Enhancing Privacy in AI: Differential Privacy in Multiparty Computation

Sikha Pentyala

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Martine De Cock, Chair

Anderson Nascimento

Ankur Teredesai

Ivan Dämgard

Program Authorized to Offer Degree:

Computer Science and Systems,
School of Engineering & Technology

University of Washington

Abstract

Enhancing Privacy in AI:
Differential Privacy in Multiparty Computation

Sikha Pentyala

Chair of the Supervisory Committee:
Martine De Cock
Computer Science and Systems

Artificial Intelligence (AI) based applications provide a lot of convenience but often rely on sensitive data of a personal nature to work well. AI pipelines raise input privacy concerns when AI models need to be trained across combined data from multiple data holders who may not be willing or even legally allowed to disclose their data to each other. Similarly, output privacy concerns arise when trained AI models are deployed in production and inadvertently leak private information about the individuals in the training data.

A popular approach to address input privacy is Federated Learning (FL), a paradigm in which models are trained in a distributed manner so that raw personal data never leaves the source. State-of-the-art techniques to mitigate output privacy risks use Differential Privacy (DP), which obfuscates the presence of individual records in the training data by adding noise. Existing solutions combining traditional FL and DP to provide input and output privacy at the same time typically cater to specific data partitions (horizontal or vertical) and sacrifice a lot of accuracy to achieve privacy.

In this dissertation, we focus on providing both input and output privacy guarantees when data is distributed across multiple data holders irrespective of the data partitioning. We provide solutions to preserve privacy when (a) training discriminative machine learning models for prediction; (b) mitigating biases in model predictions; (c) training AI models

for synthetic data generation. All our solutions are grounded in novel Secure Multi-Party Computation (MPC) protocols to provide input privacy for any data partition – offering a single solution for horizontal, vertical, or mixed partitions. To simultaneously provide output privacy while maintaining high utility, we leverage the idea of a "DP-in-MPC" paradigm through the development of MPC protocols that emulate centralized DP even when the data resides with multiple data holders in a distributed manner. Our research is characterized by the quest for such solutions that (1) enable training of high utility AI models, (2) in a manner sufficiently efficient for use in practice, (3) without compromising individual privacy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Pursuing a PhD as an international student, while adjusting to a new education system, learning the ropes of research, and raising a family, has been both challenging and deeply rewarding. It's been one of the most enriching and defining chapters of my life. This journey would not have been possible without the support, guidance, and inspiration of many people, to whom I am deeply grateful.

I feel incredibly fortunate to have had Dr. Martine De Cock as my primary advisor. Her mentorship has been a cornerstone of my academic and personal growth over these past years. She is the kind of advisor one can only hope for; someone who helps you grow without you knowing it, who simplifies the complex, and who guides with a rare mix of clarity, patience, and personal care. Her presence throughout my journey, always supportive, thoughtful, and kind, made me feel like she was part of my family. Our frequent bus rides, where we talked about research, work, kids, and life in general, only made that bond more personal. Her unwavering support in helping me adapt to a new country and academic culture made all the difference, she has been both a mentor and a grounding presence. I am deeply thankful for her thoughtful advice, steady encouragement, and the time and energy she invested in helping me become more confident and capable in my work, especially in academic writing and presentation. It is difficult to overstate how much her presence has shaped not just my work, but how I approached this journey as a whole. She has been, and continues to be, a true inspiration.

I am also grateful to Dr. Anderson Nascimento, whose support and guidance have been invaluable throughout this journey. Thanks to him, I had the opportunity to

teach, something I had always been deeply interested in, and he encouraged me every step of the way. He shared small but powerful insights on navigating life as an international student, with words that still stay with me, like: "You can best convince by giving proper reason." His stories, both personal and professional, were always a joy to listen to which reflects his experience as an amazing educator, mentor and person. I'll always remember how he looked out for me like family – checking in on me during the early days of my teaching and even staying late into the evening just to make sure I got home safely. His kindness, perspective, and quiet encouragement have quietly made a difference in my experience.

After my master's, Dr. De Cock introduced me to Dr. Golnoosh Farnadi during the COVID-19 pandemic. I had the chance to work on two projects under both their guidance, and had the privilege of joining Dr. Farnadi's lab at MILA. Although the work was remote, it proved to be a deeply enriching and transformative experience. I'm grateful to Dr. De Cock and Dr. Farnadi for making that opportunity possible. My time at MILA broadened my exposure to new research directions, lab cultures and exposed me to different perspectives of research – all of which will stay with me as I continue to grow as an independent researcher. Dr. De Cock introduced to me to the world of privacy in AI, which is the topic of my PhD, my time at Dr. Farnadi's lab exposed me to other fields of responsible AI such as fairness and explainability, which led to my interest at the intersection of privacy and other pillars of responsible AI. The research I did there became a part in itself of my dissertation. I also had the pleasure of collaborating with Nicola, mentoring Jason, and learning from the fresh perspectives of Rebecca and Khaoula and everyone else in the lab. Though most of our work was remote, meeting in person at NeurIPS 2022 was wonderful that made these collaborations even more meaningful.

The people you're surrounded by make a huge difference in shaping the PhD

experience, and I feel lucky to have been part of such a welcoming environment at UWT. The PPML research lab has always been very supportive and inclusive. I've had countless memorable moments and meaningful discussions with Steven and Daniil, from spontaneous brainstorming to thoughtful conversations. I've learned a lot from both of them, not just academically, but also personally. Their constant support, patience, and willingness to listen to my many questions and out-loud thoughts made me feel genuinely welcomed. I also enjoyed working closely with the undergraduate students, Shane and Trae, and learned a lot from their impressive hands-on skills. Geetha, who joined our group more recently and collaborated with me on the last-minute paper, brought a positive presence to the team. Beyond UWT, I had the opportunity to work with students from different universities. Collaborating with Ricardo and Jelle was a different experience all together – our late-night discussions, frantic last-minute submissions, and shared excitement made it a fun and rewarding experience.

When I got the opportunity to teach, I did not know where and how to start. I express my gratitude to Dr. Raghavi Sakpal and Professor Menaka Abraham for generously sharing their course materials and offering valuable tips and practical advice that helped me improve my teaching and classroom presence.

My internship experiences at J.P.Morgan in 2023 and 2024 offered a different but valuable perspective. I learned how research is approached in an industry setting and how problem statements are framed and refined in real-world contexts. I'm thankful to Shubham and Sanjay for their mentorship during those internships, their guidance helped me grow as an independent researcher and collaborator.

I sincerely thank my other committee members for their valuable inputs – Dr. Ankur Teredeasi for his inputs on writing and presentation during my general exam itself, Dr. Ivan Damgard for bringing his expertise in Secure Multiparty Computation

and asking insightful questions and offering simple yet effective suggestions, Dr. Dan Grossman for insightful discussions and relevant questions that made me think outside the box for my next research directions. I enjoyed and learnt a great deal from the discussions during my general and final exam.

I extend my thanks to the staff and faculty and broader community at UWT including Dean Raj Katti who have directly or indirectly supported me in my PhD journey. Rachel and Kira, who I have bugged a lot, patiently supported me and helped me navigate the administrative and logistics part of my PhD. I would like to thank Dr. Juhua Hu for giving me the opportunity to contribute to Women in Data Science (WiDS) events that helped me to engage more with the broader community. I also want to thank Dr. Afra Mashhadi from UW Bothell for her encouragement and for collaborating with me which boosted my confidence as an independent researcher and broadened my workspace. I also want to thank Dr. Rafael Dowsley for his inputs on our papers together.

None of this would have been possible without the financial support I received throughout the course of my PhD. I gratefully acknowledge J.P.Morgan Fellowship and Carwein-Andrews Fellowship for making this possible so that I could do meaningful research.

Last but not the least, I extend a big thanks to my very supportive family. My daughter, Anvi, who was supportive and surprisingly encouraged me in her own way. She listened to my presentations, offered her own sweet suggestions, and cheered me on in her own little ways. I promise to make up for the missed time with more stories, more games, and more laughter together. My husband, Sundeep, has been steady backbone through it all. Despite his own demanding work and projects, he managed everything at home and cared for our daughter with unwavering patience and love, especially during my busiest and most stressful deadlines. I couldn't have done

this without his quiet strength and constant support. My parents, Sivaparvathi and Sambaiah, who have always encouraged me and motivated me to keep going forward. My in-laws, Padmavathi and Madhusudhan, who have been silently supportive of my PhD journey. My cousin, Bhavana, who has helped me navigate life in US and supported me throughout.

My PhD has been transformative. I will always be grateful for every person who made this journey meaningful. Thank you, from the bottom of my heart.

Chapter 1

INTRODUCTION

"The important thing is not to stop questioning. Curiosity has its own reason for existing."

Albert Einstein

Every morning, Alice wakes up and checks her phone. As she does, her device silently logs her location. Her fitness tracker records her sleep patterns and heart rate. When she browses for a product online, an invisible network of advertisers competes to serve her personalized ads. Later, as she scrolls through social media, algorithms analyze her behavior to predict what content will keep her engaged the longest.

Each of these seemingly mundane activities generates data and leaves a digital trace – an imprint of habits, preferences, emotions and our intimate concerns. From ordering coffee on a mobile app to consulting a doctor via an online patient portal, nearly every facet of our modern lives takes place in the digital realm, generating an expanding web of our personal data.

Beyond our voluntary online activities, stakeholders in critical domains such as healthcare, finance, education and governance collect, store and process large amounts of our sensitive information, both digital and non-digital. Medical documents record our physical and emotional vulnerabilities, financial transactions reveal our financial position and economic stability, educational data tracks our academic performance and disciplinary history, government systems record our employment history, tax and household information, immigration status and compliance.

Such disclosure of personal information and data collection, at its core, isn't inherently problematic. Indeed, it enables access to essential services such as welfare benefits including mortgages and credit cards, medical care, legal services, and supports efficient resource allocation and societal infrastructure [6]. Data fuels innovation, driving progress across industries. Tech giants and private organizations harness information to optimize efficiency, maximize profits, and deliver better services. In return, individuals gain personalized experiences and conveniences, while retaining little control over how their data is used or who can access it.

The rise of artificial intelligence (AI) has revolutionized how this data is used, making significant advancements across various domains. AI thrives on data, extracting insights that were previously unattainable. AI analyzes electronic health records, genetic data, and real-time patient monitoring to enhance medical diagnostics and personalize treatments [7, 8, 9, 10, 11]. Financial institutions feed customers' transaction histories and spending patterns to AI systems to detect fraud and assess credit risk [12, 13, 14]. Governments leverage demographic and behavioral data for welfare distribution and predictive policing [15]. AI powers recommendation systems for personalized services based on user interactions [16, 17, 18]. These AI-based applications require large amounts of sensitive data, amplifying both the benefits and risks in our increasingly data-driven world.

The incentives for data collection are clear – data is immensely valuable. It is often said that "Data is the new Oil" [19, 20]. But at what cost?

As AI systems evolve, they rely on more granular and personal data. Our personal data, once captured, might persist indefinitely in the digital realm. It can be analyzed, shared, misused and monetized in ways that remain largely invisible to us. Privacy breaches and data misuse not only affect individuals but also impact society as a whole.

At the individual level, privacy breaches can lead to identity theft, financial fraud, discrimination, and psychological harm [21]. AI software can be built using the data to generate deepfakes that can manipulate people in harmful ways [22, 23]. Personal health data can be sold to third parties and then be used for targeted advertising or even denied coverage by

health insurers [24, 25]. Data collected from browsing history, purchase behavior, and demographic details can lead to price discrimination [26]. The very data that powers AI-driven systems also fuels mass surveillance, targeted manipulation, and algorithmic discrimination. AI driven facial recognition can be used to track individuals with or without their consent, violating personal privacy [27]. Personal data collected by social media platforms can generate targeted ads and manipulate behavior [28, 29, 30, 31].

The data collected to build AI systems often includes sensitive attributes such as race, gender, or socioeconomic status. When such data is collected and used without regulation or oversight, AI systems can exhibit biased outcomes that reflect and reinforce historical prejudices and societal inequalities. For example, AI systems may unfairly reject loan applications based on biased historical data [32], favor candidates in hiring decisions based on demographic characteristics such as gender or ethnicity, even when these are irrelevant to job performance [33, 34], or enable law enforcement algorithms that result in inequitable policing and racial profiling [35, 36]. These biases are not just technical glitches , they reveal how AI systems can unintentionally discriminate against individuals, especially when built on personal data. Sensitive attributes that should be protected become vectors for discrimination and unfairness. Even when AI systems are build without directly using sensitive attributes, disparities can still emerge due to underlying correlations in the data. While numerous mitigation strategies have been proposed to build fair AI systems and detect biases, these approaches typically require access to the very sensitive attributes that created the problem in the first place [37, 38]. This raises additional privacy concerns, ensuring fairness in AI systems often comes into tension with protecting individual privacy, especially when sensitive data is needed to audit or correct AI systems.

The consequences of privacy leakages and misuse at societal level are equally concerning. The 2018 Facebook-Cambridge Analytica scandal best illustrates this, where personal data was collected without consent and used to manipulate political opinions [39]. Such incidents reveal how privacy breaches can "*spur filter bubbles, amplify disinformation, and increase*

political polarization, with implications for democratic elections and even societal safety"[1].

The rise of generative AI (GenAI) has further heightened these risks[2]. ChatGPT[3] leaked user payment data and accidentally leaked company secrets belonging to Samsung [40, 41].

These concerns raise serious questions about data control, usage, and the meaning of privacy in this era. To fully reap the benefits of AI, we must address these concerns and foster trust. Users need confidence that AI systems will deliver value while safeguarding their privacy. We must design AI systems that are not only powerful but also responsible and trustworthy.

Regulations like the General Data Protection Regulation (GDPR) [42] and the California Consumer Privacy Act (CCPA) [43] attempt to address these concerns by enforcing data protection policies. However, legal frameworks alone are not enough. They are often reactive, hard to enforce, largely non-technical, and slow to keep up with the pace of AI development [44]. A more fundamental approach is required – one that embeds privacy into the very fabric of data collection and AI pipelines. This is especially critical in distributed environments, where data from multiple sources (such as organizations or silos) must be analyzed collectively without compromising individual privacy [45].

Synthetic data, artificially generated data that retains the statistical properties of real data without exposing any personally identifiable information, is increasingly being adopted in practice as a privacy-preserving solution to the problem of data access and safeguarding privacy of individuals [46]. When generated from data originating across multiple sources in a distributed environment, synthetic data can offer a rich and diverse alternative to any single dataset. Synthetic data is often considered privacy-preserving and can be shared or reused across systems for multiple purposes, thereby, promoting open and reproducible research, especially in critical domains such as healthcare. However, this apparent solution

[1] `https://www.cylab.cmu.edu/news/2022/02/22-roots-of-privacy.html`

[2] According to Gartner Inc, more than 40% of AI-related data breaches will be caused by the improper use of GenAI across organizations and nations by 2027.

[3] `chatgpt.openai.com`

introduces a recursive problem: synthetic data generators are themselves AI systems or statistical algorithms that typically require collecting real data from multiple sources into a single location. This centralization raises the very privacy concerns of personal data collection that synthetic data initially sought to alleviate.

As AI becomes deeply embedded in decision-making across healthcare, finance, and governance, ensuring privacy is no longer just a legal requirement, it is an ethical imperative. It has also given rise to a perceived tension between the desire to protect data privacy on one hand, and to promote an economy based on free-flowing data on the other hand [47].

This dissertation tackles the fundamental challenge in today's data-driven AI world of easing this tension: how can AI systems use sensitive and private data while safeguarding privacy of individuals – especially when data is fragmented across organizations such as hospitals, clinical research centers, or banks? Our research addresses a critical tension in society by bridging the gap between data protection regulations and the growing need for data-driven AI [47]. Through our research, we've shown that privacy protection doesn't have to hinder AI progress – it's possible to develop and apply AI to sensitive data responsibly, enabling innovation while safeguarding privacy.

1.1 Concept of Privacy

We argued above that one's privacy must be preserved when using AI-powered systems. But what is privacy? While privacy is not a new concept, its meaning has evolved over time, often intersecting with various fields such as philosophy, social sciences, and psychology. More recently, it has expanded to include social and digital dimensions, giving rise to what is often referred as "digital privacy" [48].

Since the earliest days of human evolution, humans instinctively sought out private and sheltered spaces, not only for physical protection but also to establish personal boundaries and sense of control over their environment [49, 50]. As societies evolved, the notion of privacy extended beyond physical spaces to include protection of communications, personal property, and reputation. Laws and regulations were established to incorporate this notion

of privacy, for example, the Fourth Amendment to the U.S. Constitution protects against unreasonable searches and seizures, and this protection extends to sealed mail.

With rise of technology, particularly affordable cameras, and journalism, private moments became increasingly publicized. In 1890, the seminal article "The Right to Privacy" in the Harvard Law Review demanded privacy to be called as the *right to be let alone*, which can be interpreted as shielding private lives from public exposure [51]. Later, the "Universal Declaration of Human Rights", adopted by the United Nations in 1948, addressed privacy at the international level.

The advent of the Internet and digital technology, driven by widespread data collection, further fueled privacy concerns. Alan Westin in one of his most influential books defined privacy as "*the claim of individuals ... to determine for themselves when, how and to what extent information about them is communicated*", which is very relevant in this digital age [52]. As technology and concerns surrounding security and privacy grew, so did the laws and regulations about them all around the world (such as FTCA, FERPA, Privacy Act of 1974, TCPA, HIPAA and COPPA in the U.S)[4]. One of the most impactful is the EU-General Data Protection Regulation (GDPR), which came into effect in 2018 and was based on the EU Data Protection Directive of 1995. It set the global standard for data protection and led to further laws and regulations, such as the CCPA and various U.S. state laws, Brazil's LGPD, Japan's APPI, India's DPDP Act, Canada's PIPEDA, and Australia's Privacy Act.

With the rise of big data, AI, advanced analytics, and distributed computing over the past two decades, privacy has become an increasingly complex and multifaceted concept. It is contextual in nature, varying in meaning across individuals, societies, and cultures [53]. Due to large-scale data collection and deployment of AI, issues such as data breaches, mass surveillance, and the misuse of personal information have increased global awareness of privacy and other risks leading to broader laws such as the EU AI Act, 2024 and various State AI laws in the U.S. In this era of digital and scoial media, the boundaries between

[4] `https://safecomputing.umich.edu/protect-privacy/history-of-privacy-timeline`

public and private life have become increasingly blurred. So, what does privacy mean today? It is no longer a fixed or absolute notion. It depends on context, intent, and usage.

Given how complex it is to define privacy today, this dissertation will focus on privacy from a computer science (CS) perspective which will follow pre-defined notions and goals of privacy in the CS literature detailed in the next section.

1.2 Privacy in AI

Artificial Intelligence (AI) refers to the development of computers and machines that can mimick human-level intelligence and perform tasks typically carried out by humans. Machine Learning (ML) is a major subfield of AI that enables computers to learn from data and improve their ability to perform specific tasks without being explicitly programmed[5]. After learning from data, ML systems are used to make predictions on new, unseen data.

ML based systems typically follow the pipeline illustrated in Figure 1.1 to learn from data. This pipeline involves multiple stakeholders, such as data holders, model developers and service providers, end users and service consumers, each with their own privacy requirements. Each stage of this pipeline has its own privacy vulnerabilities and risks that need to be addressed.

1. **Data Collection and Aggregation.** AI is data hungry: the more data available, the better AI models can learn and make accurate decisions. In some cases the data owned or held by a single entity is sufficient to develop an ML model. In such scenarios, the data holder itself often serves as the model owner, service provider, and consumer. An example of this is NYUTron which was developed by New York Univeristy (NYU) based on a decade's worth of clinical notes from NYU Langone Health's inpatient records and is freely available for use by all NYU Langone Health (NYULH) faculty and staff [54].

   More often than not, the data held a by a single entity is not sufficient to build better-performing ML models. For instance, a clinic aiming to analyze rare diseases such as

[5]In this dissertation, we will use AI and ML loosely to refer to the concept of "learning from data".

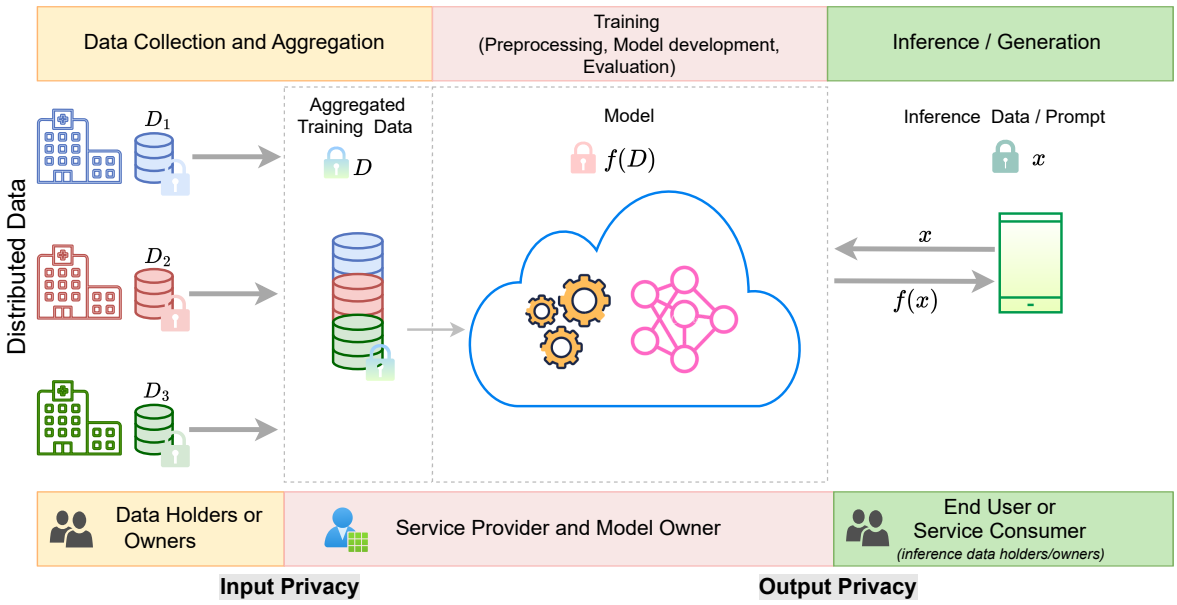


Figure 1.1: **Input and Output Privacy.** Data collection, training, and inference in ML pipelines involve multiple stakeholders, each with their own privacy requirements: data holders at inference and training stages expect **input privacy**, model owners expect **output privacy**. In the centralized paradigm, as shown here, all data holders *must trust* the single service provider and model owner.

Neurofibromatosis Type 1 (NF1), which affects roughly 1 in 3,000 individuals, using AI may lack sufficient patient data. Data then needs to be collected from multiple sources such as multiple clinics and silos, a data distribution scenario commonly referred to as cross-silo settings (i.e. across multiple organizations). This need is illustrated by efforts such as the National Institutes of Health (NIH)'s AIM-AHEAD Federated Networks program, which "seeks to empower biomedical research by establishing a decentralized, data network that respects data privacy and sovereignty" [45]. Another example where AI benefits from distributed data is in applications like smartphone keyboard apps, which learn to improve word predictions based on individual typing behavior; this is known

as a cross-device setup (i.e. across many end users). In both the setups, cross-silo and cross-device, the step of data collection and aggregation is usually carried out by a third party (or one of the participating organizations), which typically builds the (proprietary) model and provides services, therefore acting as the model owner and service provider.

At this stage of data collection and aggregation, especially when a third party is involved, there are risks of unauthorized data sharing, lack of informed consent from individuals, potential data breaches during aggregation, and the re-identification of individuals even in anonymized datasets [55, 56, 57]. Data holders aim to protect against these vulnerabilities , for example, clinics are responsible for safeguarding patient data and are regulated by laws such as GDPR, HIPAA and user agreements; similarly, users may be reluctant to share personal data from their phones, such as typing patterns.

A typical privacy expectation at this stage is that personal data, especially sensitive information such as health records or financial details, should remain protected[6]. To provide such protection, data in a distributed setting (also referred to as a federated scenario where data is fragmented across multiple organizations or individuals), remains siloed as D_1, D_2, and D_3, as shown in Figure 1.1, due to privacy concerns, regulatory constraints, or competitive interests. In such distributed scenario, the expectation is to ensure *input privacy* in AI pipelines, which guarantees that none of the data holders need to disclose their raw data to any other entity, i.e. neither the other data holders, the service providers, nor any external parties learn anything about private inputs in the pipeline.

Note on distributed settings. In a distributed scenario, data can be distributed in various ways as shown in Figure 1.2. where each data holder has records (rows) of data with the same feature set. For example, in a cross-silo setup, where data is distributed

[6]This broadly includes expectations that the data is not freely shared, and not used in ways that could harm or disadvantage individuals.

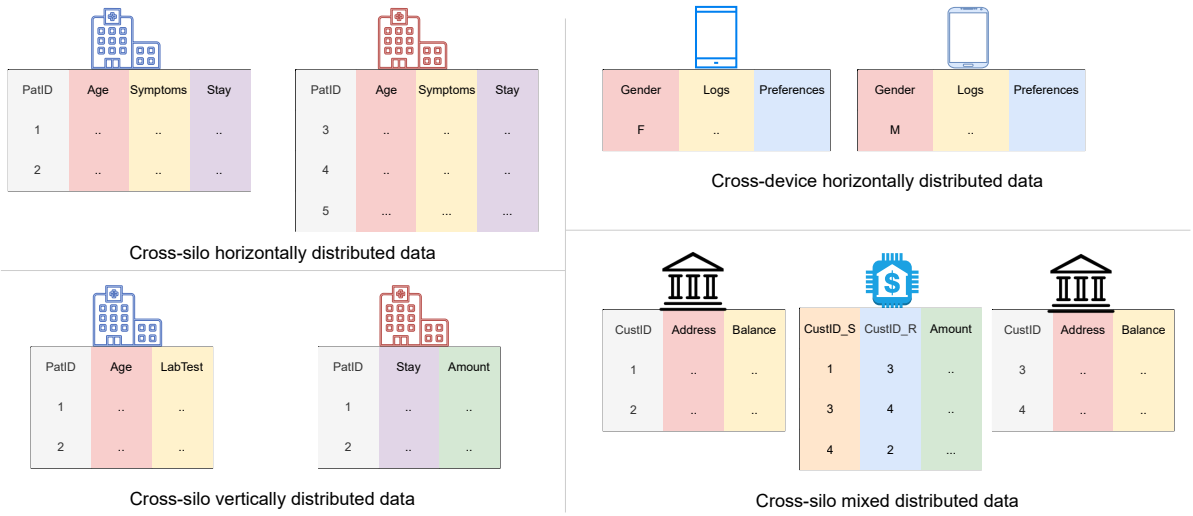


Figure 1.2: **Data Partitioning Scenarios**. Data may be spread across multiple silos or entities (cross-silo) or individuals (cross-device), leading to different data partitioning scenarios – horizontal, vertical, and mixed. We are interested in a solution that works for any of the partitions i.e. arbitrary partition.

across organizations, hospitals treating COVID-19 patients could collaborate to build a model that predicts the length of hospital stay by combining their individual patient records. In contrast, in a cross-device setup, the data resides on individual user devices (e.g. smartphones or wearables), with each holding similar types of data (such as gender, activity logs, preferences) across a large number of users. Another scenario are *vertically distributed settings*, where different data holders hold different features (columns) about the same set of individuals An example of this is an ML model that relies on lab test results as well as healthcare bill payment information about patients, which are usually managed by different departments within a hospital system. Data can also be distributed in a mixed manner that fits neither the horizontal nor the vertical partitioning scheme. In the 2023 U.S. U.K. PETs Prize competition, for instance, participants were challenged to develop a solution for financial fraud detection with data from banks and from the Society

for Worldwide Interbank Financial Telecommunication (SWIFT) [58]. In this scenario, SWIFT has data about transactions between customers across banks, while each bank has more detailed data about its own customers. Note that while cross-device settings are almost always horizontal, cross-silo setups can also involve vertical or even mixed data distributions, depending on how the data is partitioned.

2. **Training.** After the data collection and aggregation step, all the data is available in a single location as D (an aggregation of D_i's where $i = \{1, 2, 3, \ldots\}$). This is referred to as the centralized paradigm. This paradigm assumes that all the data holders *trust* the same service provider and model owner and share their private data.

   The data D is further used for model development. The data is firstly pre-processed which includes cleaning the data, handling missing values, transforming the data into appropriate inputs such as normalizing and encoding data, and feature engineering to create or select relevant features from the input data. At this point, the data is typically split into (a) training data, which is used for "learning", i.e. training an ML model, and (b) testing data, which is used to evaluate the utility of a trained model by assessing how well it makes predictions on new, unseen data, i.e. the data that it has not learned from.

   Model developers then choose a set of model architecture(s) (also called models), which are trained on the training data using the chosen training algorithm(s)[7]. During training, the model learns the underlying patterns and relationships in the data[8]. The model with best utility on testing data is then selected and deployed for service.

   Specifically, the ML models can be trained to provide accurate predictions (inference), enable unbiased predictions with higher accuracy (fair inference), or generate synthetic data (generation). These trained models can be categorized as discriminative models that

[7]Pre-trained models can also be used which are usually trained on publicly available or non-private data. These are further fine tuned on private training data.

[8]Training can lead to memorization of sensitive data or patterns.

make predictions (inference) and generative models that generate new data (generation). The training is then followed by assessing the model's utility on testing data using metrics defined for the specific task. Such as accuracy for classification tasks that assign labels (example: classifying images as cats or dogs), root mean squared error (RMSE) for regression tasks that predict numeric values (example: predicting prices of houses) and an appropriate metric like perplexity for generation tasks that generate new text (example: ChatGPT generating text, DaLLE generating images given input prompts).

During the training phase, the primary concern of service providers[9] is protecting the confidentiality and integrity of the training data (which often contains sensitive data) and ensuring that it is not exposed to unintended parties. Furthermore, service providers must ensure compliance with regulatory requirements (such as GDPR or HIPAA) regarding data handling, storage, and processing. The risks, including privacy breaches, are heightened in the centralized setup. These concerns have motivated the development of collaborative and decentralized learning approaches that minimize centralized data aggregation, such as federated learning (FL) [59, 60], where model training occurs across distributed data without centralizing raw data.

Additionally, model owners are concerned with protecting the intellectual property of the proprietary model i.e. the models need to be protected against model stealing and reconstruction attacks [61, 62]. This includes protection against unauthorized access to the model or tampering that could degrade the utility of the model (often referred to as adversarial attacks) [63]. In this dissertation, our focus is on protecting the training data.

Once a model is trained that reaches the desired level of utility as assessed through the evaluation , it is deployed as a system by the service providers for prediction or generation.

Note: One of the biggest assumptions in the training stage is the existence of a trusted en-

[9]The boundary between service providers and model owners is loosely defined and often depends on the use-case. In this dissertation, we consider both the roles to overlap.

tity, trusted by all data holders. This is a typical expectation in the centralized paradigm, where all the data is pooled together for model development. However, in practice, such a universally trusted entity does not exist [64]. Therefore, while model development proceeds under the assumption of centralized trust, the lack of a universally trusted entity increases the risk of privacy breaches and undermines trust among stakeholders in the ML pipeline.

3. **Inference.** During the inference phase, the service providers host the trained models through various interfaces such as APIs, mobile applications, or web services. End users or service consumers query the model with inputs to be predicted or prompts to be used for generation, denoted as x.

   In addition to service providers needing to protect their model and training data, end users also expect their inference data, x, to be protected. While we acknowledge the importance of protecting inference data, our primary focus in this dissertation is on preserving the privacy of individuals in the training data. An adversarial end user during this stage can perform various attacks: model stealing attempts where adversaries systematically query the model to create a functional replica [61, 62]; membership inference attacks (MIA) that aim to determine whether a specific individual's data was used in training [65, 66, 67, 68]; and reconstruction attacks that attempt to recover training examples from model outputs [69, 70]. These vulnerabilities violate data privacy regulations, and potentially expose intellectual property and compromise competitive advantages.

   To mitigate these risks during inference, service providers expect *output privacy* when communicating with end users, ensuring that the outputs of a model do not reveal information about an individual or sensitive inputs of training data, as per the given definitions of information leakage. Output privacy is often achieved through techniques like Differential Privacy during the training phase [71]. Additionally, query rate limiting can be used to prevent systematic querying and monitoring systems can detect suspicious activity patterns.

1.3 Dissertation Goals

The overall goal of this dissertation is to train effective ML models while preserving the privacy of individuals in the training dataset in a distributed paradigm. These models are typically trained on large, diverse datasets which are often distributed across multiple data holders (see Data Collection and Aggregation in Figure 1.1) i.e. a distributed/federated paradigm.

The objective of the dissertation for training ML models in federated setting are as follows.

- Providing input and output privacy guarantees while
- Achieving ML utility as in the centralized paradigm with
- No single point of trust
- Offering a single solution for any arbitrary partition, and
- Scaling for any number of data holders

The dissertation focuses on adopting privacy-enhancing techniques (PETs) to enable model training on aggregated data without exposing private information or requiring raw data sharing among entities. To achieve the objectives mentioned above, we employ two key PETs: Secure Multiparty Computation (MPC) [72] for input privacy and Differential Privacy (DP) [71] for output privacy. These methods enable effective ML model training on combined data from multiple data holders, ensuring that no party gains unauthorized access to private information.

Secure Multiparty Computation (MPC) is a cryptographic technique that allows multiple parties to jointly compute a function over their private inputs without revealing those inputs to each other or to any external party. In other words, all private inputs are encrypted (referred to as secret shared, see Chapter 2) and the computations are performed on encrypted data. MPC ensures that the only information revealed during the computation is what can be inferred from the final output, and whatever inputs an adversary already possesses. This guarantees that the individual datasets D_i remain secure throughout the process while still

allowing collaborative training of ML models on the combined dataset D.

Differential Privacy (DP) provides a mathematical framework for protecting individual records in a dataset when publishing the results of a computation on private data. DP works by adding carefully calibrated noise to the computation output, ensuring that the presence or absence of any individual's data has minimal impact on the final results. This provides plausible deniability, protecting against inference attacks and limiting what adversaries can learn about specific individuals from the trained model f.

1.4 Contributions and Outline of the Dissertation

1.4.1 Research Gaps

Current state-of-the-art ML models and generators assume the existence of a "trusted" central entity (such as the service providers and model owners) that inherently provides input privacy and trains discriminative and generative models on large and diverse private data that is distributed across multiple data holders. Output privacy, in this case, is usually provided in a centralized trust (global) model utilizing state-of-the-art Differential Privacy (DP) [73, 74, 75, 76, 77].

Various distributed learning paradigms have been proposed to provide input privacy that abide by the privacy regulations and preserve the privacy of individuals without the central entity mentioned above. Traditional federated learning [59] for the multiple data holder scenario works only for horizontally distributed data, referred to in this dissertation as Horizontal Federated Learning (HFL). HFL assumes a single honest-but-curious aggregator that collects model updates rather than raw data, which still creates a single point of failure [78]. HFL further assumes that all the data holders share the same feature subspace which is not always the case. Vertical federated learning (VFL) [79, 80] is designed specifically for vertically distributed data. Unlike this dissertation research, existing FL based paradigms do not work for any kind of arbitrary data partitioning. Moreover, achieving utility in FL (HFL or VFL) comparable to the centralized paradigm (where raw data is directly aggregated) is

particularly challenging in non-i.i.d. settings, and remains an open research question.

An ideal solution for distributed setup should support any data distribution, enabling privacy-preserving learning over pooled datasets. This allows collaboration for privacy-preserving model training across organizations and within different departments of the same organization.

Recent literature has focused on learning paradigms such as decentralized learning (DL), which do not require a trusted central entity and avoid a single point of failure, in order to provide input privacy. Peer-to-peer decentralized learning [81] removes the need of trusted entity and requires the data holders to communicate with each other. Many frameworks have been proposed to build robust and private DL approaches leveraging blockchains [82], Homomorphic Encryption (HE) [83], or Trusted Execution Environments (TEE) [84]. However, these proposed DL frameworks put a heavy communication burden on data holders to stay online for the entire training process which can affect the number of data holders that participate in the collaborative learning process.

In the above mentioned paradigms, output privacy is typically achieved using different models of DP. Local DP [85] is an alternative approach that is deployed when the data holders do not trust the "aggregator" or the other data holders, but suffers from severe utility loss. Distributed DP (e.g. [86]) has been proposed to offer higher utility but with similar guarantees as local DP. These DP based mechanisms are usually applied in HFL or VFL when the parties do not trust the aggregator or the active party. To the best of our knowledge, such approaches typically result in substantially lower utility compared to the global DP model.

Based on the literature review, there is no existing body of work that simultaneously meets all the objectives outlined in Section 1.3).

1.4.2 Research Question.

In this dissertation, we ask the key question: *"How can we train effective machine learning models across multiple private datasets, without a trusted party, while ensuring both input*

and output privacy, and scaling to arbitrary data partitions?"

Specific Research Questions. Based on the overall aim of providing input and output privacy guarantees in an arbitrary distributed data and non-trusted setting, this dissertation addresses three specific research questions below.

RQ1. How can we train accurate ML models with input and output privacy guarantees over data that is distributed in an arbitrary fashion?

RQ2. How can we achieve fairness with pre-processing or post-processing bias mitigation techniques when training privacy-preserving ML models in a distributed setting?

RQ3. How can we generate synthetic data with input and output privacy guarantees when the real data is held by multiple data holders in an arbitrary way?

1.4.3 Key Idea

We propose using Secure Multiparty Computation (MPC) to provide input privacy for any arbitrary distributed data by replacing the trusted central entity from Figure 1.1 with MPC computing servers. Unlike the traditional central paradigm, the computing servers execute MPC protocols that never see the private data of the data holders in an unencrypted manner[10].

To provide output privacy guarantees, we propose adapting Differential Privacy (DP) defined for the centralized paradigm, i.e. the global model of DP. The MPC computing servers use MPC to generate the necessary noise and add it to the result of the computations to provide DP guarantees. To do so, we designed MPC protocols to generate noise, effectively having the MPC protocol play the role of a trusted entity implementing global DP. We refer

[10]MPC protocols are cryptographic algorithms that perform the same computations as their centralized counterparts, but they operate on encrypted data, ensuring that no party involved in the computation has access to the private data in an unencrypted form (a.k.a. in-the-clear).

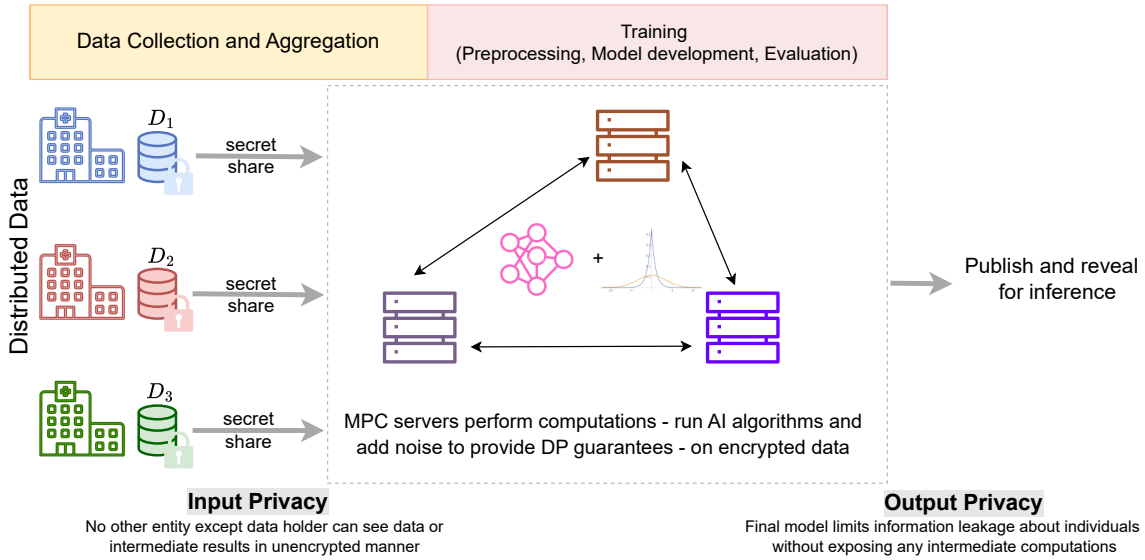


Figure 1.3: **DP-in-MPC paradigm.** The data holders send their encrypted (secret shared) private data to a set of MPC servers. These servers run MPC protocols for AI algorithms and noise generation (for DP guarantees) over encrypted (secret shared) data received from the data holders. The servers or any other entity never see the unencrypted data thus providing input privacy. The trained model is protected with DP thus providing output privacy.

to this approach as the "DP-in-MPC" paradigm as shown in Figure 1.3 and believe that it advances the state-of-the-art adoption of PETs in AI.

Our proposed approach works for any *arbitrary* distributed data – horizontal, vertical or mixed – and across all 3 research questions outlined above (RQ1, RQ2, and RQ3). In addition to the cross-silo scenarios, our research will also apply to the cross-device scenarios where the data can be curated from millions of users in a way that private, individual data would never be exposed in plaintext (i.e. without being encrypted) to any entity.

Summary of Contributions. The main contributions of this dissertation are twofold:

- The dissertation proposes a general key idea for training machine learning models under

the DP-in-MPC paradigm, aimed at achieving the objectives outlined in the dissertation statement (Section 1.3) , namely, providing both input and output privacy, removing the need for a trusted entity, and supporting arbitrary data partitioning at scale while achieving utility at par with the centralized paradigm. A key component of this general solution is designing generic MPC protocols to generate noise to satisfy DP guarantees. The required MPC protocols are built on existing efficient MPC primitives.

- The core contributions of the dissertation include the design and development of novel and generic MPC protocols that replace the role of a trusted central entity (a.k.a. aggregator). These MPC protocols are typically the adaptations of the algorithms and architectures used in the centralized paradigm.

  Computations done using MPC protocols are generally much slower than computations performed in-the-clear (i.e. non-private computations in the centralized paradigm, where the trusted central entity can access all the aggregated data). To achieve feasible computational efficiency, existing techniques from the centralized paradigm often need to be adapted to be "MPC-friendly" in a way that does not significantly affect the utility of these algorithms.

  A major challenge we address in this dissertation is the development of MPC-friendly versions of algorithms and the implementation of their corresponding MPC protocols. As part of these contributions, the dissertation presents: (i) MPC protocols for training linear models with DP, (ii) MPC protocols to achieve fairness in AI models while providing DP guarantees, and (iii) MPC protocols for generating DP synthetic data.

1.4.4 Chapter Outline

This dissertation is structured as follows to address the specific research questions and the objectives mentioned above. The contributions of each chapter are summarized as well as how they advance the state-of-the-art.

Chapter 2 provides a brief background on the PETs employed in the dissertation, specifi-

cally focusing on Secure Multiparty Computation (MPC) and Differential Privacy (DP) and how they provide input and output privacy respectively, while detailing how DP-in-MPC works. This background is essential to understand the subsequent chapters that rely on these techniques.

RQ1 Privacy-Preserving Training of Machine Learning Models

Chapter 3 addresses the literature gap in training ML models based on data distributed across multiple data holders in an arbitrary way, specifically in cross-silo settings, while achieving accuracy on par with centralized models. We demonstrate the effectiveness of the proposed DP-in-MPC paradigm in the context of training logistic regression (LR) in the distributed paradigm with DP guarantees. This work led to a 1st place in Track III of the iDASH 2021 Genome Privacy competition[11].

We design novel MPC protocols to train a DP-enabled logistic regression (LR) model using output perturbation. To the best of our knowledge, this is the first development of MPC protocols for output perturbation. We evaluate the trained ML models empirically for accuracy in both horizontal and vertical distribution settings using health datasets for classification tasks, and show better utility compared to relevant state-of-the-art techniques. We note that the protocols we design are interchangeable with all linear learners.

RQ2 Privacy-Preserving Bias Mitigation in Machine Learning Models

Chapter 4 introduces the concept of fairness and bias mitigation in AI. It briefly describes the generic techniques for bias mitigation in ML pipelines. These concepts are essential to understand Chapter 5 – 7.

[11]iDASH, supported by NIH, is a privacy and security workshop on secure genome analysis `https://www.humangenomeprivacy.org/`.

Chapter 5 explores the problem of auditing deployed machine learning (ML) models for algorithmic bias while preserving the privacy of sensitive audit data. Unlike other chapters that focus on protecting the training data throughout the ML pipeline, this chapter is an exception in that it addresses privacy at the inference stage, where auditing typically takes place between a service provider and a service consumer. Although this chapter does not use the DP-in-MPC paradigm, it serves as a foundational chapter for understanding the intersection of privacy and fairness.

ML models are known to exhibit discrimination and biases based on sensitive attributes such as gender, race, or disability. Auditing the ML models can help in deploying the less biased models for service and inference. But auditing is usually conducted with sensitive user characteristics that are subject of anti-discrimination and data protection law. Existing fairness auditing tools fail to provide any guarantees for protecting audit data. To address this, we introduce `PrivFair`, a privacy-preserving auditing library that leverages MPC to ensure the confidentiality of both the proprietary model under audit and the audit data. We design novel MPC protocols to compute fairness metrics required for assessing models for bias during auditing. We demonstrate the use of PrivFair for group fairness auditing with tabular data and image data, without requiring the investigator to disclose their data to anyone in an unencrypted manner, or the model owner to reveal their model parameters to anyone in plaintext (i.e. in-the-clear).

Chapter 6 presents novel ideas to achieve 'group fairness' in a cross-device setting while preserving the privacy of sensitive attributes of users.

Group fairness ensures that the outcome of ML based decision making systems are not biased towards a certain group of people defined by a sensitive attribute such as gender or ethnicity. Achieving group fairness in a cross-device FL setup is challenging because mitigating bias inherently requires using the sensitive at-

tribute values of all users (a.k.a. clients in FL), while FL is aimed precisely at protecting privacy by *not* giving access to the users' data. While recent research has combined FL and DP to train group fair ML models, most of them rely on in-processing bias mitigation techniques with very few exploring pre- and post-processing mitigation techniques. The latter offer advantages in certain use-cases. Moreover, none of these combine MPC and DP in effective way. In this chapter, we address this research gap by employing the DP-in-MPC paradigm to achieve group fairness without revealing sensitive user data using well known pre- and post-processing mitigation techniques in the centralized paradigm and adapting them to a federated setup. To this end, we develop novel MPC protocols by building upon the MPC protocols from Chapter 5 for reweighing techniques with DP as a pre-processing technique for group fairness and a post-processing technique that identifies different thresholds for each group with DP. We demonstrate the effectiveness of our approach on real world datasets for cross-device settings. These MPC protocols can be easily adapted to cross-silo settings too.

Chapter 7 extends on the research at the intersection of privacy and fairness for retrieval and recommendation systems in cross-device settings. These systems often display the results to end users (service consumers) in the form of ranked lists of the items (of the producers) ordered according to end user preferences. Such ranking often introduces positional biases towards the top ranked items impacting the producers of the items. To mitigate this, re-ranking systems are often deployed at inference stage before presenting the ranked list to end users and these computational methods for fair item ranking rely on disclosing end user data to a centralized server, which gives rise to privacy concerns for the end users. Ours is the first to advance research at the conjunction of producer (item) fairness and consumer (user) privacy in rankings. The focus here is to protect the inference data rather than training data. We leverage the DP-in-MPC paradigm to build

a novel framework to adapt and extend the equity of the amortized attention ranking mechanism (originally proposed for the centralized paradigm) to a cross-device federated setup. Our results using real-world datasets show that we are able to effectively preserve the privacy of users and mitigate unfairness of items without making additional sacrifices to the quality of rankings in comparison to the ranking mechanism in the centralized paradigm.

RQ3 Privacy-Preserving Synthetic Data Generation

Chapter 8 introduces the concept of synthetic data ,i.e. artificially generated data that mimics real data, and the process of synthetic data generation (SDG). It briefly highlights the research gaps in current systems, and discusses the privacy issues that arise when generating synthetic data from multiple sources. The chapter also provides an overview of a general solution and introduces the concepts that serve as the groundwork for the frameworks developed in Chapters 9 and 10.

Chapter 9 proposes a novel framework for collaborative and private generation of synthetic tabular data based on data from distributed data holders.

SDG offers an appealing solution to overcome data sharing barriers imposed by privacy regulations. However, existing SDG approaches typically assume a trusted central aggregator for training a synthetic data generator. An assumption that may not hold, as many data holders are either unwilling or legally unable to share their raw data with a central entity. In this work, we focus on a family of marginal-based SDGs, which have proven effective for tabular data, and propose a first-of-its-kind framework to ensure both input and output privacy by extending centralized algorithms to distributed settings. To achieve this, we adopt the DP-in-MPC paradigm to build our framework and propose novel MPC protocols that support the common steps of almost all marginal-based SDGs. We demonstrate their effectiveness of proposed approach on real-world datasets.

Chapter 10 extends the framework proposed in Chapter 9 to incorporate additional aspects of training a synthetic data generator: pre-processing, evaluation and hyperparameter tuning of synthetic data generator during training. These crucial aspects are often overlooked in existing privacy-preserving approaches.

While most existing techniques for distributed scenarios focus solely on synthesizer training, they assume that training data has already been pre-processed and that the synthetic data can be generated in a single step, without iterative tuning. In this chapter, we propose a novel generic framework using the DP-in-MPC paradigm to ensure input and output privacy that works with any arbitrary distribution for publishing synthetic data and that supports privacy-preserving pre-processing, training, evaluation, and hyperparameter tuning. We demonstrate the framework for the use-case of privacy-preserving synthetic genomic data publishing for leukemia research. To this end, we propose novel, specific protocols tailored to the use-case and marginal-based SDGs, building upon the protocols developed in Chapter 9.

Chapter 11 concludes the dissertation by summarizing the key takeaways, discussing open research challenges, and outlining potential directions for future research.

1.4.5 List of publications

This dissertation is based on the material from the following published works:

Conference Papers

- S. Pentyala, N. Neophytou, A. Nascimento, M. De Cock, G. Farnadi, Privacy-Preserving Group Fairness in Cross-Device Federated Learning, Proceedings of Machine Learning Research, PMLR, 2025.

- S. Pentyala, M. Pereira, M. De Cock. CaPS: Collaborative and Private Synthetic

Data Generation from Distributed Sources, 41st International Conference on Machine Learning, ICML, 2024.

- J. Sun, S. Pentyala, M. De Cock, G. Farnadi Privacy-Preserving Fair Item Ranking, 45th European Conference on Information Retrieval, ECIR, 2023.

Workshop Papers

- S. Pentyala, G. Sitaraman, T. Claar, M. De Cock. End to End Collaborative Synthetic Data Generation, AAAI-25 Workshop on Privacy-Preserving Artificial Intelligence, 2025. (Full paper, 8 pages)

- M. Pereira, S. Pentyala, A. Nascimento, R. T. de Sousa Jr., M. De Cock. Secure Multiparty Computation for Synthetic Data Generation from Distributed Data, SyntheticData4ML Workshop@NeurIPS 2022. (Full paper, 6 pages)

- S. Pentyala, D. Melanson, M. De Cock, G. Farnadi PrivFair: a Library for Privacy-Preserving Fairness Auditing, AAAI-22 Workshop on Privacy-Preserving Artificial Intelligence, 2022. (Full paper, 8 pages)

Under Review

- S. Pentyala, D. Railsback, R. Maia, R. Dowsley, D. Melanson, A. Nascimento, M. De Cock. Input and Output Privacy in Cross-Silo Federated Settings: an MPC+DP Approach. Under review. (Full paper, 16 pages)

Chapter 2

BACKGROUND ON PRIVACY ENHANCING TECHNOLOGIES

"If I have seen further, it is by standing on the shoulders of giants."

Issac Newton

Various privacy enhancing techniques (PETs) have been developed to address different aspects of data privacy, each with their own advantages, limitations, and assumptions. While some of them provide input privacy, i.e. protecting the raw data from being disclosed, others emphasize on output privacy, ensuring that the results of a computation do not leak sensitive information. In this chapter, we provide a brief background on the PETs relevant for this thesis, namely, Federated Learning (FL) in Section 2.1, Secure Multiparty Computation (MPC) in Section 2.2 and Differential Privacy (DP) in Section 2.3. We describe how each of these preserve privacy and discuss their limitations specifically in the context of distributed data settings. We further note that an ideal privacy- preserving AI system should employ a combination of multiple PETs to enhance data protection and data privacy.

2.1 Federated Learning

Federated Learning (FL) [87] is a collaborative learning paradigm designed for a distributed setup with H data holders, where each data holder i ($i = 1 \dots H$) holds a private dataset D_i. The goal is to train a ML model, often called a global model, on the collective data without requiring any data holder to share their raw data, i.e. while keeping each dataset D_i private and decentralized. In this paradigm, often referred to as a federation, the data holders are

called clients, and they coordinate with a central server (a.k.a. aggregator) to jointly learn a global model.

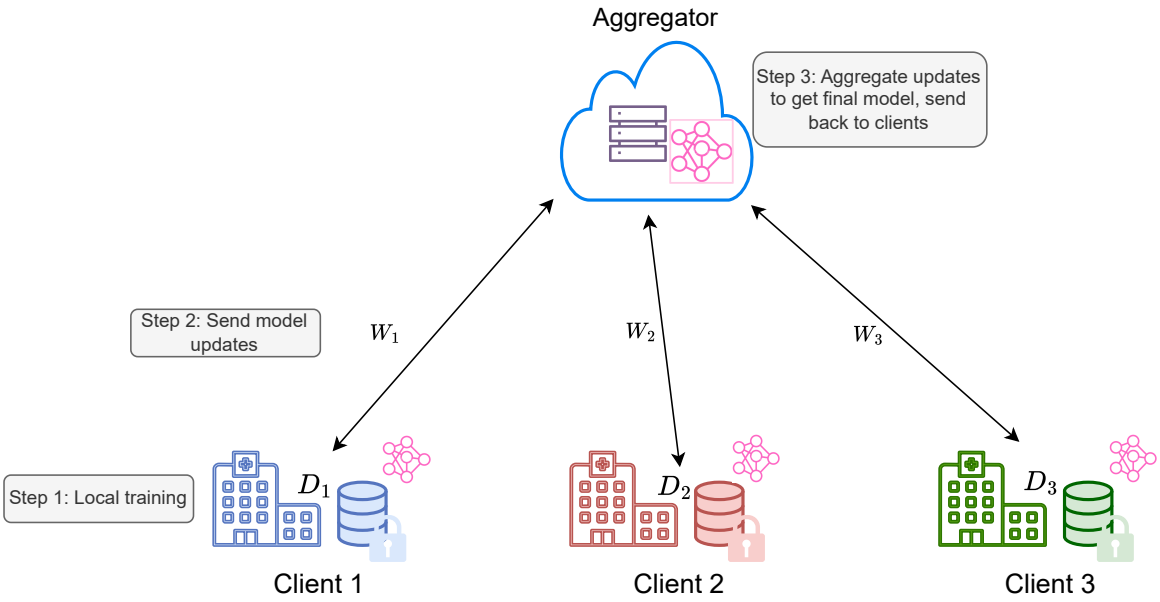


Figure 2.1: **Federated Learning**. Clients train ML models locally on their private data with or without output privacy and send the model parameters/gradients to an aggregator that aggregates the model gradients to obtain a global model. The raw data never leaves the clients.

The clients train the model on their own data locally and send only the models updates, which are typically the gradients or model parameters W_i, to the aggregator. The aggregator then combines the model updates, usually by averaging, and sends the updated global model back to the clients (see Figure 2.1). This is done iteratively for a defined number of rounds to get a final global model f that has achieved the learning objective on the combined data of the clients. This enables the clients to keep their data within the premises providing first-level privacy to the clients and hence providing *input privacy*. Note that FL relies on the principle of data minimization to preserve privacy which relates to minimal collection

and retention to the absolute minimum necessary to achieve a specific, legitimate purpose.

FL frameworks can be set up for both cross-silo and cross-device settings. In cross-silo FL, the clients are typically organizations such as hospitals, banks, or companies, each holding large datasets and having reliable network connectivity. In cross-device FL, the clients are edge devices like smartphones or IoT sensors, each contributing a small amount of data and being intermittently available due to unreliable connectivity and power constraints [59]. FL in cross-device settings has been successfully deployed in practice by Google, Apple, Microsoft and others [88, 89, 90].

Traditional FL was originally proposed for horizontally partitioned data, also known as Horizontal Federated Learning (HFL), where data across clients share the same feature space but differ in the sample space (i.e. each client holds records for different individuals with the same set of features) [87]. However, in many real-world scenarios, data is vertically partitioned, where multiple clients hold different features for the same set of individuals (i.e. same sample space, different feature space). This led to research in Vertical Federated Learning (VFL) [91]. Other extensions of FL such as split learning [92] and federated transfer learning [93] are also based on the principle of data minimization and can work with either of the partitions.

Assumptions. FL assumes an honest-but-curious and reliable aggregator, meaning the aggregator correctly follows the protocol but may attempt to learn additional information from received data such as the model updates from the clients. By design, the aggregator represents a single point of failure in traditional FL architecture. If the aggregator malfunctions or fails, the entire system breaks. Clients are assumed to be honest participants and data contributors[1], adhering to the protocol without malicious behavior. Clients are expected to have sufficient resources, including computational power, memory, and reliable network connectivity. All communications between clients and the aggregator are assumed to be secure, protecting against eavesdropping and tampering by external adversaries.

[1]There is some research to overcome these assumptions (such as [94], [95]).

Privacy Threats and Vulnerabilities. The aggregator may attempt to infer sensitive client data from the model updates or gradients it receives. Even though the aggregator is assumed to be honest-but-curious, it may still seek to extract private information about the clients' local datasets through careful analysis of aggregated model parameters or gradients [96, 97]. Malicious clients could engage in data poisoning attacks where they intentionally corrupt the training data, and model inversion attacks attempting to reconstruct sensitive information about other clients' data from the model parameters or gradients [98]. There is also risk of privacy leakage through the transmission of gradients or intermediate model parameters, which may inadvertently reveal information about the clients' datasets [99]. FL is prone to membership inference attacks [100]. Research in FL has been making progress in mitigating these vulnerabilities [101].

Limitations. FL alone cannot guarantee complete privacy. It is often combined with MPC to strengthen input privacy and with DP to provide output privacy [59, 102, 76]. Often, two variants of DP are deployed in FL. One is local DP, where each client adds noise to their data or model updates before sharing them [85]. Distributed DP, on the other hand, introduces an intermediate step, such as shuffling or secure aggregation, between the data holders and the aggregator. Distributed DP typically requires less noise than local DP to achieve comparable privacy guarantees [103]. FL's performance with or without DP is often inferior to that of centralized models. Furthermore, practical scenarios often involve a mix of horizontal and vertical partitioning, i.e. arbitrary partitioning. A single FL framework typically supports either horizontal or vertical partitioning, but not both. Other limitations that have spurred research in FL include handling non-IID data and heterogeneous settings, scalability, model personalization and adversarial attacks.

While FL is not the primary focus of this thesis, we do leverage the FL framework in Chapters 6 and 7. These chapters employ cross-device setups, which correspond to a horizontally distributed data setting, making FL a particularly suitable alternative to more computation-heavy approaches. The assumptions and implementation details specific to FL

are described in those chapters.

2.2 Secure Multiparty Computation

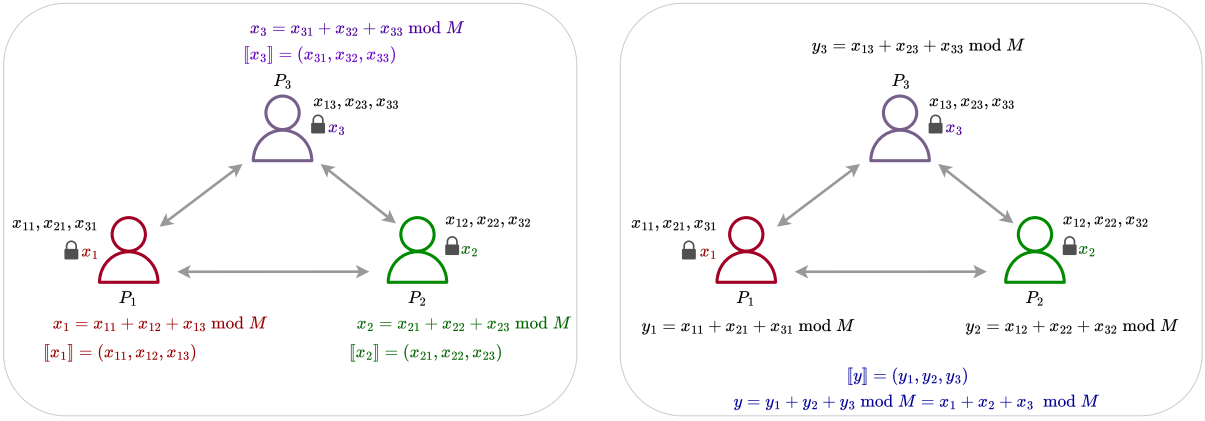
Secure Multiparty Computation (MPC) is an umbrella term for cryptographic methods that allow two or more parties to jointly perform computations to calculate the output of a previously agreed function on their combined inputs, without revealing the private information [104, 105]. In other words, the goal of MPC is to reveal nothing to an adversary beyond what is revealed by the output and any inputs already in possession of the adversary[2]. Consider computing parties $P_1, P_2, \ldots, P_n$ who respectively hold the inputs $x_1, x_2, \ldots, x_n$ and want to compute $y = f(x_1, x_2, \ldots, x_n)$ without any party, P_i, revealing its private data x_i to any other party, i.e. without revealing x_i to any party within or outside the group of computing parties. MPC provides *input privacy* by ensuring that even if some parties are compromised by an adversary, the private inputs of honest parties remain protected.

Secret-sharing. One of the most widely used techniques to achieve secure computation in MPC is secret-sharing. The basic idea is for each party to split its private input into multiple shares, based on some randomness, and distribute them among other parties such that no single share reveals any information about the original input. Formally, if n parties participate and at least $t + 1$ shares are required to reconstruct the original secret, the scheme is referred to as a (t, n) threshold secret-sharing scheme[3]. Until $t + 1$ or more shares are combined, the original input remains secured and cannot be reconstructed.

For instance, as shown in Figure 2.2.(a), P_1 secret-shares its secret (i.e. private input) x_1 with P_2 and P_3. To do so P_1,

[2]The goal of an adversary is to learn information about the private inputs of the other honest parties, modify the computations, or even cause the protocol to fail.

[3]In MPC, a scheme typically refers to a mathematical method for encoding and distributing secrets (e.g. secret-sharing), while a protocol defines the sequence of steps and communication between parties to jointly compute a function securely using such schemes.



(a) Secret Sharing

(b) Private Computation of Sum

Figure 2.2: **Secure Multiparty Computation.** Parties P_1, P_2 and P_3 hold their private data a.k.a. secrets x_1, x_2, and x_3, respectively. (a) secret-sharing: P_i secret-shares x_i with others to enable joint computations without revealing x_i, thus providing input privacy. (b) Private Computation of Sum: Parties perform local computations, thereafter holding secret-shares of sum y. None of the private inputs are exposed while computing sum, thus providing input privacy during intermediate computations.

1. converts x_1 into integer
2. generates shares by randomly splitting x_1 into x_{11}, x_{12}, and x_{13} such that $x_1 = x_{11} + x_{12} + x_{13} \mod M$ (all computations including splitting are done modulo M, see Number Representation in Section 2.2.1)
3. retains x_{11}, sends x_{12} to P_2, and sends x_{13} to P_3

P_2 and P_3 secret-share in a similar manner. This results in P_1 holding x_{11}, x_{21} and x_{31}; P_2 holding x_{12}, x_{22} and x_{32}; P_3 holding x_{31}, x_{23} and x_{33}

We denote the secret-shared value of x_i by $[\![x_i]\!]$ i.e. $[\![x_i]\!] = (x_{i1}, x_{i2}, x_{i3})$ where each x_{ij} is a share of x_i held by party j.

These shares can then be used to jointly compute the desired function without ever reconstructing x_i during the intermediate steps (reconstruction refers to combining all the

split shares to compute x_i back i.e. by summing up all secret-shares). This approach ensures that each party only sees shares of the inputs of others, never the actual data, thereby preserving input privacy throughout the computation.

Many MPC schemes rely on secret-sharing, particularly linear secret-sharing schemes (LSSS), due to their efficiency and local computability[4]. A linear secret-sharing scheme allows secret-shares to preserve linear relationships. That is, any linear operation performed on the shares directly reflects in the reconstructed secret. This linearity (or additive homomorphism) enables parties to compute linear functions over secrets locally, without any interaction or communication, by simply applying the function to their shares.

For example, Figure 2.2(b) illustrates how to securely compute the sum $y = x_1 + x_2 + x_3$, where each x_i is a private input held by party P_i. The computation is done using secret-sharing. Each party holds a share of every input, and they perform *local addition* on their shares. Specifically, party P_i computes:

$$y_i = \sum_{j=1}^{3} x_{ji} \mod M$$

where x_{ji} is the share of x_j held by P_i. As a result, P_1 holds y_1, P_2 holds y_2, and P_3 holds y_3, which together form the secret-shared value of the sum: $[\![y]\!] = (y_1, y_2, y_3)$. These shares can then be *combined* to reveal the final result. In this case, the output can be reconstructed by simply adding all the shares which is equivalent to computing the original sum:

$$y = y_1 + y_2 + y_3 \mod M = \sum_{j=1}^{3} x_{j1} + x_{j2} + x_{j3} \mod M = x_1 + x_2 + x_3 \mod M$$

Note that in MPC, computations are performed on secret-shared data in such a way that the final result is the same as if the computations had been performed directly on the original data, thereby preserving its correctness while ensuring privacy. Furthermore, while the final output value y may reveal information about the inputs (i.e. there is no output privacy), the computation itself is carried out securely without revealing any intermediate values.

[4]Not all MPC schemes use LSSS. Some protocols are based on other cryptographic primitives such as garbled circuits, oblivious transfer, or homomorphic encryption [105].

Many different MPC schemes exist, and some of them are listed in Table 2.1. The MPC protocols we propose in this thesis are sufficiently generic to work with any of the MPC schemes in general.

Table 2.1: List of a few MPC schemes for different adversarial settings

MPC scheme	Reference	#Parties (n)-PC	Domain	Adversarial Power
OT-based	[1]	2	$\mathbb{Z}_{2^k}$	Honest-but-curious
SPDZ2K	[1]	2	$\mathbb{Z}_{2^k}$	Malicious
	[2]			
Replicated (Replicated2k)	[3]	3	$\mathbb{Z}_{2^k}$	Honest-but-curious/Malicious
SPDZ-wise replicated	[4]	3	$\mathbb{Z}_{2^k}$	Malicious
Replicated (Rep4-2k)	[4]	4	$\mathbb{Z}_{2^k}$	Malicious
Shamir	[106]	3 or more	$\mathbb{Z}_p$	Honest-but-curious/Malicious

Below, we describe one widely used and efficient MPC scheme, namely Replicated secret-sharing (RSS) ([3]). Note that this scheme is different from the one used in the example shown in Figure 2.2. RSS has a security model for $n = 3$ parties and $t = 1$ (with $t < n/2$) and uses a $(1, 3)$ threshold secret-sharing scheme along with MPC protocols that operate over a ring $\mathbb{Z}_{2^k}$. This security model can tolerate one corrupted party and requires at least two parties to reconstruct the secret.

2.2.1 Replicated secret-sharing

Number representation. MPC schemes use various algebraic structures. In the MPC schemes that we use in our thesis, computations are performed over integers modulo M, i.e. $\mathbb{Z}_M$, where $M = \{0 \ldots, M-1\}$ and $\mathbb{Z}_M$ is equipped with addition and multiplication modulo M. The choice of the computation domain represented by M affects the efficiency

of the protocols used for the particular MPC scheme. When M is a prime number p, the computation domain is the field $\mathbb{Z}_p$ and when $M = 2^k$, the domain is the ring $\mathbb{Z}_{2^k}$.

Independent of the choice of computation domain, MPC protocols perform operations over discrete algebraic structures (finite fields and rings). So, it is important to map any input we want to perform computations on into such discrete structures. We consider these discrete structures as "integers" ($\mathbb{Z}_M$). The parties convert real values to integers and then secret-share them amongst each other. These real values are generally represented in floating point notation on computers but performing MPC protocols on floating point notation is expensive.

A commonly used representation for real values in MPC is a fixed-point representation [107]. The real numbers are represented as $\lfloor r \cdot 2^p \rfloor$ where r is the real value to be mapped and p is the length of the fractional part, also called precision. In other words, the real value r is mapped to an integer q scaled by a factor of 2^{-p}. We can assume that the zero point is '0'.

$$r = 2^{-p} \cdot q$$

b denotes the total bit-length used to represent the integer q in two's complement notation. This means, using b-bit integers, values can typically be represented in the range $[-2^{b-1}, 2^{b-1} - 1]$. Hence, the representable range for real numbers becomes approximately $[-2^{b-p-1}, 2^{b-p-1}]$.

With $p = 16$, $r = 5.3$ will be, for example, converted to 347340 and then secret-shared. Similarly, -5.3 will be converted as -347340 and then secret-shared. As we explain next, secure addition, subtraction, and comparison of fixed-point numbers can be performed similar to as integers. Multiplication performed on two fixed point numbers generates extra p bits leading to a product upscaled by (2^{-2p}) This requires downscaling so that the product can be represented as per the notation mentioned in the above equation (see [108, 109] for reference).

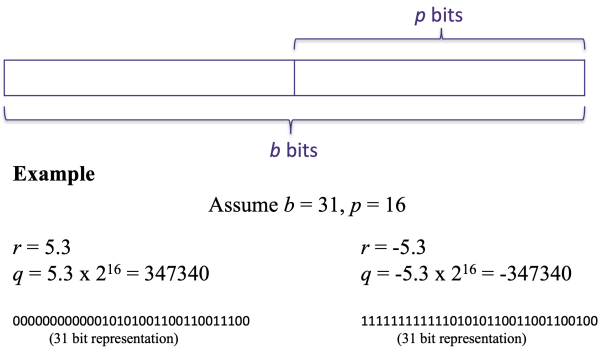


Figure 2.3: Fixed point representation: Mapping a real value to integer

RSS Scheme. In Replicated secret-sharing (RSS), a scheme for three-party computation (3PC), a private value x (assuming it is held by P_1) in $\mathbb{Z}_M$ is secret-shared among three parties P_1, P_2, and P_3, by generating uniformly random shares such that $x = x_1 + x_2 + x_3 \mod M$ where $x_i \in \mathbb{Z}_M$ are the secret-shares of x. We denote the secret-shared value of the private value x by $[\![x]\!]$ i.e. $[\![x]\!] = (x_1, x_2, x_3)$ where each x_i is a random share of x. To secret-share a value x, the dealer (i.e. the party holding the secret) uniformly selects two random values x_1, x_2 in the computation domain M. The third value is then calculated as $x_3 = x - x_1 - x_2 \mod M$.

In this scheme, the shares are typically distributed in pairs, such that each party P_i holds two out of the three shares, i.e. a pair (x_j, x_{j+1}), where the indices wrap around modulo 3, i.e. $x_4 = x_1$. For example,

$$x = x_1 + x_2 + x_3 \mod M$$
$$P_1 \text{ holds } (x_1, x_2)$$
$$P_2 \text{ holds } (x_2, x_3)$$
$$P_3 \text{ holds } (x_3, x_1)$$

To reveal the secret to all parties, each P_i sends its share x_i to P_{i+1}. After this step, every party has received all three shares and can locally reconstruct the original value x. If the secret is to be revealed to only a single party P_i, then only the missing share x_{i-1} (i.e. the

one not already held by P_i) is sent to it by P_{i-1} so that P_i can reconstruct x locally. While there are several variations in notation and share distribution, the core idea remains: each party holds two shares of the secret, and any two parties can jointly reconstruct it. Such distribution ensures that no party can alone infer or deduce any information about x. Once the secret-sharing is done, parties continue to perform computations on the secret-shared data. Linear operations such as addition, addition with a constant, and multiplication with a constant can be performed locally by all the servers.

Addition. Let us assume that $z = x + y$ is to be computed where x and y are secrets. If x is split as $x = x_1 + x_2 + x_3 \mod M$ and y is split as $y = y_1 + y_2 + y_3 \mod M$, then as per the replicated secret-sharing scheme, P_1 will hold x_1, x_2, y_1, y_2, P_2 will hold x_2, x_3, y_2, y_3 and P_3 will hold x_3, x_1, y_3, y_1. They will perform the addition on the secret-shares locally, and as a result will hold the secret-shares of $z = z_1 + z_2 + z_3 \mod M$. In other words, all parties compute $[\![x]\!] + [\![y]\!]$ generating $[\![z]\!]$, i.e. secret-share pairs from the set $(x_1 + y_1, x_2 + y_2, x_3 + y_3)$. This is illustrated below where at the end of this protocol, the parties will hold $[\![z]\!]$ as if z was directly secret-shared to the servers.

To calculate $z = x + y$

Parties hold the secret-shares

P_1 holds (x_1, x_2, y_1, y_2)

P_2 holds (x_2, x_3, y_2, y_3)

P_3 holds (x_3, x_1, y_3, y_1)

Locally compute addition:

$P_1 : (x_1 + y_1, x_2 + y_2) => (z_1, z_2)$

$P_2 : (x_2 + y_2, x_3 + y_3) => (z_2, z_3)$

$P_3 : (x_3 + y_3, x_1 + y_1) => (z_3, z_1)$

Reconstruction from any two parties:

$z = z_1 + z_2 + z_3 \mod M = \text{Sum of } (x_1 + y_1, x_2 + y_2, x_3 + y_3) = x + y$

Adding a public constant. Adding or subtracting a public constant value can be done in a similar way as illustrated below, but the secret-share pairs will be from the set $(x_1 \pm c, x_2, x_3)$.

To calculate $z = x + c$

Parties hold the secret-shares

P_1 holds (x_1, x_2)

P_2 holds (x_2, x_3)

P_3 holds (x_3, x_1)

Locally add to only one party:

$P_1 : (x_1 + c, x_2) => (z_1, z_2)$

$P_2 : (x_2, x_3) => (z_2, z_3)$

$P_3 : (x_3, x_1 + c) => (z_3, z_1)$

Reconstruction from any two parties:

$$z = (z_1 + z_2 + z_3) \mod M = \text{Sum of } (x_1 + c + x_2 + x_3) = x + c$$

Multiplying with a public constant. To multiply by a public constant c, all the parties can locally multiply their secret-shares with c as shown below. The parties hold secret-shares of the product.

To calculate $z = cx$

Parties hold the secret-shares

P_1 holds (x_1, x_2)

P_2 holds (x_2, x_3)

P_3 holds (x_3, x_1)

Locally multiply:

$P_1 : (cx_1, cx_2) => (z_1, z_2)$

$P_2 : (cx_2, cx_3) => (z_2, z_3)$

$P_3 : (cx_3, cx_1) => (z_3, z_1)$

Reconstruction from any two parties:

$$z = z_1 + z_2 + z_3 \mod M = \text{Sum of } (cx_1, cx_2, cx_3) = cx$$

Multiplication. To compute the product of two secret-shared values $z = x \cdot y$ (in the equation below) where x and y are secrets, the parties need to communicate. As before, x is split as $x = x_1 + x_2 + x_3 \mod M$ and y is split as $y = y_1 + y_2 + y_3 \mod M$, and P_1 will hold x_1, x_2, y_1, y_2, P_2 will hold x_2, x_3, y_2, y_3, and P_3 will hold x_3, x_1, y_3, y_1.

$$\begin{aligned} z &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) \\ &= x_1y_1 + x_1y_2 + x_1y_3 \\ &\quad + x_2y_1 + x_2y_2 + x_2y_3 \\ &\quad + x_3y_1 + x_3y_2 + x_3y_3 \end{aligned}$$

The protocol proceeds as per the following steps:

- P_i calculates $v_i = x_iy_i + x_iy_{i+1} + x_{i+1}y_i$ locally[5].
- A triple of random values (u_1, u_2, u_3) over computational domain $\mathbb{Z}_M$ are selected such that $u_1 + u_2 + u_3 = 0$. Each party knows only one of these values, i.e. P_1 will have u_1 only. These can be computed locally by the individual servers by generating the random shares of '0' using pseudo random functions, and secret-sharing them using a simple additive secret-sharing scheme. Having such values helps in randomizing the shares of v.
- P_i locally adds u_i to v_i to generate z_i.
- P_i sends z_i to P_{i-1}. This is done because after the above step each party holds one new share, but in replicated secret-sharing we need that secret-shares are held as pairs, i.e. all parties must hold two out of the three generated shares.
- Now P_i holds z_i and z_{i-1} similar to how it held the secret-shares of x.

This is illustrated below in the colored text and followed up with the compiled equations.

[5]Consider x_4 as x_1 and y_4 as y_1 and P_0 as P_3

To calculate $z = xy$

1. Creating secret-shares:

   $x = x_1 + x_2 + x_3 \mod M$

   $y = y_1 + y_2 + y_3 \mod M$

2. Distributing secret-shares:

   P_1 holds (x_1, x_2, y_1, y_2)

   P_2 holds (x_2, x_3, y_2, y_3)

   P_3 holds (x_3, x_1, y_3, y_1)

3. Locally compute sum of cross products:

   $P_1 : (x_1y_1 + x_1y_2 + x_2y_1) => (v_1)$

   $P_2 : (x_2y_2 + x_2y_3 + x_3y_2) => (v_2)$

   $P_3 : (x_3y_3 + x_3y_1 + x_1y_3) => (v_3)$

4. Randomize

   P_1 holds $(v_1, u_1) => (z_1 = v_1 + u_1)$

   P_2 holds $(v_2, u_2) => (z_2 = v_2 + u_2)$

   P_3 holds $(v_3, u_3) => (z_3 = v_3 + u_3)$

5. Note:

   $z = z_1 + z_2 + z_3 \mod M$

6. Resharing:

   P_1 holds (z_1, z_2)

   P_2 holds (z_2, z_3)

   P_3 holds (z_3, z_1)

$$
\begin{aligned}
z &= (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) \\
&= x_1y_1 + x_1y_2 + x_1y_3 + x_2y_1 + x_2y_2 + x_2y_3 + x_3y_1 + x_3y_2 + x_3y_3 \\
&= (x_1y_1 + x_1y_2 + x_2y_1) + (x_2y_2 + x_2y_3 + x_3y_2) + (x_3y_3 + x_3y_1 + x_1y_3) \\
&= v_1 + v_2 + v_3 \\
&= v_1 + v_2 + v_3 + 0 \\
&= v_1 + v_2 + v_3 + (u_1 + u_2 + u_3) \\
&= (v_1 + u_1) + (v_2 + u_2) + (v_3 + u_3) \\
&= z_1 + z_2 + z_3
\end{aligned}
$$

The above protocols, independent of the security setting, are usually divided in an online and an offline phase. The offline phase involves any pre-processing steps (such as generating random values) or computations that can be done independent of the inputs (x and y are inputs in $z = x + y$), whereas computations in the online phase completely depend on the inputs and cannot be executed prior to receiving inputs unlike the offline phase.

2.2.2 MPC Primitives

In Table 2.2, we list the common MPC primitives used in this thesis. These primitives are generic and defined for a variety of MPC schemes, including the RSS scheme described earlier. All of these primitives, as proposed in the literature [110], are correct and secure. The security of these MPC primitives is well-established under the Universal Composability (UC) framework [111]. In the UC model, protocols are analyzed by comparing a real-world execution with an idealized one where a trusted third party computes the functionality. The primitives in Table 2.2 thus can be securely composed to build larger protocols within the UC framework, as we do in this thesis. Next we describe the the adversarial (or threat) models under which these protocols remain secure.

2.2.3 Adversarial Models

MPC schemes are designed to prevent successful attacks by an adversary $\mathcal{A}$ that tries to corrupt one or more parties which can then: (1) passively learn the secret values or information about the secret values, the protocol (e.g. randomness in the protocol) and the intermediate inputs/outputs; or (2) cause incorrect computations by actively involving in changing or communicating wrong input or intermediate values. One can classify security models based on the number of corrupted parties and based on their behavior.

Let us assume there are n parties $P = P_1, P_2 \dots P_n$ in total and $\mathcal{A}$ can corrupt any subset of parties of size t. If the number of corrupted parties is less than half the number of total parties, i.e. $t < n/2$, it is considered an *honest majority* setting as more than half the number of parties are honest (not corrupted). Whereas, in a *dishonest majority* setting, at least half the number of parties are corrupted, i.e. $t \geq n/2$.

A corrupted party can have two different behaviors, each of which may require a different type of security. A corrupted party who follows the protocol honestly but tries to learn information from the received messages or colludes with other corrupted parties to gather private information is a *honest-but-curious* or *semi-honest* or *passive* adversary . In the *semi-*

Table 2.2: Overview of Common MPC Primitives and Notation used in Thesis

Primitive	**Notation**	**Description**
Addition	$+$	Computes the sum of two secret-shared values
Vector addition	π_{SUM}	Computes element-wise sum of two secret-shared vectors
Multiplication	π_{MUL}	Computes the product of two secret-shared values
Vector multiplication	π_{MUL}	Computes element-wise product of two secret-shared vectors
Division	π_{DIV}	Computes the division of two secret-shared values
Dot product	π_{DOT}	Computes the dot product of two secret-shared vector
Equality	π_{EQ}	Tests whether two secret-shared values are equal; returns 1 if they are, and 0 otherwise
Less than	π_{LT}	Compares two secret-shared values $[\![a]\!], [\![b]\!]$; returns 1 if $[\![a]\!] < [\![b]\!]$ 0 otherwise
Greater than equal to	π_{GTE}	Compares two secret-shared values $[\![a]\!], [\![b]\!]$; returns 1 if $[\![a]\!] \geq [\![b]\!]$ 0 otherwise
Maximum value	π_{MAX}	Returns the maximum values in a secret-shared vector
Absolute value	π_{ABS}	Computes the absolute value of secret-shared vector
Random number generation	$\pi_{\mathsf{GR-RANDOM}}(a, b)$	Generates secret-shares of a random number that lies in the interval $[a, b]$
Random bit generation	$\pi_{\mathsf{GR-RNDM-BIT}}$	Generates secret-shares of a random bit, 0 or 1
Natural logarithm	π_{LN}	Computes the natural logarithm of a secret-shared value
Base-2 logarithm	π_{LOG}	Computes the logarithm of a secret-shared value
Exponential	π_{EXP}	Computes the exponential of a secret-shared value
Square root	π_{SQRT}	Computes the square-root of a secret-shared value
Cosine	π_{COS}	Computes the cosine of a secret-shared value
Sine	π_{SIN}	Computes the sine of a secret-shared value
Softmax	π_{SOFTMAX}	Computes the softmax for a secret-shared vector

honest security model, MPC protocols achieve security by preventing leakage of information among corrupted parties (which would otherwise be leaked during collusion). They are quite

efficient and best suited when the only concern is leakage of information. On the other hand, a *malicious* or *active* adversary, may arbitrarily deviate from the protocol. Such a corrupted party can control, modify or generate messages in addition to what a *passive* adversary can. The MPC protocols to detect attacks in the *malicious* security model are relatively expensive in terms of computations and communications as they preserve privacy and ensure correctness when a corrupted party can change the values. The protocols for malicious security, in general, ensure that the output is guaranteed and correct if honest parties have received the output. This ensures that privacy is always preserved. Additionally, to detect when a party is deviating from the protocol, for the active security setting, each secret-share is tagged with another shared value known as MAC (messsage authentication code). MACs, in other words, authenticate each secret-share to ensure that the parties are sending the correct values.

Various MPC schemes are designed to handle different adversarial settings, and a few of them are listed in Table 2.1.

Assumptions. MPC operates under the assumption that no ideal trusted third party exists. All communications and computations occur solely between the participating parties over secure channels with defined properties, either synchronous or asynchronous, with assumptions about authentication and reliability. MPC defines an adversary model, typically assuming non-collusion between a certain threshold of parties. MPC protocols make assumptions about parties' resource capabilities (computational power, memory, bandwidth), their continued availability throughout execution, and their ability to be properly authenticated. Additional assumptions may include the independence of parties' inputs and whether protocols can abort if malicious behavior is detected or must guarantee output delivery regardless of adversarial actions.

Privacy Threats and Vulnerabilities. In MPC, various threats exist based on the defined adversarial setting. Curious parties attempt to learn other parties' inputs by analyzing protocol messages and intermediate values exchanged during computation. Malicious parties or ad-

versaries might additionally perform selective failure attacks, where they cause computation failures when certain input conditions are met, potentially leaking information about honest parties' inputs; and protocol transcript analysis which can leak information even when the final computation is secure, especially in protocols with many rounds of interaction. In threshold-based schemes, an adversary can enforce collusion by exceeding the security threshold. Various MPC schemes are available to mitigate all or some of these vulnerabilities. External adversaries may perform network-level attacks, including traffic analysis and timing attacks, potentially correlating communication patterns with specific inputs or operations.

Limitations. One of the strongest and most impractical limitations of MPC is the non-collusion assumption among parties, which becomes increasingly difficult to guarantee as the number of participants grows. This limitation can be partially mitigated by MPC-as-a-Service infrastructure, where a small set of non-colluding servers run the protocols on behalf of the MPC parties. We discuss this further in Section 2.4. While MPC provides strong input privacy, it generally does not ensure output privacy – the final computation result is often revealed to all parties, potentially leaking information about individual inputs. This motivates our approach of combining DP with MPC via DP-in-MPC, as discussed in Chapter 2.4, where we use DP to provide output privacy. Most MPC frameworks require participants to be simultaneously available and synchronized throughout the entire protocol execution. This requirement is impractical in distributed environments with many data holders (referred to as parties here), especially when connectivity is unreliable. Again, MPC-as-a-Service can help mitigate this by offloading protocol execution to a dedicated infrastructure. Additionally, communication complexity increases significantly with the number of data holders and the complexity of the computations. This too can be partially addressed by leveraging MPC-as-a-Service infrastructure. The performance trade-off between security levels (semi-honest vs. malicious) is also quite substantial.

The core contribution of this thesis, DP-in-MPC (Section 2.4), builds extensively upon the

MPC primitives and adversarial models discussed above. These concepts are foundational to our work in Chapters 3, 5, 6, 7, 9, and 10, with Chapter 5 focusing exclusively on MPC techniques without DP. While this section presents various threat models and security assumptions that exist in the literature, the specific assumptions employed in our DP-in-MPC framework are detailed in Section 2.4. Throughout the thesis, each chapter that utilizes MPC explicitly states the security model (e.g., semi-honest or malicious adversaries) and any additional protocol-specific assumptions.

2.3 Differential Privacy

One of the primary objectives of privacy-preserving data analysis is to ensure that the release of information does not compromise the anonymity of individuals in a dataset, commonly referred to as output privacy. This means that an adversary should not be able to infer sensitive details about any individual based on the released data.

Traditional approaches to anonymization, such as k-anonymity, l-diversity, and t-closeness [112, 113, 114], attempted to achieve this but have been shown to be vulnerable to various attacks, particularly those leveraging auxiliary information for re-identification [115, 116].

A more robust and principled approach is to treat anonymization not as a property of the data, but as a property of the process used to analyze and release information from private data. Differential Privacy (DP) is built on this perspective and has emerged as the gold standard to provide formal privacy guarantees (output privacy).

DP provides strong, mathematically rigorous guarantees that the inclusion or exclusion of an individual's data in a dataset does not significantly affect the output of any analysis. In other words, it ensures plausible deniability. An adversary cannot confidently infer whether any individual's data was used in the computation. This is achieved by introducing a controlled amount of randomness into the data analysis process.

Consider two neighboring datasets D and D' that differ in a single instance, i.e. D' can be obtained either by adding or removing an instance from D or vice-versa. A differentially private randomized algorithm $\mathcal{A}$, as shown in Figure 2.4, ensures that it generates a similar

output probability distribution on D and D' [117]. $\mathcal{A}$ can for instance be an algorithm that takes as input a dataset D of training examples and outputs an ML model; alternatively $\mathcal{A}$ can return statistics about D.

Formally, a randomized algorithm $\mathcal{A}$ is called (ϵ, δ)-DP if for all pairs of neighboring sets D and D', and for all subsets O of $\mathcal{A}$'s range,

$$\mathrm{P}(\mathcal{A}(D) \in O) \leq e^{\epsilon} \cdot \mathrm{P}(\mathcal{A}(D') \in O) + \delta. \tag{2.1}$$

where ϵ and δ are real valued, non-negative numbers; ϵ (privacy budget or privacy loss) controls the strength of the privacy guarantee, while δ represents the probability of privacy violation. The smaller these values, the stronger the privacy guarantees. When $\delta = 0$, then $\mathcal{A}$ is said to be ϵ-DP (also referred to as pure DP). An (ϵ, δ)-DP randomized algorithm $\mathcal{A}$ is commonly created out of an algorithm $\mathcal{A}^*$ by adding noise that is proportional to the *sensitivity* of $\mathcal{A}^*$, in which the sensitivity measures the maximum impact a change in the underlying dataset can have on the output of $\mathcal{A}^*$. The definition assumes that entries in the dataset are independent of one another, and the privacy guarantees hold regardless of any auxiliary information an adversary may possess.

Note: What DP protects is the presence or absence of any one individual's data in a dataset. That is, no matter how much auxiliary knowledge an adversary has, they should not be able to confidently determine whether a particular individual's data was used. In other words, as mentioned earlier, DP ensures that the results of a data analysis are roughly the same whether or not any one individual's data is included, providing plausible deniability. It is important to note that DP does not protect against all types of inference – it does not prevent learning information about the population as a whole. For example, a DP algorithm can still reveal that a majority of people in the dataset are male, or that a certain geographical region has a high incidence of diabetes. What it restricts is what can be learned about any specific individual. For instance, while it may reveal that 60% of the population has a certain trait, it does not allow an adversary to confidently infer whether Alice has that trait, even if she is part of the dataset.

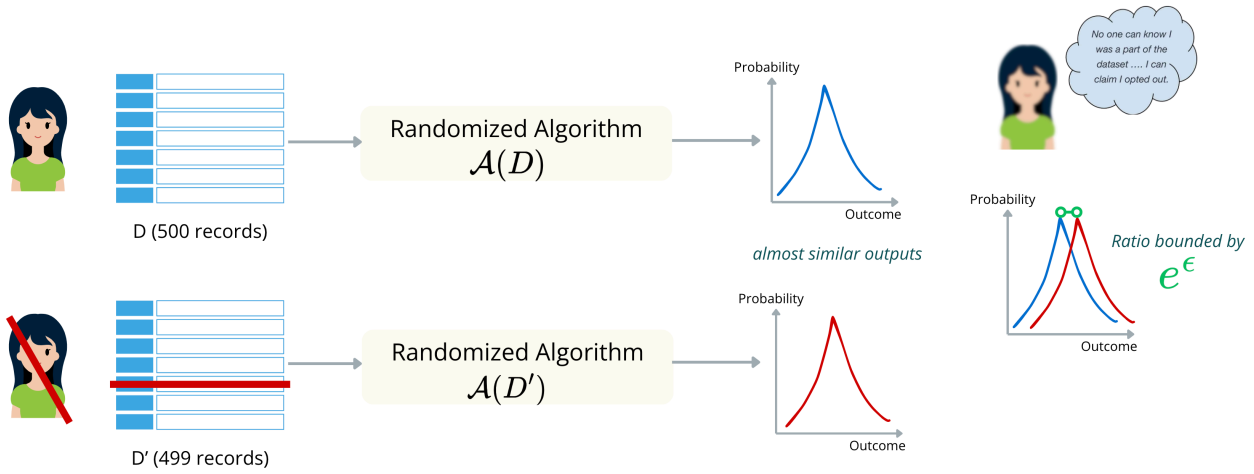


Figure 2.4: **Differential Privacy**. Neighboring datasets D and D' differ in one individual. A DP algorithm $\mathcal{A}$ produces output distributions that are indistinguishable up to e^{ϵ} thus providing output privacy.

DP provides a quantifiable trade-off between privacy and utility. While some leakage of information is inevitable for any useful analysis, DP allows this leakage to be tightly controlled and measured. This makes it one of the most widely adopted PETs in both academic research and real-world applications (e.g. by Apple, Google, the U.S. Census Bureau and National Institute of Standards and Technology (NIST)).

Unit of Privacy. DP can be applied at different granularity levels, defining what constitutes an "instance" in the neighboring datasets. We briefly mention the two most common notions in DP which are relevant to this thesis. Event-level privacy treats each individual data point (e.g. a single transaction, search query, or action) as independent, and aims to protect the privacy of that individual event. In contrast, user-level privacy protects all data associated with a single user, regardless of how many events they contribute. User-level privacy offers stronger protection, as it prevents inferences about an individual even when they have multiple data points in the dataset, but it often comes at a higher utility cost due to the increased sensitivity of user-level contributions.

Mechanisms. DP is typically achieved by adding calibrated random noise addition to the query results or computations. This noise is proportional to the sensitivity of the function – how much the output could change by modifying a single individual's data. We briefly mention the most commonly-used mechanisms to add noise that are used in the thesis.

The Laplace Mechanism is usually used for numeric queries and achieves $(\epsilon, 0)$-DP by adding noise drawn from a Laplace distribution. Given a function $\mathcal{A}^* : D \to \mathbb{R}$, and ℓ_1-sensitivity $\Delta\mathcal{A}^* = \max_{D,D'} \|f(D) - f(D')\|_1$, the Laplace mechanism adds noise as below:

$$\mathcal{A}(D) = \mathcal{A}^*(D) + \text{Lap}(\Delta\mathcal{A}^*/\epsilon)$$

where Lap(.) draws a random value from a Laplace distribution as per the given scale, i.e. $\Delta\mathcal{A}^*/\epsilon$.

The Gaussian Mechanism, on the other hand, achieves (ϵ, δ)-DP by adding noise drawn from a Gaussian distribution. For a function $\mathcal{A}$ with ℓ_2-sensitivity $\Delta\mathcal{A}^*$, this mechanism works as below:

$$\mathcal{A}(D) = \mathcal{A}^*(D) + \mathcal{N}(0, \sigma^2) \quad \text{where } \sigma \geq \frac{\sqrt{2\ln(1.25/\delta)} \cdot \Delta\mathcal{A}^*}{\epsilon}$$

where $\mathcal{N}(.)$ draws a random value from Gaussian distribution as per the given scale, i.e. σ^2.

The Exponential Mechanism is used for non-numeric outputs and allows the selection of an output value from a distinct set of candidate outputs based on a utility function. Given a utility function $u(D, s)$ that scores how good result s is for dataset D, and with sensitivity Δu, the mechanism selects an output $s \in \mathcal{R}$ with probability proportional to:

$$\Pr[\mathcal{A}^*(D) = s] \propto \exp\left(\frac{\epsilon \cdot u(D, s)}{2\Delta u}\right)$$

Each mechanism is suited to different types of queries and use cases, depending on the nature of the data and the required privacy-utility trade-off. We will use all of the above mechanisms in various chapters of the thesis.

Properties. There are three major properties of DP that help in designing DP algorithms.

Sequential composition states that if m differentially private algorithms$\mathcal{A}_1, \ldots, \mathcal{A}_m$ are applied to the same dataset, where each $\mathcal{A}_i$ satisfies ϵ_i-DP, then the combination of all m outputs satisfies $(\sum_{i=1}^{m} \epsilon_i)$-DP. This means that privacy loss accumulates linearly with each additional access to the data.

Parallel composition states that if disjoint subsets of a dataset are each processed using DP algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_m$, where each $\mathcal{A}_i$ satisfies ϵ_i-DP, then the overall privacy guarantee is determined by the $\max(\epsilon_i)$ used among them. That is, if each disjoint subset is analyzed with an ϵ-DP algorithm, the total algorithm still satisfies ϵ-DP. This allows more efficient use of the privacy budget when data is partitioned as disjoint subsets.

The *post-processing property* of DP guarantees that if $\mathcal{A}$ is ϵ-DP, then $g \circ \mathcal{A}$ is also ϵ-DP where g is an arbitrary function. In other words, any arbitrary computations performed on DP output preserves DP without any effect on the privacy budget ϵ.

Deployment Models DP can be deployed in varied settings. The traditional DP paradigm, *global or central DP*, assumes that the entire dataset resides with a central curator who, in addition to performing other computations to generate the output, also computes the noise. Alternatively, if the data originates from multiple data holders, then they can add noise before sending their information to a trusted curator for further processing. Approaches based on this *local DP* paradigm tend to have lower utility while offering stronger input privacy guarantees (as private data is protected before leaving the data holder) compared to systems that require data holders to disclose their information in an unprotected manner to a trusted curator. *Distributed DP* combines elements of central and local models through intermediate mechanisms like secure aggregation or shuffling, often achieving better utility than local DP while reducing trust requirements compared to global DP.

For more details on the properties, mechanisms and unit of privacy, please refer to the relevant chapters in [71, 118, 103, 119].

Assumptions. One of the assumptions for standard DP is the independence of instances in the dataset. This assumption enables the definition of neighboring datasets – datasets that

differ in the data of only one individual – based on which DP is defined. It also implies that the notion of neighboring datasets is well-defined, whether at the event-level or user-level. DP mechanisms also assume a correct and secure implementation, particularly regarding the unbiased sampling of random noise from appropriate distributions. Another key assumption is the accurate determination of query sensitivity, which is necessary for properly calibrating noise addition. For global DP, there's an additional assumption of a trusted curator who has access to the raw data and correctly applies the differential privacy mechanism.

Privacy Threats and Vulnerabilities. Despite its strong theoretical foundations, DP faces several practical vulnerabilities in real-world applications. When implementations use excessively large privacy budgets to improve utility, privacy guarantees can become meaningfully weaker, as observed in the 2020 U.S. Census deployment [120, 121, 122]. The compositional nature of DP creates vulnerability through privacy budget exhaustion, where multiple queries against the same dataset can rapidly deplete the privacy budget and potentially lead to early termination of access or weakened protections. Systems employing DP remain susceptible to side-channel attacks that target implementation weaknesses rather than the mathematical guarantee itself, including timing attacks, floating point attacks, memory access patterns, or compromised random number generators [123]. For highly correlated datasets, standard DP may provide insufficient protection since the privacy of one individual can be compromised through information about correlated individuals [124, 125].

Limitations. One of the major limitations DP is the inherent trade-off between privacy and utility: achieving stronger privacy (i.e. using a smaller privacy budget ϵ) typically requires adding more noise, which reduces the utility of the analysis. This trade-off is further amplified in other deployments of DP – such as local and distributed DP. DP also offers limited protection for outliers with unique data or under-represented groups, as their information may disproportionately influence outputs even under noise (a phenomenon known as disparate impact). DP faces an inherent privacy budget limitation: after a certain number of queries have been processed, the cumulative privacy loss reaches the predetermined threshold, and

the system must halt further queries or risk violating the promised privacy guarantees. It is often complex to implement DP mechanisms correctly within current computing systems [126].

Apart from these, implementing DP effectively presents challenges in defining privacy appropriately across different contexts, domains, and data modalities, particularly for complex, interconnected datasets. There are also difficulties in communicating and interpreting the notion of DP for non-experts, which can lead to misunderstandings about its practical guarantees. Despite these implementation challenges, this thesis adopts the output privacy framework provided by DP due to its mathematical rigor and proven privacy guarantees.

2.4 DP-in-MPC

In the distributed paradigm, achieving strong privacy guarantees while retaining utility remains a persistent challenge. To address this, multiple PETs are often combined, both in distributed and centralized settings. Several prior works have explored such integrations in distributed contexts, with a primary focus on combining FL (Section 2.1) and DP (Section 2.3).

We propose a paradigm, "DP-in-MPC", that combines MPC and DP to provide both input and output privacy. Formally, DP-in-MPC is a paradigm that enables privacy-preserving AI and data analysis in a distributed setting with multiple data holders, offering provable, strong input privacy guarantees through cryptographic primitives of MPC, and formal output privacy guarantees through differential privacy. This paradigm emulates a centralized setup, thus supporting arbitrary data partitioning and maintaining utility comparable to the centralized paradigm with DP, while removing the need for trust assumptions and eliminating any single point of failure[6].

Figure 2.5 illustrates the working of the proposed paradigm. It primarily consists of: (a) data holders who cannot share their data with others due to privacy constraints and regula-

[6]Henceforth, we refer to the centralized paradigm as one that provides output privacy through differential privacy.

tory restrictions, and (b) computing servers (a.k.a. MPC servers) that act as MPC parties and collaboratively run secure computation protocols over the secret-shares of the combined data provided by the data holders. We refer to such a setup as MPC-as-a-Service where data holders or entities that need to perform analysis on private distributed data outsource secure computations to computing servers. In other words, MPC-as-a-Service assumes an infrastructure setup where the MPC parties are the independent and non-colluding servers (controlled by different entities). This model allows for data from multiple data holders, i.e. the number of data holders can be different from the number of computing servers. These computing servers are equipped with MPC schemes and MPC protocols to perform AI and data analysis. The figure shows an example of this setup with three MPC servers.

The execution of the DP-in-MPC paradigm begins with each data holder secret-sharing their private data with the set of MPC servers. After this step, the data holders can go offline, eliminating the need for synchronization. The MPC servers, upon receiving the secret-shares, first run secure aggregation protocols to combine data from all the data holders – similar to the aggregation step in the centralized paradigm in Figure 1.1. But unlike the centralized setting, no trusted aggregator is needed, since MPC does not require any trusted entity. Crucially, the MPC servers never "see" or access the raw data in unencrypted form at any point in the process. Even during the aggregation phase, each MPC server only works with secret-shares, not the actual data. In this way, DP-in-MPC emulates the centralized paradigm. This is fundamentally different from the centralized paradigm, where data holders must disclose their raw data to a central curator.

Following aggregation, the MPC servers proceed to execute the pre-defined secure MPC protocols for data analysis or machine learning algorithms. In this thesis, we develop such protocols based on algorithms designed for the centralized setting. To optimize performance and preserve privacy, we design MPC-friendly versions of these algorithms ensuring efficiency.

Integrating DP with MPC. A core part of the DP-in-MPC paradigm is the integration of DP directly within MPC protocols. MPC servers jointly generate secret-shares of the noise to be added to provide DP guarantees and securely incorporate it into the computation before

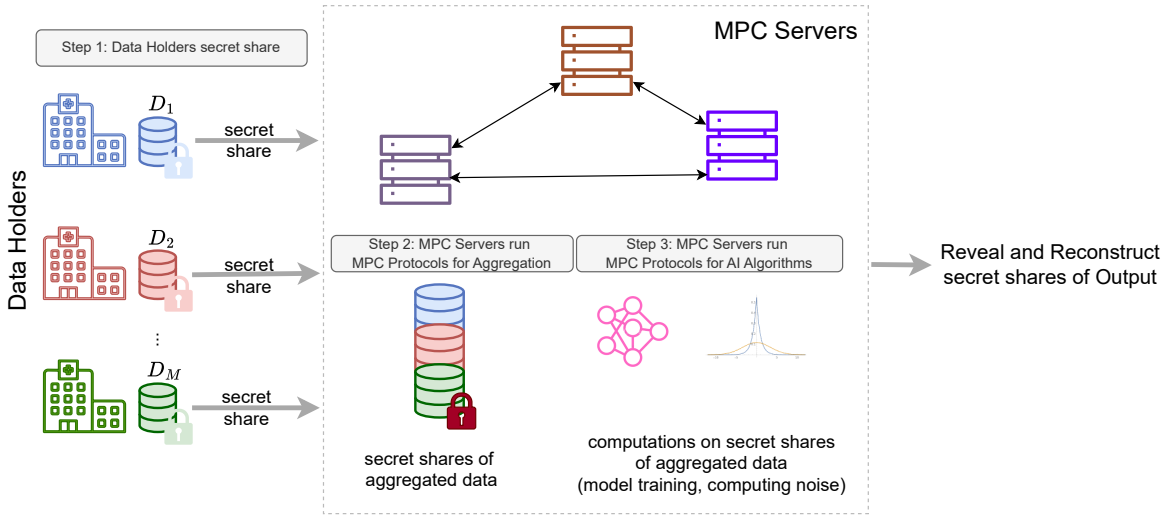


Figure 2.5: **DP-in-MPC**. Data holders send encrypted shares (secret-shares) of their private inputs to MPC servers. MPC servers run MPC protocols to merge secret-shares of data from all data holders to form secret-shares of aggregated data to emulate the centralized paradigm. Then the MPC servers run MPC protocols on the secret-shares of the aggregated data to perform AI and data analysis. Finally, the servers reveal their secret-shares of the outputs — such as the parameters of ML models or other results — which are then combined to reconstruct the final result.

revealing the final output – this design choice, though it introduces additional runtime and communication overhead, is intentional and motivated by both privacy and utility considerations. Furthermore, such an integration enables the application of global DP over the aggregated data from all data holders, similar to the centralized setting, while preserving input privacy. We next briefly mention why other choices for combining DP with MPC might not meets our goals.

A naive approach to combining DP with MPC is to rely on local DP, where each data holder adds noise to their private data prior to secret-sharing [127]. While local DP offers strong privacy guarantees without requiring trust, it typically introduces substantial noise,

especially in high-dimensional or complex data settings, severely degrading utility. This trade-off makes local DP unsuitable for many practical ML and data analysis tasks.

Other approaches that rely on data holders – distributed DP where each data holder contributes a share of the total noise to provide global DP; or when data holders locally generate noise and secret-share it with MPC servers – is often problematic [128]. Such techniques require data holders to honestly generate, calibrate, or contribute noise and be synchronized. Moreover, such approaches are not be suitable to all mechanisms[7].

An appealing alternative might be to delegate the generation of the entire DP noise component to a single MPC server. While this may appear efficient, it reintroduces a point of trust and a single point of failure, going against the reason why MPC is adopted in the first place. If this server is compromised, the DP guarantee is no longer valid.

Another alternative is to let each MPC server independently generate and add noise to the computation. As stated earlier, such a distributed noise generation approach is not suitable for all DP mechanisms. Moreover, this approach might not result in valid global DP and depends on the correctness of each server's noise generation.

The DP-in-MPC paradigm overcomes these limitations by integrating noise generation and addition directly within secure computations using MPC. By jointly generating and applying noise through the MPC protocol, the paradigm eliminates the need to trust any individual party, while enabling global DP guarantees (provided the protocols are correctly designed and implemented). This approach avoids the excessive noise associated with local DP, mitigates synchronization and trust challenges in distributed DP, and removes single points of failure. Furthermore, it supports general DP mechanisms. Though this way of integrating techniques for input privacy (MPC) and output privacy (DP) introduces some computational overhead, we believe it provides a balance between privacy protection, utility preservation, and trust minimization in distributed scenarios.

[7]For example, the sum of multiple Gaussian is a Gaussian, but this property cannot be applied to the Laplace and Exponential mechanism.

An illustrative example. To illustrate how DP-in-MPC works, we consider a simple example involving two MPC servers, S_1 and S_2, that execute an MPC protocol designed to compute a sum and add noise[8].

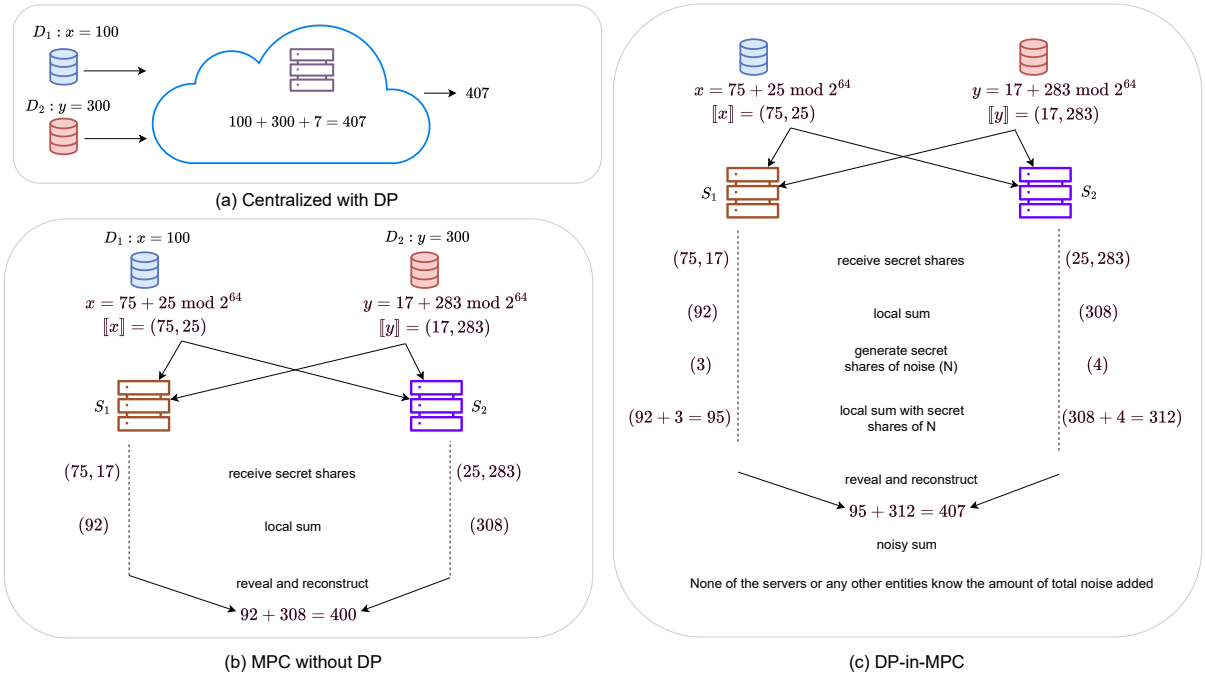


Figure 2.6: **Illustration of how DP-in-MPC works.**(a) Centralized with DP does not provide input privacy, (b) MPC without DP provides input privacy but no output privacy, (c) DP-in-MPC provides input and output privacy with results similar to centralized with DP. Note that all operations are mod 2^{64} in the example; we abstract away details for simplicity.

Suppose we have two private values that need to be summed up: $x = 100$ and $y = 300$, held by 2 data holders in D_1 and D_2 respectively. That is, $D_1 : x = 100$ and $D_2 : y = 300$. Our goal is to compute a DP version of the sum $\mathcal{A}(D_1 \cup D_2) = x+y+N$, where $\mathcal{A}^*(D_1 \cup D_2) = x + y = 400$ is the true result and N is the DP noise to be added. For simplicity, let us

[8]In this illustration, we abstract away details such as modular arithmetic and the exact mechanism of noise generation, and focus on the high-level flow.

assume a sampled noise value $N = 7$, so the final output is 407. Figure 2.6 (a) illustrates a centralized setting, where both x and y are available at an aggregated location. The central entity computes the sum and adds the DP noise to publish the final result. While this approach can ensure output privacy via DP, it requires trust in the aggregator, which is often unrealistic in distributed scenarios.

Using MPC, as shown in Figure 2.6 (b), which operates without trust assumptions, each data holder secret-shares their input. For instance, suppose $x = 100$ is secret-shared as $[\![x]\!] = (75, 25)$ and $y = 300$ is secret-shared as $[\![y]\!] = (17, 283)$. Server S_1 holds shares 75 and -17, and S_2 holds 25 and 283. The servers perform local addition: $S_1 : 75 + 17 = 92$ and $S_2 : 25 + 283 = 308$. When the shares are revealed (i.e. after the secret-shares are aggregated for publishing, also known as reconstruction), it results in the true sum 400. Without the addition of noise, the exact result is revealed, and in this case, individual values can be easily inferred. Thus, while strong input privacy is guaranteed during intermediate computations, output privacy is not preserved in the final result.

In DP-in-MPC, as shown in Figure 2.6 (c), after computing the local sum, the servers jointly generate secret-shares of the DP noise (in this case, $N = 7$). For example, they may generate $[\![N]\!] = (3, 4)$. Note that input privacy is provided even during noise generation i.e. none of the servers or any other entities learn the total value of noise. MPC servers then add noise to their local sums $S_1 : 92 + 3 = 95, S_2 : 308 + 4 = 312$. When revealing, the result is $95 + 312 = 407$, which matches the centralized DP result[9]. As evident, no single party knows the full value of the noise added, and the individual inputs remain private.

This shows how DP-in-MPC emulates the centralized DP setup, but without any trust assumptions, achieving both input privacy (via MPC) and output privacy (via DP). Moreover, since the noise is securely and jointly generated within MPC, it is not reconstructible by any party.

[9]Noise sampling is not deterministic. While, for ease of comparison, in the illustration we assumed that the exact same noise value of 7 was sampled twice, in reality it is of course much more likely to observe (slightly) different noise values.

Note on DP implementation in MPC. Keeping in mind the dangers of implementing DP with floating point arithmetic [129], we stick with the best practice of using fixed-point and integer arithmetic as recommended by, for example, OpenDP[10]. We implement all our DP mechanism using their discrete representations and use 32 bit precision to ensure correctness.

Advantages of DP-in-MPC paradigm. DP-in-MPC is designed to overcome several limitations of existing frameworks for privacy enhancing AI in federated and distributed settings (See Section 1.4.1 in Chapter 1). Through the use of this paradigm, we address multiple key shortcomings:

- Most existing paradigms and frameworks provide either input privacy or output privacy. The centralized paradigm typically offers output privacy but lacks input privacy. In distributed paradigms, FL primarily focuses on input privacy. Even when both types of privacy are addressed in FL, these approaches often suffer from significant utility loss compared to the centralized paradigm. Moreover, the input privacy provided by FL is susceptible to various privacy attacks and vulnerabilities [130, 131].

  Our adoption of MPC ensures *strong input privacy guarantees* when compared to FL in distributed scenarios (as a result of the security guarantees of MPC), while combining it with DP preserves output privacy without compromising utility.

- Achieving utility comparable to the centralized paradigm in distributed settings has been a challenge. A few FL frameworks adopt global DP, assuming a trusted central aggregator – however, this setup has been shown to be vulnerable to privacy attacks. As a result, local or distributed DP is more commonly employed in FL, but this often leads to a significant drop in utility compared to the centralized paradigm. This drop is further exacerbated in cases of non-IID data holders or data distributions in FL.

  We address this issue by employing *global DP to achieve utility similar to the centralized*

[10] `https://opendp.org/`

paradigm irrespective of non-IID data holders, while mitigating the associated privacy risks through the use of MPC. In essence, we replace the trusted central entity with an MPC protocol that operates on untrusted servers, achieving the same utility as in the global DP setting, without requiring data holders to disclose their raw data or model updates to any party. To this end, we develop MPC protocols that run various DP mechanisms, where the MPC parties collaboratively generate and add noise to ensure differential privacy, without revealing any intermediate computations, including the noise values. That is, the MPC parties execute DP mechanisms in the MPC protocol itself. Noise is added before revealing and publishing the results in MPC. This approach effectively emulates a trusted central aggregator, enabling global DP in a distributed setting while providing strong input privacy guarantees.

- Traditional DP and FL frameworks often assume the presence of a trusted or honest-but-curious central entity. This entity often introduces a single point of failure, both in terms of privacy and system robustness.

  In contrast, MPC is designed under the assumption that no single party can be trusted. Our paradigm adopts MPC to eliminate the need for a trusted entity , thereby *removing the single point of failure* and addressing a critical limitation in existing distributed approaches.

- Existing frameworks typically focus on either horizontal or vertical partitioning, with little emphasis on supporting arbitrary partitioning—including horizontal, vertical, or mixed setups (see Figure 1.2)—which are common in many real-world scenarios.

  Our approach, enabled by MPC, *securely aggregates data as if in a centralized setting* (illustrated in Figure 1.1), while preserving privacy. This is achieved through the design of MPC protocols for data aggregation, ensuring that no entity ever sees any data from any data holder in an unencrypted form. This allows our paradigm to support arbitrary data partitioning, as the final computation simulates a centralized view while providing

strong input privacy guarantees.

- Scalability with respect to the number of data holders remains a challenge in both FL and MPC, particularly in MPC due to its high communication and runtime overhead.

  To address this, we systematically use an MPC-as-a-Service model, where a fixed number of computing servers (independent of the number of data holders) act as MPC parties. Data holders secret-share their data with these servers, enabling our system to scale with any number of data holders. The data holders can then go offline, thus removing the need for synchronization as required in traditional MPC and FL setups. As a result, the computational cost depends only on the overall dimension of the combined dataset, similar to the centralized paradigm, rather than on the number of data holders. Of course, the overhead introduced by MPC still depends on the number of computing servers, but this is significantly more efficient than running MPC directly among a large number of data holders. Note that our paradigm is also applicable in settings with a small number of data holders, in which case they themselves can participate as MPC parties.

Assumptions. Our proposed paradigm relies on the assumptions of global DP and the standard security assumptions of MPC. Specifically, we assume that the data holders provide correct inputs i.e. they correctly generate and distribute secret-shares of their data. While data holders must be available and possess sufficient computational capabilities during the secret-sharing phase, they do not need to remain online thereafter. In the MPC-as-a-Service model, we assume that each MPC computing server is operated by an independent entity[11].

Privacy Threats and Vulnerabilities. The privacy vulnerabilities inherent in global DP and MPC carry over to our paradigm as described in Section 2.3 and 2.2 respectively. In addition to these, DP-in-MPC faces deployment-specific risks, particularly in the MPC-as-a-

[11]This aligns with the standard non-collusion assumption in MPC, where security is guaranteed as long as a threshold number of parties do not collude.

Service model. One critical challenge is the unavailability or untrustworthiness of independent computing servers to act as MPC servers. When a sufficient number of independent, non-colluding MPC servers cannot be reliably established or maintained, the privacy guarantees of MPC – and by extension DP-in-MPC – may be compromised. This is especially problematic in distributed or resource-constrained environments, where maintaining trust assumptions (e.g. a majority of honest servers) is more difficult.

Limitations. Apart from the inherent limitations of global DP and MPC, all data holders need to be online during the secret-sharing phase. However, this limitation can be relaxed by adapting the paradigm to accept and store the secret-shares of data holders asynchronously, as and when they become available. This may result in execution delays or necessitate a design where only a fixed number of data holders (e.g. based on a first-come-first-serve policy or data volume) are selected to participate in the computation. Another challenge lies in integrating differential privacy mechanisms into MPC, which requires the development of custom secure protocols. This process is often non-trivial and may limit the applicability of existing off-the-shelf DP tools. In this thesis, we design several such protocols, though not an exhaustive set.

Part I

PRIVACY-PRESERVING TRAINING OF MACHINE LEARNING MODELS

"The journey of a thousand miles must begin with a single step."

Lao Tzu

Chapter 3

PRIVACY-PRESERVING TRAINING OF LOGISTIC REGRESSION ON DISTRIBUTED DATA

Machine learning (ML) models are widely deployed in real-world applications such as in healthcare and finance. However, training accurate and effective ML models often requires access to large amounts of data. In many practical scenarios, relevant training data is fragmented across multiple organizations and remains siloed due to privacy concerns, regulatory constraints, and/or competitive advantage as discussed in Chapter 1.

This necessitates cross-silo federated learning approaches, where ML models can be trained collaboratively across multiple data holders (clients in federated learning) without compromising data privacy. Furthermore, in a federated scenario, data can be distributed in various ways as demonstrated in the note on distributed settings in Chapter 1. An ideal federated learning setup should support any data distribution, enabling privacy-preserving analytics over distributed datasets. This allows collaboration for privacy-preserving model training across organizations and within different departments of the same organization.

The remainder of this chapter is structured as follows. In Section 3.1, we outline the main contributions of the chapter. Section 3.2 provides preliminaries on logistic regression training and the output perturbation technique for achieving Differential Privacy (DP). In Section 3.3, we review related work and Section 3.4 presents our proposed method, detailing MPC protocols for training a logistic regression model with DP. Finally, we evaluate our approach in Section 3.5 followed by summarizing the chapter in Section 3.6.

3.1 Contributions of this Chapter

This chapter addresses Research Question 1 (RQ1) — "How can we train accurate ML models with input and output privacy guarantees over data that is distributed in an arbitrary fashion?" — as motivated in Chapter 1.

The importance of enabling privacy-preserving model training in distributed (also referred to as federated) setups has spurred a large research effort in this domain, most notably in the development and use of PETs, prominently including Federated Learning (FL) [59], Differential Privacy (DP) [117], Secure Multiparty Computation (MPC) [104], and Homomorphic Encryption (HE) [132].

Federated settings employ approaches based on (combinations of) FL, MPC, or HE, allowing data holders to train models without sharing their raw data, thereby ensuring *input privacy*. But these approaches do not guarantee protection of the underlying datasets (such as against membership inference and reconstruction attacks) once the trained model is published, i.e. after federated training, and hence they are often combined with DP. Most studies in federated settings focus on the combination of FL and DP[1] and incur severe utility loss when compared to the centralized setting. Our research aims to mitigate this utility loss by combining MPC and DP in federated settings, effectively emulating global DP from the centralized setting. While existing studies have explored MPC-DP combinations (see Section 3.3), most focus specifically on horizontally federated learning (HFL). In contrast, our proposed approach works with any arbitrary data distribution in a federated setting.

Building upon the DP-in-MPC paradigm introduced in Chapter 2, we present a concrete solution for training differentially private linear models in distributed settings, while meeting the core objectives of the thesis outlined in Section 1.3 of Chapter 1. The technical contributions of this chapter lie in the design and development of secure MPC protocols that integrate differential privacy into linear model training under the DP-in-MPC paradigm. We summarize the contributions below.

[1]Either through local DP or distributed DP; see Section 3.1 in [103].

- We propose a combination of MPC and DP to train linear models in a federated setting that yields ML utility similar to the centralized setting. Our solution can be adapted to generalized linear models [133].
- Our proposed *modular* approach works for any arbitrary distributed data setting.
- We propose a MPC protocol to sample noise from a multidimensional power exponential distribution, generating Laplace-like noise. Unlike recent works that focus on DP-SGD like mechanisms (gradient or objective perturbation), our focus is on output perturbation. We adapt an output perturbation technique from the centralized setting [77][2].
- We demonstrate the effectiveness of our approach on real-world datasets. Our solution obtained the highest accuracy in the iDASH2021 Track III competition on confidential computing, where the challenge was to propose a federated learning algorithm for training of a model to predict the risk of wild-type transthyretin amyloid cardiomyopathy using medical claims data from different hospitals, while providing DP guarantees[3].

3.2 Preliminaries

3.2.1 Training Logistic Regression

Logistic Regression (LR) is a type of linear model widely used for classification tasks. While linear regression predicts continuous outcomes by modeling a linear relationship between input features and output values, an LR model applies a non-linear transformation, specifically sigmoid function[4], to a linear combination of input features to produce probabilities in the range of $]0, 1[$ [134].

Formally, consider a set of labeled training examples $S = \{(\mathbf{x}, t)\}$, where $\mathbf{x}$ is input feature vector of length m and t is the corresponding label with $t \in \{0, 1\}$, then LR models

[2]For our datasets, our results show that output perturbation performs best; and is in general efficient for our scenario.

[3]`http://www.humangenomeprivacy.org/2021/competition-tasks.html`

[4]LR uses the sigmoid function for binary classification and the softmax function for multiclass classification.

and predicts the probability of a binary outcome using the sigmoid function $\sigma(.)$

$$P(t = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) \quad \text{where} \quad \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.1}$$

$\sigma(.)$ transforms the linear predictor $\mathbf{w}^T\mathbf{x}$ into a probability value. The vector $\mathbf{w}$ is referred to as the parameters or coefficients or weights of the LR model. The linear component, $\mathbf{w}^T\mathbf{x}$, is identical to a linear regression model, but logistic regression applies the sigmoid function to produce probabilities. An LR model classifies an input $\mathbf{x}$ as "1" if the predicted probability that $t = 1$ exceeds the predicted probability that $t = 0$.

According to Equation 3.1, this is equivalent to:

$$\frac{\sigma(\mathbf{w}^T\mathbf{x})}{1 - \sigma(\mathbf{w}^T\mathbf{x})} > 1$$

This inequality holds if and only if $\mathbf{w}^T\mathbf{x} > 0$, since the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ is monotonically increasing. Therefore, logistic regression defines a linear decision boundary, and thus corresponds to a *linear classifier*.

Training an LR model involves optimizing the model coefficients $\mathbf{w} \in \mathbb{R}^m$ to minimize the regularized logistic loss over the dataset S. The objective function $L(.)$ constitutes of multiple terms:

(i) the loss function (typically a negative log likelihood or cross-entropy loss) given by

$$l(\mathbf{w}) = -\frac{1}{|S|} \sum_{(\mathbf{x},t)\in S} [t \log(\sigma(\mathbf{w}^T\mathbf{x})) + (1 - t) \log(1 - \sigma(\mathbf{w}^T\mathbf{x}))]$$

(ii) optional regularization term to prevent overfitting[5]. This term includes a regularization constant denoted by Λ. We consider L2 regularization, which has strong convexity properties, and the regularization term is:

$$r(\mathbf{w}) = \frac{\Lambda}{2}\|\mathbf{w}\|_2^2 = \frac{\Lambda}{2} \sum_{j=1}^{m} w_j^2$$

[5]Two common regularization methods are: (a) L1 regularization: adds the sum of absolute values of coefficients as a penalty, which can drive some coefficients to exactly zero, enabling feature selection; (b) L2 regularization: adds the sum of squared coefficient values as a penalty, which shrinks coefficients toward zero but never exactly to zero.

The complete objective function with L2 regularization is thus:

$$L(\mathbf{w}) = l(\mathbf{w}) + r(\mathbf{w})$$

Training is then carried out via iterative optimization algorithms such as full batch gradient descent (GD) that finds the optimal model coefficients by minimizing the objective function, where all training examples are used in every iteration. The weights are updated over a fixed number of iterations n_{iter}, referred to as epochs, using gradient-based updates.

In particular, we compute the gradient $\Delta\mathbf{w}$ of the objective function with respect to the weights, given by[6]:

$$\Delta\mathbf{w} = -\sum_{(\mathbf{x},t)\in S} [t - \sigma(\mathbf{w}^T\mathbf{x})]\mathbf{x} + \Lambda\mathbf{w}$$

To improve convergence, we incorporate momentum and learning rate in the weight update rule:

$$\Delta\mathbf{w} \leftarrow C\Delta\mathbf{w} - \alpha\Delta\mathbf{w} - \Lambda\alpha\mathbf{w}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$$

where α is the learning rate that controls the step size of each update, C is the momentum coefficient that determines the contribution of the current gradient, and Λ is the regularization penalty.

In this chapter, we measure the utility of the LR model using its accuracy on a held-out test data, defined as the proportion of correctly classified test examples, which assesses how the model performs on unseen data[7].

3.2.2 Output Perturbation for Logistic Regression

A classical approach to training a DP LR model is output perturbation proposed in [77, 135]. This method, firstly, trains the LR model using the standard training algorithm (such as full

[6]Note that the constant $\frac{1}{|S|}$ (the size of the dataset) does not appear in the gradient computation here. This is because the optimization is invariant to this constant and logistic regression yields the same weights regardless. Omitting this constant simplifies the computation without affecting the result.

[7]We use accuracy and utility interchangeably in this chapter.

batch gradient descent) to obtain optimal weights of the model, and then adds calibrated noise to the optimal weights to achieve DP.

The weights $\mathbf{w}$ of a trained LR model are perturbed by adding a noise vector η that is sampled according to the density function

$$h(\eta) \propto e^{-\frac{n\epsilon\Lambda}{2}\|\eta\|} \tag{3.2}$$

where n is the number of instances that were used to train the LR model, ϵ is the privacy budget and Λ is the regularization constant used during training.

This technique provides ϵ-DP provided that

(C1) each input sample (i.e., each row's feature vector) has an L2 norm of at most 1; and

(C2) the LR model is trained using L2 regularization.

If **(C1)** and **(C2)** are fulfilled, then the sensitivity of LR with regularization parameter Λ is at most $\frac{2}{n\Lambda}$ [135, 77].

This technique can be formalized by defining a randomized training algorithm $\mathcal{A}$ that that outputs a perturbed model. Specifically, given a dataset S and the optimal model parameters $\mathbf{w}$ trained under L2 regularization, $\mathcal{A}$ returns:

$$\widetilde{\mathbf{w}} = \mathbf{w} + \mathbf{s} \quad \text{where} \quad \mathbf{s} \sim (p(\eta) \propto h(\eta)). \tag{3.3}$$

Here, the noise vector $\mathbf{s}$ is drawn from the distribution defined in Equation (3.2), ensuring that the overall mechanism satisfies ϵ-DP under conditions (C1) and (C2).

3.3 Related Work

The approach that we propose in this chapter preserves input privacy, i.e. it ensures that the training datasets are not exposed (except under ϵ-DP guarantees) to anyone but their original data holders during (1) model training and (2) publication or inference. As we describe below, existing methods either do not fully protect input privacy, or they do so at the cost of higher accuracy loss than our approach.

MPC/HE based Model Training Many cryptography based methods have been proposed for privacy-preserving learning of ML models with data from multiple data holders such as [136, 137]. These include training for linear regression models ([138, 139]), (ensembles of) decision trees ([140, 141, 142, 143]), and neural network architectures ([144, 145, 146, 147]).

These techniques protect input privacy during training while still, in principle, producing the same ML models that one would obtain in-the-clear (i.e. when no encryption is used). The latter is both a blessing, as there is no accuracy loss, and a problem, as upon model publication or during inference, the trained models leak the same kind of information as models trained in-the-clear [148, 149, 150, 151]. Because these methods do not provide DP guarantees, we exclude them from the comparative evaluation presented in Section 3.5.

DP and FL based Model Training Much of the literature on training DP models [76] is developed for the *global* DP (a.k.a. *central* DP) paradigm, which assumes the existence of a trusted curator (aggregator) who collects all the data and then trains a DP model over it, e.g. by adding noise to the gradients or the model coefficients. These methods do not preserve input privacy, since data holders need to disclose their datasets to the aggregator. In a *local* DP approach, privacy loss is controlled by having the data holders add noise to their input data or local models *before* disclosing it to the aggregator such as in [152] who employ local DP using DP-SGD on the client side; which results in substantial utility degradation (Row 4 in Table 4 provides an upper estimate of the accuracy that can be achieved in this way). We eliminate the need for a trusted curator by simulating this entity through MPC protocols that are run either directly by the servers themselves or as in a MPC-as-a-service model introduced in Chapter 2.

Another related existing approach combines Federated Learning (FL) with DP. In FL, each of the data holders participates in model training on their end and only exchange trained model parameters or gradients with the central server [59]. A widely adopted method to provide DP guarantees in a FL setting is through local DP, where the data holders can add noise to protect the values that they send to the central server. In Section 3.5 we

compare with such an approach in which the data holders perturb their model coefficients before sending them to the central server for aggregation. Another alternative to addressing the utility loss associated with local DP is distributed DP, which primarily works with Gaussian-like mechanisms, whereas our approach relies on variants of the Laplace mechanism. Moreover, these approaches work only in the horizontally distributed data setting, while our approach (see Section 3.4) works in the vertically distributed setting as well.

Combinations of MPC and DP The key idea of our proposed approach leverages the DP-in-MPC paradigm where we train DP models while performing as much of the computations as possible in MPC protocols in order to preserve utility. MPC and DP for ML have been well studied in isolation, but the strong privacy protections that can result from their synergy are still being explored [153, 154]. We combine MPC and DP to protect training data privacy during training *and* during inference. Using DP-in-MPC, we simulate the trusted curator present in the centralized DP model. While in the past such an approach was avoided, due to the high computational cost of training the models on top of MPC, we argue that, with advances in protocols and computing power, the higher utility that can be obtained in this way justifies its adoption in several situations. The idea to replace the trusted curator from the global DP paradigm with MPC to get better privacy at the same high utility is gaining traction. Böhler and Kerschbaum for instance have explored this idea for detecting the top k most frequent items across different datasets [155]. They let each party locally compute partial noises which are then combined, which is different from our approach of letting the servers execute an MPC protocol to jointly sample secret-shared noise.

Combining MPC with DP has been proposed in the context of FL, where the data is either *horizontally* distributed (see e.g. [156, 157, 158, 159]) or *vertically* distributed (see e.g. [160]). Existing solutions use cryptographic protocols (not necessarily MPC) and DP, such as in [159, 161, 158, 162, 163], in order to train individual models on the datasets held by the data holders and aggregate these models by averaging their coefficients. These

approaches are designed for horizontally partitioned data. Gu et al. propose a framework for combining MPC and FL, but it only works for the case of horizontally partitioned data [164]. Moreover, each (MPC) server in [164] generates noise using the Gaussian mechanism locally, secret-shares and then adds to the gradient updates, adopting distributed DP which does not achieve accuracy similar to the centralized setup. Choquette-Choo et al. propose a framework that combines HE, MPC and DP inspired by PATE to collaboratively train a model in a way that is applicable only to a horizontal setting [165]. All of the above mentioned solutions do *not* work for vertically partitioned data, unlike our method. Moreover, our solution trains the final model on the union of all the individual datasets, thus essentially obtaining the same utility that is achievable in the trusted curator scenario.

Very few recent solutions combine MPC and DP to work for any arbitrary partition, i.e. a single framework that works for horizontal, vertical, and mixed data distributions. While Das et al.'s solution could in principle work with any arbitrary distribution, their focus is on gradient perturbation, where each MPC party locally generates secret-shares of samples from a Gaussian distribution [154]. In contrast, our solution applies output perturbation and is based on a Laplace-like mechanism[8]. To the best of our knowledge, no solutions exist for training ϵ-DP linear models with output perturbation in arbitrarily partitioned scenarios that achieve accuracy at par with a centralized setup.

3.4 Secure Training of Differentially Private Logistic Regression

3.4.1 Overview

We work in the scenario described in Figure 3.1 distinguishing between the *data holders* who hold the training datasets, and the *computing servers* who run the MPC protocols for model training, noise generation, and noise addition. Our solution works in scenarios in which each

[8][154] follows the concept of distributed DP for Gaussian distributions and does not require joint noise sampling, whereas our approach relies on a Laplace-like mechanism necessitating joint sampling to provide pure DP.

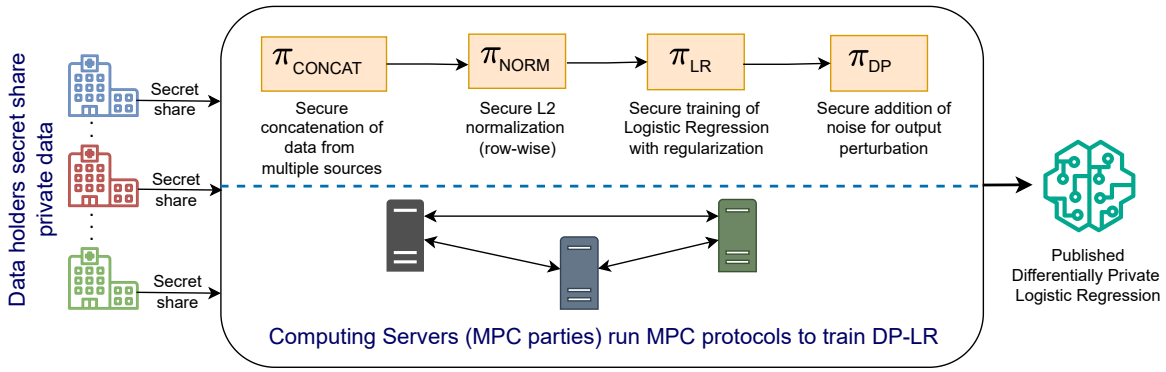


Figure 3.1: **Overview of our modular approach**: Data holders secret-share their private data with computing servers (illustrated here with 3 servers), which train an ϵ-DP model using MPC. Noise is generated and added within MPC itself to ensure DP guarantees, without reliance on a trusted entity. **For linear models**: Step 1: π_{CONCAT} merges secret-shares of data from all data holders to form secret-shares of a single union dataset, thus mimicking a centralized setup. Step 2: π_{NORM} normalizes the secret-shared data, row-wise, to satisfy L2-norm requirements for output perturbation. Step 3: π_{LR} trains an LR model with regularization on normalized secret-shares of data. Step 4: π_{DP} generates secret-shares of noise and adds it to the secret-shares of the weights of the trained model. Finally the DP model is published.

data holder (e.g. hospital or bank) is also an MPC party (i.e. owns the computing server), as well as in scenarios where the data holders outsource the computations to untrusted servers (computing servers) instead (i.e. MPC-as-a-service scenario). Though we demonstrate our solution for 2, 3 and 4 computing servers, the MPC protocols we propose are generic in nature and so our solution works with any number of computing servers as well as data holders. This can be achieved by choosing an appropriate underlying MPC scheme for the desired number of computing servers (which are MPC servers) (see Table 2.1 in Chapter 2). We build upon the cryptographic primitives available through these MPC schemes and use well-known subprotocols for division π_{DIV} of secret-shared values, square root π_{SQRT} of secret-

shared values, and generation of random values from a uniform distribution $\pi_{\mathsf{GR-RANDOM}}$ [110].

The data holders secret-share their data with a set of computing servers. In all MPC protocols used in this chapter, secret-sharings are in $\mathbb{Z}_q$ with $q = 2^\lambda$, i.e. a power 2 ring[9]. Since all computations in MPC are done over integers in $\mathbb{Z}_q$ (see Section 2.2.1), the data holders first convert the real numbers in their data to integers using a fixed-point representation [107] and subsequently split the integer values into secret-shares which are sent to the computing servers (see Figure 3.1).

Next, the computing servers run MPC protocols over the secret-shares of the combined dataset without requiring any entity to disclose its private information to anyone. These MPC protocols train an LR model over secret-shares of the combined dataset, therefore, enabling training over arbitrarily distributed datasets. After training, the computing servers run MPC protocols that generate secret-shares of noise as required and add it to the secret-shares of the model weights as per Equation 3.3. In this way, the MPC protocols effectively play the role of a trusted curator implementing global DP.

The final output of these MPC protocols is the privacy-preserving ML model. The resulting model can be used for private inference (on top of the underlying MPC protocol) or made open to the public because it is protected with ϵ-DP guarantees and preserves privacy. In particular, the computing servers:

1. Jointly run π_{CONCAT} to merge the distributed data.

2. Jointly run MPC protocol π_{LR} to L2 normalize the training data, and to subsequently train an LR model using L2 regularization from the normalized data. At the end of this protocol, the coefficients of the model are secret-shared between the servers.

3. Jointly run MPC protocol π_{DP} to add a noise vector to the secret-shared model coeffi-

[9]In Section 3.5 we present results with $\lambda = 64$ for a varying number of data holders, and for 2, 3, and 4 computing servers.

cients. At the end of this protocol, the noisy coefficients of the model are secret-shared between the servers.

4. Disclose their shares of the LR coefficients so that they can be combined in a final ϵ-DP LR model.

As the noise in step 3 is generated and added to the weights using MPC, the computing servers will not learn it, hence they will not be able to retrieve the actual model coefficients from the noisy coefficients that are disclosed in step 4.

The core of our solution is the MPC protocol π_{DP} that implements the mechanism in Section 3.2.2, specifically Equation 3.3, for providing ϵ-DP.

Note that our proposed method is modular in nature. For example, π_{CONCAT} can be an MPC protocol that simply concatenates and unions the datasets, or runs a secure alignment protocol before securely joining them [166][10]. At the end of π_{CONCAT}, the computing servers hold secret-shares of the merged training examples $[\![S]\!]$.

3.4.2 Protocol π_{LR} for Model Training

At the beginning of the LR training protocol, the computing servers have secret-shares of $S = \{([\![\mathbf{x}]\!], [\![t]\!])\}$. π_{LR} is based on an existing MPC protocol in MP-SPDZ[11] for training an LR with full batch gradient descent (GD) [110]. We extended this protocol in two ways. First, to satisfy condition **(C1)**, before the start of model training, we let the computing servers apply L2 normalization to the secret-shares of each training example by running π_{NORM} resulting in $[\![\mathbf{x}^{\mathbf{norm}}]\!]$. Pseudocode for π_{NORM} is provided separately in Protocol 2 because we also need

[10] In Section 3.5, for simplicity we assume that the records are already aligned in case of vertical partitioning and the computing servers need to simply concatenate them using the right axis. This assumption does not affect the demonstration of our novel contributions, as the results remain valid and π_{CONCAT} itself is not a novel component.

[11] Software to benchmark various MPC protocols in a variety of adversarial settings. `https://github.com/data61/MP-SPDZ/tree/master`

it as a subprotocol for π_{DP}. If the data is horizontally distributed across the data holders, then each data holder can apply sample-wise L2 normalization to their own instances before secret-sharing the training instances with the computing servers. The computing servers in this case can skip the use of π_{NORM} for this purpose, which will reduce the training runtime. Second, to comply with condition **(C2)**, we implemented regularization by changing the weight update rule, already available in MP-SPDZ, to $[\![\Delta \mathbf{w}]\!] \leftarrow C[\![\Delta \mathbf{w}]\!] - \alpha[\![\Delta \mathbf{w}]\!] - \Lambda\alpha[\![\mathbf{w}]\!]$. In this expression, $[\![\mathbf{w}]\!]$ and $[\![\Delta \mathbf{w}]\!]$ are the weights and gradients as maintained in secret-shared form throughout the model training.

Pseudocode for π_{LR} is presented in Protocol 1 which is based on an existing MPC protocol for training an LR with full batch gradient descent [110]. At the beginning of protocol π_{LR}, the computing servers have secret-shares of a set of labeled training examples. To satisfy condition **(C1)**, on Line 1–3 the computing servers first apply L2 normalization to the secret-shares of each training example by running protocol π_{NORM}; pseudocode for π_{NORM} is provided separately in Protocol 2 in this chapter.

The computing servers then begin secure training on the privately L2 normalized data from all the data holders. The training begins with initializing the secret-shares of the weights (coefficients) of the LR model using Glorot uniform initializer [167]. To this end, the computing servers execute protocol π_{INIT} on Line 4. The training is carried out for n_{iter} number of iterations (epochs), which is a public constant agreed upon by all computing servers along with the learning rate α, the regularization penalty Λ, and the momentum C. In each epoch, π_{FWD} for a secure forward pass is called on Line 6, followed by π_{BKWD} for a backward pass on Line 7. The secret-shares of the weights are then updated for every epoch using the existing module for updating the weights. We modified this module to satisfy **(C2)** with L2 regularization as per Line 8 in Protocol 1.

3.4.3 Protocol π_{DP} for Noise Generation

At the end of MPC protocol π_{LR}, the coefficients $\mathbf{w}$ of the trained LR model are secret-shared between the servers. Next, the servers run the MPC protocol π_{DP}, presented in pseudocode in

Protocol 1: π_{LR} for secure logistic regression training

Input : A set $S = \{([\![\mathbf{x}]\!], [\![t]\!])\}$ of secret-shared training examples, each consisting of a secret-shared input feature vector $\mathbf{x}$ of length m and a secret-shared label t; learning rate α; regularization penalty Λ; momentum C; number of iterations n_{iter}.

Output: A secret-shared vector $[\![\mathbf{w}]\!]$ of weights w_i that minimize the sum of squared errors over the training data.

for *training examples* $([\![\mathbf{x}]\!], [\![t]\!])$ *in* S **do**
    $[\![\mathbf{x}]\!] \leftarrow \pi_{\mathsf{NORM}}([\![\mathbf{x}]\!], m)$
end
$[\![\mathbf{w}]\!] \leftarrow \pi_{\mathsf{INIT}}$ ▷ MP-SPDZ module for Glorot uniform initializer
for $i = 1$ ***to*** n_{iter} **do**
    Run π_{FWD} ▷ MP-SPDZ module for forward pass
    Run π_{BKWD} ▷ MP-SPDZ module for backward pass
    Run π_{UPDATE} ▷ Modified MP-SPDZ module for weight updates with the modified update rule for computing $\Delta\mathbf{w}$: $[\![\Delta\mathbf{w}]\!] \leftarrow C[\![\Delta\mathbf{w}]\!] - \alpha[\![\Delta\mathbf{w}]\!] - \Lambda\alpha[\![\mathbf{w}]\!]$
end
return $[\![\mathbf{w}]\!]$

Protocol 3, to generate noise and add it to the model coefficients to provide DP guarantees. Protocol π_{DP} implements the output perturbation method (or sensitivity method) [135, 77] while providing input privacy. While the original output perturbation method relies on the fact that the model coefficients are known or disclosed to a single entity, such as a trusted curator, we do not make such an assumption. Instead, the model coefficients remain secret-shared among the computing servers, neither of which knows the true values of the model coefficients. The challenge is for the computing servers to jointly generate noise that is appropriate for the true model coefficients that they cannot see, without learning the true

Protocol 2: π_{NORM} for secure $L2$ normalization

Input : A secret-shared vector $[\![\mathbf{x}]\!]$ of length d

Output: Secret-shared L2 normalized vector $[\![\mathbf{x^{norm}}]\!]$.

1 Declare vector $[\![\mathbf{x^{norm}}]\!]$ of length d
2 $[\![S]\!] \leftarrow 0$
3 **for** $i = 1$ ***to*** d **do**
4 $\quad [\![S]\!] \leftarrow [\![S]\!] + \pi_{\mathsf{MUL}}([\![x_i]\!], [\![x_i]\!])$
5 **end**
6 $[\![v]\!] = \pi_{\mathsf{DIV}}(1, \pi_{\mathsf{SQRT}}([\![S]\!]))$
7 **for** $i = 1$ ***to*** d **do**
8 $\quad [\![x_i^{norm}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x_i]\!], [\![v]\!])$
9 **end**
10 **return** $[\![\mathbf{x^{norm}}]\!]$

value of the noise. Indeed, no entity should learn the true value of the noise, so that the noisy model coefficients can safely be disclosed at the end of the process (see step 4 in the overview at the beginning of this section), without leaking information that would violate the DP guarantees.

In the output perturbation method, sensitivity is defined using the L2 norm, and the noise vector is sampled from a particular instance of a multidimensional power exponential distribution $h(\eta) \propto e^{-\frac{n\epsilon\Lambda}{2}\|\eta\|}$. Following [168], the computing servers can obtain secret-shares of a vector $\mathbf{s}$ sampled according to the distribution $h(\eta)$, by following these steps, in which d is the length of the vector (i.e. the number of model coefficients):

1. Generate a d-dimensional Gaussian vector $\mathbf{s}$. That is, each coordinate of the vector is sampled from a Gaussian distribution with mean zero and variance one. To this end, Line 1 in Protocol 3 calls π_{GSS} (see pseudocode in Protocol 4) which relies on the transform by [169] to generate samples of the Gaussian unitary distribution, namely

Protocol 3: π_{DP} for secure output perturbation

Input : A secret-shared vector $[\![\mathbf{w}]\!]$ with d model coefficients w_i; regularization penalty Λ; total number n of training examples; privacy budget ϵ.

Output: Secret-shared vector $[\![\widetilde{\mathbf{w}}]\!]$ with perturbed model coefficients.

1 $[\![\mathbf{s}]\!] \leftarrow \pi_{\mathsf{GSS}}(d)$
2 $[\![\mathbf{s}]\!] \leftarrow \pi_{\mathsf{NORM}}([\![\mathbf{s}]\!], d)$
3 $[\![\gamma]\!] \leftarrow [\![0]\!]$
4 **for** $i = 1$ ***to*** d **do**
5 $\quad [\![u]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$
6 $\quad [\![u]\!] \leftarrow -\pi_{\mathsf{LOG}}([\![u]\!])$
7 $\quad [\![\gamma]\!] \leftarrow [\![\gamma]\!] + [\![u]\!]$
8 **end**
9 $c \leftarrow 2/(n \cdot \epsilon \cdot \Lambda)$
10 $[\![\gamma]\!] \leftarrow c[\![\gamma]\!]$
11 Initialize vector $[\![\widetilde{\mathbf{w}}]\!]$ of length d to $[\![\mathbf{0}]\!]$
12 **for** $i = 1$ ***to*** d **do**
13 $\quad [\![s_i]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_i]\!], [\![\gamma]\!])$
14 $\quad [\![\widetilde{w}_i]\!] \leftarrow [\![w_i]\!] + [\![s_i]\!]$
15 **end**
16 **return** $[\![\widetilde{\mathbf{w}}]\!]$

$\lceil d/2 \rceil$ pairs of Gaussian samples. For each pair, on Line 3–4 in Protocol 4, the computing servers securely generate secret-shares of two random numbers u and v uniformly distributed in [0,1] by executing $\pi_{\mathsf{GR-RANDOM}}$. In $\pi_{\mathsf{GR-RANDOM}}$, each server generates l random bits, where l is the fractional precision of the power 2 ring representation of real numbers, and then the servers define the bitwise XOR of these l bits as the binary representation of the random number jointly generated. On Line 5–8 in Proto-

Protocol 4: π_{GSS} for secure sampling of a vector from a Gaussian distribution

Input : Vector length d.

Output: A secret-shared vector $[\![\mathbf{s}]\!]$ of length d sampled from Gaussian distribution with mean 0 and variance 1.

1 Declare vector $[\![\mathbf{s}]\!]$ of length d
2 **for** $i = 0$ ***to*** $d/2$ **do**
3 $\quad [\![u]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$
4 $\quad [\![v]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$
5 $\quad [\![r]\!] \leftarrow \pi_{\mathsf{SQRT}}(-2\pi_{\mathsf{LOG}}([\![u]\!]))$
6 $\quad [\![\theta]\!] \leftarrow 2\pi[\![v]\!]$
7 $\quad [\![s_{2i}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![r]\!], \pi_{\mathsf{COS}}([\![\theta]\!]))$
8 $\quad [\![x_{2i+1}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![r]\!], \pi_{\mathsf{SIN}}([\![\theta]\!]))$
9 **end**
10 **if** *d is odd* **then**
11 $\quad [\![p]\!] \leftarrow \pi_{\mathsf{GSS}}(2)$
12 $\quad [\![s_{d-1}]\!] \leftarrow [\![p_0]\!]$
13 **end**
14 **return** $[\![\mathbf{s}]\!]$

col 4, the servers then jointly compute a secret-sharing of $\sqrt{-2\ln(u)} \cdot \cos(2\pi v)$ and of $\sqrt{-2\ln(u)} \cdot \sin(2\pi v)$ using MPC protocols π_{SQRT}, π_{SIN}, π_{COS}, and π_{LOG} [110]. In case d is odd, one more sample needs to be generated. The servers do so on Line 11–12 in Protocol 4 by executing π_{GSS} to sample a vector of length 2 and only retain the first coordinate.

2. Normalize $\mathbf{s}$, that is divide each coordinate of $\mathbf{s}$ by its L2 norm (Line 2 in Protocol 3). After steps 1-2, the servers have secret-shares of a random d-dimensional vector on the unit sphere (this follows from the spherical symmetry of the multivariate Gaussian

distribution).

3. In this step the computing servers change the magnitude of the vector obtained above to an appropriate value by sampling the gamma distribution $\Gamma(d, \frac{2}{n\epsilon\Lambda})$ to obtain a value γ, and multiplying each coordinate of the normalized vector produced in step 2 with γ. To generate a secret-shared sample $[\![\gamma]\!]$ from the $\Gamma(d, \frac{2}{n\epsilon\Lambda})$ distribution, on Line 3–8 in Protocol 3, the computing servers generate secret-shares of d independent samples from the exponential distribution with rate parameter one (here denoted by Exp(1)) and add them. To generate secret-shares of one such sample we use the inverse transform sampling over MPC, which consists of computing $-\ln u$, where u is a random number with precision equal to l bits generated by the computing servers within the interval $[0, 1]$:

   (a) On Line 5 the servers execute $\pi_{\mathsf{GR-RANDOM}}$ as in Protocol 4 to generate a random number with precision l in $[0, 1]$. Denote this number by u.

   (b) On Line 6 the servers compute secret-shares of $-\ln(u)$.

   Finally, on Line 9–11 the servers scale the sum by multiplying the secret-shares with the factor $\frac{2}{n\epsilon\Lambda}$. On Line 13, they then multiply each coordinate of $\mathbf{s}$ with γ to obtain the appropriate magnitude.

The obtained vector is then added to the vector of model coefficients on Line 14.

The importance of protocol π_{DP} stems from the fact that it enables the computing servers to generate secret-shares of noise, without each server learning the true value of the noise that they add to the model coefficients in Line 14 of Protocol 3. The correctness of the protocol follows from the correctness of the inverse transform sampling algorithm, and the fact that $\text{Exp}(1) = \Gamma(1, 1)$ and that $\sum_{i=1}^{d} \Gamma(1, 1) = \Gamma(d, 1)$. Moreover, it follows from the definition of the Gamma distribution that $c \cdot \Gamma(d, 1) = \Gamma(d, c)$. The security of the whole protocol follows from the security guarantees provided by the cryptographic primitives [110].

Security and Privacy of π_{LR}+π_{DP}. Below we formalize the security guarantees of π_{LR}+π_{DP}. Input privacy: The underlying MPC schemes that we use in our method implement a secure arithmetic black-box MPC. They only perform operations over secret shares, and no information is leaked during the computation over the secret shares. Moreover, we use MPC sub-protocols π_{LOG}, π_{COS}, π_{SIN}, $\pi_{\mathsf{GR-RANDOM}}$, and π_{LR} from MP-SPDZ. All these sub-protocols do not leak any information and are UC-secure. The novel protocols that we propose (π_{DP}, π_{GSS}, and π_{NORM}) therefore do not leak any information to the MPC servers or the data holders, except for the perturbed weights of the LR model. Our protocol $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$, which is a composition of π_{LR} and π_{DP}, UC-securely implements the ideal functionality $\mathcal{F}_{\mathsf{LR+DP}}$ for privacy-preserving training of a differentially private logistic regression model.

Ideal Functionality $\mathcal{F}_{\mathsf{LR+DP}}$

The functionality $\mathcal{F}_{\mathsf{LR+DP}}$ operates as follows:

- Waits to receive input datasets D_i from data holders and joins them to form D.
- Upon joining D, computes model weights $\mathbf{w} := \mathcal{A}^*(D)$ by training a logistic regression model $\mathcal{A}^*$ on D.
- Samples a noise vector $\mathbf{s} \sim (p(\eta) \propto h(\eta))$.
- Computes the privatized output $\tilde{\mathbf{w}} := \mathbf{w} + \mathbf{s}$.
- Reveals $\tilde{\mathbf{w}}$.

Output privacy: The fact that the resulting $\tilde{\mathbf{w}}$ provides $(\epsilon, 0)$-DP follows directly from the proof by [77] regarding the privacy guarantees of the output perturbation method. Although in our approach the training and noise addition procedures are executed within MPC using fixed-point representations (in our case, precision of 32-bit), prior work by [129] has shown that DP mechanisms remain valid under finite precision, even at 32 bits. Therefore, our protocol $\pi_{\mathsf{LR+DP}}$, securely computes $\tilde{\mathbf{w}}$ and satisfies the same $(\epsilon, 0)$-DP guarantees.

Remark: An MPC+DP method for models like logistic regression, as we developed, is extremely valuable. First, while deep learning is state-of-the-art for many applications, GLMs

remain a preferred method in domains such as finance and healthcare, where interpretability and regulatory compliance are essential. Second, logistic regression is often more practical and effective than deep learning in real world problems with data scarcity (e.g. training of AI models for rare diseases). The effectiveness of GLMs is supported, for instance, by our winning results on real-world datasets from the iDASH competition (Table 1) and Bailly et al. (2022) Third, and very relevant to our work, is that GLMs are significantly more efficient than deep learning. While it is technically possible to train larger neural networks in MPC [170], the computation and communication costs when doing so over encrypted data are orders of magnitude higher than for logistic regression. While there are applications where such high costs are justifiable to achieve higher accuracy, there are also many applications where logistic regression performs at par with neural networks, and training a logistic regression model in MPC is a far more economical and practical solution.

3.5 Empirical Evaluation

3.5.1 iDASH Competition Results

We submitted our approach to a competition hosted by a National Center for Biomedical Computing funded by the NIH. In Track III of the iDASH 2021 competition, participants were invited to submit solutions for learning a ML model from training data hosted by two virtual centers, while providing DP guarantees. The centers represent data holders who have medical records of respectively 831 patients and 882 patients. Both datasets have the same schema, consisting of 1,874 boolean input attributes and a boolean target variable. The goal is to train a classifier for diagnosis of transthyretin amyloid cardiomyopathy using medical claims data [171]. Solutions submitted to the competition were required to run on two machines. They were evaluated in terms of (1) training runtime on two nodes with Intel Xeon E3-1280 v5 processors (4 physical cores, hyper-threading enabled) and 64 GiB memory; (2) accuracy on a held-out test of 429 patients.

Table 3.1 contains the results for the best performing teams satisfying the ϵ-DP require-

ment (with ϵ set as 3 by the organizers). The first row corresponds to the approach presented in Section 3.4. We implemented the π_{LR} and π_{DP} protocols in MP-SPDZ, an open source framework for MPC [110]. As the underlying MPC scheme for the iDASH2021 competition, we used semi2k (a semi-honest adaptation of [1]) with mixed circuits that employ techniques using secret random bits (extended doubly-authenticated bits; edaBits) [172]. This MPC scheme enables secure 2PC against semi-honest adversaries and complied with the requirements of the competition. As the regularizer for LR training, we used $r(\mathbf{w}) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w}$, in which $\mathbf{w}$ denotes the vector of weights (coefficients) of the LR model, i.e. we used $\Lambda = 1$.

Table 3.1: Results for ϵ-DP with $\epsilon = 3$ and data from two data holders, as provided by the iDASH2021 competition organizers. π_{LR}+π_{DP} operates in 2PC passive (semi-honest, dishonest majority with a corruption threshold of 0).

	Approach	PETs	Accuracy	Runtime
1.	π_{LR}+π_{DP} (Section 3.4)	MPC & DP	86.25%	$\sim$ 15,000 sec
2.	feat. sel. and LR ensemble	DP	85.31%	31.942 sec
3.	baseline (Section 3.5.1)	DP	84.85%	0.27 sec
4.	decision tree based	DP	84.38%	0.09 sec

All methods in Table 3.1 provide ϵ-DP guarantees. The differences among the methods are in the utility (accuracy) and in the time taken to train a DP model. Our π_{LR} +π_{DP} approach achieved the highest accuracy of all methods, while taking the longest time to complete. Indeed, the runtime for the π_{LR} +π_{DP} approach is orders of magnitude larger than the runtimes for the other methods. This is because the π_{LR} +π_{DP} approach is the only method in Table 3.1 that uses MPC, while the other methods do not rely on cryptographic techniques. Approach 2 was based on feature selection and training an ensemble of LR

models on selected feature subsets, while approach 4 was based on training a decision tree in a DP manner; these approaches were not created by us, and, to the best of our knowledge, their description has not been published in the open literature. In addition to the method from Section 3.4 we submitted an MPC-free baseline method to iDASH2021. We describe this method, which corresponds to approach 3 in Table 3.1, below as we also use it for further analysis and comparison in Section 3.5.2.

Baseline Method. The baseline technique follows a FL setup with horizontally distributed data in which each data holder locally trains a model on their data and adds noise to the model parameters at their end. Each data holder then shares its noisy parameters with a central server who performs averaging of the noisy model parameters and sends the result to the data holders. At the end of this process, each data holder holds the aggregated trained model. In more detail, in the baseline technique, each data holder:

1. Applies L2 normalization to its own instances;
2. Trains an LR model on its normalized instances;[12]
3. Adds noise to the trained LR coefficients as per the output perturbation method [77].

After going through steps 1-3, the data holders can each publish their perturbed LR coefficients, which we subsequently average to create a final model. Because steps 1–3 provide ϵ-DP [77], and since the datasets do not have common entries (a case of parallel composition), the overall solution provides ϵ-DP due to the post-processing property of differential privacy.

3.5.2 Utility

Horizontally and Vertically Distributed Data. For the results in Table 3.2 we distributed the data evenly among different numbers of data holders, both horizontally and vertically. We assume that the record alignment for vertical partitioning is already done us-

[12]We used the LR implementation from sklearn for this with penalty='l2' (L2 regularization) and $C = 1$ (the inverse of Λ).

ing privacy-preserving techniques as in [173] prior to the start of the training. The baseline technique is only applicable when the data is horizontally distributed, while the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach works in the vertically distributed scenario as well. Even in the horizontally distributed scenario, the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach is preferable because it yields a higher accuracy, which becomes even more evident when the data is distributed among multiple data holders. The accuracy of the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach is independent of the number of data holders and the partitioning of data, as regardless of the partitioning, the computing servers still train a model over all the training data with π_{LR} and subsequently add noise *once* to the globally trained model coefficients with π_{DP}, effectively simulating the global DP paradigm but without the involvement of a trusted curator.

Table 3.2: 5-fold CV accuracy results for varying number of data holders for ϵ-DP with $\epsilon = 1$.

	horizontally distributed		**vertically distributed**	
# data holders	baseline	$\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$	baseline	$\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$
2	85.79%	87.98%	–	87.98%
4	83.36%	87.98%	–	87.98%
8	76.92%	87.98%	–	87.98%

The baseline technique on the other hand adheres to the local DP paradigm in which each data holder adds noise to its local model, resulting in more noise in the final aggregated model. Furthermore, the utility of the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach is independent of the number of instances and/or features owned by each individual data holder, while the accuracy of the baseline technique degrades when individual data holders do not have sufficient instances to train local models that generalize well. This is especially relevant in biomedical applications that are characterized by high-dimensional datasets with relatively few instances.

Effect of Privacy Budget ϵ on Accuracy of Models Trained with $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$. Table 3.3 shows the effect of the privacy budget ϵ on the accuracy of models trained with the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach. The accuracy is measured with 5-fold CV and over 3 iterations over the train and test data from Section 3.5.2. The training is done for 1000 epochs and $\Lambda = 1$. The results are as expected, with a larger privacy budget – i.e. less stringent privacy requirements – yielding more accurate models.

Table 3.3: Accuracy of models trained with $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ for different values of ϵ for 1-fold

ϵ	0.01	0.01	0.1	0.5	1	INF
Averaged over 5 folds	54.81%	80.66%	62.57%	89.18%	89.39%	89.47%

Comparison with Other Perturbation Techniques. For the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach (Section 3.4) and the baseline technique (Section 3.5.1), we adopted the sensitivity method that perturbs the model coefficients, i.e. the output perturbation method that was proposed as Algorithm 1 in [77]. In Table 3.4 we compare the output perturbation technique with other perturbation techniques, namely objective perturbation and gradient perturbation. For objective perturbation, we ran experiments with Algorithm 2 from [77] that adds noise to the objective function itself[13]. For gradient perturbation, we ran experiments with DP-SGD [76] that adds noise to the gradients[14]. For DP-SGD, we computed the required noise multiplier for given $\epsilon = 1, \delta = 1e-5$, batch size of 1, 300 epochs, and the number of training examples each data holder holds. This was then passed as an argument to DP-SDG optimizer along

[13]We implemented this approach using IBM's Diffprivlib library `https://github.com/IBM/differential-privacy-library`.

[14]We implemented this approach using TF-Privacy `https://www.tensorflow.org/responsible_ai/privacy/`.

with a clipping threshold of 1, learning rate of 0.1, and number of micro batches equal to the batch size.

Table 3.4: Accuracy results obtained with 5-fold CV for ϵ-DP with $\epsilon = 1$ and 2 data holders

	Perturbation	Accuracy	
Our Approach	output	$\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ (Section 3.4)	87.98%
		baseline (Section 3.5.1)	85.79%
Other Approaches	objective	baseline-OP	49.40%
	gradient	baseline-DPSGD	69.77%
	sample-wise DP	randomized response	50%

In the baseline-OP method in Table 3.4, each data holder trains a differentially private LR model locally by perturbing the objective function. The resultant coefficients of the local models are then averaged, resulting in a final DP model. The baseline-DPSGD method is entirely similar, but in this method each data holder trains a differentially private LR model by perturbing the gradients learned during training, i.e. with DP-SGD.

As can be seen in Table 3.4, contrary to what one would expect based on the analysis in [77], the accuracy results with this objective function perturbation method were not good on the iDASH2021 data, and far worse than those with the output perturbation method. We attribute this to the high-dimensional nature of the iDASH2021 data (many features and relatively few instances) which is very different from the datasets used for evaluation in [77]. Similarly, the LR models trained with DP-SGD on the iDASH2021 data are significantly less accurate than those protected with output perturbation.

We also included a baseline sample-wise DP to demonstrate the naive local DP approach. For the IDASH2021 competition dataset, since the features are binary, we used randomized response as a sample-wise DP mechanism. We studied this in a setting with

two data holders and a central entity that receives perturbed data from the data holders and trains a LR model on it. For the IDASH dataset, it is reasonable to assume that the samples correspond to different individuals and are therefore independent. So we applied parallel composition across samples (rows) and sequential composition across features. We performed 5-fold cross-validation, conducting 10 runs of randomized response per fold. The total privacy budget of $\epsilon = 1$ was distributed across features resulting in a per-value privacy budget of $\epsilon_v = 0.000533$ for each value. Our results show, that in the absence of MPC and with data holders applying sample-wise DP, the average accuracy of the model is 50.6% with $\sigma = 0.0083$ (the lowest among all approaches and the same as random guessing).

Comparison with Other Methods on Horizontally Partitioned Data. We evaluate our MPC+DP approach and compare against existing literature ([159] and [158]) that adopts a combination of PETs to train LR models and provide DP guarantees with the output perturbation technique[15]. The main distinction with our method, is that – similar as in the Baseline method we adopted in Section 3.5 – these existing approaches let each data holder train a model locally and then add noise to the averaged model parameters using MPC+DP techniques. Because each data holder is required to train a model locally, these existing methods only work in scenarios where the data is horizontally partitioned, unlike our method which is suitable for vertically partitioned scenarios as well. We also note that the amount of noise added by each technique is different.

For the results in Table 3.5 we distributed the data evenly among different numbers of data holders, in a horizontal manner. We report 5-fold CV accuracy results averaged for 100 runs of noise generation mechanism to consider the randomness in the noise generation. We observe that for 2 data holders, all the techniques have close performance in terms of accuracy. Similar as for the Baseline method in Section 3.5, the accuracy of the models trained by existing methods drops with an increase in the number of data holders. This may be because in existing approach, LR models are trained locally by the data holders, while our approach

[15] `https://github.com/bargavj/distributedMachineLearning`

Table 3.5: Accuracy results for output perturbation obtained with 5-fold CV for ϵ-DP with $\epsilon = 1$ on horizontally partitioned data

Data Holders	Privacy Technique	Accuracy
2	our approach ($\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$, Section 3.4)	87.98%
	Pathak et al.[159]	86.43%
	Jayaraman et al.[158] - MPC Grad P	86.42%
4	our approach ($\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$, Section 3.4)	87.98%
	Pathak et al.[159]	85.02%
	Jayaraman et al.[158] - MPC Grad P	85.10%
8	our approach ($\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$, Section 3.4)	87.98%
	Pathak et al.[159]	84.10%
	Jayaraman et al.[158] - MPC Grad P	84.24%

benefits from training an LR model on the combined data and learns a more generalized model. Moreover, our techniques are independent of how the data is distributed among data holders, unlike the methods in Table 3.5 that work only for horizontally distributed data.

All the experiments in Section 3.5.2 were run with 3PC passive with corruption threshold of 1.

3.5.3 Computational Efficiency

As Table 3.6 shows, the number of computing servers, the corruption threshold, and respective MPC schemes do have a substantial effect on the training time. The experiments for Table 3.6 demonstrate that our protocols are generic and can be used for a range of threat models. These experiments were run with the same training data as in Table 3.1 on co-located F32s V2 Azure virtual machines each of which contains 32 cores, 64 GiB of memory,

and network bandwidth of upto 14 Gb/s. Every computing party ran on a separate VM instance (connected with a Gigabit Ethernet network). The times reported include computing as well as communication times. The training was run for 1000 epochs. with $\epsilon = 1$, $\Lambda = 1$ and with edaBits for mixed circuit computations. The runtimes below correspond to the entire runtime of the MPC protocols, i.e. both the so-called offline and online phases. The offline phase includes any pre-processing required to begin executing the MPC protocol (such as the generation of the correlated randomness that is needed for the secure multiplication) and is independent of the specific input values; the online phase is where the MPC protocol executes on the specific inputs.

In the horizontally distributed case, the data holders can L2-normalize their instances locally while in the vertically partitioned case the computing servers need to run MPC protocol π_{NORM}; this accounts for the difference in runtime between the horizontal and vertical partitioning. As expected, the corruption threshold has the most effect on the run time. Protocols that are secure for an honest majority of players (the protocols presented in [3], and [4]) are much faster than protocols secure against a dishonest majority [1]. For the same corruption threshold, protocols secure against passive adversaries are faster than protocols secure against active adversaries. The four party protocol proposed in [4] manages to obtain good run times for the case of active adversaries by further reducing the corruption threshold to 25%, i.e. one player out of four can be corrupted by an adversary and the protocol is still secure.

Our results show that MPC implementations for honest majority in the case of realistic sized datasets for genetic studies (a few hundred patients, and a few thousand features) are practical. We can train such models and add DP guarantees on top of MPC in less than 1.3 min for the case of honest majority protocols with passive security. Even in the case of stronger adversarial models, the training can be finished in a few hours, which is still practical for many applications where the increased accuracy payoff is valuable, especially with data that is distributed across multiple holders (Table 3.2).

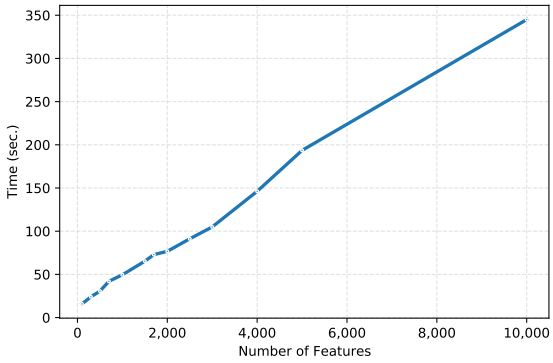
Table 3.6: Runtimes of π_{LR} +π_{DP} for different number r of computing servers.

r	Security	#Corruptions	Horizontally distributed	Vertically distributed	MPC scheme
2	Passive	1	35687 sec	38056.92 sec	Cramer et al.[1]
3	Passive	1	75.83 sec	454.83 sec	Araki et al.[3]
3	Active	1	500.28 sec	1649.07 sec	Dalskov et al.[4]
4	Active	1	160.50 sec	838.02 sec	Dalskov et al.[4]

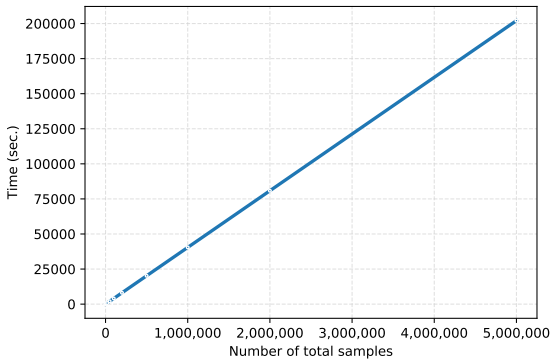
Scalability of π_{LR}+π_{DP} with Number of Computing Servers. The number of data holders in our solution is distinct from the number of computing servers. Our solution is general and works with any number of computing servers as well as data holders.

To study scalability of our approach, we generated synthetic datasets of varying dimensions as per [174]. We varied both the number of rows (#samples ranging from 100 to 5,000,000) and columns (#features ranging from 100 to 10,000). The runtimes of our approach do not depend on the number of data holders, since our approach runs MPC protocols on the secret-shares of the joined dataset and the use of the MPC-as-a-Service model decouples scalability from the number of data holders. Figure 3.2 shows that the runtimes scale linearly with the total dataset dimensions (i.e. the size of S), which is in line with the literature. We ran our MPC protocols for 3PC passive threat model to train LR for 100 epochs with batch size of 32, $\epsilon = 1$ on 2.6 GHz 6-Core Intel Core i7 and latest version of MP-SDPZ. The results carry over to other threat models as per literature [170]. The plots imply that our approach can be used in practice for smaller datasets and is feasible for larger datasets but at a high computational cost with strong privacy guarantees.

In Table 3.7, we report the runtimes and communication overheads to train an LR model with a varying number of computing servers r ranging from 3 to 7. To have a comparison of runtimes and communication overhead for different values of r, we use the same MPC



(a) Total number of samples is a constant of 200 with varying number of features from 100 to 10,000.



(b) Total number of features is a constant of 100 with varying number of samples from 100 to 5,000,000.

Figure 3.2: **Scalability Results**. The plots show runtime results for 3PC passive threat model with varying number of rows and columns of the combined dataset. Our approach is independent of number of data holders and depends on the dimensionality of the dataset only similar to centralized setting.

scheme for each security setting. The chosen MPC schemes can be used with any value of $r > 2$, and are different from the schemes that we use in Table 3.6 which were specific and efficient schemes for the given value of r. It is due to this use of different schemes that we observe a huge difference in runtimes when compared to the runtimes reported earlier. The schemes in Table 3.7 are run with secret-sharings in $\mathbb{Z}_q$ where q is a prime number[16] and with edaBits for mixed circuit computations.

The training was run on the complete training dataset from iDASH2021 consisting of 1713 training samples and 1874 features for 1000 epochs with GD, $\epsilon = 1$, and $\Lambda = 1$. The runtimes reported include computing as well as communication times. The total amount of data sent by all the computing servers is shown in the last column. The runtimes and the

[16]Defaults to None in MP-SPDZ and can be a maximum of bit length 256.

Table 3.7: Runtimes of $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ for different number r of computing servers

MPC scheme	r	#Corruptions	Runtimes	Comm. overhead
Goyal et al.[175] (Passive)	3	1	954.91 sec	57922.70 MB
	4	1	1022.16 sec	83667.90 MB
	5	2	2725.58 sec	366679.00 MB
	7	3	5064.27 sec	711226.33 MB
Cramer et al.[106] & Chida et al.[176] (Active)	3	0	21213.21 sec	5247186.89 MB
	4	1	23244.34 sec	7248822.06 MB
	5	1	68176.24 sec	25728263.60 MB
	7	2	131391.00 sec	70080327.38 MB

communication overhead increase with an increasing number of computing servers. This is because each server now needs to communicate with a higher number of servers servers, and the runtimes include communication times. Also, the active security settings take longer runtimes than their passive counterparts for a given r. These results are in line with the literature in MPC. The communication overhead in settings with a larger number of computing servers can be reduced with the use of a bulletin board functionality that enables efficient communication among many servers who are simultaneously involved in computations [139].

3.5.4 Experiments on other datasets

We further evaluate our approach on the BC-TCGA and GSE2034 datasets of the iDASH 2019 competition[17]. Both datasets contain gene expression data from breast cancer patients which are normal tissue/non-recurrence samples (negative) or breast cancer tissue/recurrence

[17] http://www.humangenomeprivacy.org/2019/competition-tasks.html

tumor samples (positive) [177]. We perform experiments with a 5-fold CV, where the training data is distributed between 2 data holders in each fold.

GSE2034. Each instance in this train dataset is characterized by 12,634 continuous input attributes and a boolean target variable. There are 895 instances in total. In each iteration of the 5-fold CV, each data holder owns 447-448 instances, 20% of which is held out for testing.

BC-TCGA. Each instance in this train dataset is characterized by 17,814 continuous input attributes and a boolean target variable. There are 1,875 instances in total. In each iteration of the 5-fold CV, each data holder owns 937-938 instances, 20% of which are held out for testing.

The secure training is run for 20 epochs for the BC-TCGA dataset and 300 epochs for the GSE2034 dataset with $\Lambda = 1$ and $\epsilon = 1$. Table 3.8 shows accuracy results obtained with a 5-fold CV. To appreciate the inherent difference in difficulty between the GSE2034 and the BC-TCGA classification tasks, as the first row of results in Table 3.8 we include the accuracies obtained with a model trained in the central learning paradigm, i.e. when all the training data resides with a single data holder, and no noise is added to the model coefficients, i.e. ϵ =INF. The other rows correspond to the federated setup from Section 3.5 with 2 data holders. The results are in line with the observation from Section 3.5 that the $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ approach provides higher utility in general, when compared to other approaches. BASELINE-OP and BASELINE-DPSGD perform very poorly on these datasets. [159] and [158] perform similarly to our approach on GSE2034 dataset but our approach outperforms all baselines for BC-TCGA. The difference in relative performance between the datasets likely relates to the distribution of the data and feature characteristics between GSE2034 and BC-TCGA, where BC-TCGA seems to be an easier classification problem overall. The results in Table 3.8 results demonstrate that our method consistently performs same or better than existing baselines.

We additionally report the runtime to train the model using $\pi_{\mathsf{LR}} + \pi_{\mathsf{DP}}$ for these datasets

Table 3.8: Accuracy averaged over 5-fold CV with $\Lambda = 1$, $\epsilon = 1$

	GSE2034	BC-TCGA
# INSTANCES n	895	1,875
# FEATURES d	12,634	17,814
CENTRAL LEARNING; 1 DATA HOLDER	65.55%	98.28%
CENTRAL LEARNING + DP; 1 DATA HOLDER	64.70%	95.83%
BASELINE (SECTION 3.5.1); 2 DATA HOLDERS	51.92%	91.37%
PATHAK ET AL. [159]; 2 DATA HOLDERS	64.55%	92.18%
JAYARAMAN ET AL.[158]; 2 DATA HOLDERS	64.55%	92.22%
BASELINE-OP; 2 DATA HOLDERS	52.72%	40.78%
BASELINE-DPSGD; 2 DATA HOLDERS	47.87%	67.10%
π_{LR}+π_{DP} (SECTION 3.4); 2 DATA HOLDERS	64.55%	95.69%
RUNTIME FOR π_{LR}+π_{DP}; PASSIVE 3PC	276.38 SEC	57.30 SEC

to illustrate the variability in runtimes with respect to the number of training samples, epochs and a number of features in the dataset. We see an increase in runtimes for per epoch when compared to the runtimes per epoch on iDASH, which is attributed to a large number of features (about 10x of iDASH2021 for BC-TCGA and 7x for GSE2034). The runtimes for other threat models will follow a similar trend. We see that for larger datasets like these it is still practical to maintain the utility of the model while providing both input and output privacy guarantees.

Implementation of DP in MPC Protocols. Our MPC protocols operate on fixed-point notation thus following the privacy-conscious choice, taken, for instance, by the OpenDP

project[18]. The accuracy of the model is another point where the precision of weights could affect the overall result. Keeping this in mind, we used 32 bits of precision, which is more than sufficient to ensure the correct behavior of the training procedure. We would like to stress that the finite precision issue is inherent to any implementation of DP on a digital computer, and it is not specific to our work on DP implemented by MPC protocols. DP theory was created, for the most part, based on continuous distributions. However, all the practical libraries implement DP using finite precision arithmetic. That includes, for example, all the implementations of DP-SGD (which is based on the Gaussian mechanism). Das et al. adopt the discrete distributed Gaussian mechanism following the properties of a Gaussian distribution [154]. Proposing a discrete version of the multidmensional power exponential distribution we use in this chapter is research in itself. We further note that our proposed approach is modular in nature and π_{GSS} can be replaced by the appropriate sampling protocols for discrete distributions.

It is legitimate to wonder if security guarantees break down in the case when continuous DP mechanisms are implemented on digital computers. However, that question, which has to be asked of all implementations of DP mechanisms based on continuous distributions, is outside the scope of this thesis.

3.6 Summary

We proposed a modular approach to train privacy-preserving linear models in a federated setting. To this end, we employed the key approach of our thesis, DP-in-MPC, which combines MPC and DP in a way that effectively offers the advantages of global DP but without the involvement of a trusted curator. We developed MPC protocols to generate calibrated random noise to provide DP guarantees.

Our approach makes no assumptions about the data partitioning scenario, the number of computing servers or data holders, or the security setting in which it is applied. On the basis

[18] `https://opendp.org/`

of linearity, π_{LR} is interchangeable with all linear learners without requiring reevaluation of noise variance. Our solution based on this approach led to 1st place in Track III of the iDASH 2021 Genome Privacy competition.

The trade-off between our DP-in-MPC approach that provides global DP and the baseline federated method with local DP can be summarized as operating cost (or running time) versus model accuracy. We empirically demonstrated the added utility of collaborative learning with MPC over the standard federated approach. The effect is particularly apparent as the number of disjoint collaborators grows. We also note that the baseline method as well as the existing methods that combine MPC with DP in FL, cannot be applied in cases where data is vertically partitioned which is a commonly-found scenario in medicine and advertising. In contrast, our DP-in-MPC approach enables collaboration across a larger space of applications.

Based on performance results, our protocol is extensible to larger datasets while remaining within a realistic time span for model training. To further improve upon accuracy, a probable research direction is to introduce MPC protocols for feature selection [178] in both horizontal and vertical partitioning schemes.

Part II

PRIVACY-PRESERVING BIAS MITIGATION IN MACHINE LEARNING MODELS

"When we identify where our privilege intersects with somebody else's oppression, we'll find our opportunities to make real change."

Ijeoma Oluo

Chapter 4

ALGORITHMIC FAIRNESS

In 2019, Jamie Heinemeier Hansson applied for an Apple Card and received a significantly lower credit limit than her husband, despite having a better credit score and sharing the same assets. When the couple reached out to Apple's customer care, the response was simply, "It's the algorithm."[1]. This incident sparked widespread concerns around algorithmic discrimination and gender bias in automated (AI) decision-making systems [32].

Jamie's case is just one of many – it reflects a larger, ongoing issue. In 2016, a landmark ProPublica investigation revealed that COMPAS[2], a risk assessment tool used across the United States to predict recidivism, showed significant racial discrimination – black defendants were far more likely to be incorrectly labeled as high risk compared to white defendants [36].

The healthcare domain is not exempt from such bias either. A widely used algorithm used to allocate extra care was found to significantly underestimate the needs of Black patients, cutting their access to care by over half due to biased reliance on healthcare costs as a proxy for health status [179].

Bias also shows up in the workplace. Amazon's AI recruiting tool was found to be systematically biased against women, as it was trained on historical hiring data that reflected male-dominated recruitment patterns[3] [180].

The list goes on. Bias in AI extends beyond these critical systems in finance, criminal justice, healthcare, and employment to everyday AI-based services. Google Photos' image

[1] `https://dhh.dk/2019/about-the-apple-card.html`

[2] COMPAS: Correctional Offender Management Profiling for Alternative Sanctions

[3] Amazon later abandoned this tool after recognizing the problem couldn't be easily fixed.

recognition algorithm incorrectly labeled photos of Black people as "gorillas" [181, 182]. Facial recognition systems have consistently shown higher error rates for women and people with darker skin tones [34].

With advancements in generative AI (GenAI) such as ChatGPT and DALL-E, issues of bias and discrimination remain pervasive despite these models being trained on large volumes of data (see references in [183, 184, 185]).

These cases make one thing clear – (AI-based) automated systems can, and do, exhibit harmful biases[4] , even without being explicitly programmed to do so. This phenomenon is known as algorithmic discrimination, when automated systems treat individuals or groups unfairly based on characteristics such as race, gender, age, or socioeconomic status (also referred to as sensitive or protected attributes). Algorithmic discrimination can emerge in automated systems due to complex interactions between historical data and biases, design choices and deployment contexts, thus amplifying and perpetuating risks and biases further.

In previous chapters, we discussed the critical importance of privacy in AI systems; we made the case that preserving individuals' privacy and enabling secure collaborative AI in distributed environments are essential to building trustworthy AI. However, privacy is just one pillar of trust.

The cases mentioned above prompt us to broaden our concerns: from simply asking "Is my privacy preserved?" to also asking "Is the system fair to me?". Building trustworthy AI systems requires not just privacy, but fairness too. Even a perfectly privacy-preserving system can still cause harm if it produces unfair, biased, or discriminatory outcomes. While privacy ensures that sensitive data is protected and not misused, fairness ensures that the AI model's decisions do not discriminate against individuals or groups.

In this part of the thesis, i.e. Chapters 5, 6 and 7, we will build AI systems that address both pillars of trust – privacy and fairness. Specifically, in this part, we will address Research

[4]More examples of bias in AI can be found in the keynote by K. Crawford at NeurIPS 2017 `https://www.youtube.com/watch?v=fMym_BKWQzk` and the documentation of the FairLearn software library `https://fairlearn.org/v0.12/user_guide/fairness_in_machine_learning.html#types-of-harms`

Question 2 (RQ2) – "How can we achieve fairness with pre-processing or post-processing bias mitigation techniques when training privacy-preserving ML models in a distributed setting?" To set the stage, in this chapter, we provide a concise overview of *algorithmic fairness* in Section 4.1 and briefly describe the bias mitigation techniques in Section 4.2.

4.1 Introduction to Algorithmic Fairness

AI systems learn patterns from data, and this data often reflects historical, societal, and institutional biases. These biases may source from underrepresentation, stereotypes, or discrimination embedded in society. When AI model are trained on such data, they can amplify these biases.

It has been shown that simply curating the data does not resolve this issue [186]. This is because biases can arise at any stage of the AI pipeline – from data collection and human annotation to model design, development and evaluation to model deployment strategies [187, 188, 189, 190]. Even well-intentioned models trained on seemingly neutral data can produce unfair outcomes due to implicit assumptions in the model architecture, optimization objectives, or evaluation metrics.

Before we dive into algorithmic fairness, we must first clarify what we mean by bias[5]. In the context of this thesis, and AI in general, bias refers to systematic and unfair discrimination against certain individuals or groups, leading to unequal treatment or outcomes. This can occur along dimensions such as race, gender, socioeconomic status, age, or geography. Algorithmic discrimination occurs when AI systems (or any automated systems)[6] exhibit biased outcomes based on sensitive attributes. Note that this discrimination can happen even when the sensitive attributes are not explicitly included as input features to the model

[5]The term 'bias' carries different meanings across domains, such as statistics, machine learning objectives, and cognitive science. In the current context, too, it is important to differentiate statistical bias (a technical property of the model) and social bias (systematic advantages or disadvantages for certain groups).

[6]Note that not all automated systems employ AI/ML algorithms, but all AI-based systems are considered automated in general. In this thesis, we focus on AI systems.

training due to correlations between these attributes and other variables in the data [37].

Just as there are regulations and laws to safeguard individuals' privacy, there are various anti-discrimination laws and regulatory policies to prevent discrimination and resulting harms such as the Fair Housing Act and the Equal Credit Opportunity Act in the U.S., the EU AI Act, the Canadian Humans Act Right, the Anti-Discrimination Regime in China, and Article 14 in Indian Constitution[7].

As mentioned in Chapter 1, laws and regulations alone are not sufficient to build trust in AI systems. Just like privacy, fairness must be embedded into the very fabric of AI pipelines.

All of the above concerns have led to the emergence of the field of *algorithmic fairness*, which focuses on developing mathematical formulations and computational techniques to ensure that algorithmic decisions are equitable and non-discriminatory. The field broadly encompasses both technical approaches (e.g. bias detection metrics, fairness-aware algorithms) and socio-technical considerations (e.g. stakeholder participation, procedural fairness, transparency).

The field of algorithmic fairness has recently aimed at achieving fairness in AI. Fairness in AI refers to ensuring that an AI systems treats individuals and groups equitably, without favoritism or unjustified disparities in outcomes. Fairness, similar to privacy, is a multifaceted and context-dependent concept, and translating it into mathematical criteria to fit into AI pipelines is non-trivial.

Many definitions, mathematical constructs, and metrics for the notion of fairness have been proposed in the literature over time [191]. There is no single agreed-upon definition or measure because fairness is subjective and relative to a task [192]. Similar to privacy, we will adhere to the formal definitions of fairness metrics that are widely used in the literature. A few broad notions have been adapted: (a) direct and indirect discrimination (see Chapter 4 and 8 in [37]) , (b) disparate impact and disparate treatment stemming directly from legal

[7]Many laws and regulations existed even before the digital era, as discrimination has long been present in society. With the adoption of AI, these laws are now being expanded to address the context and applicability of AI systems.

concepts (see Chapter 6 in [37]), and (c) individual and group fairness [191, 193]. In this thesis, we focus on the latter notion of fairness and briefly describe it below (see Figure 4.1). For the formal definitions, we refer to Section 5.2 in Chapter 5, Section 6.1 in Chapter 6 and Section 7.2 in Chapter 7.

- *Individual fairness* emphasizes that similar individuals receive similar treatment [193]. This is typically achieved by defining an appropriate similarity metric, which quantifies how similar two individuals are based on relevant attributes. The key challenge lies in defining an appropriate similarity metric for individuals. The focus is on ensuring that individuals who are similar in meaningful ways receive comparable treatment, regardless of sensitive attributes like race or gender.

Figure 4.1: Individual and Group Fairness

- *Group fairness*, on the other hand, ensures similar outcomes across predefined groups based on some sensitive attribute such as gender or race [191]. This is typically achieved by ensuring that certain statistical measures are the same (or have a small predefined delta) across the groups. These group fairness criteria focus on outcomes at the population level, ensuring that groups defined by sensitive attributes are treated similarly.

Both group and individual fairness approaches have strengths and limitations. Group fairness is relatively easier to measure and enforce but may allow for unfair treatment of individuals within groups. Certain group fairness criteria are shown to be mathematically

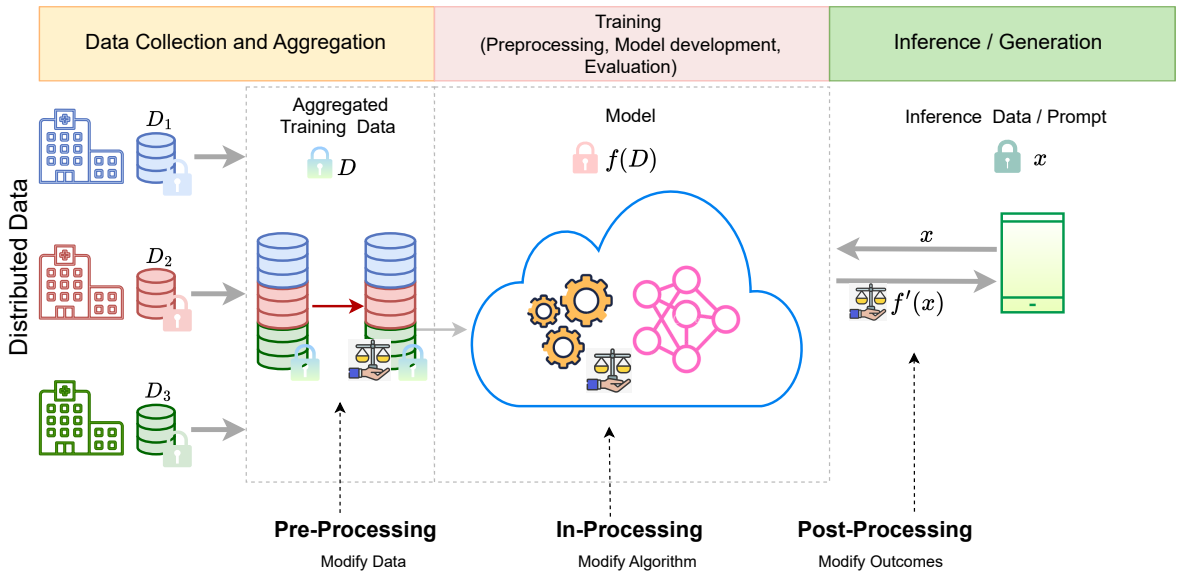


Figure 4.2: **Bias Mitigation in AI pipeline**. (a) Pre-processing: modifies the training data for better representation before training (b) In-processing: modifies the training procedure to meet fairness and utility objectives together during training (c) Post-processing: modifies the predicted outcomes to ensure fair outcomes after training

incompatible with each other [194]. Individual fairness can provide stronger guarantees at the individual level but requires defining an appropriate similarity metric, which may be subjective and context-dependent.

4.2 Bias Mitigation in AI

Like different PETs are available for preserving privacy such as FL, MPC and HE, we have different techniques to improve fairness in AI systems. These techniques can be broadly categorized into three kinds, based on when and where they are applied in the AI pipeline (see Figure 4.2) . We briefly describe these three categories below:

- **Pre-processing.** Pre-processing techniques aim to transform the training data before

the model training phase to mitigate bias. These methods modify the feature space, training instances, or class labels to create a "cleaned" dataset that is expected to lead to fairer models [195, 196, 197, 198, 199].

A prominent example, a flavor of which we also use in our thesis, is the *reweighing* method [199] in Chapter 6. It assigns different weights to training instances based on their group membership and labels to promote fairness. Some other techniques include methods such as modifying class labels of specific training examples (selected samples near the decision boundary) to reduce bias in a strategic way [195] and transforming features to ensure they have minimal correlation with sensitive attributes [197].

Pre-processing techniques are model- and metric-agnostic, i.e., they can be applied irrespective of the ML model being trained and the fairness definition and metric being used. This offers an advantage as these methods can be easily integrated into existing AI systems. These methods are designed to eliminate the biases at the source (i.e. the data itself) which can be more effective than mitigating the biases after training, especially in cases, for example, where the training instances of a certain group are underrepresented. On the other hand, these methods might not completely eliminate biases if the data has complex patterns and associations between sensitive attributes and other features of the dataset. Moreover, pre-processed data does not ensure that fairness will prevail through out the AI pipeline and the model lifecycle.

Some of the pre-processing techniques, such as those by Kamiran and Calders [195], do require access to sensitive attributes which are private in nature and that might not be easily accessible due to privacy regulations. This introduces privacy concerns, even when the training data itself is public or non-sensitive. When the training data is non-sensitive, although the model may not require private training, these bias mitigation techniques require access to sensitive attributes. When sensitive attributes are part of the training data, privacy-preserving training is necessary regardless, but pre-processing adds an extra round of access to these attributes, increasing privacy risks.

In Chapter 6, we address the challenge of access to sensitive attributes by proposing a solution that enables the use of pre-processing techniques while preserving the privacy of sensitive attributes.

- **In-processing.** In-processing techniques incorporate fairness constraints directly into the model training process, either by modifying the objective function (such as by adding a regularization term to penalize deviation from fairness) or by modifying the optimization procedure (such as by introducing fairness-aware gradient updates or constraints) [200, 201, 202, 203, 204, 205]. A few methods adapt approaches based on adversarial debiasing where a predictor is trained to make accurate predictions while an adversary attempts to predict the protected attribute from the predictor's output, thereby encouraging the predictor to be invariant to that attribute [202].

  In-processing methods often achieve a better trade-off between fairness and utility, as they directly optimize for both objectives during training. They can learn complex patterns and associations between sensitive attributes and other features in the dataset – capabilities that post-processing techniques typically lack. But, these methods are often specific to particular model architectures or learning algorithms, which limits their generalizability. Since they modify the training process, they can increase both implementation complexity and computational cost. Moreover, they often require careful hyperparameter tuning, which may involve multiple accesses to the same database, increasing the risk of privacy leakage (see sequential composition in Chapter 2). Some approaches attempt to adapt in-processing techniques under privacy constraints, such as using differentially private stochastic gradient descent (DP-SGD), which may still result in disparate impacts, i.e., amplify the biases [206]. Due to these concerns, we do not consider in-processing in this thesis. Note that if the training data is public and does not include sensitive attributes, there are no privacy concerns. If it contains sensitive attributes, privacy-preserving training is required, with only a single access to the dataset for training and bias mitigation.

- **Post-processing.** Post-processing techniques modify the predicted outcomes to ensure fair predictions without modifying the original model or training data [207, 208].

  A prominent example is the Equalized Odds post-processing method, which we use in our thesis in Chapter 6, that adjusts a model's predictions to satisfy fairness criteria, i.e., it applies different decision rules (different thresholds for classification or different decision boundaries [134]) for each group [208]. Another example is the Reject Option Classification [207], which identifies test instances near the decision boundary and applies different classification rules to them based on sensitive attributes to achieve fairness.

  A primary challenge with in-processing techniques is their tight integration with specific model architectures and training algorithms, making them difficult to apply to existing systems. They also typically require multiple accesses to the training data during hyperparameter tuning, which increases privacy leakage risks under DP constraints.Moreover, they are not easily generalizable across different model types. In contrast, post-processing techniques offer significant practical advantages: they can be applied to any existing model without retraining, making them particularly valuable for legacy systems where neither the training data nor the model can be modified[8]. This flexibility, combined with their lower implementation complexity and reduced privacy risks, makes post-processing methods an attractive choice. While post-processing approaches may yield somewhat less optimal results both in terms of bias mitigation and accuracy when compared to in-processing, their practical deployability and compatibility with privacy constraints motivates us to use these in our thesis.

  Post-processing methods, similar to pre-processing ones, require access to sensitive attributes. This access is usually independent of the training dataset and typically involves a single round of access to the sensitive attributes (and the ground truth

[8]Such legacy systems or systems where neither the training data nor the model are accessible are still prevalent in production

labels, in some cases).

Privacy and fairness must go hand-in-hand in trustworthy AI systems. Many bias mitigation techniques often require access to sensitive attributes – precisely the data that privacy protections aim to safeguard. These concerns are amplified in distributed environments where sensitive attributes are spread across multiple devices, as is common in cross-device FL. In such scenarios, balancing bias mitigation with privacy becomes essential.

When combining fairness and privacy objectives, one must carefully balance three critical dimensions: utility, privacy, and fairness. In Chapter 3, we demonstrated how DP-in-MPC can be effectively deployed for model training in FL settings to simultaneously maintain privacy and utility. While in-processing bias mitigation techniques could, in principle, also benefit from the DP-in-MPC framework, they typically require tight integration with model training and iterative gradient updates – operations that are computationally intensive and less MPC-friendly – leading to significant runtime challenges. In this part of the thesis, we instead focus on adapting pre-processing and post-processing fairness techniques, which are more modular and easier to integrate within the privacy constraints of federated setups.

In the following chapters, we explore how biases can be mitigated while preserving privacy guarantees. In Chapter 5, we develop MPC protocols for auditing ML models for biases, where both the audit data and the ML model remain private; the only chapter where we rely solely on MPC without DP. We then shift our focus to bias mitigation techniques in cross-device settings. Chapter 6 presents privacy-preserving pre- and post-processing protocols using DP-in-MPC to target group fairness. Chapter 7 extends our approach to information retrieval and recommendation systems, where we develop techniques based on DP-in-MPC to ensure fair exposure for content providers (individual fairness) while preserving user (content consumer) privacy. Throughout these chapters, we demonstrate how we leverage MPC to provide input privacy enabling bias detection and mitigation.

Chapter 5

PRIVACY-PRESERVING AUDITING OF MACHINE LEARNING MODELS

Algorithmic decision making, driven by ML, has become very prominent in applications that directly affect people's quality of life, including in healthcare, justice, and finance. ML models have made discriminatory inferences in recidivism prediction [35], credit card approval [32], advertising [30] and job matching [33], among others. In the previous chapter, we discussed how concerns over fairness in ML have prompted research into the establishment of fairness metrics and techniques to mitigate bias (see e.g. [209, 210, 204, 211, 191] and references therein). *However, mitigating biases in AI first requires detecting biases in AI.*

Auditing is a crucial step for detecting biases in existing AI systems. It allows practitioners to assess whether the defined fairness criteria, as measured by appropriate metrics, are met, or whether bias mitigation is necessary. Without systematic auditing, fairness violations may go unnoticed.

In state-of-the-art approaches based on *group fairness measures*, audits are typically conducted by an external investigator (a.k.a. the auditor) by comparing the service provider's ML model predictions for different demographic subgroups in an audit data set. For instance, a classifier satisfies the definition of *demographic parity* [193] if the subjects in the protected group and the unprotected group have equal probability of being assigned to the positive predicted class, e.g. if credit card approval is equally probable for both females and males.

Auditing ML models raises important privacy challenges. In many real-world deployments, the auditor (who is separate from the service provider and model owner) may use the audit data may include sensitive attributes – such as race or gender – that must remain confidential [212, 37]. At the same time, the model owner's (a.k.a. the auditee's) deployed ML

model may be proprietary and cannot be exposed [213]. This makes it difficult to evaluate fairness using traditional auditing approaches, which often assume access to both the model and the data. To address this, we must develop mechanisms that enable fairness assessments without compromising the privacy of either the auditor or the auditee.

In this chapter, we bridge the concepts of privacy and fairness by focusing on how to perform such privacy-preserving audits. Specifically, we propose novel MPC protocols for securely computing fairness metrics, enabling the auditing of ML models while preserving the privacy of the audit data (which acts as test data during inference, see Chapter 1) and maintaining the confidentiality of the model. As part of this work, we implement these protocols in a practical library, PrivFair, which supports private model auditing. Unlike the rest of this thesis, which focuses on the training phase using the DP-in-MPC paradigm, this chapter shifts attention to the privacy at the inference stage of the ML pipeline.

The remainder of this chapter is organized as follows. Section 5.1 introduces the problem of fairness auditing for of ML models while preserving privacy of both the auditor and the auditee. Section 5.2 formally defines the group fairness metrics that serve as the basis for determining whether a model exhibits biased behavior. The corresponding MPC protocols for computing these metrics are presented in Section 5.3. Section 5.4 provides an empirical evaluation of the proposed protocols across two data modalities (tabular and image) and two model types (logistic regression and convolutional neural networks). Finally, Section 5.5 discusses key insights and limitations, and Section 5.6 summarizes the chapter.

5.1 The Problem

In real world scenarios, a proprietary model $\mathcal{M}$ held by a company *Alice* may need to be audited by an external investigator *Bob* using sensitive audit data $\mathcal{D}$ (see Figure 5.2). For example, a bank or hospital, represented by Bob, wants to purchase use of a predictive model and needs to investigate whether the model performs fairly on their data. Alice does not want to disclose her trained model parameters as this could assist rival companies to benefit from her technology. Furthermore, ML models can memorize specific examples from the

training data [151], hence disclosing $\mathcal{M}$, or even giving just black box access through an API interface, can leak very specific information about the training data, which might be sensitive in itself. Likewise, Bob does not want to disclose the audit data $\mathcal{D}$ to Alice, because it contains sensitive attributes that are on one hand needed for the fairness audit, while on the other hand may be subject of anti-discrimination and data protection law.

While substantial research progress has been made in the area of fairness in AI and, separately, in the area of privacy-preserving ML, there is a gap in literature that addresses fairness and privacy simultaneously. Existing algorithms for fairness auditing, and their implementations in libraries such as Fairlearn [214] and AI Fairness 360 [215], operate without concern for privacy, assuming that the entity performing the fairness audit has unrestricted access to the model $\mathcal{M}$ as well as the audit data $\mathcal{D}$. This assumes that either Alice is willing to disclose $\mathcal{M}$ to Bob, or that Bob is willing to disclose $\mathcal{D}$ to Alice, or that they are both willing to disclose $\mathcal{M}$ and $\mathcal{D}$ to some trusted third party. None of these scenarios is realistic for proprietary models and sensitive audit data.

5.2 Metrics for Group Fairness

Consider the audit data $\mathcal{D}$ that contains samples of the form $(\mathbf{x}, y, a)$. $\mathbf{x}$ is the instance that needs to be classified (e.g. an image, or a row of tabular data), y is the ground truth class label, and a indicates whether the instance belongs to an unprivileged demographic subgroup. a is a value of a binary variable A where $A = 0$ designates the unprotected group and $A = 1$ designates the protected (or sensitive) group. y is a value of a variable Y that represents the actual outcome. Similarly, we use $\hat{Y}$ to denote a variable that represents the predicted outcome according to the model $\mathcal{M}$.

We recall that by fairness we mean algorithmic fairness as commonly understood in the ML literature, namely that a person's experience with an information system should not irrelevantly depend on their personal characteristics such as race, gender, sexual orientation, ethnicity, religion, or age [216]. In this chapter, we focus on fairness as defined by group-based measures. These aim to ensure that different groups of users should receive similar

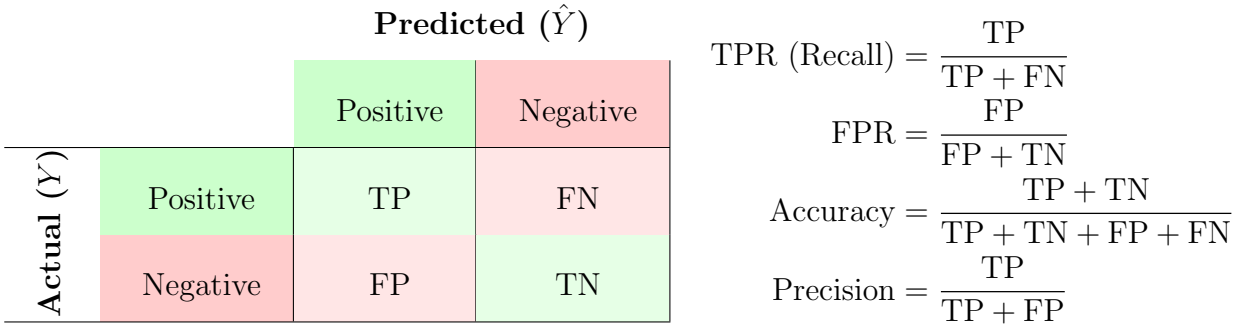


Figure 5.1: Confusion matrix and commonly used classification metrics

statistical treatment.

Many different notions of group fairness have been proposed in the literature, some of them even mathematically incompatible [194][1]. In particular, many widely adopted statistical notions of fairness rely on the components of the confusion matrix shown in Figure 5.1 when applying the ML model under investigation to an audit data set. Specifically, these components include the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These quantities form the basis of computing fairness metrics that we describe below.

- **Equalized odds.** By definition, equalized odds is satisfied if $\hat{Y}$ and A are independent conditional on Y,i.e. $\hat{Y} \perp\!\!\!\perp A \mid Y$ [208]. For a binary classifier, this is equivalent to Equation (5.1). This implies that the true positive rate TPR = TP/(TP+FN) and the false positive rate FPR = FP/(FP+TN) are the same for both the unprotected and the protected groups, i.e. ($\text{TPR}_{A=1} = \text{TPR}_{A=0}$ and $\text{FPR}_{A=1} = \text{FPR}_{A=0}$).

[1]This inherent incompatibility between fairness metrics and the choice of appropriate metrics is itself a topic of ongoing research and discussion, and it is not the focus of this thesis.

An exact equality as in Equation (5.1) may be hard to achieve, so it is common to compute the left and the right hand sides of Equation (5.1) separately and inspect their differences, sometimes reported through metrics such as the equalized odds difference.

$$\mathrm{P}(\hat{Y} = 1 \,|\, Y = y, A = 0) = \mathrm{P}(\hat{Y} = 1 \,|\, Y = y, A = 1), \quad \forall y \in \{0, 1\} \tag{5.1}$$

We extend Equation (5.1) to multi-class classification using the *one-vs-rest* approach. For a classification task with a label set $\mathcal{L} = \{0, 1, ..., C-1\}$, for each class label $c \in \mathcal{L}$, we define the equalized odds for class label c as per Equation (5.2). In this case, for each class label c, we consider two cases with $\hat{Y} = c$ and $\hat{Y} = \neg c$ (any other class). Adapting Equation (5.1) to this gives us Equation (5.2) which implies that the TPR and FPR for each class should be same for both the protected and unprotected groups.

$$\mathrm{P}(\hat{Y} = c \,|\, Y = y, A = 0) = \mathrm{P}(\hat{Y} = c \,|\, Y = y, A = 1), \quad \forall y \in \{c, \neg c\}, \forall c \in \mathcal{L} \tag{5.2}$$

- **Equal Opportunity.** By definition, equal opportunity, [208], for a binary predictor is satisfied if

$$\mathrm{P}(\hat{Y} = 1 \,|\, Y = 1, A = 0) = \mathrm{P}(\hat{Y} = 1 \,|\, Y = 1, A = 1) \tag{5.3}$$

This implies that the TPR is the same for both the unprotected and the protected group, i.e. ($\mathrm{TPR}_{A=1} = \mathrm{TPR}_{A=0}$). This fairness notion is a relaxation of equalized odds in a sense that it focuses only the positive or advantaged outcome.

Equal opportunity can be extended to a multi-class classification task, as shown in Equation (5.4). This involves computing the TPR for each individual class c for each group, implying that the TPR for each class should be same for both protected and unprotected groups.

$$\mathrm{P}(\hat{Y} = c \,|\, Y = c, A = 0) = \mathrm{P}(\hat{Y} = c \,|\, Y = c, A = 1), \quad \forall c \in \mathcal{L} \tag{5.4}$$

- **Sub-Group Accuracy.** This notion of fairness, similar to overall accuracy equality [217], when the classifier achieves equal accuracy for both the protected and unprotected groups.

  We report this metric by computing the accuracy for the protected group, the accuracy for the unprotected group, and the overall accuracy. These notions are defined as follows, for a classifier $\mathcal{M}$ and an audit data set $\mathcal{D}$ with instances of the form $(\mathbf{x}, y, a)$:

$$\text{ACC}_{\mathsf{A=1}} = \frac{|\{(\mathbf{x}, y, 1) \in \mathcal{D} \,|\, y = \mathcal{M}(\mathbf{x})\}|}{|\{(\mathbf{x}, y, 1) \in \mathcal{D}\}|} \tag{5.5}$$

$$\text{ACC}_{\mathsf{A=0}} = \frac{|\{(\mathbf{x}, y, 0) \in \mathcal{D} \,|\, y = \mathcal{M}(\mathbf{x})\}|}{|\{(\mathbf{x}, y, 0) \in \mathcal{D}\}|} \tag{5.6}$$

$$\text{ACC} = \frac{|\{(\mathbf{x}, y, a) \in \mathcal{D} \,|\, y = \mathcal{M}(\mathbf{x})\}|}{|\{(\mathbf{x}, y, a) \in \mathcal{D}\}|} \tag{5.7}$$

  Equation (5.5)–(5.7) cover both binary and multi-class classifiers.

- **Demographic parity.** Demographic parity, also known as statistical parity, is one of the best known and most widely accepted notions of group fairness [193]. A classifier satisfies demographic parity if

$$\text{P}(\hat{Y} = 1 \,|\, A = 0) = \text{P}(\hat{Y} = 1 \,|\, A = 1) \tag{5.8}$$

  This implies that both protected and unprotected groups have equal probability of receiving positive outcomes [191]. This is equivalent to the ratio of the number of positive outcomes for the group to the total number of instances belonging to the group. Note that the number of positive outcomes for a group in the audit data is the sum of the number of true positives and the number of false positives for that group, i.e. (POS = $\text{TP}_{A=a}$ + $\text{FP}_{A=a}$, where POS is the number of positive outcomes).

5.3 Secure Fairness Auditing Protocols

MPC is an appropriate choice for performing audits for ML biases without revealing $\mathcal{M}$ and the audit data $\mathcal{D}$. It enables the accurate computation of fairness metrics while providing

strong protection to both inputs, thus meeting both the privacy and auditability goals. Unlike the rest of the thesis, the focus here is solely on input privacy. Note that a limitation of this approach is that while the audit data remains protected, the outputs (i.e. fairness metrics) are revealed to the auditor. This could potentially enable attacks based on those metrics, as output privacy is not provided in this setup.

Although fairness auditing is inherently a two-party (2PC) task between *Alice* and *Bob*, in practice, either party may lack the necessary infrastructure and prefer to outsource the computation to the cloud. This motivates our proposal of MPC-as-a-Service (see Chapter 2), which supports the following two scenarios:

Scenario 1 *Alice* and *Bob* have the necessary computational setup to perform MPC computations themselves and proceed by secret-sharing their data with each other directly. In this case, MPC-as-a-Service consists of only two computing servers, where the MPC parties, namely *Alice* and *Bob*, themselves act as the MPC servers.

Scenario 2 *Alice* and *Bob* need to perform computation intensive tasks and outsource the MPC based computations, thus secret-sharing the data across a set of MPC servers. Figure 5.2 illustrates this scenario with 3 MPC servers.

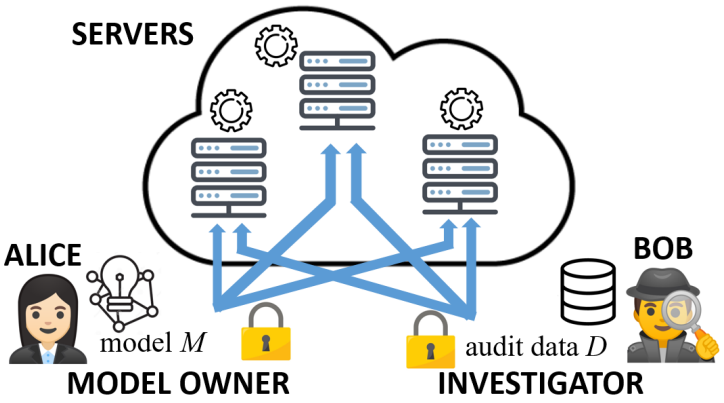
To the best of our knowledge, at the time this research was conducted, the use of MPC for private detection and mitigation of bias in ML models was so far only considered by Kilbertus et al. [218], for one group fairness measure (statistical parity). Kilbertus et al. focus on privacy-preserving training of a fair logistic regression (LR) model, in an honest-but-curious 2-server set-up in which the audit data is revealed to one of the servers. The main differences with our library is that we propose

(a) fairness auditing of a variety of ML models (LR, SVM, convolutional networks (CNNs), decision trees (DTs), and random forests (RFs)),

(b) against a variety of group fairness metrics (see Section 5.2),

(c) under passive *and* active security threat models (See Section 2.2.3 in Chapter 2),

(d) while fully protecting both the audit data *and* the model parameters.



(a) Secure fairness auditing in the 3PC scenario

```
Encrypting model parameters...
Connecting to servers...
Connected to servers...
secret-sharing parameters with servers...

Servers running MPC protocol for DP
Servers running MPC protocol for EOP

Finished protocol execution
```

(b) Model owner's terminal during audit

```
Encrypting audit data
Connecting to servers...
Connected to servers...
secret-sharing audit data with servers...

Servers running MPC protocol for DP
Servers running MPC protocol for EOP

Finished protocol execution
Aggregating results...

Demographic parity - male:  0.24
Demographic parity - female:  0.19
Equal opportunity - male:  0.399
Equal opportunity - female:  0.40
```

(c) Investigator's terminal during audit

Figure 5.2: **PrivFair**. Alice and Bob send encrypted shares of the model $\mathcal{M}$ (Alice) and the audit data $\mathcal{D}$ (Bob) to 3 servers. The servers subsequently execute MPC protocols to measure demographic parity (DP) and equal opportunity (EOP) in a privacy-preserving manner, i.e. through computations over the encrypted data.

Furthermore, the MPC protocols that we develop in this chapter can be applied in tandem with the technique proposed by Segal et al. [219] for certifying that an audited ML model is fair. In contrast with Segal et al. we propose MPC-based protocols for auditing the ML models in a privacy-preserving way, with audit data that is not disclosed in an unencrypted manner to the audit servers (a.k.a. MPC servers).

The parameter values of *Alice's* trained model $\mathcal{M}$, and the feature values of *Bob's* audit data $\mathcal{D}$, are natively often real numbers represented in a floating point format. As is common in MPC, *Alice* and *Bob* convert all data in their inputs $\mathcal{M}$ and $\mathcal{D}$ into integers in $\mathbb{Z}_{2^k}$ (See Section 2.2.1 in Chapter 2) [2].

Next, *Alice* and *Bob* secret-share the integers with the MPC servers. These MPC servers run the MPC protocols that compute the fairness metrics used to audit ML models. privacy-preserving computation of the components of the confusion matrix forms the basic building blocks of these MPC protocols. The MPC protocols we develop are generic and can be adapted to any threat model by substituting the underlying MPC primitives.

Below we describe MPC protocols to compute the metrics in Section 5.2. At the beginning of these protocols, the servers have secret-shares of the parameters of the model $\mathcal{M}$, as received from the model owner *Alice* (see Figure 5.2). Similarly, from the investigator *Bob*, the servers have received secret-shares of an audit data set $\mathcal{D}$ with N instances, including a secret-shared vector Y of length N with the ground truth labels and a secret-shared vector A of length N with the sensitive attributes.

5.3.1 Protocol π_{EOD} for Equalized Odds.

In PrivFair, the left and right hand sides of Equation (5.2) are computed in a privacy-preserving manner with protocol π_{EOD}, presented in pseudocode here as Protocol 5. Note that we abuse the notations Y and A here to denote random variables as in Equation (5.2) as well as vectors with values for those random variables as in Protocol 5.

[2] k is chosen large enough so that there is no loss of precision that would affect the utility of the ML models.

Protocol 5: π_{EOD} for computing equalized odds for multi-class classification

Input : The servers have a secret-sharing of trained model parameters $[\![\mathcal{M}]\!]$, and a secret-sharing of a data set $[\![\mathcal{D}]\!]$ with N instances and secret-sharings $[\![Y]\!]$ and $[\![A]\!]$ of the corresponding ground truth labels (drawn from a set $\mathcal{L}$ with C labels) and a binary sensitive attribute.

Output : A secret-sharing of the equalized odds metrics for each class.

$[\![Y_{\mathsf{pred}}]\!] \leftarrow \pi_{\mathsf{INFER}}([\![\mathcal{M}]\!],[\![\mathcal{D}]\!])$

for $c = 0$ ***to*** $C-1$ **do**

  for $i = 1$ ***to*** N **do**

    $[\![\text{grnd}]\!] \leftarrow \pi_{\mathsf{EQ}}([\![Y[i]]\!],\, c)$

    $[\![\text{pred}]\!] \leftarrow \pi_{\mathsf{EQ}}([\![Y_{\mathsf{pred}}[i]]\!],\, c)$

    $[\![\text{a}]\!] \leftarrow [\![A[i]]\!]$

    $[\![\text{tp}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{grnd}]\!],\, [\![\text{pred}]\!])$

    $[\![\text{ta}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{grnd}]\!],\, [\![\text{a}]\!])$

    $[\![\text{pa}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{pred}]\!],\, [\![\text{a}]\!])$

    $[\![\text{tpa}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{tp}]\!],\, [\![\text{a}]\!])$

    $[\![\text{TP}_{\mathsf{A=1}}[c]]\!] \leftarrow [\![\text{TP}_{\mathsf{A=1}}[c]]\!] + [\![\text{tpa}]\!]$

    $[\![\text{FN}_{\mathsf{A=1}}[c]]\!] \leftarrow [\![\text{FN}_{\mathsf{A=1}}[c]]\!] + ([\![\text{ta}]\!] - [\![\text{tpa}]\!])$

    $[\![\text{FP}_{\mathsf{A=1}}[c]]\!] \leftarrow [\![\text{FP}_{\mathsf{A=1}}[c]]\!] + ([\![\text{pa}]\!] - [\![\text{tpa}]\!])$

    $[\![\text{TN}_{\mathsf{A=1}}[c]]\!] \leftarrow [\![\text{TN}_{\mathsf{A=1}}[c]]\!] + ([\![\text{a}]\!] - [\![\text{ta}]\!] - [\![\text{pa}]\!] + [\![\text{tpa}]\!])$

    $[\![\text{TP}_{\mathsf{A=0}}[c]]\!] \leftarrow [\![\text{TP}_{\mathsf{A=0}}[c]]\!] + ([\![\text{tp}]\!] - [\![\text{tpa}]\!])$

    $[\![\text{FN}_{\mathsf{A=0}}[c]]\!] \leftarrow [\![\text{FN}_{\mathsf{A=0}}[c]]\!] + ([\![\text{grnd}]\!] - [\![\text{ta}]\!] - [\![\text{tp}]\!] + [\![\text{tpa}]\!])$

    $[\![\text{FP}_{\mathsf{A=0}}[c]]\!] \leftarrow [\![\text{FP}_{\mathsf{A=1}}[c]]\!] + ([\![\text{pred}]\!] - [\![\text{pa}]\!] - [\![\text{tp}]\!] + [\![\text{tpa}]\!])$

    $[\![\text{TN}_{\mathsf{A=0}}[c]]\!] \leftarrow [\![\text{TN}_{\mathsf{A=1}}[c]]\!] + (1 - [\![\text{grnd}]\!] - [\![\text{pred}]\!] - [\![\text{a}]\!] + [\![\text{tp}]\!] + [\![\text{ta}]\!] + [\![\text{pa}]\!] - [\![\text{tpa}]\!])$

  end

end

$[\![\text{TPR}_{\mathsf{A=1}}[c]]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{TP}_{\mathsf{A=1}}[c]]\!] ,[\![\text{TP}_{\mathsf{A=1}}[c]]\!] + [\![\text{FN}_{\mathsf{A=1}}[c]]\!])$

$[\![\text{TPR}_{\mathsf{A=0}}[c]]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{TP}_{\mathsf{A=0}}[c]]\!] ,[\![\text{TP}_{\mathsf{A=0}}[c]]\!] + [\![\text{FN}_{\mathsf{A=0}}[c]]\!])$

$[\![\text{FPR}_{\mathsf{A=1}}[c]]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{FP}_{\mathsf{A=1}}[c]]\!] ,[\![\text{FP}_{\mathsf{A=1}}[c]]\!] + [\![\text{TN}_{\mathsf{A=1}}[c]]\!])$

$[\![\text{FPR}_{\mathsf{A=0}}[c]]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{FP}_{\mathsf{A=0}}[c]]\!] ,[\![\text{FP}_{\mathsf{A=0}}[c]]\!] + [\![\text{TN}_{\mathsf{A=0}}[c]]\!])$

return $[\![\text{TPR}_{\mathsf{A|0,1}}]\!]$, $[\![\text{FPR}_{\mathsf{A|0,1}}]\!]$

Table 5.1: Logic for evaluating TP, FN, FP and TN

grnd	pred	Metric	Logic
1	1	TP	grnd $*$ pred
1	0	FN	grnd $*$ (1$-$pred)
0	1	FP	(1-grnd) $*$ pred
0	0	TN	(1-grnd) $*$ (1$-$pred)

On Line 1, the servers perform privacy-preserving labeling of the audit data instances in $\mathcal{D}$ with the model $\mathcal{M}$ using an MPC protocol π_{INFER} for secure inference. Such protocols have been developed by us and others for logistic regression, neural networks, decision tree ensembles etc. [220, 221, 222, 223, 108, 224] and can be used in combination with the protocols in PrivFair. After Line 1, the servers have a secret-shared vector $[\![Y_{\mathsf{pred}}]\!]$ with predicted class labels for each of the samples in the audit data.

Next, on Line 2–19, for each of the classes, the servers compute the number of true positives (TP), number of false positives (FP), number of true negatives (TN) and false negatives (FN) for the (un)protected group. To this end, on Line 4, the servers compute a secret-shared binary variable $[\![\text{grnd}]\!]$ representing if instance i belongs to the class c that is being inspected. To obtain this, the servers run an MPC protocol π_{EQ} for equality testing that takes as input the secret-shared ground truth class label $[\![Y[i]]\!]$ and the class c itself, and returns a secret-sharing of 1 if the equality test was positive, and a secret-sharing of 0 otherwise. Next, on Line 5, the servers compute in a similar way a secret-shared binary variable $[\![\text{pred}]\!]$ representing if the predicted outcome for instance i is the class c that is being inspected. On Line 6, the servers compute a secret-shared binary value $[\![\text{a}]\!]$ representing if the current sample in the audit data belongs to the protected or unprotected group.

The underlying logic for efficiently computing TP, FN, FP, and TN, is derived from the truth table shown in Table 5.1. Multiplying column 4 in Table 5.1 with a and with 1$-$a

respectively gives us the contribution of instance i to the TP, FN, FP, and TN metrics for the protected and unprotected group respectively. For example, the total number of instances of the protected group that get incorrectly classified as belonging to class c is

$$\mathrm{FP}_{\mathrm{A}=1}[c] = \sum_{i=1}^{n} (1 - \mathrm{grnd}^{(i)}) \cdot \mathrm{pred}^{(i)} \cdot \mathrm{a}^{(i)} \tag{5.9}$$

in which $\mathrm{grnd}^{(i)}$, $\mathrm{pred}^{(i)}$, and $\mathrm{a}^{(i)}$ are the secret-shared binary variables computed for instance i on Line 4–5.

This logic is modified to suit the MPC computations on Line 7–18. To this end, we rewrote the expressions in the summations as in the right hand side of Equation (5.9) to reduce the overall number of multiplications as much as possible. As a result, π_{EOD} requires only 4 multiplications per instance and per class (see Lines 7–10). On Lines 11–14 and Lines 15–18, the servers use these precomputed secret-shared products to obtain secret-shares of TP, FN, FP, and TN respectively for the protected group and for the unprotected group.

Finally, on Line 20–23 the servers execute a protocol π_{DIV} for division with secret-shared values, to obtain secret-shares of the TPR and FPR for each class for the protected and unprotected groups. At the end of the protocol, each server sends its secret-shares to the investigator Bob who can then combine the shares to learn the TPR and FPR values. The servers themselves do not learn anything about the real values of the TPR and the FPR, nor of the TP, FN, FP, and TN metrics computed along the way, and not even the number of instances of each class in the audit data.

While Protocol 5 is presented for multi-class classification, it can be trivially used for binary classification as well by removing Line 2 and 19, and running the inner for-loop only once with a fixed value of $c = 1$.

5.3.2 Protocol $\boldsymbol{\pi}_{\mathsf{EOP}}$ for Equal Opportunity

Protocol 5 can be modified directly to compute Equation (5.3) resulting in a new protocol π_{EOP}. This can be done by running the inner for-loop only once, with a fixed value of $c = 1$ (binary classification) and by removing several lines, i.e. Line 9; the lines that compute FP

and TN (Lines 13–14,17–18); and the lines that compute FPR (Lines 22–23). Through executing the resulting protocol π_{EOP}, the servers will only compute secret-shares of TPR (Line 25) for $c = 1$.

Protocol 6: π_{GACC} for computing sub-group accuracy for multi-class classification

Input : The servers have a secret-sharing of trained model parameters $[\![\mathcal{M}]\!]$, and a secret-sharing of a data set $[\![\mathcal{D}]\!]$ with N instances and secret-sharings $[\![Y]\!]$ and $[\![A]\!]$ of the corresponding ground truth labels (drawn from a set $\mathcal{L}$ with C labels) and a binary sensitive attribute.

Output : A secret-sharing of the sub-group accuracy metrics.

1 $[\![Y_{\mathsf{pred}}]\!] \leftarrow \pi_{\mathsf{INFER}}([\![\mathcal{M}]\!],[\![\mathcal{D}]\!])$

2 **for** $i = 1$ ***to*** N **do**

3 $\quad [\![\text{count}_{\mathsf{A=1}}]\!] \leftarrow [\![\text{count}_{\mathsf{A=1}}]\!] + [\![A[i]]\!]$

4 **end**

5 $[\![\text{count}_{\mathsf{A=0}}]\!] \leftarrow N - [\![\text{count}_{\mathsf{A=1}}]\!]$

6 $[\![\text{correct}_{\mathsf{A=1}}]\!] \leftarrow 0$

7 $[\![\text{correct}_{\mathsf{A=0}}]\!] \leftarrow 0$

8 **for** $i = 1$ ***to*** N **do**

9 $\quad [\![\text{corr}]\!] \leftarrow [\![Y[i]]\!] - [\![Y_{\mathsf{pred}}[i]]\!]$

10 $\quad [\![\text{iscorr}]\!] \leftarrow \pi_{\mathsf{EQ}}([\![\text{corr}]\!], 0)$

11 $\quad [\![\text{iscorr}_{\mathsf{A=1}}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{iscorr}]\!], [\![A[i]]\!])$

12 $\quad [\![\text{correct}_{\mathsf{A=1}}]\!] \leftarrow [\![\text{correct}_{\mathsf{A=1}}]\!] + [\![\text{iscorr}_{\mathsf{A=1}}]\!]$

13 $\quad [\![\text{correct}_{\mathsf{A=0}}]\!] \leftarrow [\![\text{correct}_{\mathsf{A=0}}]\!] + ([\![\text{iscorr}]\!] - [\![\text{iscorr}_{\mathsf{A=1}}]\!])$

14 **end**

15 $[\![\text{ACC}_{\mathsf{A=1}}]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{correct}_{\mathsf{A=1}}]\!] , [\![\text{count}_{\mathsf{A=1}}]\!])$

16 $[\![\text{ACC}_{\mathsf{A=0}}]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{correct}_{\mathsf{A=0}}]\!] , [\![\text{count}_{\mathsf{A=0}}]\!])$

17 $[\![\text{ACC}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{correct}_{\mathsf{A=1}}]\!] + [\![\text{correct}_{\mathsf{A=0}}]\!], 1/N)$

18 **return** $[\![\text{ACC}_{\mathsf{A|0,1}}]\!]$, $[\![\text{ACC}]\!]$

5.3.3 Protocol $\boldsymbol{\pi}_{\mathsf{GACC}}$ for Sub-Group Accuracy

On Line 1 in Protocol 6, the servers perform privacy-preserving labeling of the audit data $\mathcal{D}$ with the model $\mathcal{M}$. On Line 2-4, the servers "count" how many instances in the audit data belong to the protected group, i.e. the denominator of Equation (5.5). Note that each of the servers obtains only a secret-share of the resulting $\text{count}_{\mathsf{A}=1}$, i.e. no server finds out how many instances in the audit data belong to the protected class. Since the total number of instances N is public information, the servers can straightforwardly retrieve a secret-sharing of the number of instances belonging to the unprotected group by computing $N - [\![\text{count}_{\mathsf{A}=1}]\!]$. This is leveraged in Line 5.

Through Lines 8–14, the servers count how many predictions are correct. To this end, for each instance, on Line 9 the servers compute the difference between the actual outcome and predicted outcome. If this difference is equal to zero as per Line 10, it means it was a correct prediction. Lines 11–13 decide whether this correct prediction is to be accounted for in the total of the protected or the unprotected group. The right hand side of Line 13 for the unprotected group is written to take advantage of the multiplication that was already done for the protected group on Line 11.

On Line 15–16 the servers compute accuracy for the protected and unprotected group respectively. Line 17 computes overall accuracy by performing secure addition of number of correct instances for each subgroup followed by secure multiplication with a constant $1/N$.

5.3.4 Protocol $\boldsymbol{\pi}_{\mathsf{DP}}$ for Demographic Parity

Protocol π_{DP} can be derived by replacing Line 2 in Protocol 5 with $c = 1$ (for binary classification) and retaining only the lines required for computing TP and FP for each subgroup (Lines 3–10,11,13,15,17). Furthermore, Lines 20-23 are replaced by lines that compute the sum of TP and FP:

$$
\begin{aligned}
[\![\text{POS}_{\mathsf{A}=1}]\!] &\leftarrow [\![\text{TP}_{\mathsf{A}=1}]\!] + [\![\text{FP}_{\mathsf{A}=1}]\!] \\
[\![\text{POS}_{\mathsf{A}=0}]\!] &\leftarrow [\![\text{TP}_{\mathsf{A}=0}]\!] + [\![\text{FP}_{\mathsf{A}=0}]\!]
\end{aligned}
$$

Additionally to compute the ratio, we adopt computations similar to Protocol 6, by including lines for computing counts of the instances belonging to each group (Lines 2–5), followed by the lines to compute the ratio

$$\begin{aligned} [\![\mathrm{DP}_{\mathsf{A}=1}]\!] &\leftarrow \pi_{\mathsf{DIV}}([\![\mathrm{POS}_{\mathsf{A}=1}]\!] \, , \, [\![\mathrm{count}_{\mathsf{A}=1}]\!]) \\ [\![\mathrm{DP}_{\mathsf{A}=0}]\!] &\leftarrow \pi_{\mathsf{DIV}}([\![\mathrm{POS}_{\mathsf{A}=0}]\!] \, , \, [\![\mathrm{count}_{\mathsf{A}=0}]\!]) \end{aligned}$$

At the end, Line 25 in Protocol 5 is replaced with

$$\textbf{return } [\![\mathrm{DP}_{\mathsf{A}=1}]\!], [\![\mathrm{DP}_{\mathsf{A}=0}]\!]$$

5.4 Empirical Evaluation of Protocols

PrivFair is implemented[3] on top of MP-SPDZ [110] and is sufficiently generic to take advantage of the variety of underlying MPC schemes in MP-SPDZ.

In this section, we empirically evaluated the proposed protocols for secure fairness auditing. We perform all our experiments on virtual machines (host) on Google Cloud Platform (GCP) with 8 vCPUs, 32 GB RAM and egress bandwidth limited to 16 Gbps. We use mixed computations [225, 226, 227, 228] that switch between arithmetic ($\mathbb{Z}_{2^k}$ with $k = 64$) and binary ($\mathbb{Z}_2$) computations for efficiency. All integer additions and multiplications are performed over the arithmetic domain and any non-linear functions such as comparisons are computed over the binary domain.

We report the time taken to execute the MPC protocols in PrivFair under different security settings in Table 5.2, namely for 2PC and 3PC with passive or with active adversaries. To perform experiments for *binary classification*, we audit a logistic regression (LR) model for credit score classification on the benchmark German credit score data set [229]. The servers run MPC protocols π_{DP} and π_{EOP} from PrivFair to compute demographic parity and equality of opportunity, using gender as the sensitive attribute. For *multi-class classification*, we demonstrate PrivFair to audit a ConvNet ($\sim$ 1.48M parameters) model for

[3] `https://bitbucket.org/uwtppml/privfair`

emotion recognition from images [224]. As audit data, we use 56 images from the RAVDESS data set [230] corresponding to different emotions, namely *neutral*, *happy*, *sad*, *angry*, *fearful*, *disgust*, and *surprised*. Using gender as the sensitive attribute, the servers run MPC protocols π_{EOD} and π_{GACC} from PrivFair to compute equalized odds and subgroup accuracy.

Regarding utility we note that the LR and ConvNet models were trained in the clear, and that the π_{INFER} protocol for secure inference with these models infer the same labels as one would obtain without encryption, i.e. there is no loss of utility (accuracy). The results of the secure fairness auditing protocols are therefore also the same as one would obtain without encryption. In the remainder of this section, we therefore focus on the runtimes.

The times reported in Table 5.2 correspond to the entire runtime of the MPC protocols, i.e. both the so-called offline and online phases[4]. The offline phase includes any pre-processing required to begin executing the MPC protocol (such as the generation of the correlated randomness that is needed for the secure multiplication π_{MUL}) and is independent of the specific input values; the online phase is where the MPC protocol executes on the specific inputs.

Furthermore, the times reported in Table 5.2 encompass the time needed to infer class labels for all the instances in the audit data (such as on Line 1 in Protocol 5 and Protocol 6) and the time needed to evaluate the individual applicable fairness notions. The substantial differences in runtime for auditing the logistic regression model (binary classification) vs. the ConvNet model (multi-class classification) stem from large differences in runtime for the inference step. For example, auditing 56 images belonging to 7 classes with a ConvNet of 1.48M parameters to report EOD takes 42,745.09 sec in the active 2PC setting. Out of this, 42,515.82 sec were taken to classify the 56 images in the audit data in a privacy-preserving manner (Line 1 of Protocol 5), while execution of the rest of Protocol 5 to evaluate the fairness metric itself took 229.27 sec. Nearly all the time taken by PrivFair's protocols for auditing the image classifier are spent on inference, while the time taken to evaluate only

[4]The runtimes reported were recorded at the time of conducting this research. We note that improvements have since been made to MP-SPDZ, and it is possible to achieve better runtimes with the latest versions.

Table 5.2: Time taken to execute individual MPC protocols in PrivFair – includes time for making inferences and fairness evaluation. Binary classification corresponds to credit score classification on the German credit score data set. Multi-class classification corresponds to detecting one of 7 emotions in an image using the RAVDESS data set. The times do not include compile time but include both online and offline times. Times are an average over 5 runs. 2PC-Passive: [1], 2PC-Active: [2], 3PC-Passive: [3], 3PC-Active: [4]; all with corruption threshold of 1.

	#Samples (owned by Bob)	**#Model Parameters** (owned by Alice)		**Passive**		**Active**	
Binary	200	47 (LR)		π_{DP}	π_{EOP}	π_{DP}	π_{EOP}
			2PC	3.34 sec	3.36 sec	239.24 sec	238.69 sec
			3PC	1.37 sec	1.67 sec	6.41 sec	6.43 sec
Multi-class	56	1.48M (ConvNet)		π_{EOD}	π_{GACC}	π_{EOD}	π_{GACC}
			2PC	5,866.47 sec	5,866.46 sec	42,745.09 sec	42,742.00 sec
			3PC	29.92 sec	29.84 sec	199.02 sec	198.99 sec

the fairness metrics given the labels is near real-time and practical.

Our runtime results for the time-consuming inference (the first step in the fairness auditing protocols) align with the observations in the MPC literature [108, 4, 224] where there is also a substantial difference between protocols for passive and active adversarial settings. Providing security in the presence of active or *"malicious"* adversaries, i.e. ensuring that no such adversarial attack can succeed, comes at a much higher computational cost than in the passive case. The same holds for a dishonest-majority 2PC setting where each party (server) only trusts itself, versus a much faster 3PC honest-majority setting. We note that unlike in privacy-preserving inference applications, where fast responsiveness of the system can be of utmost importance for practical applications [108, 224], in fairness auditing use cases such as the ones we consider in this chapter, one can typically afford longer runtimes, justifying the use of the most stringent security settings.

All the above results carry over to any similar, same sized audit data sets and models. With increase in the size of the data set or the number of parameters of the model, the times for execution of the MPC protocols will increase linearly [224].

The price paid to preserve privacy during fairness auditing is an increase in runtime, which stems mostly from the computational and communication cost of the MPC protocols used for labeling the audit instances in a privacy-preserving manner. We argue that this is a reasonable price to pay when working with sensitive data. Indeed, unlike in inference applications – such as real-time speech recognition or video classification for surveillance – where fast responsiveness of the system is of utmost importance, in fairness auditing use cases with sensitive data one can typically afford longer runtimes, justifying the use of the most stringent security settings, even if they come at a higher cost.

5.5 Discussion

The current open-source implementation of PrivFair includes MPC protocols for the well established fairness metrics of equalized odds, equal opportunity, subgroup accuracy, and demographic parity. The fairness auditing protocols can be selected based on their suitability

to the application where fairness is to be guaranteed. PrivFair can be easily used and extended to report other statistical notions of fairness [191] such as treatment equality [217] and predictive equality [231]. Furthermore, while we have demonstrated the use of PrivFair for auditing of logistic regression and ConvNet models, the fairness auditing protocols can be used for any kind of model architecture for which an MPC protocol for secure inference is available, including decision tree models, random forests, support vector machines, and other kinds of neural networks [220, 221, 222, 223, 108].

The applicability and usefulness of PrivFair stretches well beyond the applications considered in our experiments. AI services that deal with sensitive data such as in healthcare, banking, predictive policing etc. not only need to protect data but also ensure that unbiased AI services are available to all. PrivFair is a first step where AI developers and AI end users can collaborate to aim for unbiased AI services, while protecting their data. As such, the impact of using PrivFair can be significant in all fields where ML models appear for automated decision making, e.g. education, housing, law-enforcement, healthcare, and banking, as well as new application domains yet to be discovered.

5.6 Summary

We presented PrivFair, a first-of-its-kind library for privacy-preserving fairness audits of ML models. We developed MPC protocols to compute the components of the confusion matrix. Building on these sub-protocols, we then designed protocols to compute metrics for group fairness that aid in auditing group fairness violations in ML-based systems.

These protocols in PrivFair allow an investigator *Bob* with audit data to evaluate the fairness of a model owner *Alice*'s ML model without requiring *Bob* or *Alice* to disclose their inputs to anyone in an unencrypted manner. To protect both the trained model parameters and the audit data, we use MPC. Specifically, MPC-as-a-Service enables 2PC scenarios in which *Alice* and *Bob* act as the 2 MPC servers themselves, as well as nPC scenarios in which *Alice* and *Bob* outsource the fairness audit to untrusted servers in the cloud. The latter service can be adapted to be used not only for synchronous but also asynchronous

MPC computations, where servers each store the corresponding secret-shares of the models that are to be investigated and an external investigator can choose the desired model to investigate.

Chapter 6

ACHIEVING GROUP FAIRNESS IN PRIVATE CROSS-DEVICE SETTINGS

In the previous chapter, we examined how group unfairness can be detected in AI systems through auditing. As explained in Chapter 4, group fairness – a widely studied notion of fairness and one often mandated by regulatory bodies in AI applications [232] – aims to ensure that ML model predictions are not biased against certain groups of people, as defined by sensitive attributes such as race, gender, or age [191].

After detecting bias, the next step is to apply methods for mitigation. Developing group-fair ML models in the centralized learning setup where a single entity has access to all data has been well studied in the literature [193, 208, 233]. In a lot of applications however, data originates from many different data holders who – out of privacy concerns – may not wish to, or may legally not be allowed to, disclose their data to a central entity (i.e. distributed or federated settings). This creates challenges for training fair ML models, particularly in Federated Learning (FL) settings, where raw data remains decentralized and does not leave the client's site (see Section 2.1 in Chapter 2). Fair model training in FL is challenging due to the intrinsic conflict between bias mitigation algorithms that often require access to data across clients, while FL is explicitly designed to protect privacy by preventing such access [234].

The emerging literature on achieving group fairness in federated settings using FL, at the time of this research, largely assumes a cross-silo setup. In this setup, each client holds data for multiple sensitive attribute values (e.g. each client represents a bank with data about both female and male customers) [235, 236, 237, 238]. These approaches often achieve fairness by sacrificing some degree of privacy. However, many FL applications naturally exhibit

a cross-device federated setup with horizontally partitioned data, where each client in the federation has one particular value for the sensitive attribute (e.g. a female client, using a mobile recommending application, will have user-item interaction data belonging only to the female group). At the time of this research, there was limited literature on achieving group fairness with strong privacy guarantees in cross-device settings.

To address this gap, in this chapter we explore how DP-in-MPC can be used to mitigate group unfairness while offering stronger privacy guarantees than traditional FL approaches, as discussed in Section 6.2. To this end, we introduce `PrivFairFL`, a privacy-preserving technique designed to enforce group fairness while training models using the traditional FL paradigm, particularly in cross-device scenarios. We do note that `PrivFairFL` can be easily adapted to both cross-silo and cross-device FL settings.

After defining the metrics we use in this chapter in Section 6.1, we briefly review existing literature in Section 6.2. We then develop MPC protocols for pre- and post-processing techniques in Section 6.3. We present empirical results in Section 6.4 and finally summarize the chapter in Section 6.5.

6.1 Group Fairness

Group fairness measures how balanced the predicted outcomes are across the groups defined by the sensitive attribute S. Many different notions of group fairness have been proposed in the literature [193, 197, 208, 194, 239].

6.1.1 Metrics

In this thesis, we consider popular statistical notions of fairness. These, as mentioned in Chapter 5, rely on computing the true positive rate (TPR_p and TPR_u), false positive rate (FPR_p and FPR_u), and the number of true positives (TP_p and TP_u) for the privileged (p) and unprivileged groups (u) respectively. We briefly outline the specific fairness metrics used in this chapter below, where N_p and N_u denote the number of samples for privileged and unprivileged group respectively. The lower the values of these metrics, the fairer the

predictions made by the model.

- **Disparate Impact (DI)** measures discrimination in the predictions by computing the recall (TPR) for the groups [29]. A value DI = 1 indicates discrimination-free predictions. We report the degree of discrimination using $|1 - \text{DI}|$.

$$|1 - \text{DI}| = |1 - \max(\text{TPR}_u/\text{TPR}_p, \text{TPR}_p/\text{TPR}_u)|$$

- **Equal opportunity (EOP)** considers the fairness of the predictions from the perspective of TPR [208]. This metric focuses only on the positive or advantaged outcome. We report this metric as equal opportunity difference (ΔEOP) by computing the difference between the TPR of the two groups.

$$\Delta\text{EOP} = |\text{TPR}_p - \text{TPR}_u|$$

- **Equalized odds (EODD)** considers the predictions fair if $\hat{Y}$ and S are independent conditional on Y [208]. This implies that both TPR and FPR are equal between the groups. We report this metric as average odds difference (ΔEODD).

$$\Delta\text{EODD} = 0.5 \cdot (|\text{TPR}_p - \text{TPR}_u| + |\text{TPR}_p - \text{TPR}_u|)$$

- **Statistical parity (SP)** considers the predictions as fair if the number of positive predictions are the same for the two groups [193]. We report statistical parity difference (ΔSP) as the difference between the ratio of positive outcomes per group.

$$\Delta\text{SP} = |(\text{TP}_p/\text{N}_p) - (\text{TP}_u/\text{N}_u)|$$

Recall from Section 4.2 in Chapter 4 that existing unfairness mitigation techniques can be categorized into three categories based on the stage in the ML pipeline they are incorporated into, namely, pre-processing, in-processing and post-processing. Unlike in-processing, the pre- and post-processing techniques are independent of the notions of fairness, learning objective, and the model being trained.

`PrivFairFL` extends pre- and post-processing to FL. Model is training done using the traditional FL paradigm. For pre-processing, we adopt a reweighting method that adjusts the loss function to penalize incorrect predictions based on weights assigned to training samples, therefore encouraging the model to learn a fair predictor across groups [240]. Specifically, it assigns different weights to data from different groups to create a balanced training distribution. For post-processing, we employ the threshold optimization technique from [208], which identifies group-specific decision thresholds by constructing ROC curves that satisfy predefined objectives.

6.2 Literature on Techniques for Group Fairness in FL

Group fairness in FL. Extending bias mitigation techniques from the centralized paradigm to FL is challenging due to an intrinsic conflict between fair model training and FL [234]: (i) evaluating the fairness of a model, or mitigating bias, requires access to the data of all clients; and (ii) FL aims at preserving data privacy by *not* giving such access. As Table 6.1 shows, methods for training group-fair models have been proposed for cross-silo setups (each client has data for multiple sensitive attribute values), and for cross-device setups (each client has data for only one sensitive attribute value). Nearly all methods are based on in-processing, hence tailored to a specific training algorithm, model architecture, and fairness notion [238, 241, 237]. Furthermore, as the "privacy" column in Table 6.1 indicates, most of the emerging literature on group-fair FL tries to work around the conflict between (i) and (ii) by sacrificing privacy for fairness. Information leaks can occur when the clients send updated model parameters, gradients, fairness metrics, or the values of sensitive attributes to the aggregator, or by analyzing the aggregated outputs. For example, during FL train-

ing, assuming that an adversary A has the model from the previous round and the gradient updates from the current round, A can infer a private training example [59]. Most works do not take into account such information leaks in FL [242, 243, 244, 245]. A can also analyze the aggregated outputs to infer knowledge about a particular client. FairFed [246] and Rodriguez et al. [247] employ SecAgg [102] to protect data leaks from information sent by the clients, but fail to protect the aggregated values. Though Rodriguez et al. [247] use a combination of MPC and DP, they reveal the aggregated gradients to the computing servers and add noise in-the-clear (i.e. in an unencrypted manner) to publish DP aggregates to the FL aggregator. Similarly, [248] protect only the discrimination indices sent by clients and fail to protect any aggregated output and gradient updates.

Private aggregation techniques. Statistics about the underlying data distribution are commonly used to improve the ML model learning process. Various works have shown that statistics about the data distribution across clients in FL can improve the utility of models or make them more fair, especially for clients with imbalanced or non-i.i.d. data [250, 237]. The challenge is to collect the statistics without infringing upon the clients' privacy. Solutions for aggregation with MPC to protect the private input data have been recently employed in FL to train group-fair models [246, 248]. However, the output of such MPC-based aggregations can still leak information about the client's data. In `PrivFairFL` we go a step further by adding noise to the aggregated values to provide DP guarantees. Existing works for aggregation do so by having the clients participate in the noise generation. Such solutions are not resilient to malicious clients, and require extensive communication between clients and the aggregator [251, 252]. Our proposal correctly generates the noise in a secure way, inside MPC protocols that are resilient to corruptions by semi-honest and malicious adversaries (see Section 6.3).

Table 6.1: Related work on Group Fairness in FL

(* indicates that `PrivFairFL` can be extended to cross-silo as discussed in Section 6.3.3)

Paper	**Scenario**		**Privacy**		**Mitigation Alg.**		
	silo	dev	DP	MPC	Pre	In	Post
Abay [236]	✓		✓	✗	✓	✓	✗
AgnosticFair [237]	✓		✗	✗	✗	✓	✗
FCFL [238]	✓		✗	✗	✗	✓	✗
Zhang [241]	✓		✗	✗	✗	✓	✗
FairFL [248]	✓		✗	✓	✗	✓	✗
FPFL [249]	✓		✓	✗	✗	✓	✗
Rodriguez [247]	✓	✗	✓	✓	✗	✓	✗
Kanaparthy [244]		✓	✗	✗	✗	✓	✗
GI-FAIR [243]		✓	✗	✗	✗	✓	✗
FADE [245]		✓	✗	✗	✗	✓	✗
FairFed [246]	✓	✓	✗	✓	✗	✓	✗
Papadaki [242]	✓	✓	✗	✗	✗	✓	✗
PrivFairFL	*	✓	✓	✓	✓	*	✓

6.3 PrivFairFL: Privacy-Preserving Group Fairness

We consider a federation with M clients, in which each client k ($k = 1 \dots M$) holds a dataset D_k with n_k training samples. $< X_{ik}, s_{ik}, y_{ik} >$ represents the i^{th} training sample held by the k^{th} client where s_{ik} is the value of a sensitive attribute, y_{ik} is the value of a class label, and X_{ik} is the set of remaining feature values. $N = \sum_{k=1}^{M} n_k$ denotes the total number of samples. S denotes the sensitive attribute and Y denotes the class label. For the sake of

simplicity, we focus on the case where S represents a binary sensitive attribute that takes u (representing the unprivileged group) and p (representing the privileged group) as its values, and Y takes 0 and 1 as its values. Our proposed methods, however, can be extended to the case of multiple sensitive attributes, including non-binary, and multi-class classification (similar to how group fairness metric computations were extended in Chapter 5).

We use $C(s, y)$ to denote the count of all training samples from all the clients with sensitive attribute value s and class label y:

$$C(s,y) = \sum_{k=1}^{M} \sum_{i=1}^{n_k} \#\{< X_{ik}, s_{ik}, y_{ik} > \mid s_{ik} = s \wedge y_{ik} = y\} \tag{6.1}$$

For example, $C(u, 1)$ is the number of training samples from all clients belonging to the unprotected group and with class label 1. The model parameters of client k are denoted as θ_k and the aggregated model parameters are represented by θ. The model θ makes predictions $\hat{Y}$.

For MPC-as-a-Service in DP-in-MPC, we consider $r \geq 2$ computing (MPC) servers that execute MPC protocols and perform other computations to aid the clients in model training in traditional FL paradigm.

We propose two strategies for bias mitigation, namely one that is applied before model training (`PrivFairFL-Pre`) and one that is applied after model training (`PrivFairFL-Post`). Both strategies are independent of the model training phase, which means that they can be combined with any technique for model training in FL, including DP-SGD [76]. While all r servers participate in the MPC protocols for bias mitigation, the aggregation of the weights or gradients during model training can either be performed by one of the servers – which then acts like the traditional central aggregator in traditional FL – or it can be implemented as a straightforward MPC protocol run by all the servers. The methods described in this section are generic and work with either aggregation setup during the model training phase.

Each strategy is based on the clients in the federation sending secret-shares of their private information to the r computing servers in an MPC-as-a-Service setting. These servers (1) run MPC protocols to perform computations for bias mitigation, including an MPC subprotocol

to add secret-shared noise to the secret-shared results of those computations to provide the desired DP guarantees[1], and (2) publish the outcomes of this process back to the clients. The two strategies differ in when the MPC protocols are executed (before or after model training), what the input and output is, and what computations are being performed. Performing bias mitigation inside MPC protocols yields the same utility and fairness as one could achieve with global DP, with the added advantage that the clients do not need to disclose their data to a trusted entity that curates the data.

6.3.1 Privacy-Preserving Pre-processing for Bias Mitigation

In `PrivFairFL-Pre`, the unified training dataset D is debiased by assigning weights to the samples based on the values of S and/or Y. Such global assignment of weights on the unified data addresses the heterogeneous data distributions in cross-device setups.

The reweighing technique, in the centralized paradigm, assigns a weight of $1/C(s, y)$ to the the training instance with sensitive attribute value s and label y, where $C(s, y)$ is the total number of examples in the training data with that sensitive attribute and label value [240]. We adopt this idea in protocol π_{RW}.

In FL, each client k has its local dataset D_k, and needs to obtain weights for the instances in D_k that are based on counts $C(s, y)$ across the entire federation. To obtain these weights, each client k starts by counting the number of negative and positive instances in D_k. We denote these local counts as $LC(0, k)$ and $LC(1, k)$ respectively, for $k = 1 \dots M$. Next each client secret-shares these local counts to the set of MPC servers P_i $(i = 1 \dots r)$, along with secret-shares of $T(k)$ which denotes whether client k belongs to the protected group $T(k) = 1$ or to the unprotected group $T(k) = 0$. None of the MPC servers can derive any information about the local counts or sensitive attribute values from the received secret-shares .

In Line 1–10 in Protocol 7, the MPC servers compute secret-shares of $C(s, y)$, for all

[1]Throughout this chapter we consider event-level DP, i.e. an entry or instance corresponds to one training example of one client in the federation, and D holds the data of all clients combined, i.e. $D = \cup_{k=1}^{M} D_k$.

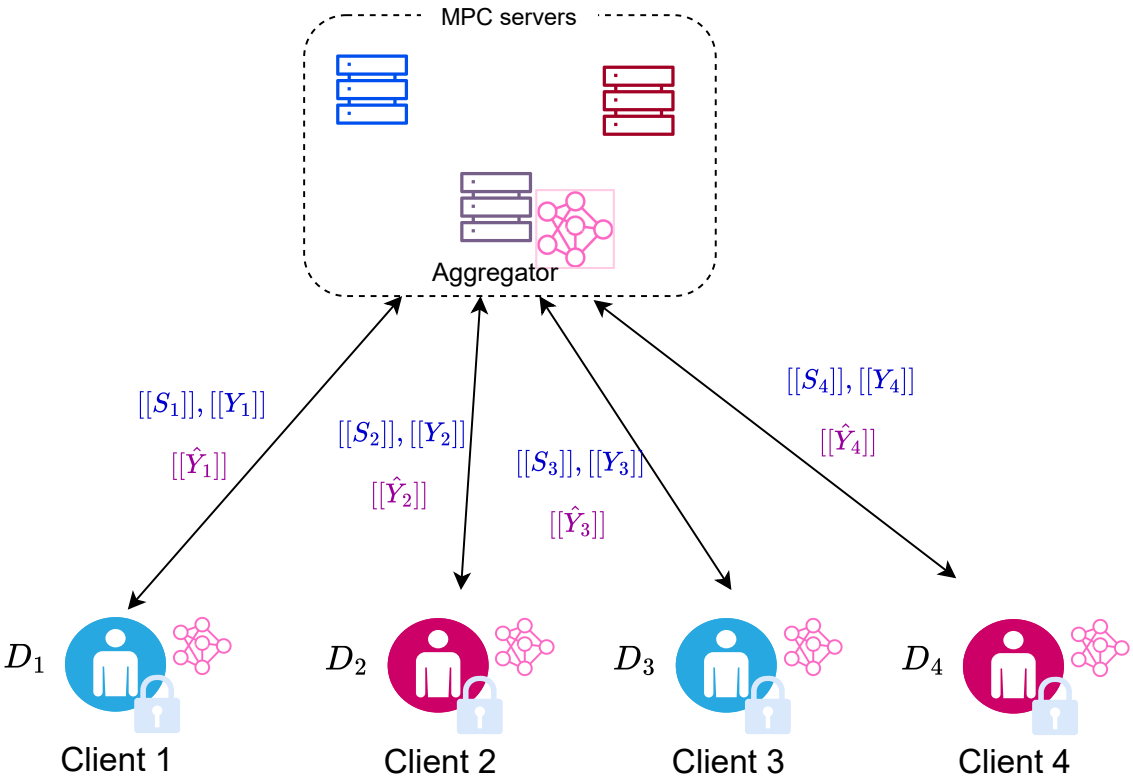


Figure 6.1: `PrivFairFL`. Clients secret-share their sensitive attributes S, ground truth label Y and predicted label $\hat{Y}$ with MPC servers. The MPC servers then run MPC protocols for pre- and post-processing group unfairness mitigation methods. These protocols leverage DP-in-MPC to compute secret-shares of statistics and ROC curves, and add noise to the computations.

Protocol 7: π_{RW} for privacy-preserving reweighing of training data

Input : Total number M of clients; secret-shares of vector T of length M denoting whether client k belongs to the protected group ($T(k) = 1$) or to the unprotected group ($T(k) = 0$); secret-shares of $2 \times M$ matrix LC with local counts of number of negative instances $LC(0, k)$ and positive instances $LC(1, k)$ for client k; ϵ privacy budget allotted for bias mitigation

Output: Secret-shares of sample weights for negative and positive instances in protected and unprotected groups

for s in $\{p, u\}$ and y in $\{0, 1\}$ **do**
    Set $C(s, y)$ to 0.
end
for $k \leftarrow 1$ **to** M **do**
    for y in $\{0, 1\}$ **do**
        $[\![LCPr(y)]\!] \leftarrow \pi_{\mathsf{MUL}}([\![T(k)]\!], [\![LC(y, k)]\!])$
        $[\![C(p, y)]\!] \leftarrow [\![C(p, y)]\!] + [\![LCPr(y)]\!]$
        $[\![C(u, y)]\!] \leftarrow [\![C(u, y)]\!] + [\![LC(y, k)]\!] - [\![LCPr(y)]\!]$
    end
end
for s in $\{p, u\}$ and y in $\{0, 1\}$ **do**
    $[\![C(s, y)]\!] \leftarrow [\![C(s, y)]\!] + \pi_{\mathsf{LAP}}\ (1/\epsilon)$
end
Set N' to 0.
for s in $\{p, u\}$ and y in $\{0, 1\}$ **do**
    $[\![N']\!] \leftarrow [\![N']\!] + [\![C(s, y)]\!]$
end
for s in $\{p, u\}$ and y in $\{0, 1\}$ **do**
    $[\![W(s, y)]\!] \leftarrow \pi_{\mathsf{DIV}}([\![N']\!], 4 \cdot [\![C(s, y)]\!])$
end
return $[\![W(s, y)]\!]$

values of s and y, from the secret-shares of the clients' local counts $LC(y, k)$.

As is common in MPC protocols, we use multiplication instead of control flow logic for

conditional assignments. In this way, the number and the kind of operations executed by the servers does not depend on the actual values of the inputs, so it does not leak information that could be exploited by side-channel attacks.

For instance, on (Line 7–8) we rewrite

$$\textbf{if } T(k) = 1 \textbf{ then } C(p, y) \leftarrow C(p, y) + LC(y, k) \textbf{ else } C(p, y) \leftarrow C(p, y)$$

as

$$C(p, y) \leftarrow C(p, y) + T(k) \cdot LC(y, k)$$

Similarly,

$$\textbf{if } T(k) = 0 \textbf{ then } C(u, y) \leftarrow C(u, y) + LC(y, k) \textbf{ else } C(u, y) \leftarrow C(u, y)$$

is rewritten as

$$C(u, y) \leftarrow C(u, y) + (1 - T(k)) \cdot LC(y, k)$$

.

The last expression can be rewritten as $C(u, y) + LC(y, k) - T(k) \cdot LC(y, k)$, allowing us to take advantage of the already computed multiplication in Line 6 to arrive at Line 8.

The counts $C(s, y)$ depend on disjoint subsets of the unified dataset D (4 such disjoint subsets, given that the sensitive attribute and the class label are binary). The results of the counts can be made public under ϵ-DP guarantee with the Laplace mechanism. The sensitivity of each count query is 1 resulting in a Laplace noise with magnitude $1/\epsilon$. Note that because counts are computed over disjoint subsets with event-level privacy, the parallel composition property of DP is applicable. To add such noise, on Line 12, the servers call protocol π_{LAP} to obtain a secret-shared value sampled from the Laplace distribution with mean 0 and median $1/\epsilon$. Pseudocode for π_{LAP} is provided separately in Protocol 8.

All the remaining computation in Protocol 7 depend only on the now noisy counts. Because of DP's post-processing property, the outcomes of those computations also satisfy

Protocol 8: π_{LAP} for secure sampling from Laplacian distribution

Input: Scale b

Output: Secret-shared value $[\![x]\!]$ drawn from Laplace distribution with mean 0 and scale b

1 $[\![u]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(-0.5, 0.5)$

2 $[\![\mathrm{sgn}_u]\!] \leftarrow \pi_{\mathsf{GTE}}([\![u]\!], 0)$

3 $[\![\mathrm{abs}_u]\!] \leftarrow \pi_{\mathsf{MUL}}([\![u]\!], [\![\mathrm{sgn}_u]\!]))$

4 $[\![\ln_u]\!] \leftarrow \pi_{\mathsf{LN}}(1 - 2 \cdot [\![\mathrm{abs}_u]\!])$

5 $[\![x]\!] \leftarrow -b \cdot \pi_{\mathsf{MUL}}([\![\ln_u]\!], [\![\mathrm{sgn}_u]\!])$

6 **return** $[\![x]\!]$

ϵ-DP. The computing servers then execute the MPC-protocol for division π_{DIV} to compute secret-shares of the weights on Line 19. The multiplication by the constant 4 (which corresponds to the number of sensitive attribute values times the number of label values) and by N' serves to rescale the weights so that they sum up to N'. Here, N' represents the sum of the noisy counts $C(s, y)$ for all combinations of s and y, and likely differs from the true total number N of training examples.

At the end of the protocol, the computing servers hold secret-shares of the computed weights, protected under DP guarantees. These weights are then published so that clients can use them for model training. π_{RW} not only protects the values of the clients' labels and sensitive attributes but also the value of the noise added to the computed weights.

Sampling from a Laplace distribution $Lap(0, b)$ with zero mean and scale b in a privacy-preserving manner is achieved by letting the servers execute protocol π_{LAP} to compute a secret-sharing of $x = -b \cdot \mathrm{sgn}(u) \cdot \ln(1 - 2|u|)$, where u is a random variable drawn from a uniform distribution in $[-0.5, 0.5]$[2]. To obtain a secret-sharing of u, on Line 1 in in Protocol 8, the servers execute $\pi_{\mathsf{GR-RANDOM}}$. In $\pi_{\mathsf{GR-RANDOM}}$, each server generates d random bits, where

[2] x has distribution $Lap(0, b)$. This follows from the inverse cumulative distribution function for $Lap(0, b)$.

d is the fractional precision of the power 2 ring representation of real numbers. Then the servers define the bitwise XOR of these d bits as the binary representation of the jointly generated random number. The rest of π_{LAP} is fairly straightforward. π_{LN} called on Line 4 is a known MPC protocol for computing the natural logarithm of a secret-shared value [110].

Extending π_{RW} to other reweighing algorithms. Protocol π_{RW} can be easily extended to other reweighing techniques [195] by computing the required statistics and adding noise to them before Line 19, and replacing Lines 11–20 to compute the weights according to the technique. For example, for reweighing techniques that balance based only on Y, the servers would compute $C(y) = \sum_s C(s, y)$ after Line 10 (which is straightforward using addition of secret-shares), which would be followed by a for-loop that iterates over y and computes $[\![W(y)]\!] \leftarrow \pi_{\mathsf{DIV}}(N', 2 \cdot [\![C(y)]\!])$.

Protocol π_{RW} can be easily extended to support other reweighing techniques [195] by computing the required statistics and adding noise to them before Line 19, and by replacing Lines 11–20 to compute the weights according to the chosen technique. For example, in reweighing methods that balance based only on Y, the servers would compute

$$C(y) = \sum_s C(s, y)$$

after Line 10 (which can be done straightforwardly using addition of secret- shares). This would be followed by a for-loop that iterates over each y and computes the weights as:

$$[\![W(y)]\!] \leftarrow \pi_{\mathsf{DIV}}(N', 2 \cdot [\![C(y)]\!]).$$

6.3.2 Privacy-Preserving Post-processing for Bias Mitigation

In `PrivFairFL-Post`, the predicted *outcomes* are debiased by finding optimal classification thresholds for each group. In the centralized paradigm, this is done by constructing ROC-curves to find thresholds that maximize the given objective [214, 208]. This requires labels and predictions over the unified training data which violates privacy in the FL scenario. Below we detail a solution for obtaining ROC-curves and optimal classification thresholds in

a privacy-preserving manner in FL. `PrivFairFL-Post` is applied *after* the training phase, so we assume that each client has obtained the last model θ. Next:

1. Each client k generates predictions (probabilities) with θ over its local training dataset. Subsequently each client secret-shares its sensitive attribute value, the predicted probabilities, and the ground truth labels for its local training examples with the MPC servers . None of the MPC servers can learn of these private inputs from the secret-shares received.

2. The MPC servers run protocol π_{ROC} (see Protocol 9) to construct secret-shares of the ROC curves for the protected and the unprotected group. Furthermore, noise is added inside the MPC protocol so that the ROC curves can be made public with ϵ-DP guarantees.

3. All MPC servers send their secret-shares of the noisy ROC curves to one of the MPC servers, which then computes optimal thresholds, and sends these to the clients for further fair inference.

In protocol π_{ROC}, the servers construct ROC curves by computing secret-sharings of FPR and TPR values for a list of predefined candidate thresholds (Line 1 in Protocol 9). On Line 3–19, they compute secret-shares of the TPR and FPR at each threshold, for the protected and the unprotected group. On Line 5–7, the servers use the comparison protocol π_{GTE} to determine for each training instance whether it would be classified as positive or negative based on the jth threshold. This information, along with the ground truth labels Y and the sensitive attribute values S is then passed to a subprotocol π_{CF} that returns secret-shares of number of TP, TN, FP and FN for each group based on the jth threshold. The code for π_{CF} is provided separately as Protocol 10.

The sensitivity of each of the returned counts is 1, and they are based on disjoint subsets of the data D, so to provide ϵ-DP, on Line 10–11 the servers draw secret-shared noise from

Protocol 9: π_{ROC} for privacy-preserving debiasing of predicted outcomes

Input : Total number of samples N as used for FL , vectors of secret-shares of true labels Y, predicted probabilities $\hat{Y}_{prob}$ and S each of length N; privacy budget ϵ allotted for bias mitigation

Output: Secret-shares of ROC curves for protected and unprotected groups

$[\![\text{thresholds}]\!] \leftarrow [\![(0.000, 0.001, 0.002, \ldots, 0.999, 1.000]\!]$; Set T to 1001

Declare $[\![\text{ROC}(p)]\!]$ and $[\![\text{ROC}(u)]\!]$ as secret-shared $3 \times T$ arrays.

Declare $\hat{Y}_{th}$ as an N-dimensional vector *holds predictions at thresholds*

for $j \leftarrow 1$ **to** T **do**

  for $i \leftarrow 1$ **to** N **do**

    $[\![\hat{Y}_{th}[i]]\!] \leftarrow \pi_{\mathsf{GTE}}([\![\hat{Y}_{prob}[i]]\!], [\![\text{thresholds}[j]]\!])$

  end

  $[\![\text{TP}(p)]\!], [\![\text{TN}(p)]\!], [\![\text{FP}(p)]\!], [\![\text{FN}(p)]\!], [\![\text{TP}(u)]\!], [\![\text{TN}(u)]\!], [\![\text{FP}(u)]\!], [\![\text{FN}(u)]\!] \leftarrow \pi_{\mathsf{CF}}([\![Y]\!], [\![\hat{Y}_{th}]\!], [\![S]\!])$

  for s in $\{p, u\}$ **do**

    $[\![\text{TP}(s)]\!] \leftarrow [\![\text{TP}(s)]\!] + \pi_{\mathsf{LAP}}\ (1/\epsilon)$; $[\![\text{TN}(s)]\!] \leftarrow [\![\text{TN}(s)]\!] + \pi_{\mathsf{LAP}}\ (1/\epsilon)$

    $[\![\text{FP}(s)]\!] \leftarrow [\![\text{FP}(s)]\!] + \pi_{\mathsf{LAP}}\ (1/\epsilon)$; $[\![\text{FN}(s)]\!] \leftarrow [\![\text{FN}(s)]\!] + \pi_{\mathsf{LAP}}\ (1/\epsilon)$

    $[\![\text{TPR}(s)]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{TP}(s)]\!], [\![\text{TP}(s)]\!] + [\![\text{FN}(s)]\!])$

    $[\![\text{FPR}(s)]\!] \leftarrow \pi_{\mathsf{DIV}}([\![\text{FP}(s)]\!], [\![\text{FP}(s)]\!] + [\![\text{TN}(s)]\!])$

    $[\![\text{ROC}(s)[1, j]]\!] \leftarrow [\![\text{FPR}(s)]\!]$

    $[\![\text{ROC}(s)[2, j]]\!] \leftarrow [\![\text{TPR}(s)]\!]$

    $[\![\text{ROC}(s)[3, j]]\!] \leftarrow [\![\text{thresholds}[j]]\!]$

  end

end

return $[\![\text{ROC}(p)]\!]$, $[\![\text{ROC}(u)]\!]$

$Lap(0, 1/\epsilon)$ with π_{LAP} (cfr. Protocol 8) and add it to the counts. All MPC servers then send their secret-shares of the lists of FPRs, TPRs, and corresponding thresholds to one of the MPC servers that finds the optimal classification threshold for each group, following the procedure in [214].

Description of π_{CF}. π_{CF}, provided separately as Protocol 10, is built on the logic developed in Section 5.3 in Chapter 5. It is designed to minimize the number of multiplications and to avoid control flow statements (`if-then-else`), which would otherwise make the number of instructions depend on the input values.

For protocol π_{CF}, the MPC servers receive as input secret-shares of a vector Y with ground truth labels, secret-shares of a vector $\hat{Y}$ with predicted labels, and secret-shares of a vector S with sensitive attribute values. All vectors have length N, which is the total number of instances. The MPC servers execute protocol π_{CF} to compute secret-shares of TP, TN, FP, and FN for each group, i.e. the protected group p and the unprotected group s.

We use the logic in the truth table shown in Table 6.2 to compute the values of TP, FN, FP, and TN and replace control flow statements (if-then-else) by multiplications, as commonly done in MPC to make the number and kind of computations done during protocol execution independent of specific values of the data. Multiplying the columns in Table 6.2 with the value $S[i]$ of the sensitive attribute as in the equations below, gives the contributions of instance i to the TP, FN, FP, and TN scores of the protected and unprotected groups:

We expand the above 8 equations and rewrite them as in Table 6.3 by precomputing the common products

$$\text{tp} = Y[i] \cdot \hat{Y}[i], \quad \text{ys} = Y[i] \cdot S[i], \quad \text{ps} = \hat{Y}[i] \cdot S[i], \quad \text{tps} = \text{tp} \cdot S[i]$$

for each instance to reduce the number of multiplications. Here $S[i] \in \{u, p\}$ is mapped to $S[i] \in \{0, 1\}$ for mathematical computations ($u = 0$ and $p = 1$).

Following the above logic, the servers, on Lines 4–17 of protocol π_{CF}, compute the contribution of an instance towards TP, FN, FP, or TN for the protected or unprotected group.

Protocol 10: π_{CF} for generation of confusion matrix for protected & unprotected group

Input : Total number of samples as used for FL N, secret-shares of vectors Y, $\hat{Y}$ and S of length N

Output: Secret-shares of confusion matrix for each group

1 **for** s in $\{p, u\}$ **do**

2 Set TP(s), FP(s), TN(s), FN(s) to 0

3 **end**

4 **for** $i \leftarrow 1$ **to** N **do**

5 $[\![\text{tp}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![Y[i]]\!], [\![\hat{Y}[i]]\!])$

6 $[\![\text{ys}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![Y[i]]\!], [\![S[i]]\!])$

7 $[\![\text{ps}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\hat{Y}[i]]\!], [\![S[i]]\!])$

8 $[\![\text{tps}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{tp}]\!], [\![S[i]]\!])$

9 $[\![\text{TP}(p)]\!] \leftarrow [\![\text{TP}(p)]\!] + [\![\text{tps}]\!]$

10 $[\![\text{TP}(u)]\!] \leftarrow [\![\text{TP}(u)]\!] + ([\![\text{tp}]\!] - [\![\text{tps}]\!])$

11 $[\![\text{FP}(p)]\!] \leftarrow [\![\text{FP}(p)]\!] + ([\![\text{ps}]\!] - [\![\text{tps}]\!])$

12 $[\![\text{FP}(u)]\!] \leftarrow [\![\text{FP}(u)]\!] + ([\![\hat{Y}[i]]\!] - [\![\text{ps}]\!] - [\![\text{tp}]\!] + [\![\text{tps}]\!])$

13 $[\![\text{FN}(p)]\!] \leftarrow [\![\text{FN}(p)]\!] + ([\![\text{ys}]\!] - [\![\text{tps}]\!])$

14 $[\![\text{FN}(u)]\!] \leftarrow [\![\text{FN}(u)]\!] + ([\![Y[i]]\!] - [\![\text{ys}]\!] - [\![\text{tp}]\!] + [\![\text{tps}]\!])$

15 $[\![\text{TN}(p)]\!] \leftarrow [\![\text{TN}(p)]\!] + ([\![S[i]]\!] - [\![\text{ys}]\!] - [\![\text{ps}]\!] + [\![\text{tps}]\!])$

16 $[\![\text{TN}(u)]\!] \leftarrow [\![\text{TN}(u)]\!] + (1 - [\![S[i]]\!] - [\![Y[i]]\!] + [\![\text{ys}]\!] - [\![\hat{Y}[i]]\!] + [\![\text{ps}]\!] + [\![\text{tp}]\!] - [\![\text{tps}]\!])$

17 **end**

18 **return** $([\![\text{TP}(s)]\!], [\![\text{TN}(s)]\!], [\![\text{FP}(s)]\!], [\![\text{FN}(s)]\!] \mid s \in \{u, p\})$

On Lines 5–8, the servers precompute the secret-shares of the common products (tp, ys, ps and tps) for each instance to reduce the number of multiplications.

On Lines 9–16, the servers compute the secret-shares of TP, FN, FP, or TN based on the formulas in Table 6.3 for the protected and unprotected groups.

Table 6.2: Logic for evaluating TP, FN, FP, and TN

$Y[i]$	$\hat{Y}[i]$	true pos. $Y[i] \cdot \hat{Y}[i]$	false neg. $Y[i] \cdot (1 - \hat{Y}[i])$	false pos. $(1 - Y[i]) \cdot \hat{Y}[i]$	true neg. $(1 - Y[i]) \cdot (1 - \hat{Y}[i])$
1	1	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
0	0	0	0	0	1

$$\text{TP(p)} = \sum_{i=1}^{N} Y[i] \cdot \hat{Y}[i] \cdot S[i] \qquad \text{TP(u)} = \sum_{i=1}^{N} Y[i] \cdot \hat{Y}[i] \cdot (1 - S[i])$$
$$\text{FP(p)} = \sum_{i=1}^{N} (1 - Y[i]) \cdot \hat{Y}[i] \cdot S[i] \qquad \text{FP(u)} = \sum_{i=1}^{N} (1 - Y[i]) \cdot \hat{Y}[i] \cdot (1 - S[i])$$
$$\text{FN(p)} = \sum_{i=1}^{N} Y[i] \cdot (1 - \hat{Y}[i]) \cdot S[i] \qquad \text{FN(u)} = \sum_{i=1}^{N} Y[i] \cdot (1 - \hat{Y}[i]) \cdot (1 - S[i])$$
$$\text{TN(p)} = \sum_{i=1}^{N} (1 - Y[i]) \cdot (1 - \hat{Y}[i]) \cdot S[i] \qquad \text{TN(u)} = \sum_{i=1}^{N} (1 - Y[i]) \cdot (1 - \hat{Y}[i]) \cdot (1 - S[i])$$

Table 6.3: Formulas for computing TP, FN, FP, and TN for protected and unprotected groups per instance

	Protected group	Unprotected group
TP	tps	tp − tps
FP	ps − tps	$\hat{Y}[i]$ − ps − tp + tps
FN	ys − tps	$Y[i]$ − tp − ys + tps
TN	$S[i]$ − ys − ps + tps	1 − $S[i]$ − $Y[i]$ + ys − $\hat{Y}[i]$ + ps + tp − tps

6.3.3 Extensions to `PrivFairFL`

Extending `PrivFairFL` to cross-silo setups is straightforward. In `PrivFairFL-Pre`, the clients will then instead secret-share local counts $LC(y, k, s)$ for *each* value of S (the count can be 0 if a certain sensitive attribute value is not present with the client), requiring only a small modification in the code for π_{RW} (Protocol 7). `PrivFairFL-Post` can be used as is even in the cross-silo setup, as it is independent of the data distribution among the clients in FL.

A straight-forward technique to extend our defined protocols for multi-class classification or multi-valued sensitive attributes is to adopt a one-vs-rest approach for each class and/or each value of sensitive attribute as done in Section 5.3 in Chapter 5. This would then require the clients to release statistics for each binary combination of S and Y in a pre-defined sequence.

Our approach can also be utilized in dynamic scenarios with client dropouts and change in local data distributions, by computing the weights to be assigned to the training samples after a set of FL rounds. This will lead to a technique that is a combination of the pre-processing and in-processing techniques.

6.4 Experimental Results

6.4.1 Datasets and Model Training

We evaluate `PrivFairFL` on two datasets described below.

ADS. The ADS dataset[3] is a collection of 300 advertisements explicitly rated by 120 users [253]. Additionally, the dataset contains demographic information, interests, personality traits and textual phrases describing the likes and dislikes of each user. The advertisements are in the form of pictures. Each user rated each advertisement on a 1–5 scale. Following the categorization in [253], we labeled any ratings above 3 as "positive" and below and equal

[3] `https://www.kaggle.com/groffo/ads16-dataset`

to 3 as "negative".

We extracted features from the advertisements (number of faces, label types, safe category, objects) using Google Vision API.[4] The features for the users are based on one-hot-encoded information on their gender, country, interests, and income category, the first two indices of the zipcode, and their normalized age, along with the word embeddings[5] for the words describing the likes and dislikes of users. Each training sample consists of the features of a user, features for an ad and the explicit rating provided by the user for the particular ad, leading to 300 samples for each user. We excluded all the users who had only negative ratings resulting in a total of 109 users and a total of 32700 training samples. We performed a stratified split over these users, keeping 80% of their data as training data and the remaining 20% data as the test data.

To predict if a user is interested in an ad ("positive" instance) we trained a dense layer with 2 units each with sigmoid activation. In the centralized setup (CL), this model is trained for 630 epochs with a batch size of 64. For FL, the models are trained for 800 rounds with each client training the local models for 2 epochs per round with a batch size of 24. For the models that we train with DP-SGD, the micro-batches are set same as the batch size with clipping threshold as 1 and noise multiplier as 0.7. We use SGD as the optimizer with glorot initializer and learning rate of 0.1, and binary cross-entropy as loss to train all the models. All the above models are trained thrice with different values of seeds (47568, 42, 1000), and the average values of the metrics are reported.

ML-1K. The Movielens dataset contains explicit 5-star movie ratings from 6040 users on 3952 movies. All users have rated at least 20 movies, but the count varies per user. Movie and user features are available, including demographic user information. The classification task is to predict whether a user would enjoy a particular movie. For this, we convert greater than 3 star reviews to a positive rating, and 3 stars or less to a negative rating, as with the

[4]`https://cloud.google.com/vision/docs/object-localizer`

[5]Based on pretrained Glove embeddings.

ADS dataset. For movie features, we include one-hot-encoded genre and time between the user rating and the movie release year. For user features, we include categorical age, gender, and one-hot-encoded occupation.

We take a sub-sample of 75 users for the experiments, within which the percentage of positive ratings is similar to the whole dataset (sub-sample: 58.11%, whole dataset: 57.52%), as is the percentage of female users (sub-sample: 32%, whole dataset: 28.29%). The model consists of a dense layer with 1 unit and sigmoid activation. In the centralized setting (CL), the model is trained for 20 epochs with a batch size of 32 and learning rate 0.015. In FL, the model is trained with 2 local epochs per user, 300 rounds, with a learning rate of 0.03. The batch size and micro-batch size for DP-SGD are set to the size of the users' training set. The clipping threshold is set to 1.0 and the noise multiplier to 1.1. The model is trained with an SGD optimizer and binary cross-entropy loss. All results are reported on averages from 3 runs, using random seeds 1, 2 and 3.

Training. We implemented the model training phase in Python using Flower [254] for FL. We use FedAvg [87] as the aggregation strategy for the weights and add noise to the model parameters locally to provide DP guarantees during FL. We train some of the FL models with DP-SGD [76] as specified in Table 6.4. We implemented our MPC protocols in MP-SDPZ [110].

We recall that our proposed techniques are independent of the model being trained. The model training times are thus the same as for training any DP-models in FL, and the cost for privacy-preserving debiasing with `PrivFairFL` is separate. Our experiments show that it takes about 32-40 minutes to train a LR model with DP-SGD on 109 clients with 800 federated runs. We also note that `PrivFairFL-Post` is independent of any hyperparamters, giving the same convergence guarantees as state-of-the-art FL+DP training techniques. `PrivFairFL-Pre` might benefit from additional rounds of FL or local epochs of the clients depending on the dataset. The number of rounds can be set by the aggregator in FL, who keeps track of the loss and utility on the held-out validation set and has knowledge

on the kind of data. Additionally, we propose to use early stopping per client and setting a large maximum number of epochs. The convergence guarantees of `PrivFairFL-Pre` thus also follow from the convergence guarantees of FL+DP.

We investigate the following research questions:

Q1: To what extent do our proposed pre-processing and post-processing approaches in `PrivFairFL` improve fairness while addressing privacy?
From the results in Table 6.4, we observe that the fairness mitigation techniques aid in achieving group fairness in FL, and that the best type of technique depends on the dataset. `PrivFairFL-Pre` benefits the highly imbalanced data in ADS, while `PrivFairFL-Post` successfully finds optimal thresholds for the almost balanced subset of data in ML-1M. We also observe that randomization provided by DP guarantees can either aid in achieving a group-fair model or worsen the bias existing in the datasets. Our DP-in-MPC approaches not only aid in achieving group fairness while providing privacy guarantees but also improve the utility of the model when compared to the baseline approach that uses local DP ([236]).

Table 6.4: Utility and fairness for $\epsilon = 1$ averaged over three runs with different values of seed

	Fairness	Privacy	**ADS**					**ML-1M**				
			Acc.	|1-DI|	ΔEOP	ΔEODD	ΔSP	Acc.	|1-DI|	ΔEOP	ΔEODD	ΔSP
CL	–	–	85.81%	1.214	0.070	0.037	0.004	62.15%	0.138	0.091	0.141	0.094
FL	–	–	85.04%	1.654	0.089	0.050	0.018	59.04%	0.096	0.081	0.096	0.091
FL-DP-SGD	–	DP-SGD	85.21%	6.606	0.070	0.043	0.023	58.30%	0.027	0.026	0.027	0.052
FL-DP-SGD	`Pre`	Local DP	83.52%	0.260	0.036	0.033	0.032	58.47%	0.045	0.042	0.052	0.061
PrivFairFL	`Pre`	MPC+DP	83.57%	0.122	0.018	0.027	0.036	58.52%	0.045	0.042	0.051	0.063
PrivFairFL	`Post`	MPC+DP	85.17%	0.572	0.008	0.005	0.001	58.46%	0.006	0.006	0.014	0.045

The strength of our proposed techniques is using MPC which provides stronger privacy to the *inputs* while maintaining the utility. We add noise to the aggregated statistics using

event-level DP. These noisy statistics are then further processed to compute *outputs* that are protected with DP guarantees due to post-processing property. These outputs see almost no variation in the ratio of the weights (PrivFairFL-Pre) and ratio of TPR and FPR (generated for PrivFair-Post). The simplicity of our approach is in secure aggregation of these statistics that aid in bias mitigation while preserving the utility of the model.

We study the trade-offs between privacy and accuracy/fairness only with respect to the privacy guarantees provided for the *outputs*, controlled by the privacy budget ϵ used in our MPC protocols to ensure differential privacy. Across a range of ϵ values from 0.25 to 50, we observe almost no loss in utility or fairness, highlighting the strength of DP-in-MPC. In contrast, for local DP, we find that even at relatively high ϵ values (5 to 50), the generated weights exhibit large standard deviations. This alters the ratios of computed statistics, negatively impacting fairness metrics. These findings are consistent with existing literature, which shows that local DP tends to degrade utility more severely.

Q2: How does `PrivFairFL` affect performance on real-world datasets?
Based on our evaluation, `PrivFairFL` can lead to less accurate models as opposed to models trained in CL and pure-FL. We think this is due to trade-offs for training fair models. Privacy-preserving techniques like DP can randomly affect the utility of the model, while our MPC+DP based approaches reduce such trade-offs and, as observed, improve utility over both the datasets.

Q3: How does `PrivFairFL` compare with existing work?
As per our evaluation, `PrivFairFL` provides improvement over the baseline local DP approach based on [236][6] w.r.t. utility and fairness. `PrivFairFL` unlike the other techniques in Table 6.1, not only reduces bias but also protects the data with provable privacy guarantees with little/no cost in utility.

[6]Adopted for cross-silo setup where each client adds noise to their local counts and uses randomized response to publish the value of their sensitive attribute.

Table 6.5: Runtimes for different MPC schemes for 2PC (dishonest majority) and 3PC/4PC (honest majority). OTSemi2k is semi-honest adapt. of [1], Replicated2k is [3], SPDZ2k is [1, 2], SPDZ-wise Replicated2k is [4], Rep4-2k is [4]; all with corruption threshold of 1

			ADS		ML-1M	
		MPC scheme	`Pre-proc.`	`Post-proc.`	`Pre-proc.`	`Post-proc.`
passive	2PC	OTSemi2k	0.10 sec	43093.79 sec	0.10 sec	13088.52 sec
	3PC	Replicated2k	0.02 sec	7993.73 sec	0.02 sec	2427.87 sec
active	2PC	SPDZ2k	3.55 sec	1199060.00 sec	3.54 sec	364180.50 sec
	3PC	SPDZ-wise Replicated2k	0.06 sec	16832.09 sec	0.05 sec	5112.27 sec
	4PC	Rep4-2k	0.02 sec	8673.63 sec	0.02 sec	2634.37 sec

Q4: How does `PrivFairFL` scale?

We study the scalability of `PrivFairFL` based on the number of clients and computing servers. Table 6.5 contains an overview of runtimes of our protocols for different MPC schemes. The runtime results show substantial difference between passive and active security settings and between honest (3PC/4PC) and dishonest (2PC) majority settings; these results align with existing literature ([108, 4]). Our experiments are evaluated over a set of 75 and 109 clients, demonstrating the scalability of our approach. As expected, `PrivFairFL-Post` takes much longer than `PrivFairFL-Pre` to complete. We argue that even the longest runtimes of `PrivFairFL-Post` are an acceptable price to pay for training fair models in a privacy-preserving way. We note that our proposed techniques are independent of the training phase of the model. The runtimes of the MPC protocols for bias mitigation are a one-time reasonable price to achieve both privacy and fairness.

`PrivFairFL-Pre` essentially collects the statistics from each client and aggregates them to publish DP-weights for further Federated Learning. This is equivalent to collecting one vector from each client (one datapoint), making the aggregation for bias mitigation dependent only

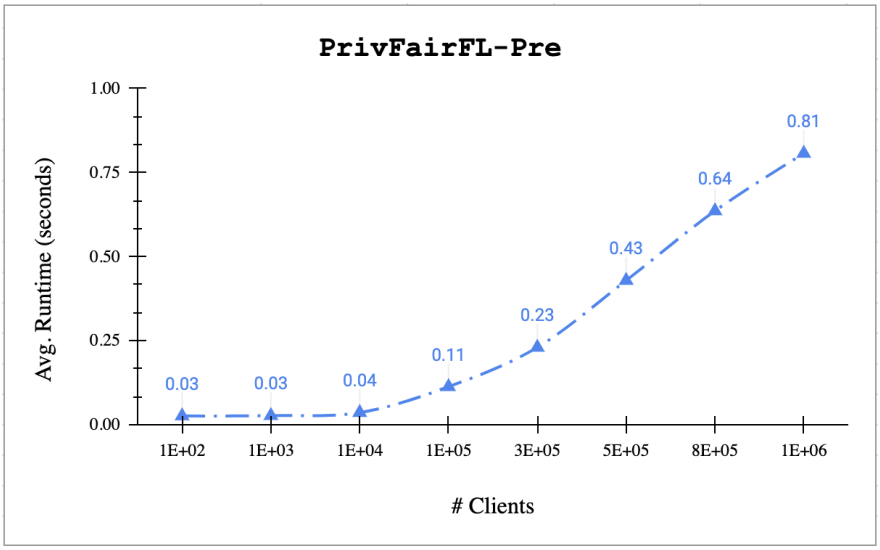


Figure 6.2: Runtimes for 3PC passive security setting with 1 corruption threshold for `PrivFairFL-Pre`

on the number of clients in the federation. Figure 6.2 shows the runtimes to execute π_{RW} with the number of clients in the federation ranging from 100 to 1,000,000, demonstrating that the runtimes increase linearly with the number of clients (note that the scale on the horizontal axis is logarithmic)[7]. We see that for hundreds of thousands of clients, it takes about 0.8 seconds to debias the distributed dataset. While these runtimes may only roughly approximate what one could expect during actual deployment in dynamic environments, they indicate that our MPC protocol for pre-processing is practical enough to achieve both privacy and fairness.

`PrivFairFL-Post` collects the training labels, the inferred labels for the training data with the (unfair) model, and the sensitive attributes from each client in the federation to compute DP ROC curves over all the training data from all the clients, where each client shares multiple datapoints. This makes our postprocessing technique dependent on the total number of training samples rather than the number of clients. Figure 6.3 shows the

[7]All the runtimes in Figure 6.2 and 6.3 are averaged over 5 runs. We simulated the large number of clients and datapoints by making copies of the 109 clients and their training samples that we used for our experiments.

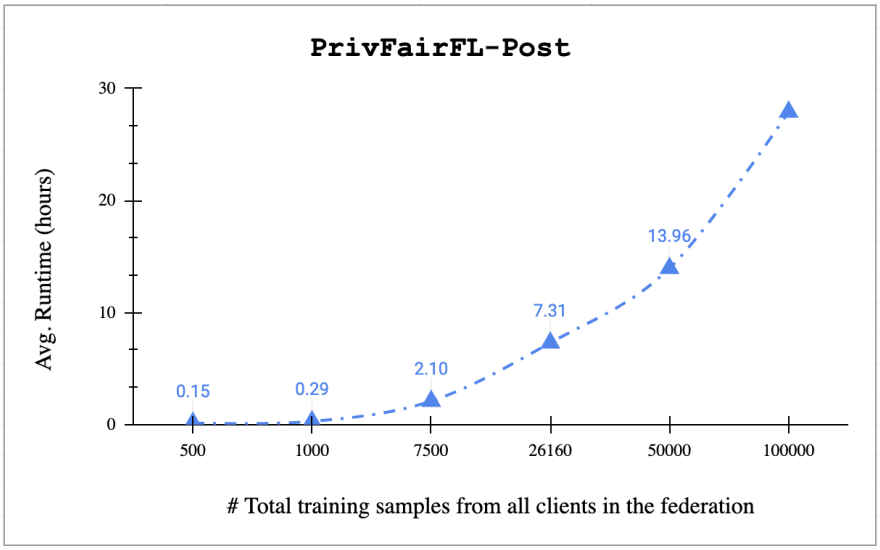


Figure 6.3: Runtimes for 3PC passive security setting with 1 corruption threshold for `PrivFairFL-Post`

runtimes to execute π_{ROC} for an increasing number of datapoints, ranging from 10 to 100,000, considered for FL to compute the confusion matrices. We see that for a hundred thousand training samples contributed by clients in the federation, it takes about 28 hours to find the optimal classification thresholds for each sensitive group. We argue that this longer runtime could be acceptable as the thresholds need to be computed only once at the end of the FL process.

The participating clients split their information (training labels, inferred labels for the training data with the learned model, sensitive attribute values) into secret shares and send them to the computing servers. The MPC overhead for the clients to contribute to model debiasing is negligible compared to the clients' model training costs. The MPC computational and communication overhead is instead carried by the computing servers who execute the MPC protocol, without requiring continued presence nor active participation of the clients. The computing servers publish DP ROC curves to the FL aggregator who can then compute the thresholds to make the current model available for inference to the clients, at no additional cost to the clients.

For practical deployment, the runtime of our solution can be further improved with optimizations of a cryptographic nature if we leave the MP-SPDZ framework. Secure multiplications are the bottