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \text{PP}_i\}$, we must have $\{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \text{PP}_i\} \backslash \text{curSS}_{\mathsf{L}}(lr^*) \neq \varnothing$.) Since $i^* \in \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \text{PP}_i\}$, there exists $j^* \in [1..\mathsf{q}_s]$ such that $lr^*.\mathsf{PP}(i^*) = (U_{j^*}, V_{j^*})$. If $\mathsf{dt}(j^*) = \perp$, $\mathcal{C}$ sets $J \leftarrow \perp$. Otherwise, let $(k, d, z) \leftarrow \mathsf{dt}(j^*)$ and $\mathcal{C}$ sets $J = k$. Then, $\mathcal{C}$ returns $(n(k^* - 1) + i^*, J, \text{Out})$, where Out consists of all variables received or generated by $\mathcal{C}$, including i^*, j^*, k^*, lr^*.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.3.4, we define IG as the algorithm that samples $2\mathsf{q}_s$ group elements uniformly from $\mathbb{G}$ and outputs them. The output J is either $\perp$ or in $[1..(n \cdot q)]$. It is not hard to see that $\mathcal{C}$ simulates the game $\text{TS-SUF-3}_{\mathsf{FROST1}}$ perfectly when all the inputs of $\mathcal{C}$ are uniformly sampled from their domain, which implies $\text{acc}(\mathcal{C}) \geqslant \mathsf{Pr}[E_2]$, where $\mathsf{Pr}[E_2]$ refers to the probability in the original $\text{TS-SUF-3}_{\mathsf{FROST1}}$ game with $\mathcal{A}$ (as in the lemma statement). By Lemma 9.3.4,

$$\text{acc}(\mathsf{Fork}_2^{\mathcal{C}}) \geqslant \frac{\mathsf{Pr}[E_2]^2}{n \cdot q(n \cdot q + 1)} \leqslant \frac{\mathsf{Pr}[E_2]^2}{2n^2q^2} .$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now give a construct of the OMDL adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$. To start with, $\mathcal{B}$ queries CHAL oracle $2\mathsf{q}_s$ times to generate U_1,V_1,...,U_{q_s},V_{q_s} as the input of $\mathsf{Fork}^{\mathcal{C}}$ and runs $\mathsf{Fork}^{\mathcal{C}}$. Without loss of generality, we can assume all the OMDL challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. All DLOG queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$ to DLOG oracle of the game $\text{OMDL}_{\mathsf{GGen}}$. Denote the event $\mathsf{BadHash}$ as $h_I \neq h'_I$, where I are outputted by the first execution of $\mathcal{C}$. Since h_I, I are independent of h'_I, we know $\mathsf{Pr}[\mathsf{BadHash}] \leqslant 1/p$.

It is left to show that if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, J, \text{Out}, \text{Out}')$ and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ can win the game $\text{OMDL}_{\mathsf{GGen}}$, which implies

$$\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) \geqslant \text{acc}(\mathsf{Fork}^{\mathcal{C}}) - \mathsf{Pr}[\mathsf{BadHash}] \geqslant \frac{\mathsf{Pr}[E_2]^2}{2n^2q^2} - 1/p .$$

We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out$'$. We first show how to compute the discrete log of U_j, V_j for $j \neq j^*$. Denote u_j, v_j as the discrete log of U_j, V_j. There are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\perp$. In this case, $\mathcal{B}$ computes u_j, v_j by directly querying oracle $\mathrm{DLOG}(U_j)$ and $\mathrm{DLOG}(V_j)$.

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\perp$. Without loss of generality, assume $\mathsf{dt}(j) = (k, d, z)$, which implies $g^z = U_j V_j^d$. $\mathcal{B}$ computes v_j by directly querying oracle $\mathrm{DLOG}(V_j)$ and computes $u_j = z - d \cdot v_j$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\perp$ and we denote $(k, d, z) \leftarrow \mathsf{dt}(j)$ and $(k', d', z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $d \neq d'$. In this case, $\mathcal{B}$ computes $v_j = \frac{z - z'}{d - d'}$, $u_j = z - d \cdot v_j$.

Case 3: $d = d'$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, since $d = d'$, $\mathcal{B}$ queries DLOG oracle only once in order to answer queries $\mathrm{DLOG}(U_j V_j^d)$ and $\mathrm{DLOG}(U_j V_j^{d'})$ from $\mathsf{Fork}^{\mathcal{C}}$.

From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (k, d, z) \neq \perp$ if and only if $\mathcal{C}$ queries DLOG on $(U_j V_j^d)$. Therefore, denote $\mathrm{DLOG}(U_j V_j^d)$ as the DLOG query associated with $\mathsf{dt}(j)$. For all the above cases, $\mathcal{B}$ queries DLOG oracle twice for simulating DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j, v_j.

We now show how to compute u_{j^*} and v_{j^*}. From the execution of $\mathsf{Fork}_2^{\mathcal{C}}$, we know $\mathsf{pk} = \mathsf{pk}'$ and $\mathrm{mapLR}(k) = \mathrm{mapLR}'(k)$ for all $k \leqslant I$, which implies $lr^* = \mathrm{mapLR}(I) = \mathrm{mapLR}'(I) = lr^{*\prime}$. Since E_2 occurs in both executions of $\mathcal{C}$, we know $\mathsf{SVf}(\mathsf{pk}, lr^*, (R^*, z^*)) = \texttt{true}$ and $\mathsf{SVf}(\mathsf{pk}, lr^*, (R^{*\prime}, z^{*\prime})) = \texttt{true}$ are valid. Therefore, $g^{z^*} = R^* g^{a_0 c}$, $R^* = \sum_{i \in lr^*.\mathsf{SS}} R_i S_i^{d_i}$, $g^{z^{*\prime}} = R^{*\prime} g^{a_0 c'}$, $R^{*\prime} = \sum_{i \in lr^*.\mathsf{SS}} R_i S_i^{d'_i}$, where $(R_i, S_i) = lr.\mathsf{PP}(i)$, $c = \mathrm{H}_2(\mathsf{pk}, M^*, R^*)$, $c' = \mathrm{H}_2'(\mathsf{pk}, M^*, R^{*\prime})$, and $d_i = \mathrm{H}_1(\mathsf{pk}, lr^*, i)$, $d'_i = \mathrm{H}_1'(\mathsf{pk}, lr^*, i)$. Since for each $i \neq i^*$ we have $d_i = h_{n(k^*-1)+i} = d'_i$, we have $g^{z^* - z^{*\prime}} = \frac{R^*}{R^{*\prime}} g^{a_0(c-c')} = S_{i^*}^{d_{i^*} - d'_{i^*}} g^{a_0(c-c')}$. Therefore, $\mathcal{C}$ can compute $v_{j^*} = \frac{z^* - z^{*\prime} - a_0(c-c')}{d_{i^*} - d'_{i^*}}$. If $J = \perp$, $\mathcal{B}$ computes u_{j^*} by querying $\mathrm{DLOG}(U_{j^*})$ directly. In this case, $\mathcal{B}$ queries DLOG only once to compute u_{j^*} and v_{j^*}. If $J \neq \perp$, let $(k, d, z) \leftarrow \mathsf{dt}(j^*) =$ and $(k', d', z') \leftarrow \mathsf{dt}(j^*)$. Then, $\mathcal{B}$ computes $u_{j^*} = z - d \cdot v_{j^*}$. Since $i^* \notin S_2(lr^*)$, we know $k \neq I$.(Otherwise, suppose $k = I$. Since $I = n(k^* - 1) + i^*$ and $\mathrm{mapLR}(k^*) = lr^*$, we know a $\mathrm{PSIGNO}(i^*, lr^*)$ is made and does not return $\perp$ during the simulation, which implies $i^* \in S_2(lr^*)$.) Thus, we have $k' = J = k \neq I$ and $d = h_J = d'$, which means $\mathcal{B}$ only needs to query DLOG once to simulate the DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$. Therefore, the total number of DLOG queries made by $\mathcal{B}$ is equal to $2\mathsf{q}_s - 1$, which implies $\mathcal{B}$ wins the game $\mathrm{OMDL}_{\mathsf{GGen}}$. $\square$

9.4 *Security of* FROST3

We show that FROST3 is TS-UF-1-secure in the ROM under the AOMDL assumption. Formally, we show the following theorem.

Theorem 9.4.1. *Let* GGen *be a group generation algorithm. For any* TS-UF-1 *adversary* $\mathcal{A}$ *making at most* q_s *queries to* PPO *and at most* q_h *queries to* RO*, there exists an AOMDL adversary* $\mathcal{B}$ *making at most* $2\mathsf{q}_s + n$ *queries to* CHAL *such that*

$$\mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{FROST3[GGen]}}(\mathcal{A}, \kappa) \leqslant \sqrt{\mathsf{q} \cdot (\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) + 3\mathsf{q}^2 \cdot 2^{-\kappa})}\ ,$$

where $\mathsf{q} = \mathsf{q}_s + \mathsf{q}_h + 1$*. Moreover,* $\mathcal{B}$ *runs in time roughly equal two times that of* $\mathcal{A}$*, plus the time to perform at most* $(4n + 2) \cdot \mathsf{q} + 2\mathsf{q}_s + 2n^2$ *exponentiations and group operations.*

The proof is almost identical to that of FROST2. The only difference is that in FROST3, the size of T_{dt} can be bounded only by $\mathrm{curSS}_{\mathsf{M}}(M^)$—which is itself bounded by $t - |CS|$ due to the winning condition of TS-UF-1—rather than by $\mathrm{curSS}_{\mathsf{L}}(lr^)$. The reason is that in FROST3, multiple distinct lr's may correspond to (M^*, R^*), i.e., $lr.\mathsf{msg} = M^*$ and $\mathsf{CompPar}(\mathsf{pk}, lr) \to (R^*, \dots)$. Therefore, T_{dt} cannot be bounded by any single lr. We will see this leads to a concrete attack for TS-UF-2-seucrity in Section 9.5 We provide the proof below for completeness.

Proof of Theorem 9.4.1. Let $\mathcal{A}$ be an adversary as described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(M^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\mathrm{H}_1(\mathsf{pk}, lr)$ prior to the query $\text{PSIGNO}(i, lr)$ for some i and lr. (This adds up to q_s additional RO queries, and we let $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$.)

We first construct an algorithm $\mathcal{C}$ compatible with the syntax in Lemma 9.2.3. The input of $\mathcal{C}$ consists of $(2\mathsf{q}_s + t)$ uniformly random group elements $A_0, \dots, A_{t-1}, U_1, V_1, \dots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s} \in \mathbb{G}$ and uniformly random integers $h_1, \dots, h_{2q} \in \mathbb{Z}_p$. Also, $\mathcal{C}$ can access an oracle DLOG, which on input $\boldsymbol{\alpha} \in \mathbb{Z}_p^{2\mathsf{q}_s + t}$ outputs $\mathrm{DLog}_{\mathbb{G},g}(A_0^{\alpha_1} \cdots A_{t-1}^{\alpha_t} U_1^{\alpha_{t+1}} V_1^{\alpha_{t+2}} \cdots U_{\mathsf{q}_s}^{t+2\mathsf{q}_s-1} V_{\mathsf{q}_s}^{t+2\mathsf{q}_s})$. (We can think of this oracle as part of $\mathcal{C}$ in the context of the Forking Lemma, as $\mathcal{C}$ does not need to be efficient.) To start with, $\mathcal{C}$ initializes all the states $\mathsf{st}_0, \dots, \mathsf{st}_n$. In addition, it initializes counters ctr_s,ctr_h to 0 and a function dt to an empty table, which are used to record the DLOG query related to each (U_j, V_j). $\mathcal{C}$ also initializes $\mathrm{curLR} \leftarrow \varnothing$ to record all leader requests that appears during the

game and initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. We also use a flag BadPPO to denote whether a bad event occurs, which are initially set to `false`. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}, \widetilde{\text{PPO}}, \widetilde{\text{PSIGNO}}, \widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: $\mathcal{C}$ initializes H to an empty table and sets $\mathsf{pk} \leftarrow A_0$, $\mathsf{pk}_i = \prod_{j=0}^{t-1} A_j^{i^j}$ for $i \in [1..n]$, and $\mathsf{sk}_i = \text{DLOG}(\mathsf{pk}_i)$ for $i \in CS$. Finally, $\mathcal{C}$ returns pk,$\mathsf{aux} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, $\{\mathsf{sk}_i\}_{i \in CS}$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_1(x)$**:** If $\mathrm{H}_1(x) \neq \perp$, $\mathcal{C}$ returns $\mathrm{H}_1(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, lr)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_1(x) \leftarrow_{\$} \mathbb{Z}_p$ and returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1, sets $\mathrm{H}_1(x) \leftarrow h_{2\mathrm{ctr}_h - 1}$, and adds lr to curLR. Also, $\mathcal{C}$ computes $R \leftarrow \prod_{i \in lr.\mathsf{SS}} R_i S_i^{h_{2\mathrm{ctr}_h - 1}}$, where $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$. If $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = \perp$, $\mathcal{C}$ sets $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = h_{2\mathrm{ctr}_h}$. In addition, define $\mathrm{mapLR}(\mathrm{ctr}_h) := lr$ and set $\mathrm{curLR} \leftarrow \mathrm{curLR} \cup \{lr\}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(x)$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_2(x)$**:** If $\mathrm{H}_2(x) \neq \perp$, $\mathcal{C}$ returns $\mathrm{H}_2(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, M, R)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_2(x) \leftarrow_{\$} \mathbb{Z}_p$ and returns $\mathrm{H}_2(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathrm{H}_2(x) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_2(x)$.

$\widetilde{\text{PPO}}(i)$ query: Same as in the game $\text{TS-SUF-2}_{\mathsf{FROST2}}$, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$, $\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (0, 0)$, and $\mathrm{ctrPP}(i, pp) \leftarrow \mathrm{ctr}_s$. In addition, BadPPO is set to `true` if there exists $lr \in \mathrm{curLR}$ such that $lr.\mathsf{PP}(i) = (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$.

$\widetilde{\text{PSIGNO}}(i, lr)$ query: Same as in the game $\text{TS-SUF-2}_{\mathsf{FROST2}}$, except in the simulation of algorithm PS, if $\mathsf{st}_i.\mathrm{mapPP}(pp) \neq \perp$, let $j \leftarrow \mathrm{ctrPP}(i, lr.\mathsf{PP}(i))$, and $\mathcal{C}$ computes $\boldsymbol{\alpha} \in \mathbb{Z}_p^{t+2\mathsf{q}_s}$ as

$$\alpha_k := \begin{cases} c \cdot \lambda_k^{lr.\mathsf{SS}} \cdot i^{k-1} \ , & \text{for } k \in [t] \ , \\ 1 \ , & \text{for } k = t + 2j - 1 \ , \\ d_i \ , & \text{for } k = t + 2j \ , \\ 0 \ , & o.w. \end{cases}$$

and sets $z_i \leftarrow \text{DLOG}(\boldsymbol{\alpha})$, which equals to $\text{DLog}_{\mathbb{G},g}\left(U_j V_j^{d_i} \mathsf{pk}_i^{c\lambda_i^{lr.\mathsf{SS}}}\right)$. In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (i, k, d_i, c\lambda_i^{lr.\mathsf{SS}}, z_i)$, where k denotes the index such that $\text{H}_1(\mathsf{pk}, lr)$ is set to h_{2k-1} during the simulation.

After receiving the output $(M^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $\perp$ if $\mathsf{BadPPO} = \texttt{true}$ or E_1 does not occur. Otherwise, $\mathcal{C}$ finds the index I such that $\text{H}_2(\mathsf{pk}, M^*, R^*)$ is set to h_I during the simulation. By our assumption of $\mathcal{A}$, we know such I must exist. Then, $\mathcal{C}$ returns (I, Out), where Out consists of all variables received or generated by $\mathcal{C}$.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.2.3, we define $S := \{2k\}_{k\in[1..q]}$ and IG as the algorithm that samples $2\mathsf{q}_s + t$ group elements uniformly from $\mathbb{G}$ and outputs them. From the simulation, we know the output index I of $\mathcal{C}$ is always in S. Also, it is clear that $\mathcal{C}$ simulates the game $\text{TS-UF-1}_{\mathsf{FROST3}}$ perfectly when all the inputs of $\mathcal{C}$ are uniformly sampled from their domain, which implies $\text{acc}(\mathcal{C}) \geqslant \mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{FROST3}}(\mathcal{A}, \kappa) - \mathsf{Pr}[\mathsf{BadPPO}]$, where $\mathsf{Pr}[E_1]$ refers to the probability in the original $\text{TS-SUF-2}_{\mathsf{FROST2}}$ game with $\mathcal{A}$ (as in the lemma statement), whereas $\mathsf{Pr}[\mathsf{BadPPO}]$ is the probability that $\mathsf{BadPPO} = \texttt{true}$ at the end of $\mathcal{C}$'s execution. Since every pair U_j, V_j is sampled uniformly from $\mathbb{G}$, for each $\text{PPO}(i)$ query, the probability BadPPO is set to $\texttt{true}$ is less than $|\text{curLR}|/|G| \leqslant \mathsf{q}_h/p$. Therefore, we have $\mathsf{Pr}[\mathsf{BadPPO}] \leqslant \mathsf{q}_s\mathsf{q}_h/p$. By Lemma 9.2.3,

$$\text{acc}(\mathsf{Fork}^{\mathcal{C}}) \geqslant \mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{FROST3}}(\mathcal{A}, \kappa)^2/\mathsf{q} - 2\mathsf{q}_s/p\ .$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now give a construct of the OMDL adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$, and the available DLOG oracle. To start with, $\mathcal{B}$ queries CHAL oracle $2\mathsf{q}_s + t$ times to generate $A_0, \ldots, A_{t-1}, U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$ as the input of $\mathsf{Fork}^{\mathcal{C}}$ and runs $\mathsf{Fork}^{\mathcal{C}}$. Without loss of generality, we can assume all the AOMDL challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. All DLOG queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$ to DLOG oracle of the game $\text{AOMDL}_{\mathsf{GGen}}$. Denote the event $\mathsf{BadHash}$ as any two of the scalars $h_1, h'_1, \ldots, h_{\mathsf{q}}, h'_{\mathsf{q}}$ generated in the execution of $\mathsf{Fork}^{\mathcal{C}}$ are same. Since $h_1, h'_1, \ldots, h_{\mathsf{q}}, h'_{\mathsf{q}}$ are sampled uniformly from $\mathbb{Z}_p$, we know $\mathsf{Pr}[\mathsf{BadHash}] \leqslant 2\mathsf{q}^2/p$.

It is left to show that if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, \text{Out}, \text{Out}')$ and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ can win the game $\text{AOMDL}_{\mathsf{GGen}}$, which implies

$$\mathsf{Adv}^{\text{aomdl}}_{\mathsf{GGen}}(\mathcal{B}, \kappa) \geqslant \text{acc}(\mathsf{Fork}^{\mathcal{C}}) - \mathsf{Pr}[\mathsf{BadHash}] \geqslant \mathsf{Adv}^{\text{ts-uf-1}}_{\mathsf{FROST3}}(\mathcal{A}, \kappa)^2/\mathsf{q} - 3\mathsf{q}^2/p\ .$$

We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out$'$. By the execution of $\mathsf{Fork}^{\mathcal{C}}$, we know $(\mathsf{pk}, M^*, R^*) = (\mathsf{pk}', M^{*\prime}, R^{*\prime})$

and $\mathsf{pk} = A_0$. Since $I \in S$, let $k^* = I/2$. It is not hard to see that $\mathrm{mapLR}(k^*) = lr^*$. (If $\mathrm{mapLR}(k^*) = \perp$, lr^* is also $\perp$.)

We first show how to compute the discrete log of $A_0, \ldots, A_{t-1}$. Denote the discrete log of $A_0, \ldots, A_{t-1}$ as $a_0, \ldots, a_{t-1}$ and define a polynomial $f(x) := \sum_{i=0}^{t-1} a_i x^i$. Since $\mathsf{BadHash}$ does not occur, we have $\mathrm{H}_2(\mathsf{pk}, M^*, R^*) = h_I \neq h'_I = \mathrm{H}'_2(\mathsf{pk}, M^*, R^*)$. Since $g^{z^*} = R^x A_0^{h_I}$, $g^{z^{*\prime}} = R^x A_0^{h'_I}$, $\mathcal{B}$ computes $f(0) = a_0 = \frac{z^* - z^{*\prime}}{h_I - h'_I}$. Define $T_{\mathsf{dt}} := \{\, j : (i,k,d,c,z) \leftarrow \mathsf{dt}(j), k = k^* \,\}$. For each $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, let $(i,k,d,c,z) \leftarrow \mathsf{dt}(j)$ and $(i',k',d',c',z') \leftarrow \mathsf{dt}'(j)$, and we have $g^z = U_j V_j^d \mathsf{pk}_i^c$, $g^{z'} = U_j V_j^{d'} \mathsf{pk}_{i'}^{c'}$. Since $\mathsf{BadPPO} = \texttt{false}$ during both execution of $\mathcal{C}$, we know (U_j, V_j) is returned by a query $\mathrm{PPO}(i)$ prior to the query $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ during the first execution of $\mathcal{C}$. Since the two executions of $\mathcal{C}$ are exactly the same prior to the query $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$, we know $i' = i$. Also, we know $d = h_k = h_{k^*} = h_{k'} = d'$. Therefore, $\mathcal{B}$ can compute $f(i) = \mathrm{DLog}_{\mathbb{G},g}(\mathsf{pk}_i) = \frac{z - z'}{c - c'}$. Denote $D := \{i\}_{j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}, (i,k,d,c,z) \leftarrow \mathsf{dt}(j)}$. Since E_1 occurs in the first execution of $\mathcal{C}$, we know $|T_{\mathsf{dt}}| = |\mathrm{curSS}_{\mathsf{M}}(M^*)| < t - |CS|$. Therefore, we know $|D| = |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| < t - |CS|$. Therefore, $\mathcal{B}$ can pick an arbitrary set $D' \in HS \backslash D$ with size $(t - |CS| - |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| - 1)$ and for each $i \in D'$, $\mathcal{B}$ queries DLOG oracle on pk_i. Therefore, $\mathcal{B}$ knows the value of $f(i)$ for $i \in CS \cup D \cup D' \cup \{0\}$. Since $|CS \cup D \cup D' \cup \{0\}| = t$, $\mathcal{B}$ can compute the value of $a_0, \ldots, a_{t-1}$ using Lagrange interpolation.

We now show how to compute the discrete log of $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$. Denote their discrete log as $u_1, v_1, \ldots, u_{\mathsf{q}_s}, v_{\mathsf{q}_s}$. From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (i,k,d,c,z) \neq \perp$ if and only if $\mathcal{C}$ queries DLOG on $U_j V_j^d \mathsf{pk}_i^c$. Therefore, denote $\mathrm{DLOG}(U_j V_j^d \mathsf{pk}_i^c)$ as the DLOG query associated with $\mathsf{dt}(j)$. For each $j \in \mathsf{q}_s$, there are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\perp$. In this case, $\mathcal{B}$ computes u_j, v_j by directly querying oracle $\mathrm{DLOG}(U_j)$ and $\mathrm{DLOG}(V_j)$.

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\perp$. Without loss of generality, assume $\mathsf{dt}(j) = (i,k,d,c,z)$, which implies $g^z = U_j V_j^d \mathsf{pk}_i^c$. $\mathcal{B}$ computes v_j by directly querying oracle $\mathrm{DLOG}(V_j)$ and computes $u_j = z - d \cdot v_j - c \cdot f(i)$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\perp$ and we denote $(i,k,d,c,z) \leftarrow \mathsf{dt}(j)$ and $(i',k',d',c',z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $k \neq k'$ or $k = k' > k^*$. In this case, we know $d = h_k \neq h'_{k'} = d'$ and $g^z = U_j V_j^d \mathsf{pk}_i^c$, $g^{z'} = U_j V_j^{d'} \mathsf{pk}_{i'}^{c'}$. Therefore, $\mathcal{B}$ computes $v_j = \frac{z - c \cdot f(i) - z' + c' \cdot f(i')}{d - d'}$, $u_j = z - d \cdot v_j - c \cdot f(i)$.

Case 3: $k = k' = k^*$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1.

Case 4: $k = k' < k^*$. $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, in this case, we have $d = d'$ and $c = c'$. Therefore, $\mathcal{B}$ queries DLOG oracle once in order to simulate the DLOG queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$.

We now count the number of DLOG queries made by $\mathcal{B}$.

- $\mathcal{B}$ queries DLOG oracle $|CS|$ times queries for simulating query $\mathrm{DLOG}(\mathsf{pk}_i)$ made by $\mathcal{C}$ for each $i \in CS$.

- $\mathcal{B}$ queries DLOG oracle $|D'|$ times queries for computing $a_0, \ldots, a_{t-1}$.

- For each $j \in \mathsf{q}_s$, $\mathcal{B}$ queries DLOG twice for simulating query associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j, v_j in case 0, 1, 2, 4 and queries 3 times in case 3.

Since the condition of case 3 is equivalent to $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, the total number of DLOG queries made by $\mathcal{B}$ is equal to $2\mathsf{q}_s + |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| + |CS| + |D'| = 2\mathsf{q}_s + t - 1$. Therefore, $\mathcal{B}$ wins the game $\mathrm{AOMDL}_{\mathsf{GGen}}$. □

9.5 Attacks for the FROST Schemes

FROST1 IS NOT TS-UF-4 SECURE. Consider the setting where $n = 20$ and $t = 3$ and the adversary $\mathcal{A}$ for the game $\text{TS-UF-4}_{\mathsf{FROST1}}$ described in Figure 9.6. We now show that $\mathsf{Adv}^{\text{ts-uf-4}}_{\mathsf{FROST1}}(\mathcal{A}, \kappa) = 1$. From the execution of PSIGNO, we know $g^{z_1} = R_1 S_1^{d_{11}} \mathsf{pk}_{11}^{\lambda_{11}^{\{11,15,20\}} \cdot c}$. The key observation here is that $\lambda_{11}^{\{11,15,20\}} = \frac{15 \cdot 20}{(15-11)(20-11)} = \frac{25}{3} = \frac{5 \cdot 10}{(5-11)(10-11)} = \lambda_{11}^{\{5,10,11\}}$. Therefore,

$$\begin{aligned} g^z &= R_1 S_1^{d_{11}} g^{r_2 + r_3 + s_2 \cdot d_{15} + s_3 \cdot d_{20}} \mathsf{pk}_{11}^{\lambda_{11}^{\{11,15,20\}} \cdot c} \mathsf{pk}_5^{\lambda_5^{\{5,10,11\}} \cdot c} \mathsf{pk}_{10}^{\lambda_{10}^{\{5,10,11\}} \cdot c} \\ &= R g^{c \cdot \sum_{i \in \{5,10,11\}} \lambda_i^{\{5,10,11\}} \cdot \mathsf{sk}_i} = R \cdot \mathsf{pk}^c \,, \end{aligned}$$

which implies $(M, (R, z))$ is valid for pk. Also, it is clear that $\mathrm{curSS}_{\mathsf{L}}(lr) = \{11\}$ and $HS \cap lr.\mathsf{SS} = \{11, 15, 20\}$, which implies the condition $\mathtt{tf}_4(lr)$ does not hold. Therefore, $\mathcal{A}$ wins the game $\text{TS-UF-4}_{\mathsf{FROST1}}$ with probability 1.

The reason why the attack is possible for FROST1 is because the honest server 11 replies to the leader request lr with tokens $lr.\mathsf{PP}(15)$ and $lr.\mathsf{PP}(20)$ not generated by the honest servers 15 and

Adversary $\mathcal{A}^{\text{INIT,PPO,PSIGNO,RO}}$:

$CS \leftarrow \{5, 10\}$; $(\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_5, \mathsf{sk}_{10}\}) \leftarrow\!\!\$\ \text{INIT}(CS)$

$(R_1, S_1) \leftarrow\!\!\$\ \text{PPO}(11)$; $s_2, r_2, s_3, r_3 \leftarrow\!\!\$\ \mathbb{Z}_p$

$lr.\mathsf{msg} \leftarrow M$; $lr.\mathsf{SS} \leftarrow \{11, 15, 20\}$

$lr.\mathsf{PP}(11) \leftarrow (R_1, S_1)$; $lr.\mathsf{PP}(15) \leftarrow (g^{r_2}, g^{s_2})$; $lr.\mathsf{PP}(20) \leftarrow (g^{r_3}, g^{s_3})$

$z_1 \leftarrow \text{PSIGNO}(11, lr)$

For $i \in \{11, 15, 20\}$ do $d_i \leftarrow \text{RO}(1, \mathsf{pk}, lr, i)$

$R \leftarrow R_1 S_1^{d_{11}} g^{r_2 + r_3 + s_2 \cdot d_{15} + s_3 \cdot d_{20}}$; $c \leftarrow \text{RO}(2, \mathsf{pk}, R, M)$

$z \leftarrow z_1 + r_2 + r_3 + s_2 \cdot d_{15} + s_3 \cdot d_{20} + c(\lambda_5^{\{5,10,11\}} \cdot \mathsf{sk}_5 + \lambda_{10}^{\{5,10,11\}} \cdot \mathsf{sk}_{10})$

Return $(M, (R, z))$

Figure 9.6: Adversary $\mathcal{A}$ that wins the game $\text{TS-UF-4}_{\mathsf{FROST1}}$, where M is a fixed message.

20 but by the adversary instead. Therefore, the attack is prevented by the general transformation from TS-SUF-3 security to TS-SUF-4 security described in Figure 8.6 since after the transformation an honest server replies to a leader request only when all the tokens within the request are authenticated by the corresponding servers, and it is not possible for the adversary to generate authenticated tokens on behalf of honest servers.

FROST2 IS NOT TS-UF-3 SECURE. Consider the setting where $n = 4$ and $t = 3$ and the adversary $\mathcal{A}$ for the game $\text{TS-UF-3}_{\mathsf{FROST2}}$ described in Figure 9.7. We now show that $\mathsf{Adv}^{\text{ts-uf-3}}_{\mathsf{FROST2}}(\mathcal{A}, \kappa) = 1$. From the execution of PSIGNO, we know $g^{z_1} = R_1 S_1^d \mathsf{pk}_1^{\lambda_1^{\{1,2,3\}} \cdot c}$. Therefore,

$$\begin{aligned} g^z &= R_1^\gamma S_1^{d \cdot \gamma} \mathsf{pk}_1^{\gamma \cdot \lambda_1^{\{1,2,3\}} \cdot c} \mathsf{pk}_3^{\lambda_3^{\{1,3,4\}} \cdot c} \mathsf{pk}_4^{\lambda_4^{\{1,3,4\}} \cdot c} \\ &= R g^{c \cdot \sum_{i \in \{1,3,4\}} \lambda_i^{\{1,3,4\}} \cdot \mathsf{sk}_i} = R \cdot \mathsf{pk}^c \ , \end{aligned}$$

which implies $(M, (R, z))$ is valid for pk. Also, it is clear that $\text{curSS}_{\mathsf{L}}(lr) = \{1\}$ and $\{i \in HS \cap lr.\mathsf{SS} : lr.\mathsf{PP}(i) \in \text{PP}_i\} = \{1, 2\}$, which implies the condition $\mathtt{tf}_3(lr)$ does not hold. Therefore, $\mathcal{A}$ wins the game $\text{TS-UF-3}_{\mathsf{FROST2}}$ with probability 1.

FROST3 IS NOT TS-UF-2 SECURE. Consider the setting where $n = 2$ and $t = 2$ and the adversary $\mathcal{A}$ for the game $\text{TS-UF-2}_{\mathsf{FROST3}}$ described in Figure 9.8. We now show that $\mathsf{Adv}^{\text{ts-uf-2}}_{\mathsf{FROST3}}(\mathcal{A}, \kappa) = 1$. From the execution of PSIGNO, we know $g^{z_1} = R_1 S_1^d \mathsf{pk}_1^{\lambda_1^{\{1,2\}} \cdot c}$ and $g^{z_2} = R_2 S_2^d \mathsf{pk}_2^{\lambda_2^{\{1,2\}} \cdot c}$.

Adversary $\mathcal{A}^{\text{INIT,PPO,PSIGNO,RO}}$:

$CS \leftarrow \{3,4\}$; $(\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_3, \mathsf{sk}_4\}) \leftarrow\!\!\$\ \text{INIT}(CS)$

$(R_1, S_1) \leftarrow\!\!\$\ \text{PPO}(1)$; $(R_2, S_2) \leftarrow\!\!\$\ \text{PPO}(2)$; $\gamma \leftarrow \lambda_1^{\{1,3,4\}}/\lambda_1^{\{1,2,3\}}$

$lr.\mathsf{msg} \leftarrow M$; $lr.\mathsf{SS} \leftarrow \{1,2,3\}$

$lr.\mathsf{PP}(1) \leftarrow (R_1, S_1)$; $lr.\mathsf{PP}(2) \leftarrow (R_2, S_2)$

$lr.\mathsf{PP}(3) \leftarrow (R_1^{\gamma^{-1}} R_2^{-1}, S_1^{\gamma^{-1}} S_2^{-1})$

$z_1 \leftarrow \text{PSIGNO}(1, lr)$

$d \leftarrow \text{RO}(1, \mathsf{pk}, lr)$; $R \leftarrow R_1^{\gamma} S_1^{\gamma \cdot d}$; $c \leftarrow \text{RO}(2, \mathsf{pk}, R, M)$

$z \leftarrow \gamma \cdot z_1 + c(\lambda_3^{\{1,3,4\}} \cdot \mathsf{sk}_3 + \lambda_4^{\{1,3,4\}} \cdot \mathsf{sk}_4)$

Return $(M, (R, z))$

Figure 9.7: Adversary $\mathcal{A}$ that wins the game $\text{TS-UF-3}_{\mathsf{FROST2}}$, where M is a fixed message.

Therefore,

$$g^z = R_1 S_1^d \mathsf{pk}_1^{\lambda_1^{\{1,2\}} \cdot c} R_2 S_2^d \mathsf{pk}_2^{\lambda_2^{\{1,2\}} \cdot c} = R_1 R_2 (S_1 S_2)^d (\mathsf{pk}_1^{\lambda_1^{\{1,2\}}} \mathsf{pk}_2^{\lambda_2^{\{1,2\}}})^c = R \cdot \mathsf{pk}^c \ ,$$

which implies $(M, (R, z))$ is valid for pk. Also, it is clear that $\text{curSS}_{\mathsf{L}}(lr_1) = \{1\}$ and $\text{curSS}_{\mathsf{L}}(lr_2) = \{2\}$, which implies the condition $\mathtt{tf}_2(lr)$ does not hold. Therefore, $\mathcal{A}$ wins the game $\text{TS-UF-2}_{\mathsf{FROST3}}$ with probability 1.

Adversary $\mathcal{A}^{\text{INIT,PPO,PSIGNO,RO}}$:

$CS \leftarrow \varnothing$; $(\mathsf{pk}, \mathsf{aux}, \perp) \leftarrow\!\!\$\ \text{INIT}(CS)$

$(R_1, S_1) \leftarrow\!\!\$\ \text{PPO}(1)$; $(R_2, S_2) \leftarrow\!\!\$\ \text{PPO}(2)$

$lr_1.\mathsf{msg} \leftarrow M$; $lr_1.\mathsf{SS} \leftarrow \{1, 2\}$; $lr_1.\mathsf{aggPP} \leftarrow (R_1R_2, S_1S_2)$

$lr_1.\mathsf{PP}(1) \leftarrow (R_1, S_1)$; $lr_1.\mathsf{PP}(2) \leftarrow\!\!\$\ (gR_2, gS_2)$

$lr_2.\mathsf{msg} \leftarrow M$; $lr_2.\mathsf{SS} \leftarrow \{1, 2\}$; $lr_2.\mathsf{aggPP} \leftarrow (R_1R_2, S_1S_2)$

$lr_2.\mathsf{PP}(1) \leftarrow (gR_1, gS_1)$; $lr_2.\mathsf{PP}(2) \leftarrow\!\!\$\ (R_2, S_2)$

$z_1 \leftarrow \text{PSIGNO}(1, lr_1)$; $z_2 \leftarrow \text{PSIGNO}(2, lr_2)$

$d \leftarrow \text{RO}(1, \mathsf{pk}, lr.\mathsf{SS}, lr.\mathsf{msg}, lr.\mathsf{aggPP})$; $R \leftarrow R_1R_2(S_1S_2)^d$; $c \leftarrow \text{RO}(2, \mathsf{pk}, R, M)$

$z \leftarrow z_1 + z_2$

Return $(M, (R, z))$

Figure 9.8: Adversary $\mathcal{A}$ that wins the game $\text{TS-UF-2}_{\mathsf{FROST}_3}$, where M is a fixed message.

Chapter 10

PARTIALLY NON-INTERACTIVE THRESHOLD SIGNATURES FROM LINEAR HASH FUNCTIONS

In this chapter, we first give the definition of linear hash functions and algebraic one-more preimage resistance (AOMPR) of a linear hash function family. Then, we show that AOMPR is implied by collision resistance of the linear hash function family, and that such linear hash function family can be instantiated from pairing-free groups based on the DL assumption. Finally, we give new threshold signature constructions based on linear hash functions.

10.1 Linear Hash Functions

The notion of linear hash functions is introduced in [80, 81], which is in turn adapted from [10]. We adapt the definition from [80] by additionally requiring the scalar set $\mathcal{S}$ to be a field and $\mathcal{D}$ and $\mathcal{R}$ to be $\mathcal{S}$-modules, which is necessary for the reduction from collision resistance to AOMPR and for our constructions in Section 10.4 to work.

Definition 10.1.1. *A linear hash function family* LHF *is a pair of algorithms* $(\mathsf{PGen}, \mathsf{F})$ *such that*

a) PGen *is a randomized algorithm that takes as input the security parameter* 1^κ *and returns the system parameter* par *that defines three sets* $\mathcal{S} = \mathcal{S}(par), \mathcal{D} = \mathcal{D}(par)$ *and* $\mathcal{R} = \mathcal{R}(par)$*, where* $\mathcal{S}$ *is a field, and* $\mathcal{D}$ *and* $\mathcal{R}$ *are* $\mathcal{S}$*-modules. Moreover, we require* $|\mathcal{S}| \geqslant 2^\kappa, |\mathcal{D}| \geqslant 2^\kappa$*, and* $|\mathcal{R}| \geqslant 2^\kappa$*.*

b) F *is a deterministic function that takes as input the system parameter* par *and an element* $x \in \mathcal{D}$ *and returns an element in* $\mathcal{R}$ *such that* $\mathsf{F}(par, \cdot) : \mathcal{D} \to \mathcal{R}$ *is a* epimorphism *of* $\mathcal{S}$*-modules. Moreover,* F *is* not a monomorphism*, which is equivalent to there exists* $z^* \in \mathcal{D}$ *such that* $z^* \neq 0$ *and* $\mathsf{F}(par, z^*) = 0$*. For simplicity, we omit* par *from the input of* F *from now on.*

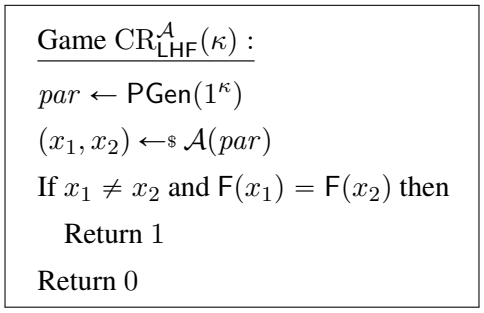

Figure 10.1: The CR security game for a linear hash family $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$.

COLLISION RESISTANCE. Collision resistance of linear hash functions is analogous to collision resistance of cryptographic hash functions, which ensures that it is hard to find two distinct inputs that map to the same output. The $\mathrm{CR}^{\mathcal{A}}_{\mathsf{LHF}}$ game is defined in Figure 10.1. The corresponding advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{LHF}}(\mathcal{A}, \kappa) := \mathsf{Pr}\left[\mathrm{CR}^{\mathcal{A}}_{\mathsf{LHF}} = 1\right]$.

10.2 Algebraic One-more Preimage Resistance

We introduce the notion of algebraic one-more preimage resistance (AOMPR) for linear hash functions, which is formally defined via the game $\mathrm{AOMPR}^{\mathcal{A}}_{\mathsf{LHF}}$, as described in Figure 10.2. It guarantees that any adversary given a description of a linear hash function $(\mathcal{S}, \mathcal{D}, \mathcal{R}, \mathsf{F})$ cannot invert $q+1$ challenges $X_1, \ldots, X_{q+1}$, where $X_i = \mathsf{F}(x_i)$ for $x_i \leftarrow\!\!\$\ \mathcal{D}$, by making at most q queries to the PI oracle that, on any input $Y \in \mathcal{R}$ that is an affine combination of the challenges, outputs an element in the preimage of Y. It is syntactically analogous to the algebraic one-more discrete logarithm (AOMDL) problem [105], where the adversary wants to compute the discrete logarithms of $q+1$ random challenges in $\mathbb{G}$ by making at most q queries to the DLOG oracle, which outputs the discrete logarithm of the input Y only when Y is an affine combination of the challenges and the combination is known to the adversary.

The following theorem, our main result on AOMPR, shows that AOMPR of a linear hash function family is implied by its collision resistance.

Theorem 10.2.1. *For any linear hash function family* LHF *and any* AOMPR *adversary* $\mathcal{A}$ *making at most* q *queries to* CHAL*, there exists an adversary* $\mathcal{B}$ *for the* $\mathrm{CR}^{\mathsf{LHF}}$ *game running in a similar*

Game $\mathrm{AOMPR}^{\mathcal{A}}_{\mathsf{LHF}}(\kappa)$:	Oracle $\textsc{Chal}()$:
$par \leftarrow\!\!\$\ \mathsf{PGen}(1^\kappa)$	$\mathrm{cid} \leftarrow \mathrm{cid} + 1$
$\mathrm{cid} \leftarrow 0\ ;\ \ell \leftarrow 0$	$x_{\mathrm{cid}} \leftarrow\!\!\$\ \mathcal{D}\ ;\ X_{\mathrm{cid}} \leftarrow \mathsf{F}(x_{\mathrm{cid}})$
$\{y_i\}_{i\in[\mathrm{cid}]} \leftarrow \mathcal{A}^{\textsc{Chal},\mathrm{PI}}(par)$	Return X_{cid}
If $\ell \geqslant \mathrm{cid}$ then return 0	Oracle $\mathrm{PI}(Y, \alpha, \{\beta_i\}_{i\in[\mathrm{cid}]})$:
If $\forall\ i \in [\mathrm{cid}]\ \mathsf{F}(y_i) = X_i$ then	Require: $Y = \mathsf{F}(\alpha) + \sum_{i\in[\mathrm{cid}]} \beta_i X_i$
Return 1	$\ell \leftarrow \ell + 1$
Return 0	Return $\alpha + \sum_{i\in[\mathrm{cid}]} \beta_i x_i$

Figure 10.2: The AOMPR game for a linear hash function family $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$. For the inputs of PI, X is in $\mathcal{R}$, α is in $\mathcal{D}$, and each β_i is in $\mathcal{S}$.

---

running time as $\mathcal{A}$ such that $\mathsf{Adv}^{\mathrm{aompr}}_{\mathsf{LHF}}(\mathcal{A}, \kappa) \leqslant 2\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{LHF}}(\mathcal{B}, \kappa)$.

Proof of Theorem 10.2.1. Given an adversary $\mathcal{A}$ for the $\mathrm{AOMPR}^{\mathsf{LHF}}$ game, without loss of generality, we assume that $\mathcal{A}$ is deterministic, queries $\textsc{Chal}$ exactly q times, and queries PI exactly $\mathsf{q}-1$ times. The construction of $\mathcal{B}$ is straightforward. After receiving par from the $\mathrm{CR}^{\mathsf{LHF}}$ game, $\mathcal{B}$ runs $\mathcal{A}$ on input par by simulating the oracles $\textsc{Chal}$ and PI exactly the same as in the $\mathrm{AOMPR}^{\mathsf{LHF}}$ game. After $\mathcal{A}$ outputs $\{y_i\}_{i\in[\mathsf{q}]}$, if

$$\exists\ i \in [\mathsf{q}] \text{ such that } \mathsf{F}(y_i) = X_i \text{ and } y_i \neq x_i, \tag{10.1}$$

where x_i and X_i are generated in the oracle $\textsc{Chal}$, then $\mathcal{B}$ outputs (x_i, y_i). Otherwise, $\mathcal{B}$ aborts.

ANALYSIS OF $\mathcal{B}$. Denote the event $\mathsf{WIN}_{\mathcal{B}}$ as after $\mathcal{A}$ returns, the condition (10.1) holds. If $\mathsf{WIN}_{\mathcal{B}}$ occurs, $\mathcal{B}$ wins the $\mathrm{CR}^{\mathsf{LHF}}$ game since $\mathsf{F}(x_i) = X_i = \mathsf{F}(y_i)$, which implies $\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{LHF}}(\mathcal{B}, \kappa) = \mathsf{Pr}[\mathsf{WIN}_{\mathcal{B}}]$.

It is left to show that $\mathsf{Pr}[\mathsf{WIN}_{\mathcal{B}}] \geqslant \frac{1}{2}\mathsf{Adv}^{\mathrm{aompr}}_{\mathsf{LHF}}(\mathcal{A}, \kappa)$. Since $\mathcal{A}$ is deterministic, the execution of $\mathcal{A}$ is fixed given the pair $(par, \boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{D}^{\mathsf{q}}$ denotes the randomness generated in the oracle $\textsc{Chal}$. Denote the event $\mathsf{WIN}_{\mathcal{A}}$ as $\mathcal{A}$ wins the $\mathrm{AOMPR}^{\mathsf{LHF}}$ game simulated by $\mathcal{B}$. Since $\mathcal{B}$ simulate the game perfectly, we know $\mathsf{Pr}[\mathsf{WIN}_{\mathcal{A}}] = \mathsf{Adv}^{\mathrm{aompr}}_{\mathsf{LHF}}(\mathcal{A}, \kappa)$. For each par, denote

$$\mathcal{W}_{\mathcal{A}} := \{\boldsymbol{x} \mid \mathsf{WIN}_{\mathcal{A}} \text{ occurs given } (par, \boldsymbol{x})\} ,$$

$$\mathcal{W}_{\mathcal{B}} := \{\boldsymbol{x} \mid \mathsf{WIN}_{\mathcal{B}} \text{ occurs given } (par, \boldsymbol{x})\} .$$

Claim 10.2.2. *For each par, there exists a* bijection $\Phi : \mathcal{W}_{\mathcal{A}} \to \mathcal{W}_{\mathcal{A}}$ *such that for any $\boldsymbol{x} \in \mathcal{W}_{\mathcal{A}}$, we have $\boldsymbol{x} \in \mathcal{W}_{\mathcal{B}} \ \vee \ \Phi(\boldsymbol{x}) \in \mathcal{W}_{\mathcal{B}}$.*

From the above claim, we can conclude the proof since

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{WIN}_{\mathcal{B}}\right] &= \mathsf{Pr}[\boldsymbol{x} \in \mathcal{W}_{\mathcal{B}}] = \frac{1}{2}\left(\mathsf{Pr}[\boldsymbol{x} \in \mathcal{W}_{\mathcal{B}}] + \mathsf{Pr}[\Phi(\boldsymbol{x}) \in \mathcal{W}_{\mathcal{B}}]\right) \\ &\geqslant \frac{1}{2}\mathsf{Pr}[\boldsymbol{x} \in \mathcal{W}_{\mathcal{B}} \ \vee \ \Phi(\boldsymbol{x}) \in \mathcal{W}_{\mathcal{B}}] \geqslant \frac{1}{2}\mathsf{Pr}[\boldsymbol{x} \in \mathcal{W}_{\mathcal{A}}] \\ &= \frac{1}{2}\mathsf{Pr}[\mathsf{WIN}_{\mathcal{A}}] = \frac{1}{2}\mathsf{Adv}^{\mathrm{aompr}}_{\mathsf{LHF}}(\mathcal{A}, \kappa) \ .\end{aligned}$$

□

Proof of Claim 10.2.2. We construct Φ as follows. For each $\boldsymbol{x} \in \mathcal{W}_{\mathcal{A}}$, consider the execution of $\mathcal{A}$ given $(par, \boldsymbol{x})$. Denote $B \in \mathcal{S}^{(\mathsf{q}-1)\times\mathsf{q}}$ as the query matrix of the execution, which is defined as follows.

Definition 10.2.3. *Given an execution of an adversary $\mathcal{A}$ for the* AOMPR *game, where $\mathcal{A}$ makes q queries to* CHAL *and ℓ queries to* PI*, define the* query matrix *of the execution as $B \in \mathcal{S}^{\ell\times q}$ such that*

$$B_{i,j} = \begin{cases} \beta_i^{(j)} \ , & i \in [\mathrm{cid}^{(j)}] \\ 0 \ , & o.w. \end{cases} \ ,$$

where $\beta_i^{(j)}$ and $\mathrm{cid}^{(j)}$ are the values of β_i and cid *when $\mathcal{A}$ makes the j-th query to* PI*.*

We now define

$$\Phi(\boldsymbol{x}) := \boldsymbol{x} + \boldsymbol{u}^{(B)} z^* \ ,$$

where $z^* \in \mathcal{D}$ and $\boldsymbol{u}^{(B)} \in \mathcal{S}^{\mathsf{q}}$ are defined in the following claim.

Claim 10.2.4. *There exists $z^* \in \mathcal{D}$ such that $\mathsf{F}(z^*) = 0$ and for any matrix $A \in \mathcal{S}^{\ell\times q}$ where $0 < \ell < q$, there exists a vector $\boldsymbol{u}^{(A)} \in \mathcal{S}^q$ and $i \in [q]$ such that*

$$A\boldsymbol{u}^{(A)} = 0 \ \wedge \ \exists\, i \in [\mathsf{q}] \ : \ u_i^{(A)} z^* \neq 0 \ . \tag{10.2}$$

Proof of Claim 10.2.4. Since F is not a monomorphism from $\mathcal{D}$ to $\mathcal{R}$, there exists a non-zero element $z^* \in \mathcal{D}$ such that $\mathsf{F}(z^*) = 0$. Since $\mathcal{S}$ is a field and A has rank at most $\ell < q$, there exists a non-zero vector $\boldsymbol{u}^{(A)} \in \mathcal{S}^q$ such that $A\boldsymbol{u}^{(A)} = 0$. Also, since $\boldsymbol{u}^{(A)}$ is non-zero, there exists $i \in [q]$ such that $u_i^{(A)} \neq 0$, and since $\mathcal{S}$ is a field and $z^* \neq 0$, we have $u_i^{(A)} z^* \neq 0$. □

ANALYSIS OF Φ. For simplicity, we use $\boldsymbol{u}$ to denote $\boldsymbol{u}^{(B)}$ in the following analysis. We first show that the executions of $\mathcal{A}$ given $(par, \boldsymbol{x})$ and given $(par, \Phi(\boldsymbol{x}))$ are identical. Since $\mathsf{F}(\Phi(\boldsymbol{x})) = \mathsf{F}(\boldsymbol{x}) + \boldsymbol{u} \cdot \mathsf{F}(z^*) = \mathsf{F}(\boldsymbol{x}) + \boldsymbol{u} \cdot 0 = \mathsf{F}(\boldsymbol{x})$, the challenges output by CHAL are the same in the two executions. For the j-th query to PI, suppose the prior views of $\mathcal{A}$ are identical. Then, $\mathcal{A}$ must make the same query $\left(X^{(j)}, \alpha^{(j)}, \{\beta_i^{(j)}\}_{i \in [\mathrm{cid}^{(j)}]}\right)$ in both executions. Since $B\boldsymbol{u} = 0$, we have $\alpha^{(j)} + \sum_{i \in [\mathrm{cid}^{(j)}]} \beta_i^{(j)} x_i = \alpha^{(j)} + \left(\boldsymbol{\beta}^{(j)}\right)^T \boldsymbol{x} = \alpha^{(j)} + \left(\boldsymbol{\beta}^{(j)}\right)^T (\boldsymbol{x} + \boldsymbol{u}z^*) = \alpha^{(j)} + \sum_{i \in [\mathrm{cid}^{(j)}]} \beta_i^{(j)} \left(\Phi(\boldsymbol{x})\right)_i$, where $\boldsymbol{\beta}^{(j)}$ denotes the j-th row of B. Therefore, $\mathcal{A}$ receives the same value from PI in both executions. By induction, the views of $\mathcal{A}$ are identical in both executions and thus $\mathcal{A}$ outputs the same values in both executions, which implies $\Phi(\boldsymbol{x}) \in \mathcal{W}_{\mathcal{A}}$ and thus Φ is a map from $\mathcal{W}_{\mathcal{A}}$ to $\mathcal{W}_{\mathcal{A}}$.

Then, it is not hard to see that $\boldsymbol{x} \in \mathcal{W}_{\mathcal{B}} \ \vee \ \Phi(\boldsymbol{x}) \in \mathcal{W}_{\mathcal{B}}$. Since the executions of $\mathcal{A}$ given $\boldsymbol{x}$ and $\Phi(\boldsymbol{x})$ are identical, the outputs $y_1, \ldots, y_{\mathsf{q}}$ of $\mathcal{A}$ are also identical in the two executions. Since there exists $i \in [\mathsf{q}]$ such that $u_i z^* \neq 0$, we have either $y_i \neq x_i$ or $y_i \neq x_i + u_i \cdot z^*$, which means $\mathsf{WIN}_{\mathcal{B}}$ occurs either in the execution given $\boldsymbol{x}$ or $\Phi(\boldsymbol{x})$.

It is left to show that Φ is a bijection. Since both the domain and range of Φ are $\mathcal{W}_{\mathcal{A}}$, which is a finite set, it is enough to show that Φ is an injection. For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{W}_{\mathcal{A}}$ such that $\Phi(\boldsymbol{x}_1) = \Phi(\boldsymbol{x}_2)$, since the execution of $\mathcal{A}$ given $\boldsymbol{x}_1$ is identical to that given $\Phi(\boldsymbol{x}_1)$ and the execution of $\mathcal{A}$ given $\boldsymbol{x}_2$ is identical to that given $\Phi(\boldsymbol{x}_2)$, we know the executions of $\mathcal{A}$ given $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are identical, which implies the query matrix B in the two executions are identical. Therefore, we have $\Phi(\boldsymbol{x}_1) = \boldsymbol{x}_1 + \boldsymbol{u}z^*$ and $\Phi(\boldsymbol{x}_2) = \boldsymbol{x}_2 + \boldsymbol{u}z^*$ for the same $\boldsymbol{u} \in \mathcal{S}^{\mathsf{q}}$, which implies $\boldsymbol{x}_1 = \boldsymbol{x}_2$. This shows that Φ is an injection. □

10.3 Instantiations From the Discrete Logarithm Problem

Following the instantiation from [80], a linear hash function family GLHF is instantiated from a group generation algorithm GGen as follows.

- On input 1^κ, PGen runs $\mathsf{GGen}(1^\kappa)$ and receives a group description $(\mathbb{G}, p, g)$. Then, PGen uniformly samples $Z \in \mathbb{G}$ and returns $\kappa \leftarrow (\mathbb{G}, p, g, Z)$.

- Given $\kappa = (\mathbb{G}, p, g, Z)$, define $\mathcal{S} := \mathbb{Z}_p$, $\mathcal{D} := \mathbb{Z}_p^2$, $\mathcal{R} := \mathbb{G}$. Also, for any $(x_1, x_2) \in \mathbb{Z}_p^2$, define $\mathsf{F}(x_1, x_2) := g^{x_1} Z^{x_2}$.

- The operation over $\mathcal{D}$ is defined as follows. For any $(x_1, y_1), (x_2, y_2) \in \mathcal{D}$ and $s \in \mathcal{S}$, $(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$ and $s \cdot (x_1, y_1) = (sx_1, sy_1)$.

- The operation over $\mathcal{R}$ is defined as follows. For any $x_1, x_2 \in \mathcal{R}$ and $s \in \mathcal{S}$, $x_1 + x_2 = x_1 x_2$, $s \cdot x_1 = x_1^s$, where $x_1 x_2$ and x_1^s are the group operations of $\mathbb{G}$.

The following theorem shows that GLHF is a linear hash function family and collision resistance of GLHF is implied by the discrete logarithm assumption. Hauck et al. [80] shows similar statements, and we also give the proof below for completeness.

Lemma 10.3.1. *For any group generation algorithm* GGen, $\mathsf{GLHF}[\mathsf{GGen}]$ *is a linear hash function family (Definition 10.1.1). Moreover, for any adversary* $\mathcal{A}$ *for the* $\mathrm{CR}^{\mathsf{GLHF}[\mathsf{GGen}]}$ *game, there exists an adversary* $\mathcal{B}$ *for the* $\mathrm{DLog}^{\mathsf{GGen}}$ *game such that* $\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{GLHF}[\mathsf{GGen}]}(\mathcal{A}, \kappa) \leqslant \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GGen}}(\mathcal{B}, \kappa)$.

Proof of Lemma 10.3.1. Given $par = (\mathbb{G}, p, g, Z)$ output from $\mathsf{PGen}(1^\kappa)$ that defines $(\mathcal{S}, \mathcal{D}, \mathcal{R}, \mathsf{F})$, we need to show

1. $\mathcal{D}$ and $\mathcal{R}$ are $\mathcal{S}$-modules.

2. F is an epimorphism from $\mathcal{D}$ to $\mathcal{R}$ but not a monomorphism.

3. The collision resistance of RLHF is implied by the RSA assumption.

PART 1. It is clear that $\mathcal{D} = \mathbb{G} \times \mathbb{G}$ and $\mathcal{R} = \mathbb{G}$ are $\mathbb{Z}_p$-modules.

PART 2. It is easy to verify F is a homomorphism of $\mathcal{S}$-modules, since for any $b \in \mathcal{S}$ and $(x_1, y_1), (x_2, y_2) \in \mathcal{D}$,

$$\begin{aligned}
\mathsf{F}((x_1, y_1) + b \cdot (x_2, y_2)) &= \mathsf{F}(x_1 + bx_1, y_1 + by_2) \\
&= g^{x_1 + bx_2} Z^{y_1 + by_2} \\
&= g^{x_1} Z^{x_1} (g^{x_2} Z^{y_2})^b \\
&= \mathsf{F}(x_1, y_1) + b\mathsf{F}(x_2, y_2) \ .
\end{aligned}$$

F is epimorphism since for any $X \in \mathcal{R}$, we have $\mathsf{F}(x, 0) = X$, where x denotes the discrete log of X to the base g. Denote z as the discrete log of Z to base g, and F is not a monomorphism, since $\mathsf{F}(z, -1) = g^z Z^{-1} = g^0$.

PART 3. For any adversary $\mathcal{A}$ for the $\mathrm{CR}^{\mathsf{GLHF}}$ game, we construct $\mathcal{B}$ for the DLog game as follows. After receiving $(\mathbb{G}, p, g, Z)$, $\mathcal{B}$ runs $\mathcal{A}$ with input $(\mathbb{G}, p, g, Z)$. If $\mathcal{A}$ wins the $\mathrm{CR}^{\mathsf{GLHF}}$ by outputting $(x_1, y_1), (x_2, y_2) \in \mathcal{D}$, such that $(x_1, y_1) \neq (x_2, y_2)$ and $\mathsf{F}(x_1, y_1) = \mathsf{F}(x_2, y_2)$, we have $g^{x_1} Z^{y_1} = g^{x_2} Z^{y_2}$. Therefore, $\mathcal{B}$ can compute $z = \frac{y_2 - y_1}{x_1 - x_2}$ and we have $g^z = Z$. □

10.4 Threshold Signatures from LHF

OUR SCHEMES. Figure 10.3 shows the protocols FROST1-H and FROST2-H that are transformed from FROST1 and FROST2, respectively. In addition to the general transformation, we need to pick an injection $\mathsf{x}_{(\cdot)} : [n] \to \mathcal{S}$. The choice of $\mathsf{x}_{(\cdot)}$ can be arbitrary, and the corresponding Lagrange coefficient for a set of index $S \subseteq [n]$ and $i \in S$ is defined as $\lambda_i^S := \prod_{j \in S \setminus \{i\}} \frac{\mathsf{x}_j}{\mathsf{x}_i - \mathsf{x}_j}$. We analyse the correctness of the scheme in Section 10.4.1 Also, similar to the multi-signature case, we optimize the schemes by sampling key shares from $\mathcal{D}_{\text{key}} \subseteq \mathcal{D}$ and setting the hash range to be $\mathcal{S}_{\text{hash}} \subseteq \mathcal{S}$.

The following theorems show that, under the AOMPR assumption, FROST2-H is TS-SUF-2-secure and FROST1-H is TS-SUF-3-secure in the random oracle model. We prove the theorems using the same techniques from Chapter 9. Here, we briefly highlight the differences:

- We need to show that $\mathcal{B}$ simulates the TS-SUF-2 and TS-SUF-3 games perfectly when no bad event occurs and that the bad events occur with a negligible probability when the secret key is sampled from $\mathcal{D}_{\text{key}}$ instead of $\mathbb{Z}_p$, and the randomness r_j is sampled from $\mathcal{D}$ instead of $\mathbb{Z}_p$.

- We need to show that $\mathcal{B}$ can compute a preimage for each challenge (Claim 10.4.5 and Claim 10.4.8) instead of the discrete logarithm to the base element. More precisely, the problem can be described as follows. Denote the challenges by $U_1, \ldots, U_\ell \in \mathcal{R}$. After the interaction with $\mathcal{A}$, $\mathcal{B}$ computes a matrix $A \in \mathcal{S}^{\ell \times \ell}$ and a vector $\boldsymbol{b} \in \mathcal{D}^\ell$ such that $A \cdot \boldsymbol{U} = \mathsf{F}(\boldsymbol{b})$, we need to show that A has full rank and thus $\mathcal{B}$ can compute a vector $\boldsymbol{u} = A^{-1}\boldsymbol{b}$ such that $\mathsf{F}(\boldsymbol{u}) = \boldsymbol{U}$.

Theorem 10.4.1. *For any* TS-SUF-2 *adversary* $\mathcal{A}$ *game making at most* q_s *queries to* PPO *and* q_h *queries to* RO*, there exists an* AOMPR *adversary* $\mathcal{B}$ *making at most* $2\mathsf{q}_s + t$ *queries to* CHAL *running in time roughly equal two times that of* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\text{ts-suf-2}}_{\mathsf{FROST2\text{-}H[LHF]}}(\mathcal{A}, \kappa) \leqslant \sqrt{\mathsf{q} \cdot (\mathsf{Adv}^{\text{aompr}}_{\mathsf{LHF}}(\mathcal{B}, \kappa) + (5\mathsf{q}^2)/2^\kappa)} \,,$$

where $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$.

Theorem 10.4.2. *For any* TS-SUF-3 *adversary* $\mathcal{A}$ *making at most* q_s *queries to* PPO *and* q_h *queries to* RO*, there exists an* AOMPR *adversary* $\mathcal{B}$ *making at most* $2\mathsf{q}_s + t$ *queries to* CHAL *running in time roughly equal two times that of* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\text{ts-suf-3}}_{\mathsf{FROST1\text{-}H[LHF]}}(\mathcal{A}, \kappa) \leqslant 4n \cdot \mathsf{q} \cdot \sqrt{(\mathsf{Adv}^{\text{aompr}}_{\mathsf{LHF}}(\mathcal{B}, \kappa) + 6\mathsf{q}/2^\kappa)} \,,$$

Algorithm $\mathsf{Setup}(1^\kappa)$:

$par \leftarrow\!\!\$\ \mathsf{PGen}(1^\kappa)$

Return par

Algorithm $\mathsf{KeyGen}()$:

For $i \in [0..t-1]$ do

  $a_i \leftarrow\!\!\$\ \mathcal{D}_{\text{key}}$

For $i \in [n]$ do

  $\mathsf{sk}_i \leftarrow\!\!\$\ \sum_{j=0}^{t-1} a_j \cdot \mathsf{x}_i^j$

  $\mathsf{pk}_i \leftarrow \mathsf{F}(\mathsf{sk}_i)$

$\mathsf{pk} \leftarrow \mathsf{F}(a_0)$

$\mathsf{aux} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

Return $\mathsf{pk}, \mathsf{aux}, \{\mathsf{sk}_i\}_{i\in[1..n]}$

Algorithm $\mathsf{SPP}(\mathsf{st}_i)$:

$r \leftarrow\!\!\$\ \mathcal{D}$; $s \leftarrow\!\!\$\ \mathcal{D}$

$pp \leftarrow (\mathsf{F}(r), \mathsf{F}(s))$

$\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (r, s)$

Return (pp, st_i)

Algorithm $\mathsf{Vf}(\mathsf{pk}, m, sig)$:

$(R, s) \leftarrow sig$

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, m, R)$

Return $(\mathsf{F}(s) = R + c \cdot \mathsf{pk})$

Algorithm $\mathsf{LPP}(i, pp, \mathsf{st}_0)$:

$\mathsf{st}_0.\mathrm{curPP}_i \leftarrow \mathsf{st}_0.\mathrm{curPP}_i \cup \{pp\}$

Return st_0

Algorithm $\mathsf{LR}(M, SS, \mathsf{st}_0)$:

If $\exists\ i \in SS : \mathsf{st}_0.\mathrm{curPP}_i = \varnothing$ then

  Return $\perp$

$lr.\mathsf{msg} \leftarrow M$; $lr.\mathsf{SS} \leftarrow SS$

For $i \in SS$ do

  Pick pp_i from $\mathsf{st}_0.\mathrm{curPP}_i$

  $lr.\mathsf{PP}(i) \leftarrow pp_i$

  $\mathsf{st}_0.\mathrm{curPP}_i \leftarrow \mathsf{st}_0.\mathrm{curPP}_i \backslash \{pp_i\}$

Return (lr, st_0)

Algorithm $\mathsf{PS}(lr, i, \mathsf{st}_i)$:

$pp_i \leftarrow lr.\mathsf{PP}(i)$

If $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \perp$ then

  Return $(\perp, \mathsf{st}_i)$

$(r_i, s_i) \leftarrow \mathsf{st}_i.\mathrm{mapPP}(pp_i)$

$\mathsf{st}_i.\mathrm{mapPP}(pp_i) \leftarrow \perp$

$(R, c, \{d_j\}_{j\in lr.\mathsf{SS}})$

  $\leftarrow \mathsf{CompPar}(\mathsf{st}_i.\mathsf{pk}, lr)$

$z_i \leftarrow r_i + d_i \cdot s_i + c \cdot \lambda_i^{lr.\mathsf{SS}} \cdot \mathsf{st}_i.\mathsf{sk}$

Return $((R, z_i), \mathsf{st}_i)$

Algorithm $\mathsf{CompPar}(\mathsf{pk}, lr)$:

$m \leftarrow lr.\mathsf{msg}$; $(R^*, s^*) \leftarrow sig$

For $i \in lr.\mathsf{SS}$ do

  $\boxed{d_i \leftarrow \mathrm{H}_1(\mathsf{pk}, lr, i)}$

  $\boxed{d_i \leftarrow \mathrm{H}_1(\mathsf{pk}, lr)}$

  $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$

$R \leftarrow \sum_{i\in lr.\mathsf{SS}} (R_i + d_i S_i)$

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, M, R)$

Return $(R, c, \{d_i\}_{i\in lr.\mathsf{SS}})$

Algorithm $\mathsf{Agg}(\mathrm{PS}, \mathsf{st}_0)$:

$R \leftarrow \perp$; $z \leftarrow 0$

For $(R', z') \in \mathrm{PS}$ do

  If $R = \perp$ then $R \leftarrow R'$

  If $R \neq R'$ then return $(\perp, \mathsf{st}_0)$

  $z \leftarrow z + z'$

Return $((R, z), \mathsf{st}_0)$

Algorithm $\mathsf{SVf}(\mathsf{pk}, lr, sig)$:

$(R^*, z^*) \leftarrow sig$

$(R, c, \{d_j\}_{j\in lr.\mathsf{SS}})$

  $\leftarrow \mathsf{CompPar}(\mathsf{st}_i.\mathsf{pk}, lr)$

Return $(R = R^*)$

  $\wedge\ (\mathsf{F}(z^*) = R + c \cdot \mathsf{pk})$

Figure 10.3: The protocol FROST1-H[LHF] and FROST1-H[LHF], where $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$ is a linear hash function family. The protocol FROST1-H contains all but the dashed box, and the protocol FROST2-H contains all but the solid box. Further, n is the number of parties, and t is the threshold of the schemes. $\mathsf{x}_{(\cdot)}$ is an injection from $[n]$ to $\mathcal{S}$ and $\lambda_i^{lr.\mathsf{SS}}$ denotes the Lagrange coefficient which is computed as $\lambda_i^{lr.\mathsf{SS}} := \prod_{j\in S\backslash\{i\}} \frac{\mathsf{x}_j}{\mathsf{x}_j - \mathsf{x}_i}$. $\mathcal{D}_{\text{key}}$ is a subset of $\mathcal{D}$ such that F is a bijection between $\mathcal{D}_{\text{key}}$ and $\mathcal{S}$.

where $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$.

To instantiate FROST1-H and FROST2-H in pairing-free groups, we set $\mathcal{D}_{\text{key}} := \{(x, 0) : x \in \mathbb{Z}\}$ and $\mathcal{S}_{\text{hash}} := \mathcal{S}$. It is clear that $\text{char}(\mathcal{S}) = p \geqslant 2^\kappa$, F is a bijection from $\mathcal{D}_{\text{key}}$ to $\mathcal{R}$, and $|\mathcal{S}_{\text{hash}}| = |\mathcal{S}| \geqslant 2^\kappa$. Also, for instantiating FROST1-H and FROST2-H, we set $\mathsf{x}_i := i$.

By combining Theorem 10.2.1 and Lemma 10.3.1 with the above theorems, we show the security of FROST1-H and FROST2-H instantiated from GLHF (defined in Section 10.3) under the discrete logarithm assumption in the random oracle model.

10.4.1 Correctness of FROST1-H *and* FROST2-H

For the correctness of the schemes, we just need to show Shamir's secret sharing scheme works over $\mathcal{D}$. More precisely, we want to show for any $S \subseteq [n]$ with size t, we have

$$\sum_{i \in S} \lambda_i^S \mathsf{sk}_i = a_0 \ , \tag{10.3}$$

where

$$\mathsf{sk}_i = \sum_{j=0}^{t-1} a_j \mathsf{x}_i^j \ , \ \lambda_i^S = \prod_{j \in S \setminus \{i\}} \frac{\mathsf{x}_j}{\mathsf{x}_j - \mathsf{x}_i},$$

for each $i \in S$ and some $a_0, \ldots, a_{t-1} \in \mathcal{D}$.

Claim 10.4.3. *For any* $S \subseteq [n]$ *with size* t *and any integer* $0 \leqslant k < t$*, we have*

$$\sum_{i \in S} \lambda_i^S \mathsf{x}_i^k = \begin{cases} 1, & k = 0 \\ 0, & k \geqslant 0 \end{cases} . \tag{10.4}$$

With the above claim, we can show (10.3) since

$$\begin{aligned} \sum_{i \in S} \lambda_i^S \mathsf{sk}_i &= \sum_{i \in S} \lambda_i^S \sum_{j=0}^{t-1} a_j \mathsf{x}_i^j = \sum_{j=0}^{t-1} a_j \sum_{i \in S} \lambda_i^S \mathsf{x}_i^j \\ &= \sum_{j=0}^{t-1} a_j \cdot \mathbf{1}\{j = 0\} = a_0 \ . \end{aligned}$$

Proof of Claim 10.4.3. Define $f(x) = x^k$. By the polynomial interpolation over the field $\mathcal{S}$, we have $\sum_{i \in S} \lambda_i^S f(\mathsf{x}_i) = f(0)$, which proves claim. □

10.4.2 *Proof of Theorem 10.4.1*

Let $\mathcal{A}$ be an adversary as described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(m^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\mathrm{H}_2(\mathsf{pk}, m^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\mathrm{H}_1(\mathsf{pk}, lr)$ prior to the query $\textsc{PSign}\mathrm{O}(i, lr)$ for some i and lr. (This adds up to q_s additional RO queries, and we let $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$.) Denote lr^* as the leader query such that $\mathrm{H}_1(\mathsf{pk}, lr^*)$ is the first query prior to the query $\mathrm{H}_2(\mathsf{pk}, m^*, R^*)$ satisfying $\mathsf{SVf}(\mathsf{pk}, lr^*, sig^*) = \mathtt{true}$. If such lr^* does not exists, lr^* is set to $\perp$. Denote the event E_1 as

$$\mathsf{Vf}(\mathsf{pk}, m^*, sig^*) \ \wedge \ (lr^* = \perp \ \vee \ \mathrm{curSS}_{\mathsf{L}}(lr^*) < t - |CS|) \ .$$

It is clear that if $\mathcal{A}$ wins the game $\text{TS-SUF-2}^{\mathsf{FROST2\text{-}H[LHF]}}$, then E_1 must occur, which implies $\mathsf{Pr}[E_1] \geqslant \mathsf{Adv}^{\text{ts-suf-2}}_{\mathsf{FROST2\text{-}H[LHF]}}(\mathcal{A})$. Therefore, the theorem will follow from the following lemma. (We isolate this statement as its own lemma also because it will be helpful in the proof of Theorem 10.4.2 below.)

Lemma 10.4.4. *There exists an adversary $\mathcal{B}$ for the* $\mathrm{AOMPR}^{\mathsf{LHF}}$ *game making at most* $2\mathsf{q}_s + t$ *queries to* CHAL *such that*

$$\mathsf{Pr}[E_1] \leqslant \sqrt{\mathsf{q} \cdot (\mathsf{Adv}^{\text{aompr}}_{\mathsf{LHF}}(\mathcal{B}) + 5\mathsf{q}^2/2^{\kappa})} \ .$$

Moreover, $\mathcal{B}$ runs in time roughly twice that of $\mathcal{A}$.

Proof of Lemma 10.4.4. We first construct an algorithm $\mathcal{C}$ compatible with the syntax in Lemma 9.2.3 and then construct $\mathcal{B}$ from $\mathsf{Fork}^{\mathcal{C}}$. The input of $\mathcal{C}$ consists of par that defines a linear hash function $(\mathcal{S}, \mathcal{D}, \mathcal{R}, \mathsf{F})$ and uniformly random elements $h_1, \ldots, h_{2\mathsf{q}} \in \mathcal{S}_{\text{hash}}$. Also, $\mathcal{C}$ can access oracles CHAL and PI defined the same as those in the $\mathrm{AOMPR}^{\mathsf{LHF}}$ game. (We can think of the oracles as part of the input of $\mathcal{C}$ in the context of the Forking Lemma.) For simplicity, when $\mathcal{C}$ makes a query $(X, \alpha, \{\beta_i\})$ to PI, we omit the coefficients $\alpha, \{\beta_i\}$ which are clear from the context. To start with, $\mathcal{C}$ makes $2\mathsf{q}_s + t$ queries to CHAL and denotes the challenges as $A_0, \ldots, A_{t-1}$, U_1, V_1, $\ldots$, U_{q_s}, $V_{\mathsf{q}_s} \in \mathcal{D}$. Then, $\mathcal{C}$ initializes all the states $\mathsf{st}_0, \ldots, \mathsf{st}_n$. In addition, it initializes counters ctr_s,ctr_h to 0 and a function dt to an empty table, which are used to record the PI query related to each (U_j, V_j). $\mathcal{C}$ also initializes $\mathrm{curLR} \leftarrow \varnothing$ to record all leader requests that appears during the game and initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. We also use a flag BadPPO to denote whether a

bad event occurs, which are initially set to `false`. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$, $\widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: $\mathcal{C}$ initializes H to an empty table and sets $\overline{\mathsf{pk}} \leftarrow A_0$, $\overline{\mathsf{pk}}_i = \prod_{j=0}^{t-1} A_j \cdot \mathsf{x}_i^j$ for $i \in [n]$, and $\overline{\mathsf{sk}}_i \leftarrow \text{PI}(\overline{\mathsf{pk}}_i)$ for $i \in CS$. $\mathcal{C}$ samples $\tilde{a}_i \leftarrow\!\!\$\ \mathcal{D}_{\text{key}}$ for $i \in [0..(t-1)]$ and sets $\mathsf{sk}_i \leftarrow \sum_{j=0}^{t-1} \tilde{a}_j \mathsf{x}_i^j$ for $i \in CS$. Then, $\mathcal{C}$ computes a polynomial $f(x) = \sum_{i=0}^{t-1} \mu_i x^i$ such that $\mu_i \in \mathcal{D}$ for $i \in [0..(t-1)]$, $f(\mathsf{x}_i) = \mathsf{sk}_i - \overline{\mathsf{sk}}_i$ for $i \in CS$, and $f(\mathsf{x}_i) = 0$ for $i \in S'$, where $S' \subseteq [n]$ denotes the set of the first $(t - |CS|)$ honest parties. [1] $\mathcal{C}$ sets $\mathsf{pk}_i \leftarrow \overline{\mathsf{pk}}_i + \mathsf{F}(f(\mathsf{x}_i))$ for $i \in [n]$. Finally, $\mathcal{C}$ returns (pk,$\mathsf{aux} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, $\{\mathsf{sk}_i\}_{i \in CS}$).

$\widetilde{\text{RO}}$ query $\text{H}_1(x)$: If $\text{H}_1(x) \neq \bot$, $\mathcal{C}$ returns $\text{H}_1(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, lr)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\text{H}_1(x) \leftarrow\!\!\$\ \mathcal{S}_{\text{hash}}$ and returns $\text{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1, sets $\text{H}_1(x) \leftarrow h_{2\text{ctr}_h - 1}$, and adds lr to curLR. Also, $\mathcal{C}$ computes $R \leftarrow \sum_{i \in lr.\mathsf{SS}} (R_i + h_{2\text{ctr}_h - 1} \cdot S_i)$, where $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$. If $\text{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = \bot$, $\mathcal{C}$ sets $\text{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = h_{2\text{ctr}_h}$. In addition, define $\text{mapLR}(\text{ctr}_h) := lr$ and set $\text{curLR} \leftarrow \text{curLR} \cup \{lr\}$. Finally, $\mathcal{C}$ returns $\text{H}_1(x)$.

$\widetilde{\text{RO}}$ query $\text{H}_2(x)$: If $\text{H}_2(x) \neq \bot$, $\mathcal{C}$ returns $\text{H}_2(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, m, R)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\text{H}_2(x) \leftarrow\!\!\$\ \mathcal{S}_{\text{hash}}$ and returns $\text{H}_2(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\text{H}_2(x) \leftarrow h_{2\text{ctr}_h}$. Finally, $\mathcal{C}$ returns $\text{H}_2(x)$.

$\widetilde{\text{PPO}}(i)$ query: Same as in the game TS-SUF-2$^{\mathsf{FROST2\text{-}H}}$, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\text{ctr}_s}, V_{\text{ctr}_s})$, $\mathsf{st}_i.\text{mapPP}(pp) \leftarrow (0, 0)$, and $\text{ctrPP}(i, pp) \leftarrow \text{ctr}_s$. In addition, BadPPO is set to `true` if there exists $lr \in \text{curLR}$ such that $lr.\mathsf{PP}(i) = (U_{\text{ctr}_s}, V_{\text{ctr}_s})$.

$\widetilde{\text{PSIGNO}}(i, lr)$ query: Same as in the game TS-SUF-2$^{\mathsf{FROST2\text{-}H}}$, except in the simulation of algorithm PS, if $\mathsf{st}_i.\text{mapPP}(pp) \neq \bot$, $\mathcal{C}$ sets

$$z_i \leftarrow \text{PI}\left(U_j + d_i V_j + c \cdot \lambda_i^{lr.\mathsf{SS}} \cdot \mathsf{pk}_i\right) ,$$

where $j \leftarrow \text{ctrPP}(i, lr.\mathsf{PP}(i))$. In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (i, k, d_i, c\lambda_i^{lr.\mathsf{SS}}, z_i)$, where k denotes the index such that $\text{H}_1(\mathsf{pk}, lr)$ is set to h_{2k-1} during the simulation.

[1] Since the degree of f is t and we fix t points of f, f is fixed and we can compute the coefficients of f by solving a linear equation.

After receiving the output $(m^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $\perp$ if $\mathsf{BadPPO} = \texttt{true}$ or E_1 does not occur. Otherwise, $\mathcal{C}$ finds the index I such that $\mathrm{H}_2(\mathsf{pk}, m^*, R^*)$ is set to h_I during the simulation. By our assumption of $\mathcal{A}$, we know such I must exist. Then, $\mathcal{C}$ returns (I, Out), where Out consists of all variables received or generated by $\mathcal{C}$.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.2.3, we define $S := \{2k\}_{k\in[1..\mathsf{q}]}$ and IG as the algorithm that runs $\mathsf{PGen}(1^\kappa)$ and outputs par. From the simulation, we know the output index I of $\mathcal{C}$ is always in S.

It is not hard to see $\mathcal{C}$ simulates the game $\text{TS-SUF-2}^{\mathsf{FROST2\text{-}H}}$ perfectly, which implies $\mathrm{acc}(\mathcal{C}) \geqslant \mathsf{Pr}[E_1] - \mathsf{Pr}[\mathsf{BadPPO}]$, where $\mathsf{Pr}[E_1]$ refers to the probability in the original $\text{TS-SUF-2}^{\mathsf{FROST2\text{-}H[LHF]}}$ game with $\mathcal{A}$ (as in the lemma statement), whereas $\mathsf{Pr}[\mathsf{BadPPO}]$ is the probability that $\mathsf{BadPPO} = \texttt{true}$ at the end of $\mathcal{C}$'s execution. Since (U_j, V_j) is sampled uniformly from $\mathcal{R}$. Therefore, for each $\mathrm{PPO}(i)$ query, the probability BadPPO is set to $\texttt{true}$ is less than $|\mathrm{curLR}|/|\mathcal{R}| \leqslant \mathsf{q}_h/2^\kappa$. Therefore, we have $\mathsf{Pr}[\mathsf{BadPPO}] \leqslant \mathsf{q}_s\mathsf{q}_h/2^\kappa$. By Lemma 9.2.3,

$$\begin{aligned}\mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) &\geqslant (\mathsf{Pr}[E_1] - \mathsf{q}_s\mathsf{q}_h/2^\kappa)^2/\mathsf{q} \geqslant \mathsf{Pr}[E_1]^2/\mathsf{q} - 2\mathsf{Pr}[E_1]\mathsf{q}_s\mathsf{q}_h/(2^\kappa \cdot q)\\ &\geqslant \mathsf{Pr}[E_1]^2/\mathsf{q} - \mathsf{q}/2^\kappa \ .\end{aligned}$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now give a construct of the AOMPR adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$, and the available CHAL and PI oracles. To start with, $\mathcal{B}$ receives par from the $\mathrm{AOMPR}^{\mathsf{LHF}}$ game and runs $\mathsf{Fork}^{\mathcal{C}}(par)$. All the CHAL queries from the first execution of $\mathcal{C}$ are relayed by $\mathcal{B}$ to its own CHAL oracle, and for all the CHAL queries from the second execution of $\mathcal{C}$, $\mathcal{B}$ answers them with the same challenges as the first execution. All the PI queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$ to its own PI oracle. Without loss of generality, we can assume all the challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. Denote the event $\mathsf{BadHash}$ as $\{h_1, \ldots, h_{2\mathsf{q}}\} \cap \{h'_1, \ldots, h'_{2\mathsf{q}}\} \neq \varnothing$, where $h_1, h'_1, \ldots, h_{2\mathsf{q}}, h'_{2\mathsf{q}}$ generated in the execution of $\mathsf{Fork}^{\mathcal{C}}$ are same. Since the hash values are sampled uniformly from $\mathcal{S}_{\mathrm{hash}}$, we know $\mathsf{Pr}[\mathsf{BadHash}] \leqslant 4\mathsf{q}^2/|\mathcal{S}_{\mathrm{hash}}| \leqslant 4\mathsf{q}^2/2^\kappa$. Then, we can conclude the proof with the following claim, which implies

$$\mathsf{Adv}^{\mathrm{aompr}}_{\mathsf{LHF}}(\mathcal{B}) \geqslant \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) - \mathsf{Pr}[\mathsf{BadHash}] \geqslant \mathsf{Pr}[E_1]^2/\mathsf{q} - 5\mathsf{q}^2/2^\kappa \ .$$

□

Claim 10.4.5. *$\mathcal{B}$ can win the game $\mathrm{AOMPR}_{\mathsf{LHF}}$, if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, \mathrm{Out}, \mathrm{Out}')$ and $\mathsf{BadHash}$ does not occur.*

Proof. We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out'. The total number of the CHAL queries is $t + 2\mathsf{q}_s$ and the corresponding challenges are $A_0, \ldots, A_{t-1}, U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$.

We first show how to compute $a_0, \ldots, a_{t-1}$ such that $\mathsf{F}(a_i) = A_i$. By the execution of $\mathsf{Fork}^{\mathcal{C}}$, we know $(\mathsf{pk}, m^*, R^*) = (\mathsf{pk}', m^{*\prime}, R^{*\prime})$ and $\mathsf{pk} = A_0$. Since $I \in S$, let $k^* = I/2$. It is not hard to see that $\mathrm{mapLR}(k^*) = lr^*$. (If $\mathrm{mapLR}(k^*) = \perp$, lr^* is also $\perp$.) Since $\mathsf{BadHash}$ does not occur, we have $\mathrm{H}_2(\mathsf{pk}, m^*, R^*) = h_I \neq h'_I = \mathrm{H}'_2(\mathsf{pk}, m^*, R^*)$. Since $\mathsf{F}(z^*) = R^* + h_I A_0$, $\mathsf{F}(z^{*\prime}) = R^* + h'_I A_0$, we have $\mathsf{F}(z^* - z^{*\prime}) = (h_I - h'_I)A_0$ and therefore $\mathcal{B}$ computes $a_0 \leftarrow \frac{z^* - z^{*\prime}}{h_I - h'_I}$.

Define $T_{\mathsf{dt}} := \{j \ : \ (i, k, d, c, z) \leftarrow \mathsf{dt}(j), k = k^*\}$. For each $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, let $(i, k, d, c, z) \leftarrow \mathsf{dt}(j)$ and $(i', k', d', c', z') \leftarrow \mathsf{dt}'(j)$, and we have $\mathsf{F}(z) = U_j + dV_j + c \cdot \mathsf{pk}_i$, $\mathsf{F}(z') = U_j + d'V_j^{d'} + c' \cdot \mathsf{pk}_{i'}$, $c = h_I \lambda_i^{lr^*.\mathsf{SS}}$, and $c' = h'_I \lambda_{i'}^{lr^*.\mathsf{SS}}$. Since $\mathsf{BadPPO} = \texttt{false}$ during both execution of $\mathcal{C}$, we know (U_j, V_j) is returned by a query $\mathrm{PPO}(i)$ prior to the query $\mathrm{H}_2(\mathsf{pk}, M^*, R^*)$ during the first execution of $\mathcal{C}$. Since the two executions of $\mathcal{C}$ are exactly the same prior to the query $\mathrm{H}_2(\mathsf{pk}, m^*, R^*)$, we know $i' = i$. Also, we know $d = h_{2k-1} = h_{2k^*-1} = h_{2k'-1} = d'$, which implies $\mathsf{F}(z - z') = \lambda_{i'}^{lr^*.\mathsf{SS}}(h_I - h'_I) \cdot \mathsf{pk}_i$. Since $h_I \neq h'_I$, $\mathcal{B}$ computes $y_i \leftarrow \frac{z - z'}{\lambda_i^{lr^*.\mathsf{SS}}(h_I - h'_I)}$, which satisfies $\mathsf{F}(y_i) = \mathsf{pk}_i$. Denote $D := \{i\}_{j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}, (i,k,d,c,z) \leftarrow \mathsf{dt}(j)}$. Since E_1 occurs in the first execution of $\mathcal{C}$, we know $|T_{\mathsf{dt}}| = |\mathrm{curSS}_{\mathsf{L}}(lr^*)| < t - |CS|$. Therefore, we know $|D| = |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| < t - |CS|$. Therefore, $\mathcal{B}$ can pick an arbitrary set $D' \in HS \backslash D$ with size $(t - |CS| - |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| - 1)$ and for each $i \in D'$, $\mathcal{B}$ sets $y_i \leftarrow \mathrm{PI}(\mathsf{pk}_i)$. Denote $D_{tot} = CS \cup D \cup D'$, and we have $|D_{tot}| = t - 1$ and for each $i \in D_{tot}$, $\mathcal{B}$ knows y_i such that $\mathsf{F}(y_i) = \mathsf{pk}_i$. Since $\mathsf{pk}_i = \mathsf{F}(f(\mathsf{x}_i)) + A_0 + \sum_{j \in [t-1]} A_j \cdot \mathsf{x}_i^j$, denote $D_{tot} = \{i_1, \ldots, i_{t-1}\}$ and we have

$$M \cdot \begin{pmatrix} A_1 \\ \cdots \\ A_{t-1} \end{pmatrix} = \begin{pmatrix} \mathsf{F}(y_{i_1} - f(\mathsf{x}_{i_1}) - a_0) \\ \cdots \\ \mathsf{F}(y_{i_{t-1}} - f(\mathsf{x}_{i_{t-1}}) - a_0) \end{pmatrix}, \text{ where } M = \begin{pmatrix} \mathsf{x}_{i_1} & \cdots & \mathsf{x}_{i_1}^{t-1} \\ \vdots & \ddots & \vdots \\ \mathsf{x}_{i_{t-1}} & \cdots & \mathsf{x}_{i_{t-1}}^{t-1} \end{pmatrix}. \tag{10.5}$$

Since M is a Vandermonde matrix, we know M has full rank and thus $\mathcal{B}$ can compute $a_1, \ldots, a_{t-1}$ from (10.5) such that $\mathsf{F}(a_i) = A_i$ for $i \in [t-1]$. Further, for $i \in [n] \backslash D_{tot}$, $\mathcal{B}$ computes $y_i \leftarrow f(\mathsf{x}_i) + \sum_{j \in [0..(t-1)]} a_j \cdot \mathsf{x}_i^j$, which satisfies $\mathsf{F}(y_i) = \mathsf{pk}_i$.

We now show how to compute $u_1, v_1, \ldots, u_{\mathsf{q}_s}, v_{\mathsf{q}_s}$ such that $\mathsf{F}(u_i) = U_i$ and $\mathsf{F}(v_i) = V_i$. From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (i, k, d, c, z) \neq \perp$ if and only if $\mathcal{C}$ queries PI on $U_j + dV_j + c \cdot \mathsf{pk}_i$. Therefore, denote $U_j + dV_j + c \cdot \mathsf{pk}_i$ as the PI query *associated* with $\mathsf{dt}(j)$. For each $j \in \mathsf{q}_s$, there are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\perp$. In this case, $\mathcal{B}$ computes u_j and v_j by directly querying oracle PI on U_j and V_j.

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\perp$. Without loss of generality, assume $\mathsf{dt}(j) = (i, k, d, c, z)$, which implies $\mathsf{F}(z) = U_j + dV_j + c \cdot \mathsf{pk}_i$. $\mathcal{B}$ computes v_j by directly querying oracle PI and computes $u_j \leftarrow z - d \cdot v_j - c \cdot y_i$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\perp$ and we denote $(i, k, d, c, z) \leftarrow \mathsf{dt}(j)$ and $(i', k', d', c', z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $k \neq k'$ or $k = k' > k^*$. In this case, we know $d = h_{2k-1} \neq h'_{2k'-1} = d'$ and $\mathsf{F}(z) = U_j + dV_j + c \cdot \mathsf{pk}_i$, $\mathsf{F}(z') = U_j + d'V_j + c' \cdot \mathsf{pk}_{i'}$. Therefore, $\mathcal{B}$ computes $v_j \leftarrow \frac{z - c \cdot y_i - z' + c' \cdot y_{i'}}{d - d'}$, and $u_j \leftarrow z - d \cdot v_j - c \cdot y_i$.

Case 3: $k = k' = k^*$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1.

Case 4: $k = k' < k^*$. $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, in this case, we have $d = d'$ and $c = c'$. Therefore, $\mathcal{B}$ queries PI oracle only once in order to simulate the PI queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$.

We now count the number of PI queries made by $\mathcal{B}$.

- $\mathcal{B}$ queries PI oracle $|CS|$ times queries for simulating query $\mathrm{PI}(\mathsf{pk}_i)$ made by $\mathcal{C}$ for each $i \in |CS|$.

- $\mathcal{B}$ queries PI oracle $|D'|$ times queries for computing $a_0, \ldots, a_{t-1}$.

- For each $j \in \mathsf{q}_s$, $\mathcal{B}$ queries PI twice for simulating query associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j and v_j in case 0, 1, 2, 4 and queries 3 times in case 3.

Since the condition of case 3 is equivalent to $j \in T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}$, the total number of PI queries made by $\mathcal{B}$ is equal to $2\mathsf{q}_s + |T_{\mathsf{dt}} \cap T_{\mathsf{dt}'}| + |CS| + |D'| = 2\mathsf{q}_s + t - 1$. Therefore, $\mathcal{B}$ wins the game $\mathrm{AOMPR}_{\mathsf{LHF}}$. $\square$

10.4.3 Proof of Theorem 10.4.2

Let $\mathcal{A}$ be the adversary described in the theorem. Denote the output message-signature pair of $\mathcal{A}$ as $(m^*, sig^* = (R^*, z^*))$. Without loss of generality, we assume $\mathcal{A}$ always queries RO on $\mathrm{H}_2(\mathsf{pk}, m^*, R^*)$ before $\mathcal{A}$ returns and always queries RO on $\mathrm{H}_1(\mathsf{pk}, lr, i)$ prior to the query $\textsc{PSign}\mathrm{O}(i, lr)$ for some i and lr. (This adds up to q_s additional RO queries, and we let $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$.) Denote lr^* as the leader query such that $\mathrm{H}_1(\mathsf{pk}, lr^*, i)$ is the first RO query prior to the $\mathrm{H}_2(\mathsf{pk}, m^*, R^*)$ query for some i satisfying $\mathsf{SVf}(\mathsf{pk}, lr^*, sig^*) = \texttt{true}$. If such lr^* does not exist, lr^* is set to $\perp$. Denote the event E_1 as

$$\mathsf{Vf}(\mathsf{pk}, m^*, sig^*) \ \wedge \ (lr^* = \perp \ \vee \ \mathrm{curSS}_{\mathsf{L}}(lr^*) < t - |CS|) \ .$$

Denote the event E_2 as

$$\mathsf{Vf}(\mathsf{pk}, m^*, sig^*) \ \wedge \ lr^* \neq \perp \ \wedge \ \mathrm{curSS}_{\mathsf{L}}(lr^*) \neq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\} \ .$$

If $\mathcal{A}$ wins the game $\text{TS-SUF-3}^{\mathsf{FROST1\text{-}H}}$ and $lr^* \neq \perp$, we know either $\mathrm{curSS}_{\mathsf{L}}(lr^*) < t - |CS|$ or $\mathrm{curSS}_{\mathsf{L}}(lr^*) \neq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\}$. Therefore, if $\mathcal{A}$ wins the game $\text{TS-SUF-3}^{\mathsf{FROST1\text{-}H}}$, then either E_1 or E_2 occurs, which implies

$$\mathsf{Adv}^{\text{ts-suf-3}}_{\mathsf{FROST1\text{-}H[LHF]}}(\mathcal{A}) \leqslant \mathsf{Pr}[E_1] + \mathsf{Pr}[E_2] \leqslant 2\max\{\mathsf{Pr}[E_1], \mathsf{Pr}[E_2]\} \ .$$

Thus, we conclude the theorem with the following two lemmas.

Lemma 10.4.6. *There exists an adversary $\mathcal{B}$ for the $\mathrm{AOMPR}^{\mathsf{LHF}}$ game making at most $2\mathsf{q}_s + t$ queries to* CHAL *such that*

$$\mathsf{Pr}[E_1] \leqslant \sqrt{\mathsf{q} \cdot (\mathsf{Adv}^{\text{aompr}}_{\mathsf{LHF}}(\mathcal{B}) + 3\mathsf{q}^2(n+1)^2/2^{\kappa})} \ ,$$

Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$.

Lemma 10.4.7. *There exists an adversary $\mathcal{B}$ for the $\mathrm{AOMPR}^{\mathsf{LHF}}$ making at most $2\mathsf{q}_s$ queries to* CHAL *such that*

$$\mathsf{Pr}[E_2] \leqslant n \cdot \mathsf{q}\sqrt{2(\mathsf{Adv}^{\text{aompr}}_{\mathsf{LHF}}(\mathcal{B}) + 1/2^{\kappa})} \ .$$

Moreover, $\mathcal{B}$ runs in time roughly equal two times that of $\mathcal{A}$.

This completes the proof of the theorem, subject to proofs of the lemmas that we discuss next.

The proof of Lemma 10.4.6 is almost the same as Lemma 10.4.4, so we omit the full proof. The only difference is that $\mathcal{C}$ takes as input $h_1, \ldots, h_{(n+1)\mathsf{q}}$ in order to simulate all RO queries. For a RO query $\mathrm{H}_1(\mathsf{pk}, lr, i)$, $\mathcal{C}$ first enumerates all $i' \in [n]$ and assigns $h_{(\mathrm{ctr}_h-1)(n+1)+i'}$ to $\mathrm{H}_1(\mathsf{pk}, lr, i')$. Then, $\mathcal{C}$ computes the nonce R for lr and assigns $h_{\mathrm{ctr}_h(n+1)}$ to $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R)$ if it is not assigned any value yet. Similarly, for a new RO query $\mathrm{H}_1(\mathsf{pk}, M, R)$, its value is set to $h_{\mathrm{ctr}_h(n+1)}$. The rest follows by a similar analysis.

Proof of Lemma 10.4.7. We first construct an algorithm $\mathcal{C}$ following the syntax of the algorithm described in Lemma 9.3.4 and then construct $\mathcal{B}$ from $\mathsf{Fork}^{\mathcal{C}}$. The input of $\mathcal{C}$ consists of par that defines a linear hash function $(\mathcal{S}, \mathcal{D}, \mathcal{R}, \mathsf{F})$ and uniform random elements $h_1, \ldots, h_{n \cdot q} \in \mathcal{S}_{\mathrm{hash}}$. Similarly to the proof of Lemma 10.4.4, $\mathcal{C}$ can oracles CHAL and PI oracle and at the beginning, start with, $\mathcal{C}$ makes $2\mathsf{q}_s$ queries to CHAL and denotes the challenges as $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s} \in \mathcal{D}$. Then, $\mathcal{C}$ initializes all the states $\mathsf{st}_0, \ldots, \mathsf{st}_n$ as in the game TS-SUF-3$^{\mathsf{FROST1\text{-}H}}$, and initializes the counters ctr_s,ctr_h to 0 and the function dt to an empty table. $\mathcal{C}$ also initializes ctrPP to an empty table, which are used to record the counter corresponding to each token generated by honest parties. Then, $\mathcal{C}$ runs $\mathcal{A}$ with access to the oracles $\widetilde{\textsc{Init}}$, $\widetilde{\textsc{PPO}}$, $\widetilde{\textsc{PSign}\mathrm{O}}$, $\widetilde{\mathrm{RO}}$, which are simulated as follows. In the following description, we use i to denote the index of parties, j to denote the index of $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$, and k to denote the index of $h_1, \ldots, h_{n \cdot \mathsf{q}}$.

$\widetilde{\textsc{Init}}(CS)$: $\mathcal{C}$ initializes H to an empty table and samples $a_0, \ldots, a_{t-1}$ uniformly from $\mathcal{D}_{\mathrm{key}}$. Define $f(x) := \sum_{i=0}^{t-1} a_i x^i$. Then, $\mathcal{C}$ sets $\mathsf{pk} \leftarrow \mathsf{F}(a_0)$, $\mathsf{pk}_i \leftarrow \mathsf{F}(f(\mathsf{x}_i))$ for $i \in [n]$, and $\mathsf{sk}_i \leftarrow f(i)$ for $i \in CS$. Finally, $\mathcal{C}$ returns pk,$\mathsf{aux} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, $\{\mathsf{sk}_i\}_{i \in CS}$.

$\widetilde{\mathrm{RO}}$ query $\mathrm{H}_1(x)$: If $\mathrm{H}_1(x) \neq \bot$, $\mathcal{C}$ returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ parses x as $(\widetilde{\mathsf{pk}}, lr, \tilde{i})$ for some $\tilde{i} \in [1..n]$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_1(x) \leftarrow_{\$} \mathcal{S}_{\mathrm{hash}}$ and returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathrm{H}_1(\mathsf{pk}, lr, i) \leftarrow h_{n(\mathrm{ctr}_h-1)+i}$ for each $i \in [n]$. In addition, let $\mathrm{mapLR}(\mathrm{ctr}_h) := lr$. Then, $\mathcal{C}$ computes $R \leftarrow \sum_{i \in lr.\mathsf{SS}} (R_i + S_i^{d_i})$, where $(R_i, S_i) \leftarrow lr.\mathsf{PP}(i)$ and $d_i = \mathrm{H}_1(\mathsf{pk}, lr, i)$. If $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) = \bot$, $\mathcal{C}$ sets $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, R) \leftarrow_{\$} \mathbb{Z}_p$. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(x)$.

$\widetilde{\mathrm{RO}}$ query $\mathrm{H}_2(x)$: If $\mathrm{H}_2(x) = \bot$, $\mathcal{C}$ sets $\mathrm{H}_2(x) \leftarrow_{\$} \mathcal{S}_{\mathrm{hash}}$. Then, $\mathcal{C}$ returns $\mathrm{H}_2(x)$.

$\widetilde{\text{PPO}}(i)$ query: Same as in the game TS-SUF-3$^{\mathsf{FROST1\text{-}H}}$, except in the simulation of algorithm SPP, $\mathcal{C}$ first increases ctr_s by 1 and sets $pp \leftarrow (U_{\mathrm{ctr}_s}, V_{\mathrm{ctr}_s})$, $\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (0,0)$, and $\mathrm{ctrPP}(i, pp) \leftarrow \mathrm{ctr}_s$.

$\widetilde{\text{PSIGNO}}(i, lr)$ query: Same as in the game TS-SUF-3$^{\mathsf{FROST1\text{-}H}}$, except in the simulation of algorithm PS, if $\mathsf{st}_i.\mathrm{mapPP}(pp) \neq \bot$, $\mathcal{C}$ computes

$$z_i \leftarrow \mathsf{PI}\,(U_j + d_i V_j) + c \cdot \lambda_i^{lr.SS} \cdot f(i)\ ,$$

where $j \leftarrow \mathrm{ctrPP}(i, pp)$. In addition, $\mathcal{C}$ sets $\mathsf{dt}(j) \leftarrow (k, d_i, z_i - c\lambda_i^{lr.\mathsf{SS}} \cdot f(i))$, where k denotes the index such that $\mathrm{H}_1(\mathsf{pk}, lr, i)$ is set to h_k during the simulation.

After receiving the output $(m^*, sig^* = (R^*, z^*))$ from $\mathcal{A}$, $\mathcal{C}$ returns $(\bot, \bot, \bot)$ if E_2 does not occur. Otherwise, we know $\mathrm{curSS}_{\mathsf{L}}(lr^*) > 0$ and $\mathrm{curSS}_{\mathsf{L}}(lr^*) \neq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\}$. Therefore, there exists k^* and i^* such that $i^* \in \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\} \backslash \mathrm{curSS}_{\mathsf{L}}(lr^*)$ and $\mathrm{mapLR}(k^*) = lr^*$. (Since $\mathrm{curSS}_{\mathsf{L}}(lr^*) \subseteq \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\}$, we must have $\{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\} \backslash \mathrm{curSS}_{\mathsf{L}}(lr^*) \neq \varnothing$.) Since $i^* \in \{i \in HS \cap lr^*.\mathsf{SS} : lr^*.\mathsf{PP}(i) \in \mathrm{PP}_i\}$, there exists $j^* \in [1..\mathsf{q}_s]$ such that $lr^*.\mathsf{PP}(i^*) = (U_{j^*}, V_{j^*})$. If $\mathsf{dt}(j^*) = \bot$, $\mathcal{C}$ sets $J \leftarrow \bot$. Otherwise, let $(k, d, z) \leftarrow \mathsf{dt}(j^*)$ and $\mathcal{C}$ sets $J = k$. Then, $\mathcal{C}$ returns $(n(k^* - 1) + i^*, J, \mathrm{Out})$, where Out consists of all variables received or generated by $\mathcal{C}$, including i^*, j^*, k^*, lr^*.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.3.4, we define IG as the algorithm that runs $\mathsf{PGen}(1^\kappa)$ and outputs par. The output J is either $\bot$ or in $[1..(n \cdot \mathsf{q})]$. It is not hard to see that $\mathcal{C}$ simulates the game TS-SUF-3$^{\mathsf{FROST1\text{-}H}}$ perfectly, which implies $\mathrm{acc}(\mathcal{C}) \geqslant \mathsf{Pr}[E_2]$, where $\mathsf{Pr}[E_2]$ refers to the probability in the original TS-SUF-3$^{\mathsf{FROST1\text{-}H}}$ game with $\mathcal{A}$ (as in the lemma statement). By Lemma 9.3.4,

$$\mathrm{acc}(\mathsf{Fork}_2^{\mathcal{C}}) \geqslant \frac{\mathsf{Pr}[E_2]^2}{n \cdot \mathsf{q}(n \cdot \mathsf{q} + 1)} \leqslant \frac{\mathsf{Pr}[E_2]^2}{2n^2\mathsf{q}^2}\ .$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now give a construct of the AOMPR adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$, and the available CHAL and PI oracles. To start with, $\mathcal{B}$ receives par from the AOMPR$^{\mathsf{LHF}}$ game and runs $\mathsf{Fork}^{\mathcal{C}}(par)$. All the CHAL queries from the first execution of $\mathcal{C}$ are relayed by $\mathcal{B}$ to its own CHAL oracle, and for all the CHAL queries from the second execution of $\mathcal{C}$, $\mathcal{B}$ answers them with the same challenges as the first execution. All the PI queries from $\mathsf{Fork}^{\mathcal{C}}$ are relayed by $\mathcal{B}$

to its own PI oracle. Without loss of generality, we can assume all the challenges are different, since otherwise, $\mathcal{B}$ can solve them trivially. Denote the event $\mathsf{BadHash}$ as $h_I \neq h'_I$, where I are outputted by the first execution of $\mathcal{C}$. Since h_I, I are independent of h'_I, we know $\mathsf{Pr}[\mathsf{BadHash}] \leqslant 1/|\mathcal{S}_{\mathrm{hash}}| \leqslant 1/2^{\kappa}$. Then, we can conclude the proof with the following claim, which implies

$$\mathsf{Adv}^{\mathrm{aompr}}_{\mathsf{LHF}}(\mathcal{B}) \geqslant \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}_2) - \mathsf{Pr}[\mathsf{BadHash}] \geqslant \frac{\mathsf{Pr}[E_2]^2}{2n^2\mathsf{q}^2} - 1/2^{\kappa} \ .$$

□

Claim 10.4.8. *$\mathcal{B}$ can win the game $\mathrm{AOMPR}_{\mathsf{LHF}}$, if $\mathsf{Fork}^{\mathcal{C}}$ returns $(I, \mathrm{Out}, \mathrm{Out}')$ and $\mathsf{BadHash}$ does not occur.*

Proof. We directly use the notations in the description of $\mathcal{C}$ to denote the variables in Out and use $(\cdot)'$ to denote the variables in Out'. The total number of the CHAL queries is $2\mathsf{q}_s$ and the corresponding challenges are $U_1, V_1, \ldots, U_{\mathsf{q}_s}, V_{\mathsf{q}_s}$.

We now show how to compute u_j, v_j for each $j \in [\mathsf{q}_s]$ such that $\mathsf{F}(u_i) = U_i$ and $\mathsf{F}(v_i) = V_i$. There are the following cases.

Case 0: Both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are $\perp$. In this case, $\mathcal{B}$ computes u_j, v_j by directly querying oracle $\mathrm{PI}(U_j)$ and $\mathrm{PI}(V_j)$.

Case 1: Exactly one of $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ is not $\perp$. Without loss of generality, assume $\mathsf{dt}(j) = (k, d, z)$, which implies $\mathsf{F}(z) = U_j + dV_j$. $\mathcal{B}$ computes v_j by directly querying oracle $\mathrm{PI}(V_j)$ and computes $u_j \leftarrow z - d \cdot v_j$.

For all the following cases, both $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ are not $\perp$ and we denote $(k, d, z) \leftarrow \mathsf{dt}(j)$ and $(k', d', z') \leftarrow \mathsf{dt}'(j)$.

Case 2: $d \neq d'$. In this case, $\mathcal{B}$ computes $v_j = \frac{z - z'}{d - d'}$, $u_j = z - d \cdot v_j$.

Case 3: $d = d'$. In this case, $\mathcal{B}$ computes v_j, u_j the same as Case 1. Also, since $d = d'$, $\mathcal{B}$ queries PI oracle only once in order to answer queries $\mathrm{PI}(U_j + dV_j)$ and $\mathrm{PI}(U_j + d'V_j)$ from $\mathsf{Fork}^{\mathcal{C}}$.

From the execution of $\mathcal{C}$, we know $\mathsf{dt}(j) = (k, d, z) \neq \perp$ if and only if $\mathcal{C}$ queries PI on $(U_j + dV_j)$. Therefore, denote $(U_j + dV_j)$ as the PI query associated with $\mathsf{dt}(j)$. For all the above cases, $\mathcal{B}$

queries PI oracle twice for simulating PI queries associated with $\mathsf{dt}(j)$ and $\mathsf{dt}'(j)$ and computing u_j, v_j.

We now show how to compute u_{j^*} and v_{j^*}. From the execution of $\mathsf{Fork}_2^{\mathcal{C}}$, we know $\mathsf{pk} = \mathsf{pk}'$ and $\mathrm{mapLR}(k) = \mathrm{mapLR}'(k)$ for all $k \leqslant I$, which implies $lr^* = \mathrm{mapLR}(I) = \mathrm{mapLR}'(I) = lr^{*\prime}$. Since E_2 occurs in both executions of $\mathcal{C}$, we know $\mathsf{SVf}(\mathsf{pk}, lr^*, (R^*, z^*)) = \mathtt{true}$ and $\mathsf{SVf}(\mathsf{pk}, lr^*, (R^{*\prime}, z^{*\prime})) = \mathtt{true}$ are valid. Therefore, $\mathsf{F}(z^*) = R^* + \mathsf{F}(a_0 c)$, $R^* = \sum_{i \in lr^*.\mathsf{SS}}(R_i + d_i S_i)$, $g^{z^{*\prime}} = R^{*\prime} + \mathsf{F}(a_0 c')$, $R^{*\prime} = \sum_{i \in lr^*.\mathsf{SS}}(R_i + d'_i S_i)$, where $(R_i, S_i) = lr.\mathsf{PP}(i)$, $c = \mathrm{H}_2(\mathsf{pk}, M^*, R^*)$, $c' = \mathrm{H}'_2(\mathsf{pk}, M^*, R^{*\prime})$, and $d_i = \mathrm{H}_1(\mathsf{pk}, lr^*, i)$, $d'_i = \mathrm{H}'_1(\mathsf{pk}, lr^*, i)$. Since for each $i \neq i^*$ we have $d_i = h_{n(k^*-1)+i} = d'_i$, we have

$$\mathsf{F}(z^* - z^{*\prime}) = R^* - R^{*\prime} + \mathsf{F}(a_0(c - c')) = (d_{i^*} - d'_{i^*})S_{i^*} + \mathsf{F}(a_0(c - c')) \ .$$

Therefore, $\mathcal{C}$ can compute $v_{j^*} = \frac{z^* - z^{*\prime} - a_0(c-c')}{d_{i^*} - d'_{i^*}}$. If $J = \perp$, $\mathcal{B}$ computes u_{j^*} by querying $\mathrm{PI}(U_{j^*})$ directly. In this case, $\mathcal{B}$ queries PI only once to compute u_{j^*} and v_{j^*}. If $J \neq \perp$, let $(k, d, z) \leftarrow \mathsf{dt}(j^*) =$ and $(k', d', z') \leftarrow \mathsf{dt}(j^*)$. Then, $\mathcal{B}$ computes $u_{j^*} = z - d \cdot v_{j^*}$. Since $i^* \notin S_2(lr^*)$, we know $k \neq I$.(Otherwise, suppose $k = I$. Since $I = n(k^* - 1) + i^*$ and $\mathrm{mapLR}(k^*) = lr^*$, we know a $\textsc{PSign}\mathrm{O}(i^*, lr^*)$ is made and does not return $\perp$ during the simulation, which implies $i^* \in S_2(lr^*)$.) Thus, we have $k' = J = k \neq I$ and $d = h_J = d'$, which means $\mathcal{B}$ only needs to query PI once to simulate the PI queries associated with $\mathsf{dt}(j^*)$ and $\mathsf{dt}'(j^*)$. Therefore, the total number of PI queries made by $\mathcal{B}$ is equal to $2\mathsf{q}_s - 1$, which implies $\mathcal{B}$ wins the game $\mathrm{AOMDL}^{\mathsf{LHF}}$. $\square$

Part III

LATTICE-BASED THRESHOLD SIGNATURES

Chapter 11

THE ALGEBRAIC ONE-MORE MISIS PROBLEM AND LATTICE-BASED THRESHOLD SIGNATURES: INTRODUCTION

One-more assumptions enable proofs of security for several interactive protocols, most notably identification schemes, threshold signatures, multi-signatures, and blind signatures. The perhaps most prominent example is the *one-more discrete logarithm assumption* (OMDL) [16], which requires the hardness of computing Q discrete logarithm instances given access to an oracle that allows the adversary to compute $Q-1$ discrete logarithms of *arbitrary* group elements. It has been used throughout several security proofs (e.g., cf. [16, 65, 105, 14]), where the oracle allows the reduction to simulate secret-key dependent behavior of honest parties without knowing the secret key. This notwithstanding, the main point of controversy is that an assumption such as OMDL is very strong–for instance, Koblitz and Menezes [89] point out that in certain groups, it is easier to break OMDL than to solve the standard discrete logarithm problem. The only available route justifying its plausible hardness on standard elliptic curves is a proof [12] in the generic-group model (GGM) [118, 101].

PROVABLY-HARD ONE-MORE PROBLEMS. A few recent works [122, 9] follow a different path: While they still leverage the power of one-more problems as an intermediate *interface* to design modular security proofs, they also instantiate the underlying mathematical structure to support a proof that the one-more problem is indeed hard based on more standard assumptions, such as the hardness of (standard) discrete log/RSA (in [122]) or of DDH (in [9]).

This is the angle pursued by this paper—we seek one-more problems which are sufficiently *expressive* to enable useful security proofs, while also enjoying *provable* hardness from standard assumptions. This paper deals specifically with *lattice-based cryptography*. Espitau, Katsumata, and Takemure (EKT) [60] recently introduced a one-more problem they refer to as *algebraic one-more MLWE (AOM-MLWE)* and show that its hardness yields the security of a two-round threshold signature. They also establish the *selective* hardness of AOM-MLWE from the standard MLWE and MSIS assumptions, which is however unfortunately not sufficient to support their application to threshold signatures. Another example—less relevant for this work, however—is the one-more

ISIS problem [7], used in the construction of lattice-based blind signatures, which has also only been validated via cryptanalysis.

OUR CONTRIBUTIONS, IN A NUTSHELL. We introduce a new variant of AOM-MLWE which we refer to as *Algebraic One-More MISIS* (AOM-MISIS, for short),[1] and for which we show two different types of results:

- ***Provability.*** We show that the hardness of AOM-MISIS follows from the hardness of MSIS and MLWE, with suitable parameters. In fact, the hardness of AOM-MISIS *implies* the hardness of AOM-MLWE, and thus our result carries over to AOM-MLWE, providing the first reduction of AOM-MLWE to standard assumptions, which was left as a main open question in [60].

- ***Expressivity.*** We show the security of the EKT threshold signature scheme from [60], as well as of the recent scheme by Chairattana-Apirom, Tessaro, and Zhu [41], assuming the hardness of AOM-MISIS. In turn, this establishes the security of these schemes from MSIS and MLWE. We obtain either better concrete security bounds compared to [41], or proofs under weaker assumptions compared to [60] for slightly larger concrete parameter sets. We also give a proof that the EKT scheme satisfies the strongest notion of security in the hierarchy of Bellare *et al.* [14].

ALGEBRAIC ONE-MORE MISIS. The definition of AOM-MISIS relies on the cyclotomic ring $R = \mathbb{Z}[X]/(X^N + 1)$, where N is a power of two, as well as the associated ring $R_q = R/qR \cong \mathbb{Z}_q[X]/(X^N + 1)$ for an odd prime q. The problem is defined via the following game:

- **Input.** The adversary is initially given a matrix $A = [\overline{A}|\mathbb{I}_k] \in R_q^{k\times m}$, where $\overline{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$, as well as Q instances $\boldsymbol{t}_i \leftarrow A\boldsymbol{s}_i$, for $i \in [Q]$, where $\boldsymbol{s}_i \leftarrow\!\!\$\ \mathscr{D}_{\sigma_i}^m$ is "small", and sampled from an m-dimensional discrete Gaussian with parameter σ_i. The instance number Q is a parameter of the game.

- **Oracle access.** The adversary can also (adaptively) query an oracle PI which takes as input a Q-dimensional vector $\boldsymbol{b} \in R^Q$, and returns $\sum_{i\in[Q]} b_i\boldsymbol{s}_i$.

[1] Our naming is due to the fact that we prefer to think of our problem as a one-more version of Inhomogenous SIS (SIS), rather than of LWE.

To win, the adversary needs to output both $\hat{\boldsymbol{b}} = (\hat{b}_1, \ldots, \hat{b}_Q) \in R^Q$ and a "short" solution $\hat{\boldsymbol{s}} \in R^m$ such that

$$\sum_{i \in [Q]} \hat{b}_i \boldsymbol{t}_i = A\hat{\boldsymbol{s}} \ .$$

To exclude trivial winning strategies, $\hat{\boldsymbol{b}}$ must however not be in the span of the vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots$ queried to PI. We in fact require something stronger, namely that in order to win, the adversary also needs to additionally output $\boldsymbol{u} \in R^Q$, along with $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{s}}$, and the following two properties need to be satisfied:

- The vector $\boldsymbol{u}$ is orthogonal to all vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots$ queried to PI, but not to $\hat{\boldsymbol{b}}$, i.e., $\boldsymbol{b}_i^\top \boldsymbol{u} = 0$, but $\hat{\boldsymbol{b}}^\top \boldsymbol{u} \neq 0$. Here, orthogonality is defined with respect to the ring R.

- Both vectors $(\hat{b}_1 \cdot \sigma_1, \ldots, \hat{b}_Q \cdot \sigma_Q)$ and $(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)$ are appropriately small, when interpreted as real vectors. Both conditions are necessary, as otherwise a simple attack exists.

Our main result is a concrete reduction showing that the AOM-MISIS problem is indeed hard if the Module LWE (MLWE) and Module SIS (MSIS) assumptions also hold with suitable parameters, related to the parameters of the above game. Our proof relies on a novel *generalization* of the technique by Tessaro and Zhu [122], which was originally used to obtain hard one-more problems based on linear hash functions. The origin of this technique goes back to the analysis of Okamoto signatures [107], and was used also in [80]. Some prior works [122, 81] can be interpreted as attempts to adapt this approach to the lattice setting in limited ways, but in our context, they would lead to worse parameters and restrictions on the generality of our game. We discuss further details in Section 11.1 below.

We note that the hardness of AOM-MISIS implies the hardness of AOM-MLWE, and thus we resolve the main open problem from [60], which only provided an analysis for *selective* adversaries issuing their oracle queries beforehand. In fact, the main difference between AOM-MISIS and AOM-MLWE is the winning condition. We require outputting a single short solution for a suitable non-trivial linear combination of the challenges, whereas AOM-MLWE requires outputting Q short solutions for all challenges, given $Q - 1$ access to a similar oracle, although with different conditions on the queries. These changes are what in part enables simpler reductions for threshold signatures. A more detailed discussion is provided in Section 12.2.

APPLICATIONS TO THRESHOLD SIGNATURES. We first recall that in a *t-out-of-n threshold signature scheme* [58, 59], n potential signers each hold a secret share of a secret signing key, with an associated public verification key. Any subset of (at least) t of these signers is able to jointly produce a signature, via interaction, whereas an adversary that controls fewer than t signers should not be able, on its own, to come up with a valid signature.

We leverage AOM-MISIS to obtain tighter analyses for state-of-the-art *two-round* lattice-based threshold signatures, based on the MSIS/MLWE assumptions, and without assuming the hardness of any ad-hoc one-more problem. We focus on two threshold signatures, which we refer to as CATZ [41] and EKT [60]. There are several small differences between the two constructions, despite the fact that they instantiate similar ideas. Both can be seen as a natural threshold version of the Fiat-Shamir-with-abort paradigm [97] that also underlies DILITHIUM [100], albeit dispensing with the actual abort, and using a sufficiently large modulus instead. In particular, EKT is a threshold version of Raccoon [55], submitted to the additional NIST call for post-quantum signatures [103]. They are natural lattice analogues of FROST [91, 14, 49], a very lightweight threshold Schnorr signature.

Taking some liberty from their formal description, in both schemes, as a result of the second round, the i-th signer produces an affine signature share

$$\boldsymbol{z}_i = \boldsymbol{r}_i + c \cdot \lambda_i \mathsf{ss}_i \ ,$$

in a suitable algebraic structure, where ss_i is the i-th signer's key share, c is a hash value associated with the signature, and λ_i is a linear reconstruction coefficient associated with the secret sharing scheme. With S being the set of signers, the final signature has format $(\boldsymbol{R}, \boldsymbol{z})$, where $\boldsymbol{R} = A \sum_{i \in S} \boldsymbol{r}_i$, ($A$ is a public matrix) and $\boldsymbol{z} = \sum_{i \in S} \boldsymbol{z}_i$. However, the two schemes differ in the following aspects:

- In CATZ, to avoid leakage of the secret shares, $\lambda_i \mathsf{ss}_i$ needs to remain sufficiently small and thus a secret sharing scheme with small reconstruction coefficients is needed. This incurs a significant cost due to the larger share size when compared to Shamir secret sharing. The security proof in [41] is fairly involved, and gives a direct reduction to MSIS.

- To enable the use of Shamir secret sharing, EKT changes the initial setup to ensure that any pair of signers shares a secret key, and these keys are used, in each execution of the signing protocol, to generates (pseudo)random masks $(\mathsf{msk}_i)_{i \in S}$ such that $\sum_{i \in S} \mathsf{msk}_i = 0$. The i-th

signer then sends $\boldsymbol{z}_i + \mathsf{msk}_i$ in the second round instead of $\boldsymbol{z}_i$, and this ensures that only the *sum* of the $\boldsymbol{z}_i$'s is leaked. The security proof in [60] relies on the direct use of the AOM-MLWE assumption, and since the constructed adversary is inherently not selective, they need to rely on a conjecture about the hardness of AOM-MLWE.

In Sections 13.1 and 13.2, we give new security analyses for both schemes based on direct reductions from AOM-MISIS, which in turn yield concrete security proofs from MLWE and MSIS. Our result for CATZ yields better parameters than the analysis of [41]. The result about EKT, in addition to now basing security on MLWE/MSIS alone, also shows a strong security property for this scheme, namely TS-UF-4 in the hierarchy of Bellare et al. [14]. The concrete efficiency numbers derived from our bounds are somewhat worse than those from EKT, but their concrete analysis relies on their own cryptanalysis of AOM-MLWE, whereas we use standard parameters for MLWE and MSIS. Closing the gap between the cryptoanalysis-driven parameter choices and those derived from our security reduction remains an interesting open problem.

The recent work on Ringtail [33] also proposes a threshold signature scheme similar in spirit to EKT and with a proof of security under MLWE. In Ringtail, unlike EKT, signers need to know already in the first round the set of involved signers S, and thus, unlike EKT and CATZ, Ringtail is not partially non-interactive in the sense of Bellare *et al.* [14]. Ringtail also needs to authenticate first-round messages. The authors of Ringtail mention removing these restrictions, while preserving comparable efficiency and provability from MLWE/MSIS, is an open problem, which we resolve here based on the EKT scheme.

Ringtail's security bound degrades with the number of random oracle queries, whereas ours degrades with the number of signing sessions. We note that the latter is a system parameter that can be enforced, whereas the former scales with the running time of the adversary.

REMARKS ON CATZ. While the concrete efficiency of CATZ is significantly worse than that of EKT, we see value in the CATZ construction because it does not rely on pairwise masks. CATZ closely resembles the structure of the original FROST protocol, making it a promising starting point for achieving identifiable aborts (constructing efficient partially non-interactive lattice-based threshold signatures with identifiable aborts is a big open problem in the field). In contrast, achieving identifiable aborts in EKT seems less likely without relying on heavyweight NIZK proofs, due to its reliance on pairwise masks.

Also, the security argument of CATZ works for a class of linear secret sharing schemes rather

than a specific scheme. Its main source of inefficiency stems from the underlying secret sharing scheme, and its efficiency could be significantly improved if better instantiations of the underlying secret sharing scheme are proposed.

OTHER RELATED WORK. We note here that there are other approaches to threshold signatures. First off, *Fully-Homomorphic Encryption* (FHE) generically yields round-optimal threshold signatures [30, 29, 8]. These require however the homomorphic evaluation of the signing algorithm, and thus come with a substantial computational overhead. Earlier works [51, 48] proposed two-round n-out-of-n threshold signatures derived from constructions for the related notion of multi-signatures. Gur *et al.* [79] proposed two-round construction based on linearly homomorphic encryption (LHE) which supports arbitrary thresholds. Both rounds are message-dependent. More recently, Pino *et al.* [54] propose a more efficient lattice-based threshold signature scheme that does not rely on FHE or the aforementioned heavy primitives, but the drawback is that the protocol has three message-dependent rounds. Recent work [87] also gives a five-round threshold signature with adaptive security based on MLWE/MSIS, whereas a threshold version of Falcon [112] was presented in [61]. The latter scheme is designed for robustness but requires four message-dependent rounds and incurs quadratic communication complexity in the threshold.

11.1 Technical Overview

The main goal of this section is to provide a detailed overview of our main result establishing the hardness of AOM-MISIS based on MSIS and MLWE. We stress that our reduction will involve concrete parameters, but we keep the discussion in this section somewhat qualitative on this front.

At a technical level, it helps to see AOM-MISIS as a lattice-based analogue of the AOMPR framework proposed by Tessaro and Zhu (TZ) [122] to define one-more problems for linear hash functions. Their framework does not cover lattice problems, however—we aim to adapt it to lattices, and this presents a number of challenges, which we explain in this section. We will also discuss how prior works have already tried to overcome such challenges, how those approaches are not sufficient for our purposes, and what we do instead.

We focus first on a variant of AOM-MISIS we denote as wAOM-MISIS, where the adversary does not output the relaxation vector $\hat{\boldsymbol{b}}$ and is instead asked to output Q solutions $\hat{\boldsymbol{s}}_1, \ldots, \hat{\boldsymbol{s}}_Q$ with small norm (instead of a single solution) such that $\boldsymbol{t}_i = A\hat{\boldsymbol{s}}_i$ for all $i \in [Q]$. Also, the constraint $\hat{\boldsymbol{b}}^T \boldsymbol{u} \neq 0$ is now changed to $\boldsymbol{u} \neq 0$ instead, while $\boldsymbol{b}_i^T \boldsymbol{u} = 0$ still must hold for the queries to PI.

This problem is not easier than AOM-MISIS: If the adversary wins the wAOM-MISIS game, there exists an index $i \in [Q]$ such that $u_i \neq 0$, and thus the adversary can win the original game by outputting $(\hat{\boldsymbol{s}}_i, \boldsymbol{e}_i, \boldsymbol{u})$, where $\boldsymbol{e}_i$ is the i-th unit vector. This is also the idea underlying the proof that AOM-MISIS hardness implies the hardness of AOM-MLWE in Section 12.2.

Reduction ideas from [122].

We now provide a self-contained review of TZ's proof framework in the context of wAOM-MISIS, and explain where it fails. In particular, given an adversary $\mathcal{A}$ for the wAOM-MISIS game, we build a simple MSIS adversary $\mathcal{B}$ that takes a random matrix $A \in R_q^{k \times m}$ as input and outputs a vector $\boldsymbol{z}$ with small norm such that $A\boldsymbol{z} = 0$. To start with, $\mathcal{B}$ runs $\mathcal{A}$ by simulating the wAOM-MISIS game faithfully using the matrix A and sampling $\boldsymbol{s}_i$ by itself. After receiving $(\hat{\boldsymbol{s}}_1, \ldots, \hat{\boldsymbol{s}}_Q, \boldsymbol{u})$, if $\mathcal{A}$ wins, $\mathcal{B}$ finds an index $i \in [Q]$ such that $\hat{\boldsymbol{s}}_i \neq \boldsymbol{s}_i$ and outputs $\boldsymbol{z} \leftarrow \hat{\boldsymbol{s}}_i - \boldsymbol{s}_i$. Otherwise, $\mathcal{B}$ aborts.

ANALYSIS OF $\mathcal{B}$. It is not hard to see that $\mathcal{B}$ wins the MSIS game if such an index i indeed exists, since $A\boldsymbol{z} = A\hat{\boldsymbol{s}}_i - A\boldsymbol{s}_i = 0$ and $\|\boldsymbol{z}\| \leqslant \|\boldsymbol{s}_i\| + \|\hat{\boldsymbol{s}}_i\|$ is bounded given both $\|\boldsymbol{s}_i\|$ and $\|\hat{\boldsymbol{s}}_i\|$ are bounded. Here $\|\cdot\|$ denotes the ℓ_2-norm. The hard part is to show that such i exists w.h.p. under the assumption that $\mathcal{A}$ wins w.h.p., and this is what we discuss next.

Following the same lines as TZ, we can assume w.l.o.g. that $\mathcal{A}$ is deterministic, and thus the view $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ of $\mathcal{A}$ (assuming the matrix A is fixed) is completely determined by the challenge secrets $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Also, we denote by $\mathbf{D}$ the set of secrets $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ such that $\mathcal{A}$ wins the wAOM-MISIS game.

While this is not the case for us, suppose for now that $\mathbf{D}$ is finite and each $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ is sampled uniformly from $\mathbf{D}$. The first step is to find a *derangement* Φ of $\mathbf{D}$, i.e., a permutation of $\mathbf{D}$ without fixed points, such that for any $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in \mathbf{D}$, $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) = \mathrm{View}_{\mathcal{A}}(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q)$ with $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) = \Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Therefore, $\mathcal{A}$ also produces the same output $(\hat{\boldsymbol{s}_1}, \ldots, \hat{\boldsymbol{s}_n}, \boldsymbol{u})$ in both cases. Since $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \neq (\boldsymbol{s}'_1, \ldots, \boldsymbol{s}_Q)$, there exists an index $i \in [Q]$ such that $\boldsymbol{s}_i \neq \boldsymbol{s}'_i$ and thus either $\boldsymbol{s}_i \neq \hat{\boldsymbol{s}}_i$ or $\boldsymbol{s}'_i \neq \hat{\boldsymbol{s}}_i$, which means $\mathcal{B}$ wins the MSIS game in at least one of the cases. Since Φ is a permutation of $\mathbf{D}$, for at least half of elements in $\mathbf{D}$ lead to $\mathcal{B}$ winning. Therefore, $\mathcal{B}$ wins with at least half of the probability that $\mathcal{A}$ wins. Crucially, note that Φ does not need to be efficient for this argument to succeed, as the algorithm $\mathcal{B}$ described above is the actual reduction.

CONSTRUCTING Φ. The first idea is to define Φ such that

$$\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) := (\boldsymbol{s}_1 + u_1 \cdot \Delta, \ldots, \boldsymbol{s}_Q + u_Q \cdot \Delta) \,, \tag{11.1}$$

where $\Delta \in R^m$ is a non-zero vector such that $A\Delta = 0$ (we will discuss below how to ensure such Δ exists), and $\boldsymbol{u}$ is the vector output by $\mathcal{A}$ in an execution with the secret value $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. It is not hard to check that $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ and $\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ produce the same view. First off, since $A(\boldsymbol{s}_i + u_i \cdot \Delta) = A\boldsymbol{s}_i + u_i \cdot (A\Delta) = A\boldsymbol{s}_i$, the input of $\mathcal{A}$ is identical in both cases. Further, for each PI query $\boldsymbol{b}$ made by $\mathcal{A}$, since $\boldsymbol{u}$ is a non-zero vector satisfying $\sum_{i\in[Q]} u_i b_i = 0$, we have $\sum_{i\in[Q]} b_i(\boldsymbol{s}_i + u_i \cdot \Delta) = \sum_{i\in[Q]} b_i \boldsymbol{s}_i + (\sum_{i\in[Q]} b_i u_i) \cdot \Delta = \sum_{i\in[Q]} b_i \boldsymbol{s}_i$, which means the response received by $\mathcal{A}$ is identical in both cases.

ISSUES AND PRIOR APPROACHES. The issue with the above approach is that in our setting the secrets are not sampled uniformly from a set. Two prior works [122, 81] have overcome this issue by sampling $\boldsymbol{s}_i$ uniformly from a bounded box $\mathscr{B}^m_{\beta_i} := \{\boldsymbol{x} \in R^m | \, \|\boldsymbol{x}\| \leqslant \beta_i\}$. Still, the challenge now is that Φ is no longer a permutation over a subset of $\mathscr{B}^m_{\beta_1} \times \cdots \times \mathscr{B}^m_{\beta_Q}$, since $\|\boldsymbol{s}_i + u_i \cdot \Delta\|$ can be larger than β_i even though $\|\boldsymbol{s}_i\| \leqslant \beta_i$.

To overcome this issue, Hauck *et al.* [81][2] proposed to set β_i to be very large such that the total faction of $\boldsymbol{s}_i \in \mathscr{B}^m_{\sigma_i}$ such that $\boldsymbol{s}_i + u_i \cdot \Delta$ falls outside $\mathscr{B}^m_{\sigma_i}$ is negligible in the security parameter κ. Then, one can still show $\mathcal{B}$ wins with nearly at least half of the winning probability of $\mathcal{A}$. However, this requires $\sigma_i = \Omega(2^\kappa \|u_i \Delta\|)$, which results in very inefficient constructions using this approach.

Chairattana-Apirom et al. [41] overcame the 2^κ barrier with a different approach. They let $\boldsymbol{s}_1$ be sampled from $\mathscr{B}^m_{\beta_1}$, whereas $\boldsymbol{s}_i$ is sampled from $\mathscr{D}^m_{\sigma_i}$. They use this in the security proof of their threshold signature scheme based on the MSIS assumption, but their technique can be massaged into a reduction from MSIS to a restricted version of the wAOM-MISIS game, where the PI queries are restricted, but still sufficient for their application.

Their proof is in some sense a probabilistic relaxation of TZ's, in that for any $\boldsymbol{s}_1 \in R^m$, they consider a *random variable* $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1)$ which represents the view of $\mathcal{A}$ given the first secret is $\boldsymbol{s}_1$, whereas the remaining secrets are sampled afresh from discrete Gaussian distributions. If for any $\boldsymbol{s}_1 \neq \boldsymbol{s}'_1 \in R^m$, such that $A\boldsymbol{s}_1 = A\boldsymbol{s}'_1$, we can show that $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1)$ and $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}'_1)$ are identically distributed, then we can once again carry out the same argument as above. They in fact generalize this argument by showing that if the Rényi divergence of $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1)$ from $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}'_1)$ is small, one

[2]We note that [81] precedes [122], but deals with a lattice-analogue of a linear-hash function based framework introduced by [80] similar in spirit to that of [122].

can still show that the winning probabilities of $\mathcal{B}$ and $\mathcal{A}$ are sufficiently related. Finally, they show the Rényi divergence is indeed bounded for the special types of queries they consider. Doing so, they ensure that σ_i depends linearly on $\sqrt{Q}$ instead of 2^κ.

Still, their reduction fails if $\mathcal{A}$ issues general PI queries, and this is because the Rényi divergence of $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1)$ from $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}'_1)$ is no longer bounded. For example, the first secret (either $\boldsymbol{s}_1$ or $\boldsymbol{s}'_1$) can be recovered from an PI query with input $(1, 0, \ldots, 0)$. Moreover, they need to ensure that almost all $\boldsymbol{s}_1$'s have one partner $\boldsymbol{s}'_i \neq \boldsymbol{s}_i \in \mathscr{B}^m_{\beta_1}$ such that $A\boldsymbol{s}_1 = A\boldsymbol{s}'_1$, and this requires $\beta_1 \geqslant 2^{\kappa/N} q^{k/m}$,[3] which leads to inefficient parameters in their threshold signature.

11.1.1 Step 1: Generalizing TZ's argument

Our goal is to provide a reduction that supports an adversary $\mathcal{A}$ without any extra restrictions on its queries to PI, and furthermore that ensures hardness even for sufficiently small parameter choices in $\mathrm{wAOM\text{-}MISIS}$. For this reason, we take a different route and generalize TZ's argument to a setting where each $\boldsymbol{s}_i$ is sampled (independently) from an arbitrary distribution $\mathcal{P}_i$, rather than uniformly from a finite set.

Let $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_Q$ be the joint distribution of $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Then, instead of requiring Φ to be a permutation of $\mathbf{D}$ (recalling that $\mathbf{D}$ is the set of secrets $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ that make $\mathcal{A}$ win the $\mathrm{wAOM\text{-}MISIS}$ game), we require that the distribution of $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ conditioned on $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in \mathbf{D}$ (denoted as $\mathcal{P}_{|\mathbf{D}}$) is identical to that of $\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ conditioned on $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in \mathbf{D}$ (denoted as $\Phi(\mathcal{P}_{|\mathbf{D}})$). The original TZ approach corresponds to the case when $\mathcal{P}$ is a uniform distribution over $\mathbf{D}$. Here, since arbitrary distributions over $\mathbf{D}$ are allowed, we no longer require $\mathbf{D}$ to be a finite set. The two other requirements for Φ remain the same, i.e., Φ has no fixed point over $\mathbf{D}$ and $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) = \mathrm{View}_{\mathcal{A}}(\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q))$.

Our key observation here is that the previous argument to lower bound the success probability of $\mathcal{B}$ still applies. To see this, it helps us to extend Φ to the space of all potential secrets (not only those in $\mathbf{D}$) by defining $\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) := (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ for all $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in R^m \backslash \mathbf{D}$. Then, consider the following two adversaries $\mathcal{B}'$ and $\mathcal{B}''$.

- $\mathcal{B}'$ is the same as $\mathcal{B}$ except that $\mathcal{B}'$ samples $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ from $\Phi(\mathcal{P})$.

- $\mathcal{B}''$ is the same as $\mathcal{B}$ except that 1. $\mathcal{B}''$ samples $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) \leftarrow\!\!\$\, \mathcal{P}$ and computes secrets as

[3] Since N is usually set to 512 or 1024, the leading term here is $q^{k/m}$.

$(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \leftarrow \Phi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$; 2. after $\mathcal{A}$ returns, $\mathcal{B}''$ outputs $\hat{\boldsymbol{s}}_i - \boldsymbol{r}_i$ if there exists $\boldsymbol{r}_i \neq \hat{\boldsymbol{s}}_i$.

We note that $\mathcal{B}'$ and $\mathcal{B}''$ do not need to be efficient. They are only used to compute the winning probability of the (efficient) adversary $\mathcal{B}$.

Denote now by $\mathsf{PWin}_{\mathcal{X}}$ the winning probability of $\mathcal{X} \in \{\mathcal{A}, \mathcal{B}, \mathcal{B}', \mathcal{B}''\}$. Then, the conclusion that $\mathsf{PWin}_{\mathcal{B}} \geqslant 1/2\mathsf{PWin}_{\mathcal{A}}$ is a straightforward corollary of the following three facts.

Fact 1. $\mathsf{PWin}_{\mathcal{B}} = \mathsf{PWin}_{\mathcal{B}'}$;

Fact 2. $\mathsf{PWin}_{\mathcal{B}} = \mathsf{PWin}_{\mathcal{B}''}$;

Fact 3. $\mathsf{PWin}_{\mathcal{B}'} + \mathsf{PWin}_{\mathcal{B}''} \geqslant \mathsf{PWin}_{\mathcal{A}}$.

Fact 1 is straightforward since $\mathcal{P}$ is identical to $\Phi(\mathcal{P})$, which means $\mathcal{B}$ and $\mathcal{B}'$ behave the same. Fact 2 is also not hard to see. Since $\mathrm{View}_A(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) = \mathrm{View}_A(\Phi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)) = \mathrm{View}_A(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$, even if $\mathcal{B}''$ sets the secrets to $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$ instead of $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q$, the output of $\mathcal{B}''$ remain the same. However, then, $\mathcal{B}''$ is identical to $\mathcal{B}$, which implies the second fact.

Finally, to prove the third fact, we can interpret the sampling process of $\mathcal{B}'$ as first sampling $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$ from $\mathcal{P}$ and then set $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \leftarrow \Phi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$. Then, the only difference between $\mathcal{B}'$ and $\mathcal{B}''$ is that $\mathcal{B}'$ check whether there exists $\boldsymbol{s}_i \neq \hat{\boldsymbol{s}}_i$, while $\mathcal{B}''$ check whether there exists $\boldsymbol{r}_i \neq \hat{\boldsymbol{s}}_i$. Since Φ has no fixed point over $\mathbf{D}$, we know $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) \neq (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ if $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) \in \mathbf{D}$. Therefore, for each $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) \in \mathbf{D}$, at least one of $\mathcal{B}'$ and $\mathcal{B}''$ wins, which implies Fact 3.

However, if we use Φ from Equation 11.1, this extended argument does not give us any benefit. Indeed, Φ as define is a bijection over $\mathbf{D}$, and therefore, the relaxed condition $\mathcal{P}_{|\mathbf{D}} = \Phi(\mathcal{P}_{|\mathbf{D}})$ still requires $\mathcal{P}$ to be a uniform distribution over $\mathbf{D}$. Also, it is unclear whether there are other ways to construct such Φ.

FURTHER RELAXATION ON THE REQUIREMENTS OF Φ. The way out from the above situation is that that the two distributions do not need to be exactly identical. In fact, this condition is only needed by Fact 1. Concretely, we relax the condition that $\mathcal{P}_{|\mathbf{D}} = \Phi(\mathcal{P}_{|\mathbf{D}})$ to requiring that the Rényi divergence of $\mathcal{P}_{|\mathbf{D}}$ from $\Phi(\mathcal{P}_{|\mathbf{D}})$ is small, denoted as $R_\alpha\left(\Phi(\mathcal{P}_{|\mathbf{D}}) \| \mathcal{P}_{|\mathbf{D}}\right)$, where $\alpha > 1$ is a parameter we can choose. Then, due to the property of Rényi divergence (formally see Lemma 11.2.5),

$$\mathsf{PWin}_{\mathcal{B}'} \leqslant \left(R_\alpha\left(\Phi(\mathcal{P}_{|\mathbf{D}}) \| \mathcal{P}_{|\mathbf{D}}\right) \cdot \mathsf{PWin}_{\mathcal{B}}\right)^{(\alpha-1)/\alpha} .$$

Combining with Fact 2 and 3, it gives

$$\mathsf{PWin}_{\mathcal{A}} \leqslant \mathsf{PWin}_{\mathcal{B}} + \left(R_{\alpha}\left(\Phi(\mathcal{P}_{|\mathbf{D}}) \| \mathcal{P}_{|\mathbf{D}}\right) \cdot \mathsf{PWin}_{\mathcal{B}}\right)^{(\alpha-1)/\alpha} ,$$

which upper bounds the winning probability of $\mathcal{A}$ by the advantage of solving the MSIS problem. We note that it is possible to get analogous statements using other distance measures, but they fail to give equally good parameters in our application scenarios.

Also, we emphasize that although both our work and [41] use Rényi divergence, the latter work bounds Rényi divergence between the *views* of $\mathcal{A}$, when run with different secrets, whereas we bound the Rényi divergence between distributions of *secrets* before and after applying the Φ map. The context where we use Rényi divergence is also very different from that of other works in lattice-based cryptography.

11.1.2 Step 2: Upper bounding $R_{\alpha}\left(\Phi(\mathcal{P}_{|\mathbf{D}}) \| \mathcal{P}_{|\mathbf{D}}\right)$

It is left to show that $R_{\alpha}\left(\Phi(\mathcal{P}_{|\mathbf{D}}) \| \mathcal{P}_{|\mathbf{D}}\right)$ is bounded, for the specific construction of Φ given in Equation 11.1, and again assuming a suitable Δ exists. The intuition that the two distributions should be close is that Φ is local in the sense that it maps each element to a nearby one given $\|(u_1\Delta, \ldots, u_Q\Delta)\|$ is small. Since each secret is sampled from a discrete Gaussian distribution, elements that are close in distance have similar probabilities of being sampled. However, turning this idea into a proof is still challenging, as both $\mathcal{P}_{|\mathbf{D}}$ and $\Phi(\mathcal{P}_{|\mathbf{D}})$ are not "well-structured" distributions, since both the set $\mathbf{D}$ and $\boldsymbol{u}$ can be arbitrarily determined by $\mathcal{A}$.

Our key insight here is that $\mathbf{D}$ can be decomposed into several disjoint sets such that the distributions conditioned on each set is a one-dimensional discrete Gaussian. Then, we can upper bound the Rényi divergence of the distributions conditioned on each set, which implies an upper bound on $R_{\alpha}\left(\Phi(\mathcal{P}_{|\mathbf{D}}) \| \mathcal{P}_{|\mathbf{D}}\right)$.

More precisely, for each $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in \mathbf{D}$, our definition of Φ in Equation 11.1 also ensures that $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ is identical to $\mathrm{View}_{\mathcal{A}}(\boldsymbol{s}_1 + ku_1 \cdot \Delta, \ldots, \boldsymbol{s}_Q + ku_Q \cdot \Delta)$ for any $k \in \mathbb{Z}$, where $\boldsymbol{u}$ is the vector output by $\mathcal{A}$ given secrets being $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Denote $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q] := \{(\boldsymbol{s}_1 + ku_1 \cdot \Delta, \ldots, \boldsymbol{s}_Q + ku_Q \cdot \Delta) | k \in \mathbb{Z}\}$. We can show that Φ is a bijection over $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]$, and all the $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]$ sets are either the same or disjoint with each other. Therefore, we just need to bound

$$R_{\alpha}\left(\Phi(\mathcal{P}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]}) \| \mathcal{P}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]}\right)$$

for each $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in \mathbf{D}$.

Now the problem is much easier since $\mathcal{P}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]}$ is well structured: each $\mathcal{P}$ is a discrete Gaussian distribution over R^{Qm}, and $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]$ is a one-dimensional lattice over R^{Qm}. In fact, we can transform $\mathcal{P}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]}$ into a discrete Gaussian over $\mathbb{Z}$ with standard deviation

$$\|(u_1/\sigma_1 \cdot \Delta, \ldots, u_Q/\sigma_Q \cdot \Delta))\|^{-1} \ ,^{4}$$

while $\Phi(\mathcal{P}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]})$ is exactly the same distribution shifted by 1. Therefore, we can use the upper bound from [121] (see also Lemma 11.2.7) of the Rényi divergence.

We refer to Section 12.3 for the proof details.

Further optimizations. We explain in this subsection two further optimizations we do to the reduction, which give us better parameters.

OUTPUTTING ONLY ONE SOLUTION. We observe that the reduction $\mathcal{B}$ uses only one of the solutions output by $\mathcal{A}$, which means most of the solutions are actually redundant. In fact, the reduction still works if $\mathcal{A}$ outputs a single solution $(i, \hat{\boldsymbol{s}})$ for the i-th challenge satisfying $u_i \neq 0$ and $\mathcal{B}$ just outputs $\hat{\boldsymbol{s}} - \boldsymbol{s}_i$ if $\hat{\boldsymbol{s}} \neq \boldsymbol{s}_i$. This is because for each $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in \mathbf{D}$ and $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) = \Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$, we have $\boldsymbol{s}_i \neq \boldsymbol{s}_i + u_i \cdot \Delta = \boldsymbol{s}'_i$ if $u_i \neq 0$, and thus $\mathcal{B}$ wins in at least one of the cases if $\mathcal{A}$ outputs a solution for the i-th challenge. This leads us to define our AOM-MISIS problem, where $\mathcal{A}$ is only required to output a single solution $\hat{\boldsymbol{s}}$ together with a relaxation vector $\hat{\boldsymbol{b}}$ such that $\sum_{i\in[Q]} \hat{b}_i \boldsymbol{t}_i = A\hat{\boldsymbol{s}}$. Similarly, the requirement on the special solution becomes $\sum_{i\in[Q]} u_i \hat{b}_i \neq 0$. The relaxation vector $\hat{\boldsymbol{b}}$ is needed when reducing the security of the threshold signatures to the hardness of AOM-MISIS.

REDUCING THE NORM OF Δ. One question we do not answer in the above discussion is how to guarantee the existence of a small non-zero vector $\Delta \in R^m$ such that $A\Delta = 0$. Such Δ must exist in $\mathscr{B}^m_\beta$ if $|\mathscr{B}^m_{\beta/2}| > q^{kN}$, which implies $\beta = \Omega(q^{k/m})$. However, this results in a large $\|\Delta\|$ negatively impacting both the Rényi divergence bound and σ_i, which depends linearly on $\|\Delta\|$.

We can get a smaller $\|\Delta\|$ by relying on the MLWE assumption to embed a small Δ into A as a (very minimal) trapdoor. We sample A as $[\boldsymbol{d}|D|\mathbb{I}_k]$, where $D \leftarrow\!\!\$\ R_q^{(m-k-1)\times k}$ and $\boldsymbol{d} = D\boldsymbol{a} + \boldsymbol{e}$ with $\boldsymbol{a} \leftarrow\!\!\$\ \mathscr{B}^{m-k-1}_{\beta_{\mathsf{lwe}}}$ and $\boldsymbol{e} \leftarrow\!\!\$\ \mathscr{B}^{k}_{\beta_{\mathsf{lwe}}}$. Then, we can let $\Delta = (1, -\boldsymbol{a}, -\boldsymbol{e})$, which satisfies $A\Delta = 0$, and $\|\Delta\| \leqslant \sqrt{m}\beta_{\mathsf{lwe}}$, which is much smaller than $q^{k/m}$. By the MLWE assumption, $[\boldsymbol{d}|D]$ is computationally indistinguishable from a matrix uniformly sampled from $R_q^{(m-k)\times k}$.

---

[4] This is the reason why we bound $\|u_1/\sigma_1, \ldots, u_Q/\sigma_Q\|$ in the winning condition.

Two remarks are in order. The first is that this trick would not have worked to improve the parameters of [41] directly, as they need something stronger than the existence of a small Δ. Second, the way we use MLWE here is different than its use to improve parameters in prior works on lattice-based signatures and threshold signatures (e.g. [98, 34, 54]). It is really tailored at supporting our tighter analysis of the reduction via the embedding of a small enough Δ.

11.2 Preliminaries

11.2.1 Polynomial Rings

Let q be an odd prime and N be a power of 2. We denote the ring $R := \mathbb{Z}[X]/(X^N + 1)$, contained in the cyclotomic field $K := \mathbb{Q}[X]/(X^N + 1)$, and let $R_q := R/qR \cong \mathbb{Z}_q[X]/(X^N + 1)$. Denote $K_{\mathbb{R}} := \mathbb{R} \otimes K \cong \mathbb{R}[X]/(X^N + 1)$. For an element $v \in K_{\mathbb{R}}$, where $v = \sum_{i=0}^{N-1} v_i X^i$, we denote its conjugate as $v^* = \sum_{i=0}^{N-1} -v_i X^{N-i}$. We use ϕ to denote the coefficient embedding that embeds $K_{\mathbb{R}}$ in $\mathbb{R}^N$, and ϕ maps v to vector $(v_0, \ldots, v_{N-1}) \in \mathbb{R}^N$. When applying ϕ to a vector $\boldsymbol{v} \in K_{\mathbb{R}}^m$, ϕ maps $\boldsymbol{v}$ to a vector in $\mathbb{R}^{mN}$ by applying ϕ to each entry of $\boldsymbol{v}$. The map ϕ is a bijection, and we denote its inverse by ϕ^{-1}. An ℓ_p-norm of $\boldsymbol{v} \in K_{\mathbb{R}}^m$ is given by

$$\|\boldsymbol{v}\|_p := \|\phi(\boldsymbol{v})\|_p = \left(\sum_{i=1}^{m} \sum_{j=0}^{N-1} |v_{i,j}|^p \right)^{\frac{1}{p}},$$

where $v_{i,j}$ denotes the coefficient of X^j of the i-th entry of $\boldsymbol{v}$. Additionally, the ℓ_∞-norm of $\boldsymbol{v}$ is defined as $\|\boldsymbol{v}\|_\infty := \max_{i \in [m], j \in [0..N-1]} |v_{i,j}|$. For the ℓ_2-norm, we omit the subscript and denote $\|\boldsymbol{v}\|$ as the ℓ_2-norm of $\boldsymbol{v}$. Denote the conjugate transpose of $\boldsymbol{v} \in K_{\mathbb{R}}^m$ as $\boldsymbol{v}^\dagger := (\boldsymbol{v}^*)^T$. We define the inner product of two vectors $\boldsymbol{v}, \boldsymbol{v}' \in K_{\mathbb{R}}^m$ as $\langle \boldsymbol{v}, \boldsymbol{v}' \rangle := \phi(\boldsymbol{v})^T \phi(\boldsymbol{v}') = \langle \phi(\boldsymbol{v}), \phi(\boldsymbol{v}') \rangle$. We have $\|\boldsymbol{v}\| = \langle \boldsymbol{v}, \boldsymbol{v} \rangle$. We say $\boldsymbol{v}$ is a unit vector if $\|\boldsymbol{v}\| = 1$.

For any integer $p > 0$ and any $x \in \mathbb{Z}_p$, denote $\bar{x} \in \mathbb{Z}$ to be the lift of x such that $\bar{x} \in [0..p-1]$ and $\bar{x} = x \mod p$. We use $\lfloor \cdot \rceil : \mathbb{R} \to \mathbb{Z}$ to denote the rounding operator that maps any $x \in \mathbb{R}$ to $\lfloor x + 1/2 \rfloor$. For any integers $v > 0$ and $q > 2^v$, denote $q_v = \lfloor q/2^v \rfloor$ and denote $\lfloor \cdot \rceil_v : \mathbb{Z}_q \to \mathbb{Z}_{q_v}$ a function that maps $x \in \mathbb{Z}_q$ to $\lfloor \bar{x}/2^v \rceil \in \mathbb{Z}_{q_v}$. These operations can be extended an element x in R or R_q by applying them to each coefficient of x.

Also, we define a map ϕ_{M} that maps each element in $K_{\mathbb{R}}$ to a matrix in $\mathbb{R}^{N \times N}$ as follows. Let $M_X := \begin{pmatrix} \mathbf{0} & -1 \\ \mathbb{I}_{N-1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^N$, where I_{N-1} is the identity matrix in $\mathbb{R}^{N-1}$. For each $v \in K_{\mathbb{R}}$,

$\phi_{\mathsf{M}}(v) := \sum_{i=0}^{N-1} v_i M_X^i$, which can be viewed as the matrix representation of v. In particular, for ϕ and ϕ_{M}, the following properties hold: for any $v, v' \in K_{\mathbb{R}}$, $\phi_{\mathsf{M}}(v^*) = \phi_{\mathsf{M}}(v)^T$, $\phi_{\mathsf{M}}(vv') = \phi_{\mathsf{M}}(v)\phi_{\mathsf{M}}(v')$ and $\phi_{\mathsf{M}}(v)\phi(v') = \phi(vv')$. We extend the above definitions to R_q by representing each $v \in R_q$ as $v = \sum_{i=0}^{N-1} v_i X^i$, where $v_i \in \{-(q-1)/2, \ldots, (q-1)/2\}$.

For a matrix $M \in K_{\mathbb{R}}^{m \times m}$, we denote its conjugate transpose as $M^\dagger = (M^*)^T$, and we say M is *hermitian* if $M = M^\dagger$. We say M is *positive definite* if and only if M is hermitian and for all $\boldsymbol{x} \in K_{\mathbb{R}}^m \backslash \{\mathbf{0}\}$, $\langle \boldsymbol{x}, M\boldsymbol{x} \rangle > 0$, or equivalently, $\phi_{\mathsf{M}}(M)$ is positive definite. Also, denote $\sigma_{\min}(M) := \inf_{\boldsymbol{x} \in K_{\mathbb{R}}^m, \|\boldsymbol{x}\|=1} \langle \boldsymbol{x}, M\boldsymbol{x} \rangle$ as the smallest singular value of M and $\sigma_{\max}(M) := \sup_{\boldsymbol{x} \in K_{\mathbb{R}}^m, \|\boldsymbol{x}\|=1} \langle \boldsymbol{x}, M\boldsymbol{x} \rangle$ as the largest singular value of M.

We borrow the following lemma from [22], which establishes the property of the set of signed monomials $\mathcal{S}_{\mathsf{b}} := \{\pm 1, \ldots, \pm X^{N-1}\} \subseteq R_q$.

Lemma 11.2.1 (Lemma 3.1 of [22])**.** *Let $\mathcal{S}_{\mathsf{b}} := \{\pm 1, \ldots, \pm X^{N-1}\} \subseteq R_q$. For any $b, \bar{b} \in \mathcal{S}_{\mathsf{b}}$ such that $b \neq \bar{b}$, there exists $\gamma \in R$ such that $(b - \bar{b})\gamma = 2 \mod q$ and γ is a polynomial with coefficients only in $\{-1, 0, 1\}$.*

11.2.2 Lattices and Discrete Gaussian Distributions

In this subsection, we give definitions for lattices and discrete Gaussian distributions over $\mathbb{R}$ and $K_{\mathbb{R}}$. An m-dimensional lattice Λ over $\mathbb{Z}$ (resp. R) is a discrete additive subgroup of $\mathbb{Z}$ (resp. R). Equivalently, $\Lambda = \mathcal{L}(\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}) := \{\sum_{i \in [k]} x_i \boldsymbol{b}_i : x_i \in \mathbb{Z}\}$ for a set of linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \in \mathbb{Z}^m$ (resp. R^m), which is referred to as a basis of Λ. The size k is the *rank* of the lattice Λ. We say Λ is a *full rank* lattice if $k = m$ (resp. $k = mN$ for Λ over R). For any $a \in \mathbb{Z}^m$ (resp. R^m), $\Lambda + a$ is a *coset* of Λ. The *dual lattice* of Λ is denoted as $\Lambda^* = \{\boldsymbol{x} \in \mathsf{Span}(\Lambda) : \forall\ \boldsymbol{y} \in \Lambda, \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}\}$. A Λ-subspace is the linear span of some subset of Λ, i.e., a subspace S such that $S = \mathsf{Span}(S \cap \Lambda)$.

For a matrix $A \in R_q^{k \times m}$, we define the R-lattice $\Lambda_q^\perp(A) \subseteq R^m$ as

$$\Lambda_q^\perp(A) := \{\boldsymbol{x} \in R^m : A\boldsymbol{x} = 0 \bmod q\} .$$

We know $\Lambda_q^\perp(A)$ has full-rank since $qR^m \subseteq \Lambda_q^\perp(A)$.

For a positive definite matrix $\Sigma \in \mathbb{R}^{m \times m}$ (resp. an invertible positive definite matrix $\Sigma \in K_{\mathbb{R}}^{m \times m}$) and a vector $\boldsymbol{c} \in \mathbb{R}^n$ (resp. $K_{\mathbb{R}}^m$), we define the function $\rho_{\Sigma, \boldsymbol{c}}$ over $\mathbb{R}^m$ (resp. $K_{\mathbb{R}}^m$) as

$$\rho_{\Sigma, \boldsymbol{c}}(\boldsymbol{x}) := \exp\left(-\pi \left\langle \boldsymbol{x} - \boldsymbol{c}, \Sigma^{-1}(\boldsymbol{x} - \boldsymbol{c}) \right\rangle\right) .$$

Then, we denote $\mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}$ as the discrete Gaussian distribution over a lattice coset $\Lambda + \boldsymbol{a} \subseteq \mathbb{Z}^m$ (resp. R^m) with covariance matrix Σ, centered at $\boldsymbol{c} \in \mathbb{R}^m$, where for $\boldsymbol{x} \in \Lambda + \boldsymbol{a}$, we define

$$\mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}(\boldsymbol{x}) := \mathsf{Pr}[\boldsymbol{x} \leftarrow_{\$} \mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}] = \frac{\rho_{\Sigma,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{\Sigma,\boldsymbol{c}}(\Lambda+\boldsymbol{a})}$$

where $\rho_{\Sigma,\boldsymbol{c}}(\Lambda+\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\Lambda+\boldsymbol{a}} \rho_{\Sigma,\boldsymbol{c}}(\boldsymbol{x})$. For $\Lambda+\boldsymbol{a} \subseteq R^m$, we denote $\mathscr{D}^{m,\bmod q}_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}(\boldsymbol{x})$ as the distribution of $(\boldsymbol{x} \bmod q) \in R^m_q$ for $\boldsymbol{x}$ sampled from $\mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}$.

Also, we make some remarks about the notations we will use throughout the paper. When $\Sigma = \sigma^2\mathbb{I}_m$ for $\sigma \in \mathbb{R}$, we will use $\rho_{\sigma,\boldsymbol{c}}$ and $\mathscr{D}^m_{\Lambda+\boldsymbol{a},\sigma,\boldsymbol{c}}$ as $\rho_{\Sigma,\boldsymbol{c}}$ and $\mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}$, respectively. If the center $\boldsymbol{c} = 0$, then we omit the subscript $\boldsymbol{c}$ from $\rho_{\Sigma,\boldsymbol{c}}$ and $\mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}$. Moreover, when $\Lambda + \boldsymbol{a} = \mathbb{Z}^m$ (resp. $\Lambda + \boldsymbol{a} = R^m$), we omit $\Lambda + \boldsymbol{a}$ from the subscript of $\mathscr{D}^m_{\Lambda+\boldsymbol{a},\Sigma,\boldsymbol{c}}$.

The smoothing parameter of a lattice Λ with respect to $\varepsilon > 0$, denoted by $\eta_\varepsilon(\Lambda)$, is the smallest $s > 0$ such that $\rho_{1/s}(\Lambda^*\backslash\{0\}) \leqslant \varepsilon$. Throughout the paper, we set $\varepsilon = 2^{-2\kappa}$.

We borrow the following lemma from [6] that bounds the ℓ_2-norm of discrete Gaussian random variables and adapt it to lattices over $K_{\mathbb{R}}$.

Lemma 11.2.2 (Lemma 3 of [6] adapted to $K_{\mathbb{R}}$)**.** *For any $\varepsilon \in (0,1)$, a lattice $\Lambda \subseteq R^m$, $\boldsymbol{c} \in K^m_{\mathbb{R}}$, and $\sigma \geqslant \eta_\varepsilon(\Lambda)$, then*

$$\mathsf{Pr}[\|\boldsymbol{x} - \boldsymbol{c}\| \geqslant \sigma\sqrt{mN} \ : \ \boldsymbol{x} \leftarrow_{\$} \mathscr{D}_{\Lambda,\sigma,\boldsymbol{c}}] \leqslant \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-mN} \ .$$

We also borrow the following lemma from [76] that show upper bounds of smoothing parameters for general lattices.

Lemma 11.2.3 (Lemma 2.6 of [76])**.** *For any full-rank lattice Λ in $\mathbb{R}^m$ and $\varepsilon > 0$, $\eta_\varepsilon(\Lambda) \leqslant \frac{\sqrt{\log(2m(1+1/\varepsilon))/\pi}}{\lambda^\infty_1(\Lambda^*)}$, where $\lambda^\infty_1(\Lambda^*)$ denotes the ℓ_2 norm of the shortest non-zero vector in the ℓ_∞ norm in the dual lattice Λ^*.*

11.2.3 Assumptions

We recall the module short integer solution (MSIS) problem and the module learning with error (MLWE) problem (defined in Figure 11.1). The advantage of $\mathcal{A}$ for the MSIS problem is defined as

$$\mathsf{Adv}^{\mathrm{msis}}_{q,N,k,m,\beta}(\mathcal{A}) := \mathsf{Pr}\left[\mathrm{MSIS}^{\mathcal{A}}_{q,N,k,m,\beta} = 1\right] \ .$$

The advantage of $\mathcal{A}$ for the MLWE problem is defined as

$$\mathsf{Adv}^{\mathrm{mlwe}}_{q,N,k,m,\beta}(\mathcal{A}) := \mathsf{Pr}\left[\mathrm{MLWE}^{\mathcal{A}}_{q,N,k,m,\beta} = 1\right] \ .$$

Game $\mathrm{MSIS}^{\mathcal{A}}_{q,N,k,m,\beta}$:	Game $\mathrm{MLWE}^{\mathcal{A}}_{q,N,k,m,\beta}$:
$\overline{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$	$\overline{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$; $A \leftarrow [\overline{A}\|\mathbb{I}_k]$
$A \leftarrow [\overline{A}\|\mathbb{I}_k]$	$\boldsymbol{s} \leftarrow\!\!\$\ \mathscr{B}_\beta^m$; $\boldsymbol{t}_0 \leftarrow A\boldsymbol{s}$; $\boldsymbol{t}_1 \leftarrow\!\!\$\ R_q^k$
$\boldsymbol{x} \leftarrow \mathcal{A}(A)$	$b \leftarrow\!\!\$\ \{0,1\}$
Return $(\|\boldsymbol{x}\| \leqslant \beta \ \wedge\ A\boldsymbol{x} = 0)$	$\hat{b} \leftarrow \mathcal{A}(A, t_b)$
	Return $(\hat{b} = b)$

Figure 11.1: The module-SIS and module-LWE problems, where $R := \mathbb{Z}[X]/(X^N+1)$, $R_q := R/qR$ and $\mathscr{B}^m_{\beta_{\mathsf{lwe}}} := \{\boldsymbol{x} \in R^m |\ \|\boldsymbol{x}\|_\infty \leqslant \beta_{\mathsf{lwe}}\}$.

11.2.4 Rényi Divergence

We define the notion of Rényi Divergence between two distributions P, Q which we will use in our analysis of the scheme.

Definition 11.2.4 (Rényi Divergence)**.** *Let P, Q be two discrete probability distributions such that $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$. We define the Rényi Divergence of order α, for $\alpha \in (1,\infty)$ as $R_\alpha(P\|Q) := \left(\sum_{x\in Supp(P)} \frac{P(x)^\alpha}{Q(x)^{\alpha-1}}\right)^{\frac{1}{\alpha-1}}$.*

The following lemma gives basic properties of the Rényi Divergence.

Lemma 11.2.5 (Lemma 4.1 of [93])**.** *Let $\alpha \in (1,\infty)$ and P, Q be discrete probability distributions with $\mathrm{Supp}(P) \subseteq \mathrm{Supp}(Q)$. Then, the following properties hold:*

- ***Data Processing Inequality:*** *$R_\alpha\left(P^f\|Q^f\right) \leqslant R_\alpha(P\|Q)$ for any function f, where P^f (and Q^f) denotes the distribution which samples $x \leftarrow\!\!\$\ P$ ($x \leftarrow\!\!\$\ Q$) and outputs $f(x)$.*

- ***Probability Preservation:*** *Let $E \subseteq \mathrm{Supp}(Q)$ be an arbitrary event. Then, for $\alpha \in (1,\infty)$,*

$$\mathsf{Pr}_{x \leftarrow\!\!\$\ P}[E] \leqslant \left(\mathsf{Pr}_{x \leftarrow\!\!\$\ Q}[E] R_\alpha(P\|Q)\right)^{(\alpha-1)/\alpha}.$$

Also, we show the following property.

Lemma 11.2.6. *Let P and Q denote two distributions over the union of a countable number of disjoint sets $\{S_i\}_{i\in U}$ such that $P(S_i) = Q(S_i)$ for any $i \in U$. Then, for any $\alpha > 1$, if there exists δ such that $R_\alpha\left(P_{|S_i} \| Q_{|S_i}\right) \leqslant \delta$ for any $i \in U$, then $R_\alpha\left(P\|Q\right) \leqslant \delta$.*

Proof.

$$\begin{aligned} R_\alpha\left(P\|Q\right)^{\alpha-1} &= \sum_{x\in\mathrm{Supp}(P)} \frac{P(x)^\alpha}{Q(x)^{\alpha-1}} = \sum_{i\in U}\sum_{x\in S_i} \frac{P(x)^\alpha}{Q(x)^{\alpha-1}} \\ &= \sum_{i\in U} Q(S_i) \sum_{x\in S_i} \frac{(P(x)/P(S_i))^\alpha}{(Q(x)/Q(S_i))^{\alpha-1}} \\ &= \sum_{i\in U} Q(S_i) R_\alpha\left(P_{|S_i}\|Q_{|S_i}\right)^{\alpha-1} \leqslant \sum_{i\in U} Q(S_i)\delta^{\alpha-1} = \delta^{\alpha-1}\,, \end{aligned}$$

which concludes the lemma. □ □

We borrow the following lemma from [121], which upperbounds the Rényi Divergence between two discrete Gaussian distributions with different centers.

Lemma 11.2.7 (Lemma 5 of [121])**.** *For any m-dimensional lattice $\Lambda \subseteq \mathbb{R}^m$, $\sigma > 0$, and two vectors $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^m$, let $P = \mathscr{D}^m_{\Lambda,\sigma,\mathbf{c}}$ and $Q = \mathscr{D}^m_{\Lambda,\sigma,\mathbf{c}'}$. If $\mathbf{c}, \mathbf{c}' \in \Lambda$, set $\varepsilon = 0$. Otherwise, fix $\varepsilon \in (0,1)$ and assume $\sigma \geqslant \eta_\varepsilon(\Lambda)$. Then,*

$$R_\alpha\left(P\|Q\right) \leqslant \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\frac{\alpha}{\alpha-1}} \exp\left(\alpha\pi\frac{\|\mathbf{c}-\mathbf{c}'\|^2}{\sigma^2}\right)\,.$$

Chapter 12

THE ALGEBRAIC ONE-MORE MISIS PROBLEM

In this chapter, we first formally define the algebraic one-more MISIS (AOM-MISIS) problem, then provide a comparison with the AOM-MLWE problem, and finally present a formal reduction from standard lattice problems to AOM-MISIS.

12.1 Definition of AOM-MISIS

The algebraic one-more MISIS game is defined in Figure 12.1. The game is defined implicitly over the cyclotomic ring $R = \mathbb{Z}[X]/(X^N+1)$, where N is a power of two, as well as the associated ring $R_q = R/qR \cong \mathbb{Z}_q[X]/(X^N+1)$ for an odd prime q. The adversary $\mathcal{A}$ is given Q MISIS challenges $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_Q$ and the goal is to output a short solution $\hat{\boldsymbol{s}}$ for a linear combination of the challenges $\sum_{i\in[Q]} \hat{b}_i \boldsymbol{t}_i$, where the norm of $\hat{\boldsymbol{b}}$ is suitably bounded. The adversary can also issue queries to the oracle PI which reveal linear combinations of the secrets. To win, the adversary also needs to output an additional vector $\boldsymbol{u}$ which is orthogonal to all queries to PI, but not to the vector $\hat{\boldsymbol{b}}$. We can also view the existence of such $\boldsymbol{u}$ as a constraint on the PI queries. We refer the reader to the introduction for some additional intuition. Here, for $\mathsf{par} = (q, N, k, m, Q, (\sigma_i)_{i\in[Q]}, \beta_{\mathsf{s}}, \beta_{\mathsf{b}}, \beta_{\mathsf{u}})$, we define the *advantage* of an adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{aom-misis}}_{\mathsf{par}}(\mathcal{A}, \kappa) = \mathsf{Pr}\left[\text{AOM-MISIS}^{\mathcal{A}}_{\mathsf{par}}(\kappa) = 1\right] . \tag{12.1}$$

We note that we slightly abuse notation in the asymptotic definition of the game, since it is understood that all parameters grow with κ, including Q, and thus the notation $(\sigma_i)_{i\in Q}$ is not entirely well-defined. This will not be an issue in actual use cases, and we dispense with a more rigorous definition.

Remark 12.1. *We note that for each $i \in [Q]$, it is equivalent to sample $\boldsymbol{s}_i$ from $\mathscr{D}^{m,\bmod q}_{\sigma_i}$ instead of $\mathscr{D}^m_{\sigma_i}$, since the view of $\mathcal{A}$ remains unchanged. With this insight, it is natural to allow σ_i to be ∞, in which case $\boldsymbol{s}_i$ is sampled uniformly from R^m_q, $1/\sigma_i = 0$, and we require $\hat{b}_i = 0$ in the winning condition (since, o.w., $\left\|\hat{b}_i \cdot \sigma_i\right\|_1$ is ∞). This extension is used, in particular, in our security*

Game AOM-MISIS$^{\mathcal{A}}_{q,N,k,m,Q,(\sigma_i)_{i\in[Q]},\beta_{\mathsf{s}},\beta_{\mathsf{b}},\beta_{\mathsf{u}}}(\kappa)$:

$B \leftarrow \varnothing$

$\overline{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$; $A \leftarrow [\overline{A}|\mathbb{I}_k]$

For $i \in [Q]$,

$\boldsymbol{s}_i \leftarrow \mathscr{D}^m_{\sigma_i}$; $\boldsymbol{t}_i \leftarrow A\boldsymbol{s}_i \bmod q$

$(\hat{\boldsymbol{s}}, \hat{\boldsymbol{b}}, \boldsymbol{u}) \leftarrow \mathcal{A}^{\mathrm{PI}}(A, \{\boldsymbol{t}_i\}_{i\in[Q]})$ // $\hat{\boldsymbol{s}} \in R^m, \hat{\boldsymbol{b}}, \boldsymbol{u} \in R^Q$

Return $(\forall \boldsymbol{b} \in B\ :\ \boldsymbol{b}^T\boldsymbol{u} = 0 \mod q) \ \wedge\ \hat{\boldsymbol{b}}^T\boldsymbol{u} \neq 0$

$\wedge\ \|(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)\|_2 \leqslant \beta_{\mathsf{u}}$

$\wedge\ \|\hat{\boldsymbol{s}}\|_2 \leqslant \beta_{\mathsf{s}}\ \wedge\ \left\|(\hat{b}_1\cdot\sigma_1, \ldots, \hat{b}_Q\cdot\sigma_Q)\right\|_1 \leqslant \beta_{\mathsf{b}}$

$\wedge\ \sum_{i\in[Q]} \hat{b}_i\boldsymbol{t}_i = A\hat{\boldsymbol{s}} \mod q$

Oracle PI($\boldsymbol{b} \in R^Q$) :

$B \leftarrow B \cup \{\boldsymbol{b}\}$

Return $\sum_{i\in[Q]} b_i\boldsymbol{s}_i \mod q\ (\in R_q^m)$

Figure 12.1: The AOM-MISIS game, where $R := \mathbb{Z}[X]/(X^N+1)$ and $R_q := R/qR$.

---

reduction of the CATZ construction (see Section 13.1.1), where the random values for generating the key shares are sampled uniformly from R_q^m.

12.2 Comparison with prior work

We present in Figure 12.2 the AOM-MLWE problem proposed by Espitau et al. [60]. In our formulation, the PI oracle corresponds to the $\mathcal{O}_{\mathsf{solve}}$ oracle in their notation, and the constraints on PI queries are explicitly stated within the game. More precisely, the constraints are:

- The number of PI queries is exactly $Q-1$.
- Denote $\begin{pmatrix}\boldsymbol{v}^T\\ \underline{D}\end{pmatrix} = [\boldsymbol{d}_1|\cdots|\boldsymbol{d}_{Q-1}]$, where $\boldsymbol{d}_i$ denotes the i-th PI query. $\underline{D}$ is an invertible matrix.
- Let $\boldsymbol{w} \leftarrow (\boldsymbol{v}^T\underline{D}^{-1})^T$. The ℓ_2-norm of each entry of $\boldsymbol{w}$ is bounded by β_{d}.

Game $\text{AOM-MLWE}^{\mathcal{A}}_{q,k,m,Q,(\sigma_i)_{i\in[Q]},\beta_{\mathsf{s}},\beta_{\mathsf{e}},\beta_{\mathsf{b}},\beta_{\mathsf{d}}}(\kappa)$:

$\phi_{\mathcal{M}} \leftarrow 0$

$\overline{A} \leftarrow_{\$} R_q^{k\times(m-k)}$; $A \leftarrow [\overline{A}|\mathbb{I}_k]$

For $i \in [Q]$ do $\boldsymbol{s}_i \leftarrow \mathscr{D}^m_{\sigma_i}$; $\boldsymbol{t}_i \leftarrow A\boldsymbol{s}_i \bmod q$

$(\hat{\boldsymbol{b}}, \hat{\boldsymbol{s}}_1, \ldots, \hat{\boldsymbol{s}}_Q) \leftarrow \mathcal{A}^{\text{PI}}(A, \{\boldsymbol{t}_i\}_{i\in[Q]})$ // $\hat{\boldsymbol{s}} \in R^m, \hat{\boldsymbol{b}}$

If $\phi_{\mathcal{M}} \neq Q-1$ then return 0

Parse $\begin{pmatrix} \boldsymbol{v}^T \\ \underline{D} \end{pmatrix} \leftarrow [\boldsymbol{d}_1|\cdots|\boldsymbol{d}_{Q-1}]$

If $\underline{D}^{-1}$ does not exist then return 0

$\boldsymbol{w} \leftarrow (\boldsymbol{v}^T\underline{D}^{-1})^T$

Return $(\forall i \in [Q-1] \ : \ \|w_i\| \leqslant \beta_{\mathsf{d}})$

$\quad \wedge \ \left\|(\hat{s}_{1,[m-k]}, \ldots, \hat{s}_{Q,[m-k]})\right\|_2 \leqslant \beta_{\mathsf{s}}$

$\quad \wedge \ \left\|(\hat{s}_{1,[(m-k+1)..m]}, \ldots, \hat{s}_{Q,[(m-k+1)..m]})\right\|_2 \leqslant \beta_{\mathsf{e}}$

$\quad \wedge \ (\forall i \in [Q] \ : \ b_i \neq 0 \ \wedge \ \|b_i\| \leqslant \beta_{\mathsf{b}} \ \wedge \ \hat{b}_i\boldsymbol{t}_i = A\hat{\boldsymbol{s}}_i \bmod q)$

Oracle $\text{PI}(\boldsymbol{b} \in R^Q)$:

$\phi_{\mathcal{M}} \leftarrow \phi_{\mathcal{M}} + 1$

$\boldsymbol{d}_{\phi_{\mathcal{M}}} \leftarrow \boldsymbol{b}$

Return $\sum_{i\in[Q]} b_i\boldsymbol{s}_i$

Figure 12.2: The AOM-MLWE game.

We also define, for $\mathsf{par} = (q, N, k, m, Q, (\sigma_i)_{i\in[Q]}, \beta_{\mathsf{s}}, \beta_{\mathsf{e}}, \beta_{\mathsf{b}}, \beta_{\mathsf{d}})$, the corresponding advantage

$$\mathsf{Adv}^{\text{aom-mlwe}}_{\mathsf{par}}(\mathcal{A}, \kappa) = \mathsf{Pr}\left[\text{AOM-MLWE}^{\mathcal{A}}_{\mathsf{par}}(\kappa) = 1\right] .$$

Compared with the AOM-MLWE problem, our problem differs in the following key aspects:

1. Our problem only asks the adversary to output one special solution $\hat{\boldsymbol{s}}$ for a linear combination of the challenges, while the AOM-MLWE problem demands one solution for each challenge. This makes our problem inherently easier to solve, as noted at the beginning of Section 11.1. Moreover, this relaxation simplifies the security reduction, as it only needs to extract one solution rather than multiple, making it easier to reduce from our assumption. Additionally, it enables better parameter selections, since the norm bound applies to a single solution instead of n solutions.

2. The constraints on the PI queries differ between AOM-MLWE and our problem. In particular, their constraints are a special case of ours: given the PI queries satisfying their constraints, we can verify that these queries also satisfy our constraints by setting $\boldsymbol{u} = (1, -\boldsymbol{w})$, since $\boldsymbol{u} \begin{pmatrix} \boldsymbol{v}^T \\ \underline{D} \end{pmatrix} = 0$. We will show this formally later in this section.

3. The norm bounding approach is different. In AOM-MLWE, the norm of each entry of $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{w}}$ is bounded individually. In contrast, we bound the norm of the entire vectors $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{u}}$, with each entry weighted by the standard deviation of the corresponding challenge. This adjustment leads to better parameter selections.

In short, our problem is easier to reduce from and enables better parameter selections. We formally show in Lemma 12.2.1 that the hardness of our problem implies the hardness of theirs, and we discuss the improvement in the parameter selections in Section 13.2.1, Remark 13.1.

Remark 12.2. *We note that Espitau et al. [60] propose an alternative way of defining constraints, which, like ours, requires the existence of a nonzero vector $\boldsymbol{u}$ that is orthogonal to all* PI *queries. The key difference is that they impose a bound on the norm of each entry of $\boldsymbol{u}$. We can show that the hardness of our problem also implies the hardness of theirs under these constraints. However, we omit a formal analysis here, as the proof idea is very similar, and this alternative version is not used to establish the security of their threshold signature scheme.*

Lemma 12.2.1. *For any* $\mathsf{par} = (q, N, k, m, Q, (\sigma_i)_{i\in[Q]}, \beta_\mathsf{s}, \beta_\mathsf{e}, \beta_\mathsf{b}, \beta_\mathsf{d})$ *and any adversary $\mathcal{A}$ playing the game* $\text{AOM-MLWE}^{\mathcal{A}}_{\mathsf{par}}$*, there exists an* AOM-MISIS *adversary $\mathcal{B}$ running in time roughly the same as $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{aom-mlwe}}_{\mathsf{par}}(\mathcal{A}, \kappa) \leqslant \mathsf{Adv}^{\text{aom-misis}}_{\mathsf{par}'}(\mathcal{B}, \kappa) \ ,$$

where $\mathsf{par}' = (q, N, k, m, Q, (\sigma_i)_{i\in[Q]}, \beta'_\mathsf{s} = \beta_\mathsf{s} + \beta_\mathsf{e}, \beta'_\mathsf{b} = \sqrt{N}\beta_\mathsf{b}\sigma_1, \beta_\mathsf{u})$ *and* $\beta_\mathsf{u} = 1/\sigma_1 + \beta_\mathsf{d}\sqrt{Q}/(\min_{i\in[2..Q]} \sigma_i)$.

Roughly, the proof idea is that for an adversary $\mathcal{A}$ that wins the game AOM-MLWE, $\mathcal{A}$ can win the game AOM-MISIS by outputting only the solution to the first challenge (i.e., outputting $\hat{s}_1$ and $(\hat{b}_1, 0, \ldots, 0)$) and outputting $\boldsymbol{u} \leftarrow (1, -\boldsymbol{v}^T\underline{D}^{-1})$. It is not hard to check $\boldsymbol{u}$ satisfies the constraints of the game AOM-MISIS. We do this explicitly in the following proof.

Proof of Lemma 12.2.1. For any adversary $\mathcal{A}$ described in the lemma, we construct $\mathcal{B}$ as follows. To start with, $\mathcal{B}$ runs $\mathcal{A}$ on its input $A, \{\boldsymbol{t}_i\}_{i\in[Q]}$ by forwarding all PI queries from $\mathcal{A}$ to its own PI oracle. After receiving $\mathcal{A}$'s output $(\hat{\boldsymbol{b}}, \hat{\boldsymbol{s}}_1, \ldots, \hat{\boldsymbol{s}}_Q)$, if $\mathcal{A}$ wins the AOM-MLWE game simulated by $\mathcal{B}$, $\mathcal{B}$ returns $((\hat{b}_1, 0, \ldots, 0), \hat{\boldsymbol{s}}_1, (1, -\boldsymbol{w}^T))$, where $\boldsymbol{w}$ is defined in the AOM-MLWE game. Otherwise $\mathcal{B}$ aborts.

We now show $\mathcal{B}$ wins the AOM-MISIS game given $\mathcal{A}$ wins by checking all the winning conditions of $\mathcal{B}$.

- For the first condition, since $(1, -\boldsymbol{w}^T) \cdot \begin{pmatrix} \boldsymbol{v}^T \\ \underline{D} \end{pmatrix} = \boldsymbol{v}^T - \boldsymbol{v}^T \underline{D}^{-1} \underline{D} = 0$, for any $i \in [Q-1]$, $(1, -\boldsymbol{w}^T) \cdot \boldsymbol{d}_i = 0$, so the first condition is satisfied.

- For the second condition, $(\hat{b}_1, 0, \ldots, 0) \begin{pmatrix} 1 \\ -\boldsymbol{w} \end{pmatrix} = \hat{b}_1 \neq 0$.

- For the third condition, $\|(1/\sigma_1, w_1/\sigma_2, \ldots, w_{Q-1}/\sigma_Q)\| \leqslant \frac{1}{\sigma_1} + \sum_{i\in[Q-1]} \frac{\|w_i\|}{\sigma_i} \leqslant \frac{1}{\sigma_1} + \frac{\beta_{\mathsf{d}}}{\min_{i\in[2..Q]} \sigma_i} = \beta_{\mathsf{u}}$.

- For the fourth condition, $\|\hat{\boldsymbol{s}}_1\| \leqslant \left\|\hat{s}_{1,[m-k]}\right\| + \left\|\hat{s}_{1,[(m-k+1)..k]}\right\| \leqslant \beta_{\mathsf{s}} + \beta_{\mathsf{e}} = \beta'_{\mathsf{s}}$.

- For the fifth condition, $\left\|(\hat{b}_1 \cdot \sigma_1, 0, \ldots, 0)\right\|_1 \leqslant \left\|\hat{b}_1 \cdot \sigma_1\right\|_1 \leqslant \sqrt{N}\sigma_1 \left\|\hat{b}_1\right\| = \beta'_{\mathsf{b}}$.

- For the final condition, $\hat{b}_1 \boldsymbol{t}_1 = A\hat{\boldsymbol{s}} \mod q$.

□

12.3 Reduction from MSIS and MLWE

This section shows our main result establishing hardness of AOM-MISIS from the hardness of MLWE and MSIS. (We remind the reader that we also provided a detailed overview of this proof in Section 11.1 above.)

Theorem 12.3.1. *For any $\varepsilon \in (0,1)$, $\alpha > 1$, any $\mathsf{par} = (q, N, k, m, Q, (\sigma_i)_{i\in[Q]}, \beta_{\mathsf{s}}, \beta_{\mathsf{b}}, \beta_{\mathsf{u}})$ such that $mN \geqslant 2\kappa$ and $\sigma_i \geqslant \sqrt{\log(6mN)/\pi}$, and any* AOM-MISIS *adversary $\mathcal{A}$, there exist a* MSIS

adversary $\mathcal{B}$ and two MLWE *adversaries $\mathcal{C}$ and $\mathcal{D}$, such that*

$$\mathsf{Adv}^{\text{aom-misis}}_{\mathsf{par}}(\mathcal{A},\kappa) \leqslant 2\delta_\alpha \left(\mathsf{Adv}^{\text{msis}}_{q,k,m,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{D},\kappa) + Q\cdot 2^{-2\kappa+2}\right)^{\frac{\alpha}{\alpha-1}} + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{C},\kappa)\ ,$$

where $\delta_\alpha = \frac{1+\varepsilon}{1-\varepsilon}\cdot\exp\left((\alpha-1)\pi^2/\log(2(1+1/\varepsilon))\right)$, $\beta_{\mathsf{sis}} = \beta_{\mathsf{s}} + \sqrt{mN}\beta_{\mathsf{b}}$, and $\beta_{\mathsf{lwe}} = 1/(\beta_{\mathsf{u}}\sqrt{mN}\sqrt{\log(2(1+1/\varepsilon))/\pi})$. The three adversaries have, roughly the same running time as that of $\mathcal{A}$.

Proof. We prove the theorem via the following hybrid.

$\mathbf{G}_0$: Same as the AOM-MISIS game.

$\mathbf{G}_1$: Same as $\mathbf{G}_0$ except that A is sampled with a short solution embedded, i.e., $A \leftarrow [D\boldsymbol{a}+\boldsymbol{e}|D|\mathbb{I}_k]$ for $(\boldsymbol{a},\boldsymbol{e}) \leftarrow\!\!\$\ \mathscr{B}^{m-1}_{\beta_{\mathsf{lwe}}}$, where $\mathscr{B}^{m-1}_{\beta_{\mathsf{lwe}}} := \{\boldsymbol{x} \in R^{m-1} |\ \|\boldsymbol{x}\|_\infty \leqslant \beta_{\mathsf{lwe}}\}$, and $D \leftarrow\!\!\$\ R_q^{k\times(m-k-1)}$. Since the only difference between $\mathbf{G}_0$ and $\mathbf{G}_1$ is that $\overline{A}$ is replaced with a MLWE challenge, there exists a MLWE adversary $\mathcal{C}$ such that

$$\mathsf{Adv}^{\mathbf{G}_0}(\mathcal{A},\kappa) \leqslant \mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A},\kappa) + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{C},\kappa)\ . \tag{12.2}$$

Consider a variant of the MSIS game, where A is sampled with a short solution embedded (same as the above $\mathbf{G}_1$), i.e., $A \leftarrow [D\boldsymbol{a}+\boldsymbol{e}|D|\mathbb{I}_k]$ for $(\boldsymbol{a},\boldsymbol{e}) \leftarrow\!\!\$\ \mathscr{B}^{m-1}_{\beta_{\mathsf{lwe}}}$, where $\mathscr{B}^{m-1}_{\beta_{\mathsf{lwe}}} := \{\boldsymbol{x} \in R^{m-1} |\ \|\boldsymbol{x}\|_\infty \leqslant \beta_{\mathsf{lwe}}\}$, and $D \leftarrow\!\!\$\ R_q^{k\times(m-k-1)}$. We refer to the game as td-MSIS. We construct $\mathcal{B}$ playing the td-MSIS game as follows. Given the td-MSIS challenge $A' \in R_q^{k\times(m-k)}$, $\mathcal{B}$ runs $\mathcal{A}$ by simulating the game $\mathbf{G}_1$ with $\mathcal{A}$ faithfully except that $\mathcal{B}$ sets $A \leftarrow A'$ instead of sampling A by itself. After receiving the output $(\hat{\boldsymbol{s}},\hat{\boldsymbol{b}},\boldsymbol{u})$ from $\mathcal{A}$, if $\mathcal{A}$ wins the game $\mathbf{G}_1$ simulated by $\mathcal{B}$, $\mathcal{B}$ outputs $\boldsymbol{x} \leftarrow \hat{\boldsymbol{s}} - \sum_{i\in[Q]}\hat{b}_i\boldsymbol{s}_i$. Otherwise, $\mathcal{B}$ aborts.

ANALYSIS OF $\mathcal{B}$. Similar to the hybrid between $\mathbf{G}_0$ and $\mathbf{G}_1$, the td-MSIS game is computationally indistinguishable from the MSIS game under the MLWE assumption. Therefore, there exists a MLWE adversary $\mathcal{D}$ such that

$$\mathsf{Adv}^{\text{td-msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) \leqslant \mathsf{Adv}^{\text{msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{D},\kappa)\ . \tag{12.3}$$

Claim 12.3.2. $\mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A},\kappa) \leqslant 2\delta_\alpha\left(\mathsf{Adv}^{\text{td-msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) + Q\cdot 2^{-2\kappa+2}\right)^{(\alpha-1)/\alpha}$.

We can conclude the proof since

$$\begin{aligned}\mathsf{Adv}^{\text{aom-misis}}_{\mathsf{par}}(\mathcal{A},\kappa) &\leqslant \mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A},\kappa) + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{C},\kappa)\\ &\leqslant 2\delta_\alpha\left(\mathsf{Adv}^{\text{td-msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) + Q\cdot 2^{-2\kappa+2}\right)^{(\alpha-1)/\alpha} + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{C},\kappa)\\ &\leqslant 2\delta_\alpha\left(\mathsf{Adv}^{\text{msis}}_{q,k,m,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{D},\kappa) + Q\cdot 2^{-2\kappa+2}\right)^{\frac{\alpha}{\alpha-1}}\\ &\quad + \mathsf{Adv}^{\text{mlwe}}_{q,k,m-1,\beta_{\mathsf{lwe}}}(\mathcal{C},\kappa)\ ,\end{aligned}$$

where the first inequality follows from Equation (12.2), the second inequality follows from the claim, and the third inequality follows from Equation (12.3). □

Proof of Claim 12.3.2. We first show that, except for negligible probability, $\mathcal{B}$ wins if its output vector $\boldsymbol{x}$ is non-zero. It is clear that if $\mathcal{A}$ wins the game $\mathbf{G}_1$ simulated by $\mathcal{B}$, by the winning conditions of the game, it holds that $A\boldsymbol{x} = A\hat{\boldsymbol{s}} - \sum_{i\in[Q]}\hat{b}_i\boldsymbol{t}_i = 0$.

Also, $\|\boldsymbol{x}\|$ is small except for negligible probability. Denote BNonce as the event that there exists $i \in [Q]$ such that $\|\boldsymbol{s}_i\| > \sigma_i\sqrt{mN}$ for $(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q)$ sampled as in the game $\mathbf{G}_1$. If BNonce does not occurs, we know

$$\begin{aligned}\|\boldsymbol{x}\| &\leqslant \|\hat{\boldsymbol{s}}\| + \sum_{i\in[Q]}\left\|\hat{b}_i\right\|_1\cdot\|\boldsymbol{s}_i\| \leqslant \beta_{\mathsf{s}} + \sum_{i\in[Q]}\left\|\hat{b}_i\right\|_1\cdot(\sqrt{mN}\sigma_i)\\ &= \beta_{\mathsf{s}} + \sqrt{mN}\left\|(\hat{b}_1\cdot\sigma_1,\ldots,\hat{b}_Q\cdot\sigma_Q)\right\|_1 = \beta_{\mathsf{s}} + \sqrt{mN}\beta_{\mathsf{b}} \leqslant \beta_{\mathsf{sis}}\ .\end{aligned}$$

Therefore, $\mathcal{B}$ wins if BNonce does not occur, which means $\mathsf{Adv}^{\text{td-msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) \geqslant \mathsf{Pr}[\mathcal{B}\text{ outputs }\boldsymbol{x}\neq 0] - \mathsf{Pr}[\mathsf{BNonce}]$. To bound $\mathsf{Pr}[\mathsf{BNonce}]$, since $\boldsymbol{s}_i$ is sampled from $\mathscr{D}^m_{\sigma_i}$ and $\sigma_i \geqslant \sqrt{\log(6mN)/\pi} = \eta_{\varepsilon'}(R^m)$ with $\varepsilon' = 1/2$, by Lemma 11.2.2, $\mathsf{Pr}[\|\boldsymbol{s}_i\| > \sigma_i\sqrt{mN}] \leqslant 3\cdot 2^{-mN} \leqslant 2^{-2\kappa+2}$. By the Union Bound, $\mathsf{Pr}[\mathsf{BNonce}] \leqslant \sum_{i\in[Q]}\mathsf{Pr}[\|\boldsymbol{s}_i\| > \sigma_i\sqrt{mN}] \leqslant Q\cdot 2^{-2\kappa+2}$. Therefore,

$$\mathsf{Adv}^{\text{td-msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B},\kappa) \geqslant \mathsf{Pr}[\mathcal{B}\text{ outputs }\boldsymbol{x}\neq 0] - Q\cdot 2^{-2\kappa+2}\ . \tag{12.4}$$

To bound the probability that the output of $\mathcal{B}$ is non-zero, consider a fixed randomness $(\boldsymbol{a},\boldsymbol{e},D)$ of td-MSIS. Also, w.l.o.g. assume $\mathcal{A}$ is deterministic. Then, the execution of $\mathcal{B}$ is determined by $\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q$. We define a map $\Phi_{\boldsymbol{a},\boldsymbol{e},D,\mathcal{A}} : R^{Qm} \to R^{Qm}$ as follows such that the view of $\mathcal{A}$ given $(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q)$ is exactly the same as that given $\Phi_{\boldsymbol{a},\boldsymbol{e},D,\mathcal{A}}(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q)$. Consider the execution given $(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q)$. To simplify notation, we omit the subscript of Φ for the rest of the proof. If $\mathcal{A}$ does not win the $\mathbf{G}_1$ simulated by $\mathcal{B}$, we set $\Phi(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q) = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q)$. Otherwise, we set

$\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) = (\boldsymbol{s}_1 + u_1\Delta, \ldots, \boldsymbol{s}_Q + u_Q\Delta)$, where $\boldsymbol{u} \in R^Q$ is output by $\mathcal{A}$ and $\Delta = (1, -\boldsymbol{a}, -\boldsymbol{e}) \in R^m$.

It is not hard to see that the view of $\mathcal{A}$ given $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ is identical to that given $\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. In particular, in the case that $\mathcal{A}$ wins the game $\mathbf{G}_1$, $\Phi(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ leads to the same view since $\sum_{i\in[Q]} b_i(\boldsymbol{s}_i + u_i\Delta) = \sum_{i\in[Q]} b_i\boldsymbol{s}_i + \Delta\sum_{i\in[Q]} b_i u_i = \sum_{i\in[Q]} b_i\boldsymbol{s}_i$ for any $(b_1, \ldots, b_Q) \in B$ and $A(\boldsymbol{s}_i + u_i\Delta) = A\boldsymbol{s}_i$ for any $i \in [Q]$ due to the fact that $A \cdot \Delta = [D\boldsymbol{a} + \boldsymbol{e}|D|\mathbb{I}_k] \cdot (1, -\boldsymbol{a}, -\boldsymbol{e})^T = \mathbf{0}$.

Also, it is not hard to see that Φ is a bijection. For $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \in R^{mQ}$, suppose $\Phi(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Since $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q)$ leads to the same view of $\mathcal{A}$ as $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$, we have either $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ in case that $\mathcal{A}$ does not win the game $\mathbf{G}_1$, or $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) = (\boldsymbol{s}_1 - u_1\Delta, \ldots, \boldsymbol{s}_Q - u_Q\Delta)$ in case that $\mathcal{A}$ wins the game $\mathbf{G}_1$ by outputting $\boldsymbol{u}$ given $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Therefore, such $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q)$ must exist uniquely, which means that Φ is a bijection.

In the game $\mathbf{G}_1$, the distribution of $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ is $\mathscr{D}_\Sigma^{mQ}$, where $\Sigma = \mathbb{I}_m \otimes \mathrm{diag}(\sigma_1^2, \ldots, \sigma_Q^2)$. Denote $P = \Phi(\mathscr{D}_\Sigma^{mQ})$. Following the idea described in Section 11.1.1, consider the following two adversaries $\mathcal{B}'$ and $\mathcal{B}''$.

- $\mathcal{B}'$ is the same as $\mathcal{B}$ except that $\mathcal{B}'$ samples $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ from P.

- $\mathcal{B}''$ is the same as $\mathcal{B}$ except that 1. $\mathcal{B}''$ samples $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) \leftarrow\!\!\$\ \mathscr{D}_\Sigma^{mQ}$ and computes secrets as $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \leftarrow \Phi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$; 2. after $\mathcal{A}$ returns, $\mathcal{B}''$ outputs $\boldsymbol{x} \leftarrow \hat{\boldsymbol{s}} - \sum_{i\in[Q]} \hat{b}_i\boldsymbol{r}_i$.

We note that $\mathcal{B}'$ and $\mathcal{B}''$ do not need to be efficient. They are only used to compute the winning probability of the (efficient) adversary $\mathcal{B}$.

Denote now by $\mathsf{PVal}_\mathcal{X}$ the probability that $\mathcal{X} \in \{\mathcal{B}, \mathcal{B}', \mathcal{B}''\}$ outputs a non-zero vector. Then, we will show the following three facts.

Fact 1. $\mathsf{PVal}_{\mathcal{B}'} \leqslant \delta_\alpha \mathsf{PVal}_\mathcal{B}^{(\alpha-1)/\alpha}$;

Fact 2. $\mathsf{PVal}_\mathcal{B} = \mathsf{PVal}_{\mathcal{B}''}$;

Fact 3. $\mathsf{PVal}_{\mathcal{B}'} + \mathsf{PVal}_{\mathcal{B}''} \geqslant \mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A}, \kappa)$.

We can conclude the proof from the facts since

$$\begin{aligned}\mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A}, \kappa) &\leqslant \mathsf{PVal}_{\mathcal{B}'} + \mathsf{PVal}_{\mathcal{B}''} \leqslant \delta_\alpha \mathsf{PVal}_\mathcal{B}^{(\alpha-1)/\alpha} + \mathsf{PVal}_\mathcal{B} \\ &\leqslant 2\delta_\alpha \mathsf{PVal}_\mathcal{B}^{(\alpha-1)/\alpha} = 2\delta_\alpha \left(\mathsf{Adv}^{\text{td-msis}}_{q,k,m-1,\beta_{\mathsf{sis}}}(\mathcal{B}, \kappa) + Q \cdot 2^{-2\kappa+2}\right)^{(\alpha-1)/\alpha},\end{aligned}$$

where the last inequality is due to the fact that $\delta_\alpha \geqslant 1$ and $(\alpha-1)/\alpha < 1$ and the last equation is due to Equation (12.4).

We now show the above three facts. For Fact 1, since the only difference between $\mathcal{B}$ and $\mathcal{B}'$ is the distribution of $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$, by Lemma 11.2.5,

$$\mathsf{PVal}_{\mathcal{B}'} \leqslant \left(\mathsf{PVal}_{\mathcal{B}} \cdot R_\alpha\left(P \| \mathscr{D}_\Sigma^{mQ}\right)\right)^{(\alpha-1)/\alpha} .$$

Therefore, Fact 1 follows from the following lemma, which we will prove below.

Lemma 12.3.3. *For the discrete Gaussian distribution $\mathscr{D}_\Sigma^{mQ}$ (defined in Section 11.2.2), where $\Sigma = \mathbb{I}_m \otimes \mathrm{diag}(\sigma_1^2, \ldots, \sigma_Q^2)$, and the distribution $P = \Phi(\mathscr{D}_\Sigma^{mQ})$,*

$$R_\alpha\left(P \| \mathscr{D}_\Sigma^{mQ}\right) \leqslant \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\frac{\alpha}{\alpha-1}} \cdot \exp\left(\alpha\pi^2/\log(2(1+1/\varepsilon))\right) .$$

The arguments for Facts 2 and 3 follow exactly as in Section 11.1.1. We repeat here for completeness. For Fact 2, since $\mathrm{View}_A(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q) = \mathrm{View}_A(\Phi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)) = \mathrm{View}_A(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$, even if $\mathcal{B}''$ sets the secrets to $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$ instead of $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q$, the output of $\mathcal{B}''$ remains the same. However, then, $\mathcal{B}''$ is identical to $\mathcal{B}$, which implies the second fact.

Finally, to prove the third fact, we can interpret the sampling process of $\mathcal{B}'$ as first sampling $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$ from $\mathcal{P}$ and then setting $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \leftarrow \Phi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_Q)$. Then, the only difference between $\mathcal{B}'$ and $\mathcal{B}''$ is that $\mathcal{B}'$ returns $\hat{\boldsymbol{s}} - \sum_{i\in[Q]} \hat{b}_i \boldsymbol{s}_i$, while $\mathcal{B}''$ returns $\hat{\boldsymbol{s}} - \sum_{i\in[Q]} \hat{b}_i \boldsymbol{r}_i$. If $\mathcal{A}$ wins the game $\mathbf{G}_1$ simulated by $\mathcal{B}'$ or $\mathcal{B}''$, we know $\sum_{i\in[Q]} \hat{b}_i \boldsymbol{r}_i = \sum_{i\in[Q]} \hat{b}_i(\boldsymbol{s}_i - u_i\Delta) = \sum_{i\in[Q]} \hat{b}_i \boldsymbol{s}_i - \Delta \cdot \hat{\boldsymbol{b}}^T \boldsymbol{u} \neq \sum_{i\in[Q]} \hat{b}_i \boldsymbol{s}_i$ by the definition of Φ, and thus, at least one of $\mathcal{B}'$ and $\mathcal{B}''$ outputs a non-zero vector, which implies Fact 3. □

Proof of Lemma 12.3.3. We first partition the support R^{mQ} into disjoint sets, then show that the Rényi divergence conditioning on each set is small, and finally use Lemma 11.2.6 to conclude the lemma. Denote $\mathcal{S}_{\mathsf{L}}$ as the set of $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ such that the event AWin does not occur. Since Φ is the identity function over $\mathcal{S}_{\mathsf{L}}$, $\mathscr{D}_\Sigma^{mQ}(\mathcal{S}_{\mathsf{L}}) = P(\mathcal{S}_{\mathsf{L}})$. For each $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q) \notin \mathcal{S}_{\mathsf{L}}$, denote $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q] := \{(\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta)\}_{k\in\mathbb{Z}}$, where $\boldsymbol{u}$ is output by $\mathcal{A}$ given $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. By a similar argument as above, we know any $(\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) \in \mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]$ leads to the same view of $\mathcal{A}$. Therefore, $\Phi(\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta) = (\boldsymbol{s}_1 + (k+1)u_1\Delta, \ldots, \boldsymbol{s}_Q + (k+1)u_Q\Delta)$, and Φ is a bijection over $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]$, which implies $\mathscr{D}_\Sigma^{mQ}(\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]) = P(\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q])$. Also, for any $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q), (\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q) \notin \mathcal{S}_{\mathsf{L}}$, $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]$ and $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_Q]$ are either equal

to disjoint. Therefore, R^{mQ} can be partitioned into disjoint sets $\{\mathcal{S}_{\mathsf{L}}\} \cup \{\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]\}_{(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q) \in R^{mQ} \setminus \mathcal{S}_{\mathsf{L}}}$, and for each set, the probability that $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$ falls in the set is equal under both distributions $\mathscr{D}_{\Sigma}^{mQ}$ and P.

Therefore, by Lemma 11.2.6, we just need to show the Rényi divergence conditioning on each set is small. For $\mathcal{S}_{\mathsf{L}}$, since Φ is the identity function over $\mathcal{S}_{\mathsf{L}}$, $R_\alpha\left(P_{|\mathcal{S}_{\mathsf{L}}} \| \mathscr{D}_{\Sigma}^{mQ}{}_{|\mathcal{S}_{\mathsf{L}}}\right) = 1$.

For any $\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q]$, we show in the following that the conditioned distribution is identical to a one-dimensional discrete Gaussian distribution under a linear transformation. Let $\sqrt{\Sigma}^{-1} = \mathbb{I}_m \otimes \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_Q)$. Denote $\boldsymbol{X} := \sqrt{\Sigma}^{-1}(u_1\Delta, \ldots, u_Q\Delta)$ and $\boldsymbol{S} := \sqrt{\Sigma}^{-1}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_Q)$. Denote $\boldsymbol{S}_\perp := \boldsymbol{S} - s_0\boldsymbol{X}$, where $s_0 := \langle \boldsymbol{S}, \boldsymbol{X}\rangle / \langle \boldsymbol{X}, \boldsymbol{X}\rangle \in \mathbb{R}$, and we have $\langle \boldsymbol{S}_\perp, \boldsymbol{X}\rangle = \langle \boldsymbol{S}, \boldsymbol{X}\rangle - s_0 \langle \boldsymbol{X}, \boldsymbol{X}\rangle = 0$. Then, for any $k \in \mathbb{Z}$,

$$\begin{aligned}
\rho_\sigma(\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta) &= \exp\left(-\pi \left\|\sqrt{\Sigma}^{-1}(\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta)\right\|^2\right) \\
&= \exp\left(-\pi \left\|\boldsymbol{S} + k\boldsymbol{X}\right\|^2\right) \\
&= \exp\left(-\pi \left(\left\|\boldsymbol{S}_\perp\right\|^2 + (s_0 + k)^2 \left\|\boldsymbol{X}\right\|^2\right)\right) \\
&\propto \exp\left(-\pi(-s_0 - k)^2 \left\|\boldsymbol{X}\right\|^2\right) \ .
\end{aligned}$$

Therefore, $\mathscr{D}_{\Sigma}^{mQ}{}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]} = T\left(\mathscr{D}_{\|\boldsymbol{X}\|^{-1}, -s_0}\right)$, where $T : \mathbb{Z} \to R^{mQ}$ maps k to $(\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta)$. Since $\Phi(\boldsymbol{s}_1 + (k-1)u_1\Delta, \ldots, \boldsymbol{s}_Q + (k-1)u_Q\Delta) = (\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta)$, we know $P(\boldsymbol{s}_1 + ku_1\Delta, \ldots, \boldsymbol{s}_Q + ku_Q\Delta) = \mathscr{D}_{\Sigma}^{mQ}(\boldsymbol{s}_1 + (k-1)u_1\Delta, \ldots, \boldsymbol{s}_Q + (k-1)u_Q\Delta)$. By a similar argument as above, $P_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]} = T\left(\mathscr{D}_{\|\boldsymbol{X}\|^{-1}, -s_0+1}\right)$. Since $\|\boldsymbol{X}\|^2 = \sum_{i\in[Q]}(u_i/\sigma_i)^2 \|\Delta\|^2 = \|(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)\|^2 \|\Delta\|^2 \leqslant \sqrt{mN}\beta_{\mathsf{u}}\beta_{\mathsf{lwe}}$, we have

$$\|\boldsymbol{X}\|^{-1} \geqslant 1/(\sqrt{mN}\beta_{\mathsf{u}}\beta_{\mathsf{lwe}}) = \sqrt{\log(2(1 + 1/\varepsilon))/\pi} \geqslant \eta_\varepsilon(\mathbb{Z}) \ .$$

Therefore, by Lemma 11.2.7,

$$\begin{aligned}
R\left(\mathscr{D}_{\Sigma}^{mQ}{}_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]} \| P_{|\mathcal{S}_{\mathsf{W}}[\boldsymbol{s}_1,\ldots,\boldsymbol{s}_Q]}\right) &\leqslant \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\frac{\alpha}{\alpha-1}} \cdot \exp\left(\alpha\pi \left\|\boldsymbol{X}\right\|^2\right) \\
&\leqslant \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\frac{\alpha}{\alpha-1}} \cdot \exp\left(\alpha\pi mN\beta_{\mathsf{u}}^2\beta_{\mathsf{lwe}}^2\right) \\
&= \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^{\frac{\alpha}{\alpha-1}} \cdot \exp\left(\alpha\pi^2/\log(2(1+1/\varepsilon))\right) \ .
\end{aligned}$$

Therefore, we can conclude the lemma by Lemma 11.2.6. □

Chapter 13

TWO-ROUND LATTICE-BASED THRESHOLD SIGNATURES

13.1 Analysis of the CATZ Construction

As our first application to threshold signatures, this section applies AOM-MISIS to the analysis of the CATZ construction [41].

We adopt the following definition from [41], simplifying the small coefficient property by bounding only the term required in the security proof (Section 13.1.3).

Definition 13.1.1 (Linear Threshold Secret Sharing with Small Coefficients)**.** *Let $1 < t \leqslant n$ and B_{ss} be positive integers and $\mathbb{G}$ be an abelian group. A t-out-of-n linear threshold secret sharing scheme $\mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$ for $\mathbb{G}$ consists of two algorithms $(\mathsf{Share}, \mathsf{Recon})$ with the following syntax:*

- $\mathsf{Share}(s \in \mathbb{G}; \boldsymbol{\rho} \in \mathbb{G}^K) \Rightarrow (\mathsf{ss}_j)_{j\in[L]} \in \mathbb{G}^L$*: takes as input a secret $s \in \mathbb{G}$ and a randomness vector $\boldsymbol{\rho} \in \mathbb{G}^K$ (sampled uniformly from $\mathbb{G}^K$), and returns the secret shares $(\mathsf{ss}_j)_{j\in[L]}$. We note that each party $i \in [n]$ has a subset of indices $T_i \subseteq [L]$ such that the share of party i is $(\mathsf{ss}_j)_{j\in T_i}$. We say that the individual share size of party i is $|T_i|$, the **total share size** is L, and the **randomness size** is K.*

- $\mathsf{Recon}(U, (\mathsf{ss}_j)_{j\in\bigcup_{i\in U} T_i}) \Rightarrow s \in \mathbb{G}$*: takes as input a set $U \subseteq [n]$ with $|U| \geqslant t$ and the secret shares corresponding to each party in U, and returns the reconstructed secret s.*

We require that $\mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$ satisfies the following properties:

- ***Linearity**: The sharing algorithm Share can be written as an integer matrix $M \in \mathbb{Z}^{L\times(K+1)}$ mapping a vector $\boldsymbol{v} = (s, \rho_1, \ldots, \rho_K)^T \in \mathbb{G}^{K+1}$ to $M\boldsymbol{v} \in \mathbb{G}^L$. We refer to M as the **sharing matrix** of $\mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$. Moreover, for any $U \subseteq [n]$ denote M_U as the matrix M restricted to the rows indexed with $\bigcup_{i\in U} T_i$, the following is also true:*

  - *For any $U \subseteq [n], |U| \geqslant t$, there exists a **reconstruction coefficient** vector $\boldsymbol{\lambda}^U \in \mathbb{Z}^L$ such that $\lambda^U_j = 0$ for $j \notin \bigcup_{i\in U} T_i$ and $(\boldsymbol{\lambda}^U)^T M = (1, 0, \ldots, 0)$. Then, the output of $\mathsf{Recon}(U, \cdot)$ on input $(\mathsf{ss}_j)_{j\in\bigcup_{i\in U} T_i}$ can be written as $\sum_{i\in U}\sum_{j\in T_i} \lambda^U_j \mathsf{ss}_j$. Hence, for*

$(\mathsf{ss}_j)_{j\in[L]} \leftarrow \mathsf{Share}(s;\boldsymbol{\rho})$ *for any* $s \in \mathbb{G}$ *and* $\boldsymbol{\rho} \in \mathbb{G}^K$*, we have that* $\sum_{i\in U}\sum_{j\in T_i}\lambda_j^U \mathsf{ss}_j = s$.

– *For any* $CS \subseteq [n]$ *with* $|CS| < t$*, there exists a vector* $\boldsymbol{w}^{CS} \in \mathbb{Z}^{K+1}$ *such that* $w_1 = 1$ *and* $M_{CS}\boldsymbol{w}^{CS} = \mathbf{0}$*. We call such* $\boldsymbol{w}^{CS}$ *the* ***sweeping vector*** *of* M_{CS}.

- ***Small Coefficients****: For any* $U \subseteq [n]$ *with* $|U| \geqslant t$*, any* $i \in U$ *and, any* $CS \subset [n]$ *with* $|CS| < t$*, it holds that* $\sum_{(\hat{i},\hat{j})\in T_i\times[K+1]} \lambda_{\hat{i}}^U M_{\hat{i},\hat{j}} w_{\hat{j}}^{CS} \leqslant B_{\mathsf{ss}}$.

Also, [41] shows the existence of such secret sharing scheme from the generic construction by Benaloh and Leichter [21], which can be stated as the following lemma.

Lemma 13.1.2 ([41])**.** *For any* $1 < t \leqslant n$*, there exists a* t*-out-of-*n *linear threshold secret sharing with small coefficients with total share size* $L = O(t'^{4.3} n \log n)$ *making the individual share size* $|T_i| \leqslant O(t'^{4.3} n \log n)$ *for* $t' = \min(t, n-t)$ *and the small coefficient bound* $B_{\mathsf{ss}} = O(t'^{4.3} n (\log n)^2)$.

13.1.1 Construction and main security theorem

We present the scheme $\mathsf{CATZ}[\mathsf{SecSha}]$ in Figure 13.1, where $\mathsf{SecSha} = \mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$ is a linear threshold secret sharing scheme with small coefficients, which is defined in Definition 13.1.1. In Figure 13.2, we give the description of the parameters used in the protocol. One small change here is that the signing key is sampled from a discrete Gaussian distribution with standard deviation σ_{sk} instead of a uniform distribution over the set of vectors with ℓ_∞-norm bounded by σ_{sk}. We note that this does not affect the correctness of the scheme, as the ℓ_2-norm of $\widehat{\mathsf{sk}}$ remains bounded by $\sqrt{mN}\sigma_{\mathsf{sk}}$ except for a negligible probability, which is the exact property needed in the correctness proof.

For unforgeability, we show that TS-UF-0 security of CATZ is implied by the hardness of the AOM-MISIS problem in the random oracle model, which is formally stated in the following theorem. In particular, we consider an extension of AOM-MISIS (See Remark 12.1 for details) in which some σ_i can be ∞. The full proof of Theorem 13.1.3 is given in Section 13.1.3.

Theorem 13.1.3 (TS-UF-0 of CATZ)**.** *For any integers* $q = q(\kappa), k = k(\kappa), m = m(\kappa)$*, any linear threshold secret sharing scheme* $\mathsf{SecSha} = \mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$ *with small coefficients, (see Definition 13.1.1) and any* TS-UF-0 *adversary* $\mathcal{A}$ *making at most* $\mathsf{q}_s = \mathsf{q}_s(\kappa)$ *queries to* PPO *and*

Algorithm $\mathsf{Setup}(1^\kappa)$:

$\overline{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$; $A \leftarrow [\overline{A}|\mathbb{I}_k]$

Return A

Algorithm $\mathsf{KeyGen}()$:

$\widehat{\mathsf{sk}} \leftarrow\!\!\$\ \mathscr{D}^m_{\sigma_{\mathsf{sk}}}$; $\mathsf{pk} \leftarrow A\widehat{\mathsf{sk}} \bmod q$

$\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K \leftarrow\!\!\$\ R_q^m$

$(\mathsf{ss}_1, ..., \mathsf{ss}_L)^T \leftarrow M \cdot (\widehat{\mathsf{sk}}, \boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K)^T$

For $i \in [n]$ do $\mathsf{sk}_i \leftarrow (\mathsf{ss}_j)_{j\in T_i}$

Return $(\mathsf{pk}, (\mathsf{sk}_i)_{i\in[n]})$

Algorithm $\mathsf{SPP}(\mathsf{st}_i)$:

For $j \in [0..\ell]$ do $\boldsymbol{r}_j \leftarrow\!\!\$\ \mathscr{D}^m_{\sigma_{\mathsf{r}}}$

For $j \in [0..\ell]$ do $\boldsymbol{R}_j \leftarrow A\boldsymbol{r}_j \bmod q$

$pp \leftarrow (\boldsymbol{R}_j)_{j\in[0..\ell]}$

$\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (\boldsymbol{r}_j)_{j\in[0..\ell]}$

Return (pp, st_i)

Algorithm $\mathsf{CompPar}(\mathsf{pk}, lr)$:

$\mu \leftarrow lr.\mathsf{msg}$

For $i \in lr.\mathsf{SS}$ do

  $(b_j)_{j\in[\ell]} \leftarrow \mathrm{H}_1(\mathsf{pk}, lr)$

  $(\boldsymbol{R}_{i,j})_{j\in[0..\ell]} \leftarrow lr.\mathsf{PP}(i)$

$\boldsymbol{R} \leftarrow \sum_{i\in lr.\mathsf{SS}} \left(\boldsymbol{R}_{i,0} + \sum_{j\in[\ell]} b_j \boldsymbol{R}_{i,j}\right)$

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, \mu, \boldsymbol{R})$

Return $(\boldsymbol{R}, c, (b_j)_{j\in[\ell]})$

Algorithm $\mathsf{PS}(lr, i, \mathsf{st}_i)$:

$pp_i \leftarrow lr.\mathsf{PP}(i)$

If $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \perp$ then return $(\perp, \mathsf{st}_i)$

$(\boldsymbol{r}_j)_{j\in[0..\ell]} \leftarrow \mathsf{st}_i.\mathrm{mapPP}(pp_i)$

$\mathsf{st}_i.\mathrm{mapPP}(pp_i) \leftarrow \perp$

$(\boldsymbol{R}, c, (b_j)_{j\in[\ell]}) \leftarrow \mathsf{CompPar}(\mathsf{st}_i.\mathsf{pk}, lr)$

$(\mathsf{ss}_j)_{j\in T_i} \leftarrow \mathsf{st}_i.\mathsf{sk}$

$\boldsymbol{z} \leftarrow \boldsymbol{r}_0 + \sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_j$
$\quad +2c \cdot \sum_{j\in T_i} \lambda_j^{lr.\mathsf{SS}} \mathsf{ss}_j \bmod q$

Return $((\boldsymbol{R}, \boldsymbol{z}), \mathsf{st}_i)$

Algorithm $\mathsf{Agg}(\mathrm{PS}, \mathsf{st}_0)$:

$\boldsymbol{R} \leftarrow \perp$; $\boldsymbol{z} \leftarrow 0$

For $(\boldsymbol{R}', \boldsymbol{z}') \in \mathrm{PS}$ do

  If $\boldsymbol{R} = \perp$ then $\boldsymbol{R} \leftarrow \boldsymbol{R}'$

  If $\boldsymbol{R} \neq \boldsymbol{R}'$ then return $(\perp, \mathsf{st}_0)$

  $\boldsymbol{z} \leftarrow \boldsymbol{z} + \boldsymbol{z}'$

Return $((\boldsymbol{R}, \boldsymbol{z}), \mathsf{st}_0)$

Algorithm $\mathsf{Vf}(\mathsf{pk}, \mu, sig)$:

$(\boldsymbol{R}, \boldsymbol{z}) \leftarrow sig$

If $\|\boldsymbol{z}\| > \beta_{\mathsf{z}}$ then return 0

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, \mu, \boldsymbol{R})$

Return $(A\boldsymbol{z} = \boldsymbol{R} + 2c \cdot \mathsf{pk} \bmod q)$

Figure 13.1: Lattice-based t-out-of-n threshold signatures CATZ[SecSha], where SecSha $=$ $\mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$ is t-out-of-n a linear secret sharing scheme with small coefficients (see Definition 13.1.1). In particular, $\lambda_j^{lr.\mathsf{SS}}$ denotes the reconstruction coefficient. Also, $\mathrm{H}_1 : \{0,1\}^* \to \mathcal{S}_{\mathsf{b}}^\ell$ and $\mathrm{H}_2 : \{0,1\}^* \to \mathcal{S}_{\mathsf{c}}$. The algorithms LPP and LR are defined the same as in Figure 10.3.

Parameters	Description
κ	Security parameter
n	Number of signers
t	Threshold for signing
N	$N \geqslant 2\kappa$, power of two defining the ring R
q	Prime modulus
k	Number of rows of A
m	$m > k$, number of columns of A,
$\ell + 1$	$\ell = 2\kappa/\log(2N)$, number of nonces for each signer
$\mathcal{S}_{\mathsf{b}}$	$\mathcal{S}_{\mathsf{b}} = \{\pm 1, \pm X, \ldots, \pm X^{N-1}\}$, set for the aggregating coefficients b_j
β_{c}	Satisfying $2^{\beta_{\mathsf{c}}}\binom{N}{\beta_{\mathsf{c}}} \geqslant 2^{2\kappa}$, the ℓ_1-norm of the challenge c,
$\mathcal{S}_{\mathsf{c}}$	$\mathcal{S}_{\mathsf{c}} = \{c \in R : \|c\|_\infty = 1, \|c\|_1 = \beta_{\mathsf{c}}\}$, set of the challenges c
σ_{sk}	Standard deviation of the signing key sk
σ_{r}	Standard deviation of the nonces $\boldsymbol{r}_i$
ν_{pk}	Only for EKT: satisfying $\lfloor q/2^{\nu_{\mathsf{pk}}} \rceil = \lfloor q/2^{\nu_{\mathsf{pk}}} \rceil$, number of bits saved on pk
ν_{r}	Only for EKT: satisfying $\lfloor q/2^{\nu_{\mathsf{r}}} \rceil = \lfloor q/2^{\nu_{\mathsf{r}}} \rceil$, number of bits saved on $\boldsymbol{h}$
$q_{\nu_{\mathsf{pk}}}, q_{\nu_{\mathsf{r}}}$	Only for EKT: $(q_{\nu_{\mathsf{pk}}}, q_{\nu_{\mathsf{r}}}) = (\lfloor q/2^{\nu_{\mathsf{pk}}} \rceil, \lfloor q/2^{\nu_{\mathsf{r}}} \rceil)$ the rounded moduli
β_{z}	ℓ_2-norm bound of a valid signature vector $\boldsymbol{z}$
	(or $(\boldsymbol{z}, 2^{\nu_{\mathsf{pk}}}\boldsymbol{h})$ for EKT)

Figure 13.2: Parameters for CATZ and EKT. Some parameters only apply to EKT.

$\mathsf{q}_h = \mathsf{q}_h(\kappa)$ *queries to* RO*, there exists an* AOM-MISIS *adversary* $\mathcal{B}$ *running in time roughly two times that of* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\text{ts-uf-0}}_{\mathsf{CATZ}}(\mathcal{A}, \kappa) \leqslant \sqrt{\mathsf{qAdv}^{\text{aom-misis}}_{\mathsf{par}}(\mathcal{B}, \kappa) + 8\mathsf{q}^3 2^{-2\kappa}}\,.$$

where $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$ *and* $\mathsf{par} = (q, k, m, Q = 1 + K + \mathsf{q}_s(1+\ell), (\sigma_i)_{i\in[Q]}, \beta_{\mathsf{s}} = 2\beta_{\mathsf{z}}, \beta_{\mathsf{b}} = 4\sigma_{\mathsf{sk}}\beta_{\mathsf{c}}, \beta_{\mathsf{u}} = 1/\sigma_{\mathsf{sk}} + \beta_{\mathsf{c}} B_{\mathsf{ss}}\sqrt{8N\mathsf{q}_s}/\sigma_{\mathsf{r}})$ *with* $\sigma_1 = \sigma_{\mathsf{sk}}$*,* $\sigma_{1+i} = \infty$ *for* $i \in [K]$*,* $\sigma_{1+K+i} = \sigma_{\mathsf{r}}$ *for* $i \in [\mathsf{q}_s(\ell+1)]$.

For completeness, we recall the correctness theorem from [41], which is needed for parameter selections later.

Theorem 13.1.4 (Correctness of CATZ [41]). *For any integers $1 < t \leqslant n$, any linear threshold secret sharing scheme $\mathsf{SecSha} = \mathsf{SecSha}_{t,n,B_{\mathsf{ss}}}$ with small coefficients, (see Definition 13.1.1) given $\sigma_{\mathsf{r}} \geqslant 2\sqrt{6mN\log(2mN)\kappa/\pi}$ and $\beta_{\mathsf{z}} \geqslant \sqrt{mN}(2\beta_{\mathsf{c}}\sigma_{\mathsf{sk}} + \sigma_{\mathsf{r}}\sqrt{n(1+\ell)})$, the threshold signature scheme $\mathsf{CATZ}[\mathsf{SecSha}]$ is correct with correctness error $\varepsilon_{\mathsf{cor}} \leqslant (2 + 4n(\ell+1)) \cdot 2^{-2\kappa}$.*

13.1.2 Parameter selection

In this section, we first discuss the asymptotic parameters selection derived from the security theorems and the hardness of AOM-MISIS, then compare these parameters with those proposed in [41], and finally estimate the concrete efficiency based on the parameter selections.

ASYMPTOTIC PARAMETER SELECTIONS. Denote β_{lwe} as the norm of the underlying MLWE assumption. Initially, we select $N, m, k, \beta_{\mathsf{lwe}}$ such that N is a power of $N \geqslant 2\kappa$, $m, k = \mathsf{poly}(\kappa)$, and $\beta_{\mathsf{lwe}} \geqslant m\log(N)$.[1] (We note that when estimating the concrete efficiency, we will enumerate through plausible $(N, m, k, \beta_{\mathsf{lwe}})$ tuples and pick the one that yields the best efficiency.) Then, we set other parameters as follows.

- Set β_{c} as the smallest integer such that $2^{\beta_{\mathsf{c}}}\binom{N}{\beta_{\mathsf{c}}} \geqslant 2^{2\kappa}$.
- $\sigma_{\mathsf{sk}} = \max\{2\beta_{\mathsf{lwe}}\sqrt{mN}, \sqrt{\log(6mN)/\pi}\}$. The first term is usually the leading term.
- $\sigma_{\mathsf{r}} = \max\{\sigma_{\mathsf{sk}}\beta_{\mathsf{c}}B_{\mathsf{ss}}\sqrt{8N\mathsf{q}_s}, 2\sqrt{6mN\log(2mN)\kappa/\pi}\}$. The first term is usually the leading term.
- $\beta_{\mathsf{z}} = \sqrt{mN}(2\beta_{\mathsf{c}}\sigma_{\mathsf{sk}} + \sigma_{\mathsf{r}}\sqrt{n(1+\ell)})$
- Denote $\beta_{\mathsf{sis}} = 2\beta_{\mathsf{z}} + 4\sigma_{\mathsf{sk}}\beta_{\mathsf{c}}\sqrt{mN}$.
- Select q such that the problem $\mathrm{MSIS}_{q,k,m,\beta_{\mathsf{sis}}}$ and the problem $\mathrm{MLWE}_{q,N,k,m,\beta_{\mathsf{lwe}}}$ are assumed to be exponentially hard in κ.

By Theorem 13.1.3 and Theorem 12.3.1 with $\varepsilon = 1/2$ and $\alpha = 2$ (we can further optimize the concrete bound by adjusting α), TS-UF-0 of CATZ is implied by the hardness of $\mathrm{MSIS}_{q,k,m,\beta_{\mathsf{sis}}}$ and $\mathrm{MLWE}_{q,N,k,m,\beta_{\mathsf{lwe}}}$.

[1] This is for guaranteeing the underlying MLWE is hard.

	$\log_2(q)$	k	m	σ_{sk}	σ_{r}	β_{z}	$\lvert\mathsf{pk}\rvert$	$\lvert sig\rvert$	Comm.
[41]	119	8	50	2^{18}	$2^{105.06}$	$2^{117.25}$	60.73KB	440.32KB	2.02MB
This thesis	94.1	7	17	$2^{9.1}$	$2^{80.5}$	$2^{91.96}$	42.14KB	144.47KB	1.24MB

Figure 13.3: The concrete parameters and estimated efficiency for $\kappa = 128$ and $n = 32$ in [41] and this work. We set $(N, \ell, \beta_{\mathsf{c}}) = (512, 26, 64)$. The last second column denotes the communication complexity per signer.

COMPARED WITH [41] The differences of the parameter selections between [41] and ours shows in the selections of σ_{sk} and σ_{r}. In particular, the prior work requires

- $\sigma_{\mathsf{sk}} = q^{k/m} 2^{2\kappa/(Nm)}$.
- $\sigma_{\mathsf{r}} = \max\{N\beta_{\mathsf{c}} B_{\mathsf{ss}} \sigma_{\mathsf{sk}} \sqrt{32\pi \mathsf{q}_s mN}, \frac{16N\sqrt{3m}}{\sqrt{\pi}} q^{\frac{k}{m}} \sqrt{N(\log(2mN) + 2\kappa)}\}$. The first term is usually the leading term.

For σ_{sk}, $q^{k/m}$ is significantly larger than $2\beta_{\mathsf{lwe}}\sqrt{mN}$, which influences the choice of k and m. In particular, one needs to set m to be several times larger than k to ensure σ_{sk} does not cause the parameters to grow excessively. For σ_{r}, we can see a factor of $N\sqrt{m}$ improvement by comparing the first term inside the maximization.

CONCRETE EFFICIENCY. We show a set of concrete parameters and estimated efficiency for $\kappa = 128$ and $n = 32$, and compare them with those from [41] in Figure 13.3, where we can see improvements in both signature sizes and communication complexity. We derive the parameters following our parameter selections mentioned above, where SecSha are instantiated using Lemma 13.1.2. To estimate the concrete hardness of MSIS and MLWE, we use the state-of-art tool, lattice estimator,[2] which is also used in other prior works [54, 60, 33].

13.1.3 Security reduction of CATZ

The proof structure generally follows from the security proof of FROST [20], which reduces the security of FROST to the algebraic one-more discrete logarithm assumption, as the CATZ con-

[2] https://github.com/malb/lattice-estimator

struction can be seen as a lattice analog of FROST. The key differences lie in how the reduction computes $\boldsymbol{u}$ and how we bound the norm of the output solution $(\hat{\boldsymbol{s}}, \hat{\boldsymbol{b}})$ and $\boldsymbol{u}$.

To prove Theorem 13.1.3, we slightly generalize the forking lemma (Lemma 9.2.3) such that the hash values $(h_1, \ldots, h_1)$ are independently sampled by an algorithm HG instead of uniformly sampled from a set H.

Proof of Theorem 13.1.3. Let $\mathcal{A}$ be a TS-UF-0 adversary as described in the theorem. W.l.o.g. we assume that $\mathcal{A}$ is deterministic and corrupts exactly $t-1$ signers. Also, we assume if $\mathcal{A}$ returns $(\mu^*, (\boldsymbol{R}^*, \boldsymbol{z}^*))$, the RO query $\mathrm{H}_2(\mathsf{pk}, \mu^*, \boldsymbol{R}^*)$ was made by $\mathcal{A}$, which adds at most one RO query. Also, since the game makes at most one RO query to H_1 and H_2 respectively for each signing query, the total number of RO queries to each of H_1 and H_2 is bounded $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$. We first construct an algorithm $\mathcal{C}$ compatible with the syntax in Lemma 9.2.3 and construct $\mathcal{B}$ from $\mathsf{Fork}^{\mathcal{C}}$.

CONSTRUCTION OF $\mathcal{C}$. The input of $\mathcal{C}$ consists of $A, (\boldsymbol{t}_i)_{i\in[K+1+\mathsf{q}_s(\ell+1)]}$, and a list of hash values $h_1, \ldots, h_{2\mathsf{q}}$, where $(A, (\boldsymbol{t}_i)_{i\in[K+1+\mathsf{q}_s(\ell+1)]})$ are sampled following the AOM-MISIS game, and for each $i \in [\mathsf{q}]$, h_{2i-1} is sampled uniformly from $\mathcal{S}_{\mathsf{b}}$ and h_{2i} is sampled uniformly from $\mathcal{S}_{\mathsf{c}}$. To start with, $\mathcal{C}$ sets $par \leftarrow A$, initializes $\mathsf{st}_i.\mathrm{mapPP} \leftarrow ()$ for $i \in [n]$, and in addition, initializes a counter $\mathrm{ctr}_h \leftarrow 0$ for counting the number of random oracle queries. Then, $\mathcal{C}$ runs $\mathcal{A}$ on input par with access to oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$ and $\widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: $\mathcal{C}$ sets $\mathsf{pk} \leftarrow \boldsymbol{t}_1$ and views $\boldsymbol{t}_{1+\hat{j}}$ as $A\boldsymbol{\rho}_{\hat{j}}$ for $\hat{j} \in [K]$. Then, for each $i \in CS$ and $j \in T_i$, $\mathcal{C}$ computes $\mathsf{ss}_j \leftarrow \mathrm{PI}(\boldsymbol{d})$ with

$$d_{\hat{j}} = \begin{cases} M_{j,\hat{j}}\ , & \hat{j} \in [K+1]\ , \\ 0\ , & o.w. \end{cases} \tag{13.1}$$

Finally, $\mathcal{C}$ returns $(\mathsf{pk}, (\mathsf{sk}_i = (\mathsf{ss}_j)_{j\in T_i})_{i\in CS})$.

$\widetilde{\text{PPO}}(i)$: For the j-th query, $\mathcal{C}$ sets $\boldsymbol{R}_{\hat{j}} \leftarrow \boldsymbol{t}_{K+1+(j-1)(\ell+1)+\hat{j}+1}$ for $\hat{j} \in [0..\ell]$. Since $\mathcal{C}$ does not sample $\{\boldsymbol{r}_{\hat{j}}\}_{\hat{j}\in[0..\ell]}$, $\mathcal{C}$ uses $\mathsf{st}_i.\mathrm{mapPP}$ to store the index j instead, i.e., $\mathcal{C}$ sets $\mathsf{st}_i.\mathrm{mapPP}(\{\boldsymbol{R}_{\hat{j}}\}_{\hat{j}\in[0..\ell]}) \leftarrow j$.

$\widetilde{\text{PSIGNO}}(i, lr)$: The same as $\text{PSIGNO}(i, lr)$ except that $\mathcal{C}$ computes $\boldsymbol{z}$ using the PI oracle as fol-

lows. Let $j \leftarrow \mathsf{st}_i.\mathrm{mapPP}(lr.\mathsf{PP}(i))$ and $\boldsymbol{d}$ be a vector in R^Q such that

$$d_{\hat{j}} = \begin{cases} 2c \cdot \sum_{\hat{i} \in T_i} \lambda_{\hat{i}}^{lr.\mathsf{SS}} M_{\hat{i},\hat{j}} \ , & \hat{j} \in [K+1] \ , \\ 1 \ , & \hat{j} = K + 1 + (j-1)(\ell+1) + 1 \ , \\ b_{j'} \ , & \hat{j} = K + 1 + (j-1)(\ell+1) + 1 + j', j' \in [\ell] \ , \\ 0 \ , & o.w. \end{cases} \tag{13.2}$$

$\mathcal{C}$ computes $\boldsymbol{z} \leftarrow \mathrm{PI}(\boldsymbol{d})$.

$\widetilde{\mathbf{RO}}$ query $\mathrm{H}_1(x)$**:** If $\mathrm{H}_1(x) \neq \perp$, $\mathcal{C}$ returns $\mathrm{H}_1(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, lr)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_1(x) \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{b}}^{\ell}$ and returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1, sets $\mathrm{H}_1(x) \leftarrow h_{2\mathrm{ctr}_h - 1}$. Also, $\mathcal{C}$ computes $\boldsymbol{R} \leftarrow \sum_{i \in lr.\mathsf{SS}} (\boldsymbol{R}_{i,0} + \sum_{j \in [\ell]} b_j \cdot \boldsymbol{R}_{i,j})$, where $(\boldsymbol{R}_{i,j})_{j \in [0..\ell]} \leftarrow lr.\mathsf{PP}(i)$ and $\{b_j\}_{j \in [\ell]} \leftarrow h_{2\mathrm{ctr}_h - 1}$. If $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R}) = \perp$, $\mathcal{C}$ sets $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R}) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(x)$.

$\widetilde{\mathbf{RO}}$ query $\mathrm{H}_2(x)$**:** If $\mathrm{H}_2(x) \neq \perp$, $\mathcal{C}$ returns $\mathrm{H}_2(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, \mu, \boldsymbol{R})$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_2(x) \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{c}}$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathrm{H}_2(x) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_2(x)$.

After receiving the output $(\mu^*, (\boldsymbol{R}^*, \boldsymbol{z}^*))$ from $\mathcal{A}$, $\mathcal{C}$ aborts if $\mathcal{A}$ does not win the TS-UF-0 game. Otherwise, $\mathcal{C}$ finds the index I such that $\mathrm{H}_2(\mathsf{pk}, \mu^*, \boldsymbol{R}^*)$ is set to h_I during the simulation. By our assumption of $\mathcal{A}$, we know such I must exist. Then, $\mathcal{C}$ returns $(I, \mathrm{Out} = (\mu^*, \boldsymbol{R}^*, \boldsymbol{z}^*))$.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.2.3, we define $S := \{2i\}_{i \in [\mathsf{q}]}$, IG as the algorithm that samples $(A, (\boldsymbol{t}_i)_{i \in [K+1+\mathsf{q}_s(\ell+1)]})$ following the AOM-MISIS game, and HG as the algorithm that samples h_{2i-1} uniformly from $\mathcal{S}_{\mathsf{b}}$ and samples h_{2i} uniformly from $\mathcal{S}_{\mathsf{c}}$ for each $i \in [\mathsf{q}]$. From the simulation, we know that the output index I of $\mathcal{C}$ is always in S. Also, it is not hard to check that $\mathcal{C}$ simulates the game TS-SUF-0 perfectly, which implies $\mathrm{acc}(\mathcal{C}) \geqslant \mathsf{Adv}^{\text{ts-uf-0}}_{\mathsf{CATZ}}(\mathcal{A}, \kappa)$. By Lemma 9.2.3, we have that

$$\mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) \geqslant \mathsf{Adv}^{\text{ts-uf-0}}_{\mathsf{CATZ}}(\mathcal{A}, \kappa)^2 / \mathsf{q} \ .$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now construct the AOM-MISIS adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$. To start with, $\mathcal{B}$ receives $(A, \{\boldsymbol{t}_i\}_{i \in [Q]})$ from the AOM-MISIS game with $Q = K + 1 + \mathsf{q}_s(\ell + 1)$ and runs $\mathsf{Fork}^{\mathcal{C}}(A, \{\boldsymbol{t}_i\}_{i \in [Q]})$ with access to the PI oracle from the AOM-MISIS game. If $\mathsf{Fork}^{\mathcal{C}}$

outputs $(I, \mathrm{Out} = (\mu^*, \boldsymbol{R}^*, \boldsymbol{z}^*), \overline{\mathrm{Out}} = (\bar{\mu}^*, \bar{\boldsymbol{R}}^*, \bar{\boldsymbol{z}}^*))$, we know $A\boldsymbol{z}^* = \boldsymbol{R}^* + 2h_I\mathsf{pk} \bmod q$ and $A\bar{\boldsymbol{z}}^* = \bar{\boldsymbol{R}}^* + 2\bar{h}_I\mathsf{pk} \bmod q$, which implies $A(\boldsymbol{z}^* - \bar{\boldsymbol{z}}^*) = 2(h_I - \bar{h}_I)\mathsf{pk} \bmod q$. Therefore, $\mathcal{B}$ sets $\hat{\boldsymbol{s}} \leftarrow \boldsymbol{z}^* - \bar{\boldsymbol{z}}^*$ and $\hat{\boldsymbol{b}} \leftarrow (2(h_I - \bar{h}_I), 0, \ldots, 0)$, and it holds that $\sum_{i\in[Q]} \hat{b}_i \boldsymbol{t}_i = 2(h_I - \bar{h}_I)\boldsymbol{t}_0 = 2(h_I - \bar{h}_I)\mathsf{pk} = A(\boldsymbol{z}^* - \bar{\boldsymbol{z}}^*) = \hat{\boldsymbol{s}} \bmod q$.

We now show how $\mathcal{B}$ sets $\boldsymbol{u}$ such that $\hat{\boldsymbol{b}}^T\boldsymbol{u} \neq 0$ and $\boldsymbol{d}^T\boldsymbol{u} = 0$ for any oracle query $\mathrm{PI}(\boldsymbol{d})$.

By the linearity of SecSha, there exists a sweeping vector $\boldsymbol{w} \in \mathbb{Z}^{K+1}$ such that $M_{CS}\boldsymbol{w} = \boldsymbol{0}$ and $w_1 = 1$, and we set $u_{[K+1]} = \boldsymbol{w}$. Therefore, $\hat{\boldsymbol{b}}^T\boldsymbol{u} = \hat{b}_1 u_1 = \hat{b}_1 \neq 0$. Also, for each PI query made during the execution of $\widetilde{\textsc{Init}}$, the query is of the form $\boldsymbol{d} = (M_{j,1}, \ldots, M_{j,K+1}, 0, \ldots, 0)$, where $j \in \bigcup_{i\in CS} T_i$, and thus $\boldsymbol{d}^T\boldsymbol{u} = (M_{j,1}, \ldots, M_{j,K+1}) \cdot \boldsymbol{w} = 0$.

For $j \in [\mathsf{q}_s]$, $\mathcal{B}$ sets $u_{K+1+(j-1)(\ell+1)+[\ell+1]}$ as follows. To simplify notation in the following analysis, we use $\boldsymbol{v}$ to denote the vector $u_{K+1+(j-1)(\ell+1)+[\ell+1]} \in R^{\ell+1}$. We say a $\widetilde{\textsc{PSign}}\mathrm{O}$ query (i, lr) corresponds to the j-th token if and only if it is the valid query with $\mathsf{st}_i.\mathrm{mapPP}(lr.\mathsf{PP}(i)) = j$, where a valid query means the $\widetilde{\textsc{PSign}}\mathrm{O}$ oracle does not return $\perp$. From the simulation, there is at most one $\widetilde{\textsc{PSign}}\mathrm{O}$ query corresponding to the j-th token during each execution of $\mathcal{A}$. Therefore, there are the following cases:

Case 1: No query corresponds to the j-th token during both executions. In this case, $\mathcal{B}$ set $\boldsymbol{v} \leftarrow \boldsymbol{0}$.

Case 2: Only one query corresponds to the j-th token during the two executions. Denote $\boldsymbol{d}$ as the PI query made during the execution of the $\widetilde{\textsc{PSign}}\mathrm{O}$ query corresponding to the j-th token, where $\boldsymbol{d}$ follows the form given in Equation (13.2). $\mathcal{B}$ sets $\boldsymbol{v} \leftarrow (-\sum_{\hat{j}\in[K+1]} d_{\hat{j}}u_{\hat{j}}, 0, \ldots, 0)$, which implies $\boldsymbol{d}^T\boldsymbol{u} = \sum_{\hat{j}\in[K+1]} d_{\hat{j}}u_{\hat{j}} + v_1 = 0$.

Case 3: There is one query corresponding to the j-th token during each of the two executions. Denote $\boldsymbol{d}$ (resp. $\bar{\boldsymbol{d}}$) as the PI query made during the execution of the $\widetilde{\textsc{PSign}}\mathrm{O}$ query corresponding to the j-th token before (resp. after) rewinding. If $\boldsymbol{d} = \bar{\boldsymbol{d}}$, then $\mathcal{C}$ sets $\boldsymbol{v}$ in the same way as Case 2. Otherwise, let $\hat{k} \in [\ell]$ be the index such that $d_{K+1+(j-1)(\ell+1)+1+\hat{k}} \neq \bar{d}_{K+1+(j-1)(\ell+1)+1+\hat{k}}$. (If such $\hat{k}$ does not exist, $\mathcal{B}$ aborts.) Denote $b := d_{K+1+(j-1)(\ell+1)+1+\hat{k}}$ and $\bar{b} := d_{K+1+(j-1)(\ell+1)+1+\hat{k}}$. By Equation (13.2), we know that 2 divides $d_{\hat{j}}$ and $\bar{d}_{\hat{j}}$ for $\hat{j} \in [K+1]$. Therefore, denote $\Delta := \sum_{\hat{j}\in[K+1]}(d_{\hat{j}}/2)u_{\hat{j}}$ and $\bar{\Delta} := \sum_{\hat{j}\in[K+1]}(\bar{d}_{\hat{j}}/2)u_{\hat{j}}$. Since

$b, \bar{b} \in \mathcal{S}_{\mathsf{b}}$, by Lemma 11.2.1, there exists $\gamma \in R$ such that $\gamma(b - \bar{b}) = 2 \mod q$. $\mathcal{C}$ sets

$$v_{\hat{j}} \leftarrow \begin{cases} -2\Delta + b\gamma(\Delta - \bar{\Delta}) \ , & \hat{j} = 1 \ , \\ -\gamma(\Delta - \bar{\Delta}) \ , & \hat{j} = \hat{k} \ , \\ 0 \ , & o.w. \end{cases}$$

Then, it holds that $\boldsymbol{d}^T \boldsymbol{u} = \sum_{\hat{j} \in [K+1]} d_{\hat{j}} u_{\hat{j}} + v_1 + b v_{\hat{k}} = 2\Delta - 2\Delta + b\gamma(\Delta - \bar{\Delta}) - b\gamma(\Delta - \bar{\Delta}) = 0$ and $\bar{\boldsymbol{d}}^T \boldsymbol{u} = \sum_{\hat{j} \in [K+1]} \bar{d}_{\hat{j}} u_{\hat{j}} + v_1 + \bar{b} v_{\hat{k}} = 2\bar{\Delta} - 2\Delta + b\gamma(\Delta - \bar{\Delta}) - \bar{b}\gamma(\Delta - \bar{\Delta}) = 2\bar{\Delta} - 2\Delta + (b - \bar{b})\gamma(\Delta - \bar{\Delta}) = 2\bar{\Delta} - 2\Delta + 2(\Delta - \bar{\Delta}) = 0$.

ANALYSIS OF $\mathcal{B}$. Denote $\mathsf{BadHash}$ as the event that $h_1, \bar{h}_1, \ldots, h_{2\mathsf{q}}, \bar{h}_{2\mathsf{q}}$ are *not* all distinct. We now show that $\mathcal{B}$ wins the AOM-MISIS game if $\mathsf{Fork}^{\mathcal{C}}$ returns and $\mathsf{BadHash}$ does not occur.

We first show that if $\mathsf{Fork}^{\mathcal{C}}$ returns and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ does not abort. Suppose $\mathsf{Fork}^{\mathcal{C}}$ returns and $\mathsf{BadHash}$ does not occur. Then, the only step where $\mathcal{B}$ might abort is in Case 3 above. In Case 3, suppose $\boldsymbol{d} \neq \bar{\boldsymbol{d}}$ and $d_{K+1+(j-1)(\ell+1)+1+[\ell]} = \bar{d}_{K+1+(j-1)(\ell+1)+1+[\ell]}$ (in which case $\mathcal{B}$ aborts). Let (i, lr) be the $\widetilde{\text{PSIGNO}}$ query that corresponds to the j-th token *before* rewinding. Then, there exists $J \in [\mathsf{q}]$ such that $h_{2J-1} = \mathrm{H}_1(\mathsf{pk}, lr) = d_{K+1+(j-1)(\ell+1)+1+[\ell]}$. Since $\mathsf{BadHash}$ does not occur, the only possibility is that the $2J - 1 < I$ and (i, lr) is also the $\widetilde{\text{PSIGNO}}$ query that corresponds to the j-th token *after* rewinding. Let $\boldsymbol{R}$ be the aggregated nonce computed from $\mathsf{CompPar}(\mathsf{pk}, lr)$. Denote $\hat{J} \in [\mathsf{q}]$ be the index such that $h_{2\hat{J}} = \mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R})$ before rewinding. From the simulation of the random oracles, it holds that $\hat{J} \leqslant J$ and thus $2\hat{J} \leqslant 2J \leqslant I$. Also, since $lr.\mathsf{msg} \neq \mu^*$ (o.w., $\mathcal{C}$ would not win the game), we have $2\hat{J} \neq I$ and thus $2\hat{J} < I$. Therefore, $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R})$ in the second execution of $\mathcal{C}$ is also $h_{2\hat{J}}$. This implies $\boldsymbol{d} = \bar{\boldsymbol{d}}$, which contradicts our assumption.

If $\mathcal{B}$ does not abort, we have $\|\hat{\boldsymbol{s}}\| \leqslant \|\boldsymbol{z}^*\| + \|\bar{\boldsymbol{z}}^*\| \leqslant 2\beta_{\mathsf{z}} = \beta_{\mathsf{s}}$ and

$$\left\| (\hat{b}_1 \cdot \sigma_1, \ldots, \hat{b}_Q \cdot \sigma_Q) \right\|_1 = \left\| 2(h_I - \bar{h}_I)\sigma_{\mathsf{sk}} \right\|_1 \leqslant 4\sigma_{\mathsf{sk}}\beta_{\mathsf{c}} = \beta_{\mathsf{b}} \ .$$

It is left to bound $\|(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)\|$. For $j \in [\mathsf{q}_{\mathsf{s}}]$, denote $\boldsymbol{v} := u_{K+1+(j-1)(\ell+1)+[\ell+1]}$. There are three cases as mentioned in the construction of $\mathcal{B}$. For the first case, $\|\boldsymbol{v}\| = 0$. For the second case, $\|\boldsymbol{v}\| = \left\| 2c \cdot \sum_{(\hat{i},\hat{j}) \in T_i \times [K+1]} \lambda_{\hat{i}}^{lr.\mathsf{SS}} M_{\hat{i},\hat{j}} w_{\hat{j}} \right\| \leqslant 2B_{\mathsf{ss}} \|c\| \leqslant 2B_{\mathsf{ss}}\sqrt{\beta_{\mathsf{c}}}$. For the third case, $\|\Delta\|_1 = \left\| c \cdot \sum_{(\hat{i},\hat{j}) \in T_i \times [K+1]} \lambda_{\hat{i}}^{lr.\mathsf{SS}} M_{\hat{i},\hat{j}} w_{\hat{j}} \right\|_1 \leqslant B_{\mathsf{ss}}\beta_{\mathsf{c}}$ and similarly $\left\|\bar{\Delta}\right\|_1 \leqslant B_{\mathsf{ss}}\beta_{\mathsf{c}}$. Since $\|\gamma\| \leqslant \sqrt{N}$

(by Lemma 11.2.1),

$$\begin{aligned}\|\boldsymbol{v}\|^2 &= \left\|\gamma(\bar{b}\Delta - b\bar{\Delta}), 0, \ldots, 0, -\gamma(\Delta - \bar{\Delta}), 0, \ldots, 0)\right\|^2 \\ &\leqslant 2(\|\Delta\|_1 + \left\|\bar{\Delta}\right\|_1)^2 \|\gamma\|^2 \leqslant 8NB_{\mathsf{ss}}^2\beta_{\mathsf{c}}^2 \ .\end{aligned}$$

Therefore, $\|(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)\| \leqslant 1/\sigma_{\mathsf{sk}} + \sqrt{\sum_{j\in[\mathsf{q}_s]} \|\boldsymbol{v}/\sigma_{\mathsf{r}}\|^2} \leqslant 1/\beta_{\mathsf{s}} + \beta_{\mathsf{c}}B_{\mathsf{ss}}\sqrt{8N\mathsf{q}_s}/\sigma_{\mathsf{r}} \leqslant \beta_{\mathsf{u}}$.

Since each of $h_1, h_3, \ldots, h_{2\mathsf{q}-1}$ and $h_2, h_4, \ldots, h_{2\mathsf{q}}$ are sampled uniformly from $\mathcal{S}_{\mathsf{b}}^{\ell}$ and $\mathcal{S}_{\mathsf{c}}$ respectively, $\mathsf{Pr}[\mathsf{BadHash}] \leqslant (2\mathsf{q})^2/|\mathcal{S}_{\mathsf{b}}|^{\ell} + (2\mathsf{q})^2/|\mathcal{S}_{\mathsf{c}}| \leqslant 8\mathsf{q}^2 2^{-2\kappa}$. Therefore,

$$\begin{aligned}\mathsf{Adv}_{\mathsf{par}}^{\text{aom-misis}}(\mathcal{B}, \kappa) &\geqslant \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) - \mathsf{Pr}[\mathsf{BadHash}] \\ &\geqslant \mathsf{Adv}_{\mathsf{CATZ}}^{\text{ts-uf-0}}(\mathcal{A}, \kappa)^2/\mathsf{q} - 8\mathsf{q}^2 2^{-2\kappa} \ ,\end{aligned}$$

which concludes the theorem. □

13.2 Analysis of the EKT Construction

We apply AOM-MISIS to the analysis of the EKT construction and discuss the parameter selections in this section.

13.2.1 Construction and main security theorem

We recall the EKT construction from [60] in Figure 13.4. A summary of the parameters used by the scheme is provided in Figure 13.2. The scheme also depends on a pseudorandom function PRF with suitable domain and range. The Lagrange coefficient $\lambda_{U,j} \in \mathbb{Z}_q$ for any subset $U \subseteq [n]$ and $j \in U$ is defined as $\lambda_{U,j} := \prod_{i\in U\setminus\{j\}} \frac{-i}{j-i}$.

The following theorem establishes TS-UF-4 security for EKT from the hardness of AOM-MISIS. In the theorem statement, $\mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{prf}}(\mathcal{C}, \kappa)$ refers to the standard PRF advantage of an adversary $\mathcal{C}$. The full proof of Theorem 13.2.1 is given in Section 13.2.3.

Theorem 13.2.1 (TS-UF-4 of EKT)**.** *For any integers $q = q(\kappa), k = k(\kappa), m = m(\kappa)$, any pseudorandom function* PRF*, and any* TS-UF-4 *adversary $\mathcal{A}$ making at most $\mathsf{q}_s = \mathsf{q}_s(\kappa)$ queries to* PPO *and $\mathsf{q}_h = \mathsf{q}_h(\kappa)$ queries to* RO*, given $\sigma_{\mathsf{r}} > 2N \cdot q^{\frac{1}{m}+\frac{1}{N(m-k)}}$, there exist an* AOM-MISIS *adversary $\mathcal{B}$ and a* PRF *adversary $\mathcal{C}$ running in time roughly two times that of $\mathcal{A}$ such that*

$$\mathsf{Adv}_{\mathsf{EKT}[\mathsf{PRF}]}^{\text{ts-uf-4}}(\mathcal{A}, \kappa) \leqslant \sqrt{\mathsf{q}\mathsf{Adv}_{\mathsf{par}}^{\text{aom-misis}}(\mathcal{B}, \kappa) + 8\mathsf{q}^3 2^{-2\kappa}} + n^2 \cdot \mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{prf}}(\mathcal{C}, \kappa) + \mathsf{q}_s^2 \cdot 2^{-2\kappa+1} \ .$$

Algorithm $\mathsf{Setup}(1^\kappa)$:

$\widehat{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$; $A \leftarrow [\widehat{A}|\mathbb{I}_k]$

Return A

Algorithm $\mathsf{KeyGen}()$:

$\widehat{\mathsf{sk}} \leftarrow\!\!\$\ \mathscr{D}_{\sigma_{\mathsf{sk}}}^m$; $\mathsf{pk} \leftarrow \left\lfloor 2A \cdot \widehat{\mathsf{sk}} \right\rceil_{\nu_{\mathsf{pk}}}$

For $(i,j) \in [n] \times [n]$ do $\mathsf{seed}_{i,j} \leftarrow \{0,1\}^\kappa$

$\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{t-1} \leftarrow\!\!\$\ R_q^{m-k}$

For $i \in [n]$ do

  $\mathsf{ssk}_i \leftarrow \widehat{\mathsf{sk}}_{[m-k]} + \sum_{j=1}^{t-1} \boldsymbol{a}_j i^j$

  $\mathsf{sk}_i \leftarrow (\mathsf{ssk}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[n]})$

Return $(\mathsf{pk}, (\mathsf{sk}_i)_{i\in[n]})$

Algorithm $\mathsf{SPP}(\mathsf{st}_i)$:

For $j \in [0..\ell]$ do $\boldsymbol{r}_j \leftarrow\!\!\$\ \mathscr{D}_{\sigma_{\mathsf{r}}}^m$

For $j \in [0..\ell]$ do $\boldsymbol{R}_j \leftarrow A \cdot \boldsymbol{r}_j$

$\mathsf{st}_i.\mathrm{mapPP}((\boldsymbol{R}_j)_{j\in[0..\ell]}) \leftarrow (\boldsymbol{r}_j)_{j\in[0..\ell]}$

Return $((\boldsymbol{R}_j)_{j\in[0..\ell]}, \mathsf{st}_i)$

Algorithm $\mathsf{CompPar}(\mathsf{pk}, lr)$:

For $i \in lr.\mathsf{SS}$ do

  $(b_j)_{j\in[\ell]} \leftarrow \mathrm{H}_1(\mathsf{pk}, lr)$

  $(\boldsymbol{R}_{i,j})_{j\in[0..\ell]} \leftarrow lr.\mathsf{PP}(i)$

$\boldsymbol{R} \leftarrow \left\lfloor \sum_{i\in lr.\mathsf{SS}} \left(\boldsymbol{R}_{i,0} + \sum_{j\in[\ell]} b_j \boldsymbol{R}_{i,j} \right) \right\rceil_{\nu_{\mathsf{r}}}$

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R})$

Return $(\boldsymbol{R}, c, (b_j)_{j\in[\ell]})$

Algorithm $\mathsf{PS}(lr, i, \mathsf{st}_i)$:

$pp_i \leftarrow lr.\mathsf{PP}(i)$

If $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \perp$ then return $(\perp, \mathsf{st}_i)$

$(\boldsymbol{r}_j)_{j\in[0..\ell]} \leftarrow \mathsf{st}_i.\mathrm{mapPP}(pp_i)$

$\mathsf{st}_i.\mathrm{mapPP}(pp_i) \leftarrow \perp$

$(\boldsymbol{R}, c, (b_j)_{j\in[\ell])}) \leftarrow \mathsf{CompPar}(\mathsf{st}_i.\mathsf{pk}, lr)$

$(\mathsf{ssk}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[n]}) \leftarrow \mathsf{st}_i.\mathsf{sk}$

$\mathsf{mask} \leftarrow \sum_{j\in lr.SS} \mathsf{PRF}(\mathsf{seed}_{i,j}, (\mathsf{pk}, lr))$

$\mathsf{mask}' \leftarrow \sum_{j\in lr.SS} \mathsf{PRF}(\mathsf{seed}_{j,i}, (\mathsf{pk}, lr))$

$\boldsymbol{z} \leftarrow \boldsymbol{r}_{0,[m-k]} + \sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_{j,[m-k]}$

  $+2c \cdot \lambda_{lr.\mathsf{SS},j} \cdot \mathsf{ssk}_j + \mathsf{mask} - \mathsf{mask}' \bmod q$

Return $((\boldsymbol{R}, \boldsymbol{z}), \mathsf{st}_i)$

Algorithm $\mathsf{Agg}(\mathrm{PS}, \mathsf{st}_0)$:

$\boldsymbol{R} \leftarrow \perp$; $\boldsymbol{z} \leftarrow 0$

For $(\boldsymbol{R}', \boldsymbol{z}') \in \mathrm{PS}$ do $\boldsymbol{R} \leftarrow \boldsymbol{R}'$; $\boldsymbol{z} \leftarrow \boldsymbol{z} + \boldsymbol{z}'$

$c \leftarrow \mathrm{H}_2(\mathsf{pk}, \mu, \boldsymbol{R})$

$\boldsymbol{h} \leftarrow \boldsymbol{R} - \left\lfloor \widehat{A}\boldsymbol{z} - 2^{\nu_{\mathsf{pk}}} \cdot c \cdot \overline{\mathsf{pk}} \right\rceil_{\nu_{\mathsf{r}}}$ // $\boldsymbol{h} \in R_{q_{\nu_{\mathsf{r}}}}^k$

Return $((c, \boldsymbol{z}, \boldsymbol{h}), \mathsf{st}_0)$

Algorithm $\mathsf{Vf}(\mathsf{pk}, \mu, sig)$:

$(c, \boldsymbol{z}, \boldsymbol{h}) \leftarrow sig$

If $\left\|(\boldsymbol{z}, 2^{\nu_{\mathsf{r}}}\overline{\boldsymbol{h}} \mod q)\right\|_2 > \beta_{\mathsf{z}}$ then return 0

$c' \leftarrow \mathrm{H}_2(\mathsf{pk}, \mu, \left\lfloor \widehat{A}\boldsymbol{z} - 2^{\nu_{\mathsf{pk}}} \cdot c \cdot \overline{\mathsf{pk}} \right\rceil_{\nu_{\mathsf{r}}} + \boldsymbol{h})$

Return $(c' = c)$

Figure 13.4: Lattice-based t-out-of-n threshold signatures $\mathsf{EKT}[\mathsf{PRF}]$, where PRF is a pseudo-random function. Here, $\mathrm{H}_1 : \{0,1\}^* \to \mathcal{S}_{\mathsf{b}}^\ell$ and $\mathrm{H}_2 : \{0,1\}^* \to \mathcal{S}_{\mathsf{c}}$. Also, $\lambda_{lr.\mathsf{SS},j}$ denotes the Lagrange coefficient, and $\overline{\mathsf{pk}}, \overline{\boldsymbol{h}} \in R^k$ denote the lift (see Section 11.2.1 for more details) of pk and $\boldsymbol{h}$ respectively. The algorithms LPP and LR are defined the same as in Figure 10.3.

where $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$ *and* $\mathsf{par} = (q, k, m, Q = 1 + \mathsf{q}_s(1+\ell), (\sigma_i)_{i\in[Q]}, \beta_{\mathsf{s}} = (2^{\nu_{\mathsf{r}}+2} + \beta_{\mathsf{c}} \cdot 2^{\nu_{\mathsf{pk}}+1}) \cdot \sqrt{Nk} + 4\beta_{\mathsf{z}}, \beta_{\mathsf{b}} = 4\sigma_{\mathsf{sk}}\beta_{\mathsf{c}}, \beta_{\mathsf{u}} = 1/\sigma_{\mathsf{sk}} + \beta_{\mathsf{c}}\sqrt{8N\mathsf{q}_s}/\sigma_{\mathsf{r}})$ *with* $\sigma_1 = \sigma_{\mathsf{sk}}$, $\sigma_{1+i} = \sigma_{\mathsf{r}}$ *for* $i \in [\mathsf{q}_s(\ell+1)]$.

Remark 13.1. *We note that we can directly establish* TS-UF-0 *security of* EKT *from* AOM-MISIS *by applying the unforgeability theorem from [60] and Lemma 12.2.1. However, the resulting parameters are worse than those in Theorem 13.2.1. Specifically,* β_{u} *becomes* $1/\sigma_{\mathsf{sk}} + 2\beta_{\mathsf{c}}\sqrt{\ell N\mathsf{q}_s}/\sigma_{\mathsf{r}}$, *and* β_{b} *becomes* $4\sigma_{\mathsf{sk}}\sqrt{\beta_{\mathsf{c}}N}$.

Also, we recall the correctness theorem from [60], which is needed for parameter selection later.

Theorem 13.2.2 (Correctness of EKT [60])**.** *For any integers* $1 < t \leqslant n$, *any pseudorandom function* PRF, *given* $\sigma_{\mathsf{r}} \geqslant \sqrt{(\log(2Nm) + \kappa)/\pi}$ *and* $\beta_{\mathsf{z}} \geqslant (\beta_{\mathsf{c}}2^{\nu_{\mathsf{pk}}} + 2^{\nu_{\mathsf{r}}})\sqrt{mN} + e^{1/4}(2\beta_{\mathsf{c}}\sigma_{\mathsf{sk}} + \sigma_{\mathsf{r}}\sqrt{n(1+\ell)})\sqrt{N}(\sqrt{k} + \sqrt{m-k})$, *the threshold signature scheme* EKT[PRF] *is correct with correctness error negligible in* κ.

13.2.2 Parameter selection

In this section, we first discuss the asymptotic parameters selection derived from the security theorems and the hardness of AOM-MISIS, then compare these parameters with those proposed in [60], and finally estimate the concrete efficiency based on the parameter selections. We also discuss how our parameters are compared to the parameters of Ringtail [33].

ASYMPTOTIC PARAMETER SELECTIONS. Denote β_{lwe} as the norm of the underlying MLWE assumption. Initially, we select $N, m, k, \beta_{\mathsf{lwe}}$ such that N is a power of $N \geqslant 2\kappa$, $m, k = \mathsf{poly}(\kappa)$, and $\beta_{\mathsf{lwe}} \geqslant m\log(N)$.[3] (We note that when estimating the concrete efficiency, we will enumerate through plausible $(N, m, k, \beta_{\mathsf{lwe}})$ tuples and pick the one that yields the best efficiency.) Then, we set other parameters as follows.

- Set β_{c} as the smallest integer such that $2^{\beta_{\mathsf{c}}}\binom{N}{\beta_{\mathsf{c}}} \geqslant 2^{2\kappa}$.
- $\sigma_{\mathsf{sk}} = \max\{2\beta_{\mathsf{lwe}}\sqrt{mN}, \sqrt{\log(6mN)/\pi}\}$. The first term is usually the leading term.
- $\sigma_{\mathsf{r}} = \max\{\sigma_{\mathsf{sk}}\beta_{\mathsf{c}}\sqrt{8N\mathsf{q}_s}, \sqrt{(\log(2Nm) + \kappa)/\pi}, 2N \cdot q^{\frac{1}{m} + \frac{1}{N(m-k)}}\}$. The first term is usually the leading term.

[3] This is for guaranteeing the underlying MLWE is hard.

- $\nu_{\mathsf{pk}} = \log_2(\sigma_{\mathsf{r}}/\beta_{\mathsf{c}})$ and $\nu_{\mathsf{r}} = \log_2(\sigma_{\mathsf{r}})$.
- β_{z} is set as shown in Theorem 13.2.2
- Denote $\beta_{\mathsf{sis}} = 2\beta_{\mathsf{z}} + 4\sigma_{\mathsf{sk}}\beta_{\mathsf{c}}\sqrt{mN}$.
- Select q such that the problem $\mathrm{MSIS}_{q,k,m,\beta_{\mathsf{sis}}}$ and the problem $\mathrm{MLWE}_{q,N,k,m,\beta_{\mathsf{lwe}}}$ are assumed to be exponentially hard in κ.

By Theorem 13.2.1 and Theorem 12.3.1 with $\varepsilon = 1/2$ and $\alpha = 2$ (we can further optimize the concrete bound by adjusting α), TS-UF-4 of EKT is implied by the hardness of $\mathrm{MSIS}_{q,k,m,\beta_{\mathsf{sis}}}$ and $\mathrm{MLWE}_{q,N,k,m,\beta_{\mathsf{lwe}}}$.

COMPARISON WITH [60]. Although the prior work does not give a security reduction to standard lattice assumptions, they provide candidate asymptotic parameters based on the heuristic assumption that the selective version of AOM-MLWE is as hard as the adaptive version. Still, our asymptotic parameters are slightly better than their candidate asymptotic parameters provided in [60]. The key difference lies in the choice of σ_{r}, which significantly impacts efficiency. In particular, the prior work requires $\sigma_{\mathsf{r}} = \Omega(\beta_{\mathsf{c}}\beta_{\mathsf{lwe}}N\sqrt{\mathsf{q}_s Nk})$, while we require $\sigma_{\mathsf{r}} = \Omega(\beta_{\mathsf{c}}\beta_{\mathsf{lwe}}N\sqrt{\mathsf{q}_s m})$. Therefore, there is roughly a factor of $\sqrt{N}$ improvement.

COMPARISON WITH RINGTAIL [33]. Ringtail is very close to EKT. The main differences are that the output space of hash function H_1 changes and the nonces $\boldsymbol{r}_j$ are sampled from unbalanced discrete Gaussian distributions. In particular, in Ringtail, the first $m-k$ entries of each nonce $\boldsymbol{r}_{j,[m-k]}$ is sampled from $\mathscr{D}^{m-k}_{\sigma_{\mathsf{r}}}$, while the rest $\boldsymbol{r}_{j,[(m-k+1)..m]}$ is sampled from $\mathscr{D}^{k}_{\sigma'_{\mathsf{r}}}$ with $\sigma'_{\mathsf{r}} \neq \sigma_{\mathsf{r}}$. Therefore, it is possible to compare the parameter selections directly.

The key difference still lies in the choice of σ_{r}. In particular, Ringtail requires $\sigma_{\mathsf{r}} = \Omega(\beta_{\mathsf{c}}\sqrt{\mathsf{q}_h})$, where q_h denotes the number of random oracle queries. Here we offer different trade-offs. A key drawback of their parameters is that σ_{r} depends on q_h, which is typically assumed to be much larger than q_s. This is because q_s refers to the number of online signing queries and is a system parameter that can be enforced, while q_h scales with the offline computational power of the adversary. However, if we set $\mathsf{q}_h = \mathsf{q}_s$, their σ_{r} is smaller than ours roughly by a factor of $\beta_{\mathsf{lwe}}N\sqrt{m}$.

CONCRETE EFFICIENCY. We show a set of concrete parameters and estimated efficiency for $\kappa \in \{128, 192, 256\}$ and $n = 1024$ in Figure 13.5. We derive the parameters following our parameter selections mentioned above, and similar to the CTZ scheme, we estimate the concrete hardness of

κ	$\log_2(q)$	k	m	σ_{sk}	σ_{r}	β_{z}	$\lvert\mathsf{pk}\rvert$	$\lvert sig\rvert$	Comm.
128	69.3	6	11	$2^{10.6}$	$2^{52.6}$	$2^{66.2}$	8.72KB	30.88KB	766.83KB
192	69.6	8	15	$2^{10.6}$	$2^{52.6}$	$2^{66.5}$	11.74KB	42.90KB	1.01MB
256	70.2	10	18	$2^{11.17}$	$2^{53.2}$	$2^{67.1}$	14.76KB	50.72KB	1.29MB

Figure 13.5: The concrete parameters and estimated efficiency of the EKT scheme for $\kappa = 128, 192, 256$ and $n = 1024$. We set $(N, \ell, \beta_{\mathsf{c}}) = (512, 26, 64)$. The last second column denotes the communication complexity per signer.

MSIS and MLWE using the lattice estimator. We note that our concrete parameters are worse than those given in [60], although in a similar ballpark. However, worse parameters are to be expected. This is because their parameter selection is based on their direct cryptanalysis of AOM-MLWE, whereas we rely on a reduction from two standard lattice assumptions. In a similar spirit, our parameters are worse than those claimed for Ringtail in [33], however this is to be expected, too, as they heuristically assume $\mathsf{q}_h = \mathsf{q}_s$ (also see the above discussion) to set parameters. In practice, however, we expect q_h to be best approximated conservatively by the running time of the adversary, and this can be as high as 2^{256}, whereas q_s could typically be 2^{60}. The authors of Ringtail were aware of this fact, and their choice was motivated by their conjecture that a better dependency would be possible. We confirm their conjecture for the case of EKT.

13.2.3 Security reduction of EKT

Unlike other security analyses [54, 60] of lattice-based threshold signatures that use masking techniques, our reduction follows a two-step approach, which we believe has independent interest.

In the first step, we reduce the TS-UF-4 game of EKT[PRF] to an *ideal* unforgeability game Ideal-TUF for threshold signatures (defined in Figure 13.6). In this game, no secret sharing of signing key or masking is involved. Moreover, the adversary directly obtains an *aggregation* of all partial signatures from honest parties in $lr.\mathsf{SS}$ via a second-round signing oracle SIGNO, provided that the tokens in lr for honest signers are all valid. Intuitively, the ideal game captures the information hidden by the masks. In particular, all the secret key shares and the partial signatures

Game Ideal-TUF$^{\mathcal{A}}(\kappa)$:

$\widehat{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$; $A \leftarrow [\widehat{A}|\mathbb{I}_k]$

$\mathrm{H} \leftarrow\!\!\$\ \mathsf{TS.HF}$; $\mathrm{S} \leftarrow \varnothing$

For $i \in [n]$ do

  $\mathsf{st}_i.\mathrm{mapPP} \leftarrow ()$

$(\mu, sig) \leftarrow \mathcal{A}^{\text{INIT,PPO,PSIGNO,RO}}(A)$

Return $(\mu \notin \mathrm{S} \ \wedge\ \mathsf{Vf}(\mathsf{pk}, \mu, sig) = 1)$

Oracle INIT(CS) :

Require: $CS \subseteq [n]$ and $|CS| < t$

$HS \leftarrow [n] \backslash CS$

$\mathsf{sk} \leftarrow\!\!\$\ \mathscr{D}^m_{\sigma_{\mathsf{sk}}}$

$\mathsf{pk} \leftarrow \lfloor 2A \cdot \mathsf{sk} \rceil_{\nu_{\mathsf{pk}}}$

Return pk

Oracle RO(x) :

Return $\mathrm{H}(x)$

Oracle PPO(i) :

Require: $i \in HS$

$\boldsymbol{r}_j \leftarrow\!\!\$\ \mathscr{D}^m_{\sigma_{\mathsf{r}}}$

For $j \in [0..\ell]$ do $\boldsymbol{R}_j \leftarrow A \cdot \boldsymbol{r}_j$

$pp \leftarrow (\boldsymbol{R}_j)_{j\in[0..\ell]}$

$\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (\boldsymbol{r}_j)_{j\in[0..\ell]}$

Return pp

Oracle SIGNO(lr) :

Require: $lr.\mathsf{SS} \subseteq [n]$ and $|lr.\mathsf{SS}| \geqslant t$

$\mathsf{hon} \leftarrow lr.\mathsf{SS} \cap HS$

If $\exists\ i \in \mathsf{hon} : \mathsf{st}_i.\mathrm{mapPP}(lr.\mathsf{PP}(i)) = \perp$ then

  Return $\perp$

$\mathrm{S} \leftarrow \mathrm{S} \cup \{lr.\mathsf{msg}\}$

$(c, b_1, \ldots, b_\ell) \leftarrow \mathsf{CompPar}(\mathsf{pk}, lr)$

$\boldsymbol{z} \leftarrow 2c \cdot \mathsf{sk}_{[m-k]}$

For $i \in \mathsf{hon}$ do

  $(\boldsymbol{r}_j)_{j\in[0..\ell]} \leftarrow lr.\mathsf{PP}(i)$

  $\mathsf{st}_i.\mathrm{mapPP}(lr.\mathsf{PP}(i)) \leftarrow \perp$

  $\boldsymbol{z} \leftarrow \boldsymbol{z} + \boldsymbol{r}_{0,[m-k]} + \sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_{j,[m-k]}$

Return $\boldsymbol{z}$

Figure 13.6: The Ideal-TUF game, where the algorithms CompPar and Vf are defined in Figure 13.4.

(except their aggregation) are entirely hidden by the masks.

In the second step, we establish the hardness of the ideal unforgeability game based on AOM-MISIS. The approach is cleaner than prior proofs, as it clearly separates the effects of masks from the main security reduction to the AOM-MISIS problem. In particular, the first step relies only on the security of PRF, while the second step does not involve masking at all.

Concretely, Theorem 13.2.1 is a corollary from the following two lemmas.

Lemma 13.2.3. *For any integers $q = q(\kappa), k = k(\kappa), m = m(\kappa)$, any PRF scheme* PRF*, and any* TS-UF-4 *adversary $\mathcal{A}$ making at most $\mathsf{q}_s = \mathsf{q}_s(\kappa)$ queries to* PPO *and $\mathsf{q}_h = \mathsf{q}_h(\kappa)$ queries to* RO*, given $\sigma_\mathsf{r} > 2N \cdot q^{\frac{1}{m}+\frac{1}{N(m-k)}}$, there exists an* Ideal-TUF *adversary $\mathcal{B}$ making at most q_s queries to* PPO *and q_h queries to* RO *and an* PRF *adversary $\mathcal{C}$ running in time roughly the same as $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{ts-uf-4}}_{\mathsf{EKT[PRF]}}(\mathcal{A},\kappa) \leqslant \mathsf{Adv}^{\text{ideal-tuf}}(\mathcal{B},\kappa) + n^2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{C},\kappa) + \mathsf{q}_s^2 \cdot 2^{-2\kappa+1} \ .$$

where n denotes the number of signers.

Lemma 13.2.4. *For any integers $q = q(\kappa), k = k(\kappa), m = m(\kappa)$ and any* Ideal-TUF *adversary $\mathcal{A}$ making at most q_s queries to* PPO *and q_h queries, there exists an* AOM-MISIS *adversary $\mathcal{B}$ running in time roughly two times that of $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{ideal-tuf}}(\mathcal{A},\kappa) \leqslant \sqrt{\mathsf{q}\mathsf{Adv}^{\text{aom-misis}}_{\mathsf{par}}(\mathcal{B},\kappa) + 8\mathsf{q}^3 2^{-2\kappa}} \ .$$

where $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$ and $\mathsf{par} = (q, k, m, Q = 1 + \mathsf{q}_s(1+\ell), (\sigma_i)_{i\in[Q]}, \beta_\mathsf{s} = (2^{\nu_\mathsf{r}+2} + \beta_\mathsf{c} \cdot 2^{\nu_\mathsf{pk}+1}) \cdot \sqrt{Nk} + 4\beta_\mathsf{z}, \beta_\mathsf{b} = 4\sigma_\mathsf{sk}\beta_\mathsf{c}, \beta_\mathsf{u} = 1/\sigma_\mathsf{sk} + \beta_\mathsf{c}\sqrt{8N\mathsf{q}_s}/\sigma_\mathsf{r})$ with $\sigma_1 = \sigma_\mathsf{sk}$, $\sigma_{1+i} = \sigma_\mathsf{r}$ for $i \in [\mathsf{q}_s(\ell+1)]$.

The rough idea behind the first reduction (Lemma 13.2.3) is as follows. Due to masking, the reduction can simulate the PSIGNO oracle by responding with a uniformly random vector, unless the adversary has made PSIGNO queries to all honest party in $lr.\mathsf{SS}$. In this case, the reduction queries its own SIGNO oracle to obtain an aggregated signature and derives the requested partial signature from it. Thus, the reduction only queries SIGNO for message $lr.\mathsf{msg}$ when all honest signers in $lr.\mathsf{SS}$ were queried. This implies that the set S of messages considered signed in TS-UF-4 is exactly the same as the set S defined in Ideal-TUF. The second reduction (Lemma 13.2.4) is the similar to the proof of Theorem 13.1.3, and its proof is provided in Section 13.2.4.

Proof of Lemma 13.2.3. Let $\mathcal{A}$ be a TS-UF-4 adversary described in the theorem. We show the lemma via the following series of games.

$\mathbf{G}_0$: This is the same as TS-UF-4. The game is formally defined in Figure 13.7

$\mathbf{G}_1$: The same as $\mathbf{G}_0$ except that in the oracle PSIGNO, the response z is computed in a different way, and the game aborts if there are two valid PSIGNO queries for the same input (i, lr) (denoted as BNonce). The game is formally defined in Figure 13.8. We first

Game $\mathbf{G}_0^{\mathcal{A}}(\kappa)$:

$\widehat{A} \leftarrow\!\!\$\ R_q^{k\times(m-k)}$; $A \leftarrow [\widehat{A}|\mathbb{I}_k]$

$\mathrm{H} \leftarrow\!\!\$\ \mathsf{TS.HF}$; $\mathrm{S} \leftarrow \varnothing$; $\mathrm{curSS}_{\mathsf{L}} \leftarrow ()$

$(\mu^*, sig^*) \leftarrow \mathcal{A}^{\text{INIT,PPO,PSIGNO,OPEN,RO}}(A)$

Return $(\mu^* \notin \mathrm{S} \ \wedge\ \mathsf{Vf}(\mathsf{pk}, \mu^*, sig^*) = 1)$

Oracle $\text{INIT}(CS)$:

$HS \leftarrow [n]\backslash CS$

$\widehat{\mathsf{sk}} \leftarrow\!\!\$\ \mathscr{D}^m_{\sigma_{\mathsf{sk}}}$; $\mathsf{pk} \leftarrow \left\lfloor 2A \cdot \widehat{\mathsf{sk}} \right\rceil_{\nu_{\mathsf{pk}}}$

For $(i,j) \in [n] \times [n]$ do

  $\mathsf{seed}_{i,j} \leftarrow \{0,1\}^\kappa$

$\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{t-1} \leftarrow\!\!\$\ R_q^{m-k}$

For $i \in [n]$ do

  $\mathsf{ssk}_i \leftarrow \widehat{\mathsf{sk}}_{[m-k]} + \sum_{j=1}^{t-1} \boldsymbol{a}_j i^j$

  $\mathsf{sk}_i \leftarrow (\mathsf{ssk}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[n]})$

Return $(\mathsf{sk}_i)_{i\in CS}$

Oracle $\text{PPO}(i)$:

Require: $i \in HS$

For $j \in [0..\ell]$ do $\boldsymbol{r}_j \leftarrow\!\!\$\ \mathscr{D}^m_{\sigma_{\mathsf{r}}}$

For $j \in [0..\ell]$ do $\boldsymbol{R}_j \leftarrow A \cdot \boldsymbol{r}_j$

$pp \leftarrow (\boldsymbol{R}_j)_{j\in[0..\ell]}$

$\mathsf{st}_i.\mathrm{mapPP}(pp) \leftarrow (\boldsymbol{r}_j)_{j\in[0..\ell]}$

Return pp

Oracle $\text{PSIGNO}(i, lr)$:

Require: $lr.\mathsf{SS} \subseteq [n]$ and $i \in HS \cap lr.\mathsf{SS}$

$pp_i \leftarrow lr.\mathsf{PP}(i)$

If $\mathsf{st}_i.\mathrm{mapPP}(pp_i) = \perp$ then return $\perp$

If $\mathrm{curSS}_{\mathsf{L}}(lr) = \perp$ then

  $\mathrm{curSS}_{\mathsf{L}}(lr) \leftarrow \{i\}$

Else $\mathrm{curSS}_{\mathsf{L}}(lr) \leftarrow \mathrm{curSS}_{\mathsf{L}}(lr) \cup \{i\}$

If $\mathrm{curSS}_{\mathsf{L}} = lr.\mathsf{SS} \cap HS$ then

  $\mathrm{S} \leftarrow \mathrm{S} \cup \{lr.\mathsf{msg}\}$

$(\boldsymbol{r}_j)_{j\in[0..\ell]} \leftarrow \mathsf{st}_i.\mathrm{mapPP}(pp_i)$

$\mathsf{st}_i.\mathrm{mapPP}(pp_i) \leftarrow \perp$

$(\boldsymbol{R}, c, (b_j)_{j\in[\ell])}) \leftarrow \mathsf{CompPar}(\mathsf{pk}, lr)$

$(\mathsf{ssk}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[n]}) \leftarrow \mathsf{sk}_i$

$\mathsf{mask} \leftarrow \sum_{j\in lr.SS} \mathsf{PRF}(\mathsf{seed}_{i,j}, (\mathsf{pk}, lr))$

$\mathsf{mask}' \leftarrow \sum_{j\in lr.SS} \mathsf{PRF}(\mathsf{seed}_{j,i}, (\mathsf{pk}, lr))$

$\boldsymbol{z} \leftarrow \boldsymbol{r}_{0,[m-k]} + \sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_{j,[m-k]}$
$+2c \cdot \lambda_{lr.\mathsf{SS},j} \cdot \mathsf{ssk}_j + \mathsf{mask} - \mathsf{mask}' \bmod q$

Return $(\boldsymbol{R}, \boldsymbol{z})$

Oracle $\text{RO}(x)$:

Return $\mathrm{H}(x)$

Figure 13.7: The $\mathbf{G}_0$ game, where the algorithms CompPar and Vf are defined in Figure 13.4.

---

show that if BNonce does not occur, the game is identical to $\mathbf{G}_0$. From the description of $\mathbf{G}_1$, if $\mathrm{curSS}_{\mathsf{L}}(lr) \neq \mathsf{hon}$, we have $\boldsymbol{z} = \boldsymbol{v} + \mathsf{mask}_{\mathsf{c}} = \boldsymbol{r}_{0,[m-k]} + (\sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_{j,[m-k]}) + 2c \cdot \lambda_{lr.\mathsf{SS},i} \cdot \mathsf{ssk}_i + \mathsf{mask}_{\mathsf{h}} + \mathsf{mask}_{\mathsf{c}}$, which is exactly the same as $\mathbf{G}_0$. Otherwise, $\boldsymbol{z} =$

Game $\boxed{\mathbf{G}_1^{\mathcal{A}}(\kappa)}$, $\mathbf{G}_2^{\mathcal{A}}(\kappa)$, $\boxed{\mathbf{G}_3^{\mathcal{A}}(\kappa)}$:

Oracle $\text{PSIGNO}(i, lr)$:

Require: $lr.\mathsf{SS} \subseteq [n]$ and $i \in HS \cap lr.\mathsf{SS}$; $pp_i \leftarrow lr.\mathsf{PP}(i)$

If $\mathsf{st}_i.\text{mapPP}(pp_i) = \perp$ then return $\perp$

If $i \in \text{curSS}_{\mathsf{L}}(lr)$ then the game aborts // Bad event BNonce

$\mathsf{hon} \leftarrow lr.SS \backslash CS$; $\mathsf{cor} \leftarrow lr.SS \cap CS$

$\text{curSS}_{\mathsf{L}}(lr) \leftarrow \text{curSS}_{\mathsf{L}}(lr) \cup \{i\}$

If $\text{curSS}_{\mathsf{L}} = \mathsf{hon}$ then $\text{S} \leftarrow \text{S} \cup \{lr.\mathsf{msg}\}$

$(\boldsymbol{R}, c, (b_j)_{j\in[\ell])}) \leftarrow \mathsf{CompPar}(\mathsf{pk}, lr)$

$\mathsf{mask_c} \leftarrow \sum_{j\in\mathsf{cor}} \mathsf{PRF}(\mathsf{seed}_{i,j}, (\mathsf{pk}, lr)) - \sum_{j\in\mathsf{cor}} \mathsf{PRF}(\mathsf{seed}_{j,i}, (\mathsf{pk}, lr))$

$(\boldsymbol{r}_j)_{j\in[0..\ell]} \leftarrow \mathsf{st}_i.\text{mapPP}(pp_i)$; $\mathsf{st}_i.\text{mapPP}(pp_i) \leftarrow \perp$

$\text{curSumR}(lr) \leftarrow \text{curSumR}(lr) + \boldsymbol{r}_{0,[m-k]} + \sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_{j,[m-k]}$

If $\text{curSS}_{\mathsf{L}}(lr) \neq \mathsf{hon}$ then // i is not the last queried honest party in $lr.\mathsf{SS}$

$(\mathsf{ssk}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[n]}) \leftarrow \mathsf{sk}_i$

$\mathsf{mask_h} \leftarrow\!\!\$\ \sum_{j\in\mathsf{hon}}(\mathsf{PRF}(\mathsf{seed}_{i,j}, (\mathsf{pk}, lr)) - \mathsf{PRF}(\mathsf{seed}_{j,i}, (\mathsf{pk}, lr)))$

$\mathsf{mask_h} \leftarrow\!\!\$\ R_q^{m-k}$

$\boldsymbol{v} \leftarrow \boldsymbol{r}_{0,[m-k]} + (\sum_{j\in[\ell]} b_j \cdot \boldsymbol{r}_{j,[m-k]}) + 2c \cdot \lambda_{lr.\mathsf{SS},i} \cdot \mathsf{ssk}_i + \mathsf{mask_h}$

$\boldsymbol{v} \leftarrow\!\!\$\ R_q^{m-k}$

$\text{curSum}(lr) \leftarrow \text{curSum}(lr) + \boldsymbol{v}$

Else // i is the last queried honest signer

$\boldsymbol{v} \leftarrow \text{curSumR}(lr) + 2c \cdot \widehat{\mathsf{sk}}_{[m-k]} - \text{curSum}(lr) - \sum_{j\in\mathsf{cor}} 2c \cdot \lambda_{lr.\mathsf{SS},j} \cdot \mathsf{ssk}_j$

$\boldsymbol{z} \leftarrow \boldsymbol{v} + \mathsf{mask_c}$

Return $(\boldsymbol{R}, \boldsymbol{z})$

Figure 13.8: The PSIGNO oracle of the games $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$, where $\mathbf{G}_1$ only contains dashed boxes, $\mathbf{G}_2$ only contains highlighted boxes, and $\mathbf{G}_3$ only contains solid boxes. In addition, each entry of the tables curSum and curSumR is initialized to 0. The rest of each game is identical to $\mathbf{G}_0$.

$\mathrm{curSumR}(lr) + 2c \cdot \widehat{\mathsf{sk}}_{[m-k]} - \mathrm{curSum}(lr) - \sum_{j \in \mathsf{cor}} 2c \cdot \lambda_{lr.\mathsf{SS},j} \cdot \mathsf{ssk}_j + \mathsf{mask}_\mathsf{c}$. Here, $\mathrm{curSumR}(lr) = \sum_{i' \in \mathsf{hon}} \boldsymbol{r}^{(i')}_{0,[m-k]} + (\sum_{j \in [\ell]} b_j \cdot \boldsymbol{r}^{(i')}_{j,[m-k]})$, where $\{\boldsymbol{r}^{(i')}_j\}_{j \in [0..\ell]}$ denotes the nonces for signer $i' \in \mathsf{hon}$, and $\mathrm{curSum}(lr) = \sum_{i' \in \mathsf{hon} \setminus \{i\}} \boldsymbol{v}^{(i')} = \sum_{i' \in \mathsf{hon} \setminus \{i\}} (\boldsymbol{r}^{(i')}_{0,[m-k]} + (\sum_{j \in [\ell]} b_j \cdot \boldsymbol{r}^{(i')}_{j,[m-k]}) + 2c \cdot \lambda_{lr.\mathsf{SS},i'} \cdot \mathsf{ssk}_{i'} + {\mathsf{mask}_\mathsf{h}}^{(i')})$, where $(\cdot)^{(i')}$ denotes the value computed during the query (i', lr). Since $\widehat{\mathsf{sk}} = \sum_{i' \in lr.\mathsf{SS}} \lambda_{lr.\mathsf{SS},i'} \cdot \mathsf{ssk}_{i'}$ and $\sum_{i' \in \mathsf{hon}} {\mathsf{mask}_\mathsf{h}}^{(i')} = \sum_{i' \in \mathsf{hon}} \sum_{j \in \mathsf{hon}} (\mathsf{PRF}(\mathsf{seed}_{i',j}, (\mathsf{pk}, lr)) - \mathsf{PRF}(\mathsf{seed}_{j,i'}, (\mathsf{pk}, lr))) = 0$, we have $\boldsymbol{z} = \boldsymbol{r}^{(i)}_{0,[m-k]} + (\sum_{j \in [\ell]} b_j \cdot \boldsymbol{r}^{(i)}_{j,[m-k]}) + 2c \cdot \lambda_{lr.\mathsf{SS},i} \cdot \mathsf{ssk}_i + {\mathsf{mask}_\mathsf{h}}^{(i)} + \mathsf{mask}_\mathsf{c}$, which is identical to $\mathbf{G}_0$.

We now argue that BNonce occurs with a negligible probability. Since $\mathsf{st}_i.\mathrm{mapPP}(pp_i)$ is set to $\perp$ after the query (i, lr), BNonce occurs only if the PPO oracle generates a new token $(\boldsymbol{R}_0, \ldots, \boldsymbol{R}_\ell)$ that is exactly the same as pp_i. Therefore, by the following lemma from [54], the probability that this occurs is at most $\mathsf{q}_s^2 \cdot 2^{-N+1}$, and thus, since $N \geqslant 2\kappa$,

$$\mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A}, \kappa) \geqslant \mathsf{Adv}^{\mathbf{G}_0}(\mathcal{A}, \kappa) - \mathsf{q}_s^2 \cdot 2^{-2\kappa+1} \,. \tag{13.3}$$

Lemma 13.2.5 (Lemma 3.8, [54]). *For any integers $q, m, k > 0$, any real number $\sigma > 0$ and any matrix $A \in R_q^{k \times m}$, denote a distribution $\mathcal{D}(A) := \{[A|\mathbb{I}_k] \cdot \boldsymbol{s} \mid \boldsymbol{s} \leftarrow\!\!\$\ \mathscr{D}_\sigma^{m+k}\}$. If $\sigma > 2N \cdot q^{\frac{1}{k+m} + \frac{1}{Nm}}$,*

$$\mathsf{Pr}_{A \leftarrow\!\!\$\ R_q^{k \times m}}[H_\infty(\mathcal{D}(A)) \geqslant N - 1] \geqslant 1 - 2^{-N+1} \,,$$

where $H_\infty(\mathcal{D}(A)) := -\log_2(\max_{\boldsymbol{x}' \in R_q^k} \mathsf{Pr}_{\boldsymbol{x} \leftarrow\!\!\$\ \mathcal{D}(A)}[\boldsymbol{x} = \boldsymbol{x}'])$ denotes the min-entropy of $\mathcal{D}(A)$.[4]

$\mathbf{G}_2$: The same as $\mathbf{G}_1$ except in the oracle PSIGNO, the aggregated mask mask_h computed from honest parties' seeds are replaced with a uniformly random value. The game is formally defined in Figure 13.8. We can show this game is computationally close to $\mathbf{G}_1$ by first replacing each $\mathsf{PRF}(\mathsf{seed}_{i,j}, \cdot)$ for honest parties i and j with a truly random function, which incurs at most a reduction loss of $n^2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{C}, \kappa)$. Then, the game is identical to $\mathbf{G}_1$, since for each lr, denoting $\mathsf{hon} = lr.\mathsf{SS} \cap HS$ and ${\mathsf{mask}_\mathsf{h}}^{(i)} = \sum_{j \in \mathsf{hon}} (\mathsf{PRF}(\mathsf{seed}_{i,j}, (\mathsf{pk}, lr)) - \mathsf{PRF}(\mathsf{seed}_{j,i}, (\mathsf{pk}, lr)))$, we have $\{{\mathsf{mask}_\mathsf{h}}^{(i)}\}_{i \in \mathsf{hon} \setminus \{i'\}}$ is uniformly distributed over $R_q^{k(|\mathsf{hon}|-1)}$ for any $i' \in \mathsf{hon}$. Therefore,

$$\mathsf{Adv}^{\mathbf{G}_2}(\mathcal{A}, \kappa) \geqslant \mathsf{Adv}^{\mathbf{G}_1}(\mathcal{A}, \kappa) - n^2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathcal{C}, \kappa) \,. \tag{13.4}$$

[4] We omit the bit dropping from the original lemma, as it is not needed here and it only reduces the min-entropy.

$\mathbf{G}_3$: The same as $\mathbf{G}_2$ except the value $\boldsymbol{v}$ is uniformly sampled from R_q^{m-k} if i is not the last queried honest party in $lr.\mathsf{SS}$. The game is formally defined in Figure 13.8. Since mask_h is sampled uniformly from R_q^{m-k} and only used to mask $\boldsymbol{v}$, the distribution of $\boldsymbol{v}$ is identical in both games. Therefore, we have

$$\mathsf{Adv}^{\mathbf{G}_3}(\mathcal{A}, \kappa) = \mathsf{Adv}^{\mathbf{G}_2}(\mathcal{A}, \kappa) \ . \tag{13.5}$$

We construct an Ideal-TUF adversary $\mathcal{B}$ as follows. To start with, after receiving A from the Ideal-TUF game, $\mathcal{B}$ initializes $\mathsf{st}_i.\mathrm{mapPP}$ for $i \in [n]$, tables $\mathrm{curSS}_\mathsf{L}$ and curSum following the game $\mathbf{G}_3$, then initializes a map curLR to an empty map, recording whether a lr request has made, and runs $\mathcal{A}$ with input A and access to oracles $\widetilde{\textsc{Init}}$, $\widetilde{\textsc{PPO}}$, $\widetilde{\textsc{PSign}}\textsc{O}$ and $\widetilde{\textsc{RO}}$, which are simulated as follows.

$\widetilde{\textsc{Init}}(CS)$: $\mathcal{B}$ queries $\mathsf{pk} \leftarrow \textsc{Init}(CS)$. For each $i \in CS$, $\mathcal{B}$ samples $\mathsf{ssk}_i \leftarrow\!\!\$\ R_q^{m-k}$ and $\mathsf{seed}_{i,j} \leftarrow\!\!\$\ \{0,1\}^\kappa$ for each $j \in [n]$. Finally, $\mathcal{B}$ returns $(\mathsf{ssk}_i, (\mathsf{seed}_{i,j}, \mathsf{seed}_{j,i})_{j\in[n]})_{i\in CS}$.

$\widetilde{\textsc{PPO}}, \widetilde{\textsc{RO}}$: $\mathcal{B}$ forwards queries directly to PPO and RO respectively.

$\widetilde{\textsc{PSign}}\textsc{O}(i, lr)$: The same as $\textsc{PSign}\textsc{O}(i, lr)$ in $\mathbf{G}_3$ except that when $\mathrm{curSS}_\mathsf{L}(lr) = \mathsf{hon}$, $\mathcal{B}$ queries $\hat{\boldsymbol{z}} \leftarrow \textsc{Sign}\textsc{O}(lr)$ and sets $\boldsymbol{z} \leftarrow \hat{\boldsymbol{z}} - \mathrm{curSum}(lr) - \sum_{j\in\mathsf{cor}} 2c \cdot \lambda_{lr.\mathsf{SS},j} \cdot \mathsf{ssk}_j + \mathsf{mask}_\mathsf{c}$. Also, $\mathcal{B}$ does not need to retrieve $\{\boldsymbol{r}_{i,j}\}_{j\in[0..\ell]}$ and update $\mathrm{curSumR}$, as the table $\mathrm{curSumR}$ is not used anymore.

After $\mathcal{A}$ returns, $\mathcal{B}$ outputs the output of $\mathcal{A}$.

We observe that $\mathcal{B}$ wins the Ideal-TUF game if $\mathcal{A}$ wins the game $\mathbf{G}_3$, since the message $lr.\mathsf{msg}$ is added to S in $\mathbf{G}_3$ if and only if $\mathrm{curSS}_\mathsf{L}(lr) = \mathsf{hon}$, which is exactly the scenario where $\mathcal{B}$ makes a $\textsc{Sign}\textsc{O}$ query on lr. Also, since $\mathcal{B}$ simulates the game $\mathbf{G}_3$ perfectly, it follows that $\mathsf{Adv}^{\text{ideal-tuf}}(\mathcal{B}, \kappa) \geqslant \mathsf{Adv}^{\mathbf{G}_3}(\mathcal{A}, \kappa)$. Therefore, we can conclude the lemma by Equations (13.3) to (13.5). □ □

13.2.4 Proof of Lemma 13.2.4

Let $\mathcal{A}$ be a Ideal-TUF adversary as described in the lemma. W.l.o.g. we assume that $\mathcal{A}$ is deterministic. Also, we assume if $\mathcal{A}$ returns $(\mu^*, (\boldsymbol{R}^*, \boldsymbol{z}^*))$, the RO query $\mathrm{H}_2(\mathsf{pk}, \mu^*, \boldsymbol{R}^*)$ was made by $\mathcal{A}$, which adds at most one RO query. Also, since the game makes at most one RO query to H_1 and H_2 respectively for each signing query, the total number of RO queries to each of H_1 and

H_2 is bounded $\mathsf{q} = \mathsf{q}_h + \mathsf{q}_s + 1$. We now construct an algorithm $\mathcal{C}$ compatible with the syntax in Lemma 9.2.3 and construct $\mathcal{B}$ from $\mathsf{Fork}^{\mathcal{C}}$.

CONSTRUCTION OF $\mathcal{C}$. The input of $\mathcal{C}$ consists of $A, (\boldsymbol{t}_i)_{i\in[1+\mathsf{q}_s(\ell+1)]}$, and a list of hash values $h_1, \ldots, h_{2\mathsf{q}}$, where $(A, (\boldsymbol{t}_i)_{i\in[1+\mathsf{q}_s(\ell+1)]})$ are sampled following the AOM-MISIS game, and for each $i \in [\mathsf{q}]$, h_{2i-1} is sampled uniformly from $\mathcal{S}_{\mathsf{b}}$ and h_{2i} is sampled uniformly from $\mathcal{S}_{\mathsf{c}}$. To start with, $\mathcal{C}$ initializes $\mathsf{st}_i.\mathrm{mapPP} \leftarrow ()$ for $i \in [n]$, and in addition, initializes a counter $\mathrm{ctr}_h \leftarrow 0$ for counting the number of random oracle queries. Then, $\mathcal{C}$ runs $\mathcal{A}$ on input A with access to oracles $\widetilde{\text{INIT}}$, $\widetilde{\text{PPO}}$, $\widetilde{\text{PSIGNO}}$ and $\widetilde{\text{RO}}$, which are simulated as follows.

$\widetilde{\text{INIT}}(CS)$: The same as $\text{INIT}(CS)$ except $\mathcal{C}$ sets $\mathsf{pk} \leftarrow \lfloor 2\boldsymbol{t}_0 \rceil_{\nu_{\mathsf{pk}}}$.

$\widetilde{\text{PPO}}(i)$: For the j-th query, $\mathcal{C}$ sets $\boldsymbol{R}_{\hat{j}} \leftarrow \boldsymbol{t}_{1+(j-1)(\ell+1)+\hat{j}+1}$ for $\hat{j} \in [0..\ell]$ and sets

$$\mathsf{st}_i.\mathrm{mapPP}((\boldsymbol{R}_{\hat{j}})_{\hat{j}\in[0..\ell]}) \leftarrow j \ .$$

Note that since $\mathcal{C}$ does not sample $(\boldsymbol{r}_{\hat{j}})_{\hat{j}\in[0..\ell]}$, $\mathcal{C}$ uses $\mathsf{st}_i.\mathrm{mapPP}$ to store the index j instead.

$\widetilde{\text{SIGNO}}(lr)$: The same as $\text{SIGNO}(lr)$ except that $\mathcal{C}$ computes $\boldsymbol{z} \leftarrow \mathrm{PI}(\boldsymbol{d})$, where

$$d_{\hat{j}} = \begin{cases} 2c \ , & \hat{j} = 1 \ , \\ 1 \ , & \hat{j} = 1 + (j-1)(\ell+1) + 1 \ , j \in \mathsf{honR} \\ b_{j'} \ , & \hat{j} = 1 + (j-1)(\ell+1) + 1 + j', j' \in [\ell], j \in \mathsf{honR} \ , \\ 0 \ , & o.w. \ , \end{cases} \tag{13.6}$$

and $\mathsf{honR} := \{\mathsf{st}_i.\mathrm{mapPP}(lr.\mathsf{PP}(i))\}_{i\in\mathsf{hon}}$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_1(x)$: If $\mathrm{H}_1(x) \neq \bot$, $\mathcal{C}$ returns $\mathrm{H}_1(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, lr)$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_1(x) \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{b}}^{\ell}$ and returns $\mathrm{H}_1(x)$. Otherwise, $\mathcal{C}$ increases ctr_h by 1, sets $\mathrm{H}_1(x) \leftarrow h_{2\mathrm{ctr}_h-1}$. Also, $\mathcal{C}$ computes $\boldsymbol{R} \leftarrow \sum_{i\in lr.\mathsf{SS}}(\boldsymbol{R}_{i,0} + \sum_{j\in[\ell]} b_j \cdot \boldsymbol{R}_{i,j})$, where $(\boldsymbol{R}_{i,j})_{j\in[0..\ell]} \leftarrow lr.\mathsf{PP}(i)$ and $\{b_j\}_{j\in[\ell]} \leftarrow h_{2\mathrm{ctr}_h-1}$. If $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R}) = \bot$, $\mathcal{C}$ sets $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R}) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(x)$.

$\widetilde{\text{RO}}$ query $\mathrm{H}_2(x)$: If $\mathrm{H}_2(x) \neq \bot$, $\mathcal{C}$ returns $\mathrm{H}_2(x)$. Otherwise, parse x as $(\widetilde{\mathsf{pk}}, \mu, \boldsymbol{R})$. If the parsing fails or $\widetilde{\mathsf{pk}} \neq \mathsf{pk}$, $\mathcal{C}$ sets $\mathrm{H}_2(x) \leftarrow\!\!\$\ \mathcal{S}_{\mathsf{c}}$. Otherwise, $\mathcal{C}$ increases ctr_h by 1 and sets $\mathrm{H}_2(x) \leftarrow h_{2\mathrm{ctr}_h}$. Finally, $\mathcal{C}$ returns $\mathrm{H}_2(x)$.

After receiving the output $(\mu^*, (c^*, \boldsymbol{z}^*, \boldsymbol{h}^*))$ from $\mathcal{A}$, $\mathcal{C}$ aborts if $\mathcal{A}$ does not win the Ideal-TUF game. Otherwise $\mathcal{C}$ computes $\boldsymbol{R}^* \leftarrow \left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}} \cdot c^* \cdot \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}} + \boldsymbol{h}^*$ and finds the index I such that $\mathrm{H}_2(\mathsf{pk}, \mu^*, \boldsymbol{R}^*)$ is set to h_I during the simulation. By our assumption of $\mathcal{A}$, we know such I must exist. Then, $\mathcal{C}$ returns $(I, \mathrm{Out} = (\mu^*, \boldsymbol{R}^*, c^*, \boldsymbol{z}^*, \boldsymbol{h}^*))$.

ANALYSIS OF $\mathcal{C}$. To use Lemma 9.2.3, we define $S := \{2j\}_{j\in[\mathsf{q}]}$ and IG as the algorithm that samples $(A, (\boldsymbol{t}_i)_{i\in[1+\mathsf{q}_s(\ell+1)]})$ following the AOM-MISIS game, and HG as the algorithm that samples h_{2i-1} uniformly from $\mathcal{S}_{\mathsf{b}}$ and samples h_{2i} uniformly from $\mathcal{S}_{\mathsf{c}}$ for each $i \in [\mathsf{q}]$. From the simulation, we know that the output index I of $\mathcal{C}$ is always in S. Also, it is not hard to check that $\mathcal{C}$ simulates the game Ideal-TUF perfectly, which implies $\mathrm{acc}(\mathcal{C}) \geqslant \mathsf{Adv}^{\text{ideal-tuf}}(\mathcal{A}, \kappa)$. By Lemma 9.2.3, we have that

$$\mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) \geqslant \mathsf{Adv}^{\text{ideal-tuf}}(\mathcal{A}, \kappa)^2/\mathsf{q} \ .$$

CONSTRUCT $\mathcal{B}$ FROM $\mathsf{Fork}^{\mathcal{C}}$. We now construct the AOM-MISIS adversary $\mathcal{B}$ using $\mathsf{Fork}^{\mathcal{C}}$. To start with, $\mathcal{B}$ receives $(A, \{\boldsymbol{t}_i\}_{i\in[Q]})$ from the AOM-MISIS game with $Q = K + 1 + \mathsf{q}_s(\ell + 1)$ and runs $\mathsf{Fork}^{\mathcal{C}}(A, \{\boldsymbol{t}_i\}_{i\in[Q]})$ with access to the PI oracle from the AOM-MISIS game. If $\mathsf{Fork}^{\mathcal{C}}$ outputs $(I, \mathrm{Out} = (\mu^*, \boldsymbol{R}^*, c^*, \boldsymbol{z}^*, \boldsymbol{h}^*), \widetilde{\mathrm{Out}} = (\widetilde{\mu}^*, \widetilde{\boldsymbol{R}}^*, \widetilde{c}^*, \widetilde{\boldsymbol{z}}^*, \widetilde{\boldsymbol{h}}^*))$ and $c^* \neq \widetilde{c}^*$, $\mathcal{B}$ sets $\hat{\boldsymbol{s}} \leftarrow (\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*, 2(c^* - \widetilde{c}^*)\boldsymbol{t}_0 - \widehat{A}(\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*))$ and $\hat{\boldsymbol{b}} \leftarrow (2(c^* - \widetilde{c}^*), 0, \ldots, 0)$. Otherwise, $\mathcal{B}$ aborts. It is clear that $A\hat{\boldsymbol{s}} = \widehat{A}(\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*) + 2(c^* - \widetilde{c}^*)\boldsymbol{t}_0 - \widetilde{A}(\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*) = 2(c^* - \widetilde{c}^*)\boldsymbol{t}_0 = \sum_{i\in[Q]} \hat{b}_i \boldsymbol{t}_i$.

We now show how $\mathcal{B}$ sets $\boldsymbol{u}$ such that $\hat{\boldsymbol{b}}^T\boldsymbol{u} \neq 0$ and $\boldsymbol{d}^T\boldsymbol{u} = 0 \mod q$ for any oracle query $\mathrm{PI}(\boldsymbol{d})$. Note that $\mathcal{B}$ only makes PI queries while simulating oracle $\widetilde{\mathrm{SIGNO}}$. We set $u_1 = 1$ and thus $\hat{\boldsymbol{b}}^T\boldsymbol{u} = 2(c^* - \widetilde{c}^*) \neq 0$.

Enumerating j from 1 to q_s, $\mathcal{C}$ sets $u_{1+(j-1)(\ell+1)+[\ell+1]}$ such that

$$\sum_{i\in[1+j(\ell+1)]} u_i d_i = 0 \text{ for each PI query } \boldsymbol{d} \text{ with } d_{1+[j(\ell+1)]} \neq \boldsymbol{0} \ . \tag{13.7}$$

To simplify notation in the following analysis, we use $\boldsymbol{v}$ to denote the vector $u_{1+(j-1)(\ell+1)+[\ell+1]} \in R^{\ell+1}$. Concretely, $\mathcal{C}$ sets $\boldsymbol{v}$ as follows. Suppose Equation (13.7) holds for $j-1$ (except when $j = 1$, i.e., no condition is required for the case $j = 1$). We say a $\widetilde{\mathrm{SIGNO}}$ query lr corresponds to the j-th token if and only if it is the valid query with $\mathsf{st}_i.\mathrm{mapPP}(lr.\mathsf{PP}(i)) = j$, where a valid query means the $\widetilde{\mathrm{SIGNO}}$ oracle does not return $\perp$. From the simulation, there is at most one $\widetilde{\mathrm{SIGNO}}$ query corresponding to the j-th token during each execution of $\mathcal{A}$. Therefore, there are the following cases:

Case 1: No query corresponds to the j-th token during both executions. In this case, $\mathcal{B}$ set $\boldsymbol{v} \leftarrow \mathbf{0}$.

Case 2: Only one query corresponds to the j-th token during the two executions. Denote $\boldsymbol{d}$ as the PI query made during the execution of the $\widetilde{\text{SIGNO}}$ query corresponding to the j-th token, where $\boldsymbol{d}$ follows the form given in Equation (13.6). $\mathcal{B}$ sets $\boldsymbol{v} \leftarrow (-\sum_{i\in[1+(j-1)(\ell+1)]} u_i d_i, 0, \ldots, 0)$, and it follows $\sum_{i\in[1+j(\ell+1)]} u_i d_i = \sum_{i\in[1+(j-1)(\ell+1)]} u_i d_i + v_1 = 0$. Also, since $\boldsymbol{d}$ is the PI query with $d_{1+(j-1)(\ell+1)+[\ell+1]} \neq \mathbf{0}$, it follows that Equation (13.7) holds.

Case 3: There is one query corresponding to the j-th token during each of the two executions. Denote $\boldsymbol{d}$ (resp. $\widetilde{\boldsymbol{d}}$) as the PI query made during the execution of the $\widetilde{\text{SIGNO}}$ query corresponding to the j-th token before (resp. after) rewinding. If $\boldsymbol{d} = \widetilde{\boldsymbol{d}}$, then $\mathcal{C}$ sets $\boldsymbol{v}$ in the same way as Case 2.

Otherwise, let $\hat{k} \in [\ell]$ be the index such that $d_{1+(j-1)(\ell+1)+1+\hat{k}} \neq \widetilde{d}_{1+(j-1)(\ell+1)+1+\hat{k}}$. (If such $\hat{k}$ does not exist, $\mathcal{B}$ aborts.) Denote $b := d_{1+(j-1)(\ell+1)+1+\hat{k}}$ and $\widetilde{b} := d_{1+(j-1)(\ell+1)+1+\hat{k}}$. Since Equation (13.7) holds for $j-1$, we have either $\sum_{i\in[1+(j-1)(\ell+1)]} u_i d_i = 0$ or $\sum_{i\in[1+(j-1)(\ell+1)]} u_i d_i = d_1 \in 2\mathcal{S}_{\mathsf{c}}$ by Equation (13.6), where $2\mathcal{S}_{\mathsf{c}} := \{2c \mid c \in \mathcal{S}_{\mathsf{c}}\}$. Therefore, denote $\Delta := \frac{1}{2}\sum_{i\in[1+(j-1)(\ell+1)]} u_i d_i$ and $\widetilde{\Delta} := \frac{1}{2}\sum_{i\in[1+(j-1)(\ell+1)]} u_i \widetilde{d}_i$. Since $b, \widetilde{b} \in \mathcal{S}_{\mathsf{b}}$, by Lemma 11.2.1, there exists $\gamma \in R$ such that $\gamma(b - \widetilde{b}) = 2 \mod q$. $\mathcal{C}$ sets

$$v_{\hat{j}} \leftarrow \begin{cases} -2\Delta + b\gamma(\Delta - \widetilde{\Delta}) \ , & \hat{j} = 1 \ , \\ -\gamma(\Delta - \widetilde{\Delta}) \ , & \hat{j} = \hat{k} \ , \\ 0 \ , & o.w. \end{cases}$$

Then, it holds that $\sum_{i\in[1+j(\ell+1)]} u_i d_i = \sum_{i\in[1+(j-1)(\ell+1)]} u_i d_i + v_1 + b v_{\hat{k}} = 2\Delta - 2\Delta + b\gamma(\Delta - \bar{\Delta}) - b\gamma(\Delta - \bar{\Delta}) = 0$ and $\sum_{i\in[1+j(\ell+1)]} u_i \widetilde{d}_i = \sum_{i\in[1+(j-1)(\ell+1)]} u_i \widetilde{d}_i + v_1 + \widetilde{b} v_{\hat{k}} = 2\widetilde{\Delta} - 2\Delta + b\gamma(\Delta - \widetilde{\Delta}) - \widetilde{b}\gamma(\Delta - \widetilde{\Delta}) = 2\widetilde{\Delta} - 2\Delta + (b - \widetilde{b})\gamma(\Delta - \widetilde{\Delta}) = 2\widetilde{\Delta} - 2\Delta + 2(\Delta - \widetilde{\Delta}) = 0$. Finally, since $\boldsymbol{d}$ and $\widetilde{\boldsymbol{d}}$ are the PI queries with $d_{1+(j-1)(\ell+1)+[\ell+1]} \neq \mathbf{0}$, it follows that Equation (13.7) holds.

ANALYSIS OF $\mathcal{B}$. Denote $\mathsf{BadHash}$ as the event that $h_1, \widetilde{h}_1, \ldots, h_{2\mathsf{q}}, \widetilde{h}_{2\mathsf{q}}$ are *not* all distinct. We now show that $\mathcal{B}$ wins the AOM-MISIS game if $\mathsf{Fork}^{\mathcal{C}}$ returns and $\mathsf{BadHash}$ does not occur.

We first show that if $\mathsf{Fork}^{\mathcal{C}}$ returns and $\mathsf{BadHash}$ does not occur, $\mathcal{B}$ does not abort. (The following argument is similar to the one provided in the security reduction of the CTZ protocol.) Suppose

$\mathsf{Fork}^{\mathcal{C}}$ returns and BadHash does not occur. Then, the only step where $\mathcal{B}$ might abort is in Case 3 above. In Case 3, suppose $\boldsymbol{d} \neq \widetilde{\boldsymbol{d}}$ and $d_{1+(j-1)(\ell+1)+1+[\ell]} = \widetilde{d}_{1+(j-1)(\ell+1)+1+[\ell]}$ (in which case $\mathcal{B}$ aborts). Let lr be the $\widetilde{\text{SIGNO}}$ query that corresponds to the j-th token *before* rewinding. Then, there exists $J \in [\mathsf{q}]$ such that $h_{2J-1} = \mathrm{H}_1(\mathsf{pk}, lr) = d_{1+(j-1)(\ell+1)+1+[\ell]}$. Since BadHash does not occur, the only possibility is that the $2J-1 < I$ and lr is also the $\widetilde{\text{SIGNO}}$ query that corresponds to the j-th token *after* rewinding. Let $\boldsymbol{R}$ be the aggregated nonce computed from $\mathsf{CompPar}(\mathsf{pk}, lr)$. Denote $\hat{J} \in [\mathsf{q}]$ be the index such that $h_{2\hat{J}} = \mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R})$ before rewinding. From the simulation of the random oracles, it holds that $\hat{J} \leqslant J$ and thus $2\hat{J} \leqslant 2J \leqslant I$. Also, since $lr.\mathsf{msg} \neq \mu^*$ (o.w., $\mathcal{C}$ would not win the game), we have $2\hat{J} \neq I$ and thus $2\hat{J} < I$. Therefore, $\mathrm{H}_2(\mathsf{pk}, lr.\mathsf{msg}, \boldsymbol{R})$ in the second execution of $\mathcal{C}$ is also $h_{2\hat{J}}$. This implies $\boldsymbol{d} = \bar{\boldsymbol{d}}$, which contradicts our assumption.

Suppose $\mathcal{B}$ does not abort. From the way $\mathcal{B}$ sets $\boldsymbol{u}$, Equation (13.7) holds for $j = \mathsf{q}_s$. Then, since for each PI query $\boldsymbol{d}$, the component $d_{1+[\mathsf{q}_s(\ell+1)]} \neq \mathbf{0}$, it follows that $\boldsymbol{d}^T\boldsymbol{u} = \sum_{i\in[1+\mathsf{q}_s(\ell+1)]} u_i d_i = 0$. It is left to bound the norms of vectors $\hat{\boldsymbol{s}}$, $\hat{\boldsymbol{b}}$ and $\boldsymbol{u}$.

We first bound the norm of $\hat{\boldsymbol{s}}$. We have

$$\|\hat{\boldsymbol{s}}[m-k]\| = \|\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*\| \leqslant 2\beta_{\mathsf{z}} \tag{13.8}$$

and, by Lemma 13.2.6 (proved in [60]),

$$\begin{aligned}
&\|\hat{\boldsymbol{s}}[(m-k+1)..m]\| = \left\|2(c^* - \widetilde{c}^*)\boldsymbol{t}_0 - \widehat{A}(\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*)\right\| \\
&\leqslant \left\|(c^* - \widetilde{c}^*)2^{\nu_{\mathsf{pk}}}\overline{\mathsf{pk}} - \widehat{A}(\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*)\right\| + \left\|(c^* - \widetilde{c}^*)\cdot(2\boldsymbol{t}_0 - 2^{\nu_{\mathsf{pk}}}\overline{\mathsf{pk}})\right\| \\
&\leqslant \left\|(c^* - \widetilde{c}^*)2^{\nu_{\mathsf{pk}}}\mathsf{pk} - \widehat{A}(\boldsymbol{z}^* - \widetilde{\boldsymbol{z}}^*)\right\| + \beta_{\mathsf{c}}2^{\nu_{\mathsf{pk}}+1}\sqrt{Nk} \quad \text{(by Lemma 13.2.6)} \\
&\leqslant \left\|2^{\nu_{\mathsf{r}}}\overline{\left\lfloor \widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}}\widetilde{c}^*\mathsf{pk}\right\rceil_{\nu_{\mathsf{r}}}} - 2^{\nu_{\mathsf{r}}}\overline{\left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}}c^*\mathsf{pk}\right\rceil_{\nu_{\mathsf{r}}}} \mod q\right\| \\
&\quad + \left\|(\widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}}c^*\mathsf{pk}) - 2^{\nu_{\mathsf{r}}}\overline{\left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}}c^*\mathsf{pk}\right\rceil_{\nu_{\mathsf{r}}}}\right\| \\
&\quad + \left\|(\widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}}\widetilde{c}^*\mathsf{pk}) - 2^{\nu_{\mathsf{r}}}\overline{\left\lfloor \widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}}\widetilde{c}^*\mathsf{pk}\right\rceil_{\nu_{\mathsf{r}}}}\right\| + \beta_{\mathsf{c}}2^{\nu_{\mathsf{pk}}+1}\sqrt{Nk} \\
&\leqslant \left\|2^{\nu_{\mathsf{r}}}\left(\overline{\left\lfloor \widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}}\widetilde{c}^*\mathsf{pk}\right\rceil_{\nu_{\mathsf{r}}}} - \overline{\left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}}c^*\mathsf{pk}\right\rceil_{\nu_{\mathsf{r}}}}\right) \mod q\right\| \\
&\quad + 2^{\nu_{\mathsf{r}}+1}\sqrt{Nk} + \beta_{\mathsf{c}}2^{\nu_{\mathsf{pk}}+1}\sqrt{Nk}\,. \quad \text{(by Lemma 13.2.6)}
\end{aligned} \tag{13.9}$$

Lemma 13.2.6 (Lemma 3.14 [60]). *For any integers $v \geqslant 4$ and $q > 2^v$, let $q_v = \lfloor q/2^v \rceil$. Moreover,*

assume q and v satisfies $q_v = \lfloor q/2^v \rceil$. Then, for any $x \in \mathbb{Z}_q$, we have

$$\left| x - 2^v \cdot \overline{\lfloor x \rceil_v} \right| \leqslant 2^v - 1 \ .$$

Since $\left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}} c^* \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}} + \boldsymbol{h}^* = \boldsymbol{R}^* = \widetilde{\boldsymbol{R}}^* = \left\lfloor \widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}} \widetilde{c}^* \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}} + \widetilde{\boldsymbol{h}}^*$, there exists $\boldsymbol{\delta} \in R^k$ such that $\|\boldsymbol{\delta}\|_\infty \leqslant 2$ and $\overline{\left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}} c^* \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}}} + \overline{\boldsymbol{h}^*} = \overline{\left\lfloor \widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}} \widetilde{c}^* \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}}} + \overline{\widetilde{\boldsymbol{h}}^*} + q_{\nu_{\mathsf{r}}} \cdot \boldsymbol{\delta}$. Therefore,

$$\begin{aligned} &\left\| 2^{\nu_{\mathsf{r}}} \left(\overline{\left\lfloor \widehat{A}\widetilde{\boldsymbol{z}}^* - 2^{\nu_{\mathsf{pk}}} \widetilde{c}^* \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}}} - \overline{\left\lfloor \widehat{A}\boldsymbol{z}^* - 2^{\nu_{\mathsf{pk}}} c^* \mathsf{pk} \right\rceil_{\nu_{\mathsf{r}}}} \right) \mod q \right\| \\ &\leqslant \left\| 2^{\nu_{\mathsf{r}}} \overline{\boldsymbol{h}^*} \mod q \right\| + \left\| 2^{\nu_{\mathsf{r}}} \overline{\widetilde{\boldsymbol{h}}^*} \mod q \right\| + \left\| 2^{\nu_{\mathsf{r}}} q_{\nu_{\mathsf{r}}} \cdot \boldsymbol{\delta} \mod q \right\| \\ &\leqslant 2\beta_{\mathsf{z}} + 2^{\nu_{\mathsf{r}}} \|\boldsymbol{\delta}\| \leqslant 2\beta_{\mathsf{z}} + 2^{\nu_{\mathsf{r}}+1} \sqrt{Nk} \ . \end{aligned} \tag{13.10}$$

Therefore, by Equations (13.8) to (13.10), $\|\hat{\boldsymbol{s}}\| \leqslant (2^{\nu_{\mathsf{r}}+2} + \beta_{\mathsf{c}} \cdot 2^{\nu_{\mathsf{pk}}+1}) \cdot \sqrt{Nk} + 4\beta_{\mathsf{z}} \leqslant \beta_{\mathsf{s}}$.

Also, $\left\| (\hat{b}_1 \cdot \sigma_1, \ldots, \hat{b}_Q \cdot \sigma_Q) \right\|_1 = \|2(c^* - \widetilde{c}^*)\sigma_{\mathsf{sk}}\|_1 \leqslant 4\sigma_{\mathsf{sk}}\beta_{\mathsf{c}} \leqslant \beta_{\mathsf{b}}$.

It is left to bound $\|(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)\|$. For $j \in [\mathsf{q}_s]$, denote $\boldsymbol{v} := u_{K+1+(j-1)(\ell+1)+[\ell+1]}$. There are three cases as mentioned in the construction of $\mathcal{B}$. For the first case, $\|\boldsymbol{v}\| = 0$. For the second case, since $v_1 \in 2\mathcal{S}_{\mathsf{c}} \cup \{0\}$, $\|\boldsymbol{v}\| = \|\boldsymbol{v}\| \leqslant 2\sqrt{\beta_{\mathsf{c}}}$. For the third case, since $\Delta, \widetilde{\Delta} \in \mathcal{S}_{\mathsf{c}} \cup \{0\}$, $\|\Delta\|_1 \leqslant \beta_{\mathsf{c}}$ and $\left\|\tilde{\Delta}\right\|_1 \leqslant \beta_{\mathsf{c}}$. Since $\|\gamma\| = \sqrt{N}$ (by Lemma 11.2.1),

$$\begin{aligned} \|\boldsymbol{v}\|^2 &= \left\| \gamma(\widetilde{b}\Delta - b\widetilde{\Delta}), 0, \ldots, 0, -\gamma(\Delta - \widetilde{\Delta}), 0, \ldots, 0) \right\|^2 \\ &\leqslant 2(\|\Delta\|_1 + \left\|\bar{\Delta}\right\|_1)^2 \|\gamma\|^2 \leqslant 8N\beta_{\mathsf{c}}^2 \ . \end{aligned}$$

Therefore, $\|(u_1/\sigma_1, \ldots, u_Q/\sigma_Q)\| \leqslant u_1/\sigma_{\mathsf{sk}} + \sqrt{\sum_{j \in [\mathsf{q}_s]} \|\boldsymbol{v}/\sigma_{\mathsf{r}}\|} \leqslant 1/\sigma_{\mathsf{sk}} + \beta_{\mathsf{c}}\sqrt{8N\mathsf{q}_s}/\sigma_{\mathsf{r}} \leqslant \beta_{\mathsf{u}}$. The above shows that $\mathcal{B}$ wins the AOM-MISIS game, given that $\mathsf{Fork}^{\mathcal{C}}$ returns and BadHash does not occur.

Finally, since each of $h_1, h_3, \ldots, h_{2\mathsf{q}-1}$ and $h_2, h_4, \ldots, h_{2\mathsf{q}}$ are sampled uniformly from $\mathcal{S}_{\mathsf{b}}^\ell$ and $\mathcal{S}_{\mathsf{c}}$ respectively, $\Pr[\mathsf{BadHash}] \leqslant (2\mathsf{q})^2/|\mathcal{S}_{\mathsf{b}}|^\ell + (2\mathsf{q})^2/|\mathcal{S}_{\mathsf{c}}| \leqslant 8\mathsf{q}^2 2^{-2\kappa}$. Therefore,

$$\begin{aligned} \mathsf{Adv}_{\mathsf{par}}^{\text{aom-misis}}(\mathcal{B}, \kappa) &\geqslant \mathrm{acc}(\mathsf{Fork}^{\mathcal{C}}) - \Pr[\mathsf{BadHash}] \\ &\geqslant \mathsf{Adv}^{\text{ideal-tuf}}(\mathcal{A}, \kappa)^2/\mathsf{q} - 8\mathsf{q}^2 2^{-2\kappa} \ , \end{aligned}$$

which concludes the lemma. □

BIBLIOGRAPHY

[1] PCM: Click fraud prevention and attribution sent to advertiser. https://webkit.org/blog/11940/pcm-click-fraud-prevention-and-attribution-sent-to-advertiser/. Accessed: 2021-09-30.

[2] Trust tokens. https://developer.chrome.com/docs/privacy-sandbox/trust-tokens/. Accessed: 2022-01-11.

[3] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Berlin, Heidelberg, May 2001.

[4] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, Berlin, Heidelberg, November 1996.

[5] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Berlin, Heidelberg, August 2000.

[6] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete Gaussian leftover hash lemma over infinite domains. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 97–116. Springer, Berlin, Heidelberg, December 2013.

[7] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53. ACM Press, November 2022.

[8] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.

[9] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from DDH with full adaptive security. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 429–459. Springer, Cham, May 2024.

[10] Matilda Backendal, Mihir Bellare, Jessica Sorrell, and Jiahao Sun. The fiat-shamir zoo: relating the security of different signature variants. In *Nordic Conference on Secure IT Systems*, pages 154–170. Springer, 2018.

[11] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013.

[12] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 587–617. Springer, Cham, December 2021.

[13] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Berlin, Heidelberg, August 2009.

[14] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022.

[15] Mihir Bellare, Wei Dai, and Lucy Li. The local forking lemma and its application to deterministic encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 607–636. Springer, Cham, December 2019.

[16] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[17] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

[18] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[19] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Berlin, Heidelberg, May / June 2006.

[20] Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger security for non-interactive threshold signatures: BLS and FROST. Cryptology ePrint Archive, Report 2022/833, 2022.

[21] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 27–35. Springer, New York, August 1990.

[22] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 551–572. Springer, Berlin, Heidelberg, December 2014.

[23] Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. SPRINT: High-throughput robust distributed Schnorr signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 62–91. Springer, Cham, May 2024.

[24] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Cham, October 2021.

[25] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012.

[26] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short blind signatures. *J. Comput. Secur.*, 21(5):627–661, 2013.

[27] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003.

[28] Dan Boneh, Rosario Gennaro, and Steven Goldfeder. Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In Tanja Lange and Orr Dunkelman, editors, *LATINCRYPT 2017*, volume 11368 of *LNCS*, pages 352–377. Springer, Cham, September 2019.

[29] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, August 2018.

[30] Dan Boneh, Rosario Gennaro, Steven Goldfeder, and Sam Kim. A lattice-based universal thresholdizer for cryptographic systems. Cryptology ePrint Archive, Report 2017/251, 2017.

[31] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

[32] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[33] Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 149–164, Los Alamitos, CA, USA, May 2025. IEEE Computer Society.

[34] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Cham, August 2022.

[35] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Berlin, Heidelberg, August 1994.

[36] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Berlin, Heidelberg, May 2001.

[37] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Berlin, Heidelberg, August 2004.

[38] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

[39] Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 3–31. Springer, Cham, August 2022.

[40] Rutchathon Chairattana-Apirom and Anna Lysyanskaya. Compact cut-and-choose: Boosting the security of blind signature schemes, compactly. Cryptology ePrint Archive, Report 2022/003, 2022.

[41] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 268–302. Springer, Singapore, December 2024.

[42] David Chaum. Verification by anonymous monitors. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 138–139. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.

[43] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[44] David Chaum. Blind signature system. In David Chaum, editor, *CRYPTO'83*, page 153. Plenum Press, New York, USA, 1983.

[45] David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 177–182. Springer, Berlin, Heidelberg, May 1988.

[46] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, New York, August 1990.

[47] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993.

[48] Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 716–747. Springer, Cham, August 2023.

[49] Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood. Two-Round Threshold Schnorr Signatures with FROST. Internet-Draft draft-irtf-cfrg-frost-10, Internet Engineering Task Force, September 2022. Work in Progress.

[50] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove Schnorr assuming Schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021.

[51] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Cham, May 2021.

[52] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

[53] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994.

[54] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

[55] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 409–444. Springer, Cham, August 2024.

[56] Frank Denis, Frederic Jacobs, and Christopher A. Wood. RSA Blind Signatures. Internet-Draft draft-irtf-cfrg-rsa-blind-signatures-02, Internet Engineering Task Force, August 2021. Work in Progress.

[57] Frank Denis, Frederic Jacobs, and Christopher A. Wood. RSA Blind Signatures. RFC 9474, October 2023.

[58] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer, Berlin, Heidelberg, August 1988.

[59] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, New York, August 1990.

[60] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024.

[61] Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In Leonid Reyzin and Douglas

Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 425–458. Springer, Cham, August 2024.

[62] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Cham, August / September 2016.

[63] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Berlin, Heidelberg, August 2015.

[64] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

[65] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Cham, May 2020.

[66] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 244–251. Springer, Berlin, Heidelberg, December 1993.

[67] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, Berlin, Heidelberg, May 2014.

[68] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Berlin, Heidelberg, August 2011.

[69] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1179–1194. ACM Press, October 2018.

[70] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 156–174. Springer, Cham, June 2016.

[71] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 354–371. Springer, Berlin, Heidelberg, May 1996.

[72] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure applications of Pedersen's distributed key generation protocol. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 373–390. Springer, Berlin, Heidelberg, April 2003.

[73] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007.

[74] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. Robust and efficient sharing of RSA functions. *Journal of Cryptology*, 13(2):273–300, March 2000.

[75] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. Erratum: Robust and efficient sharing of RSA functions. *Journal of Cryptology*, 20(3):393, July 2007.

[76] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[77] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Cham, April 2017.

[78] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[79] Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 266–300. Springer, Cham, June 2024.

[80] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Cham, May 2019.

[81] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Cham, August 2020.

[82] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Private Access Tokens. Internet-Draft draft-private-access-tokens-01, Internet Engineering Task Force, October 2021. Work in Progress.

[83] Nicholas Hopper. Proving security of tor's hidden service identity blinding protocol. https://www-users.cse.umn.edu/~hoppernj/basic-proof.pdf, 2013.

[84] Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 414–443. Springer, Cham, November 2020.

[85] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 468–497. Springer, Cham, March 2022.

[86] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 404–434. Springer, Cham, October 2021.

[87] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.

[88] Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 468–492. Springer, Cham, December 2021.

[89] Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and diffie-hellman problems. *J. Math. Cryptol.*, 2(4):311–326, 2008.

[90] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007.

[91] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020.

[92] Chelsea Komlo, Ian Goldberg, and T Wilson-Brown. Two-Round Threshold Signatures with FROST. Internet-Draft draft-irtf-cfrg-frost-01, Internet Engineering Task Force, August 2021. Work in Progress.

[93] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, Berlin, Heidelberg, May 2014.

[94] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014.

[95] Yehuda Lindell. Simple three-round multiparty Schnorr signing with full simulatability. *CiC*, 1(1):25, 2024.

[96] Yehuda Lindell, Ariel Nof, and Samuel Ranellucci. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. Cryptology ePrint Archive, Report 2018/987, 2018.

[97] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.

[98] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012.

[99] Vadim Lyubashevsky, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehle, and Shi Bai. CRYSTALS – DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms.

[100] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[101] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005.

[102] National Institute of Standards and Technology. Multi-Party Threshold Cryptography, 2018–Present. https://csrc.nist.gov/Projects/threshold-cryptography.

[103] National Institute of Standards and Technology. Post-Quantum Cryptography: Additional Digital Signature Schemes, 2022–Present. https://csrc.nist.gov/projects/pqc-dig-sig.

[104] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994.

[105] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Cham.

[106] Miyako Ohkubo and Masayuki Abe. Security of some three-move blind signature schemes reconsidered. In *The 2003 Symposium on Cryptography and Information Security*, 2003.

[107] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Berlin, Heidelberg, August 1993.

[108] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Berlin, Heidelberg, August 1992.

[109] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Berlin, Heidelberg, August 1992.

[110] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[111] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, National Institute of Standards and Technology, 2022. https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms.

[112] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[113] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. ROAST: Robust asynchronous Schnorr threshold signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2551–2564. ACM Press, November 2022.

[114] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990.

[115] Claus-Peter Schnorr. Efficient identification and signatures for smart cards (abstract) (rump session). In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 688–689. Springer, Berlin, Heidelberg, April 1990.

[116] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[117] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, November 2001.

[118] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

[119] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, Berlin, Heidelberg, May 2000.

[120] Douglas R. Stinson and Reto Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *ACISP 01*, volume 2119 of *LNCS*, pages 417–434. Springer, Berlin, Heidelberg, July 2001.

[121] Katsuyuki Takashima and Atsushi Takayasu. Tighter security for efficient lattice cryptography via the Rényi divergence of optimized orders. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 412–431. Springer, Cham, November 2015.

[122] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Cham, April 2023.

[123] Benedikt Wagner, Lucjan Hanzlik, and Julian Loss. PI-cut-choo! Parallel instance cut and choose for practical blind signatures. Cryptology ePrint Archive, Report 2022/007, Version 20220107:165256, 2022.

[124] Hoeteck Wee. Threshold and revocation cryptosystems via extractable hash proofs. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 589–609. Springer, Berlin, Heidelberg, May 2011.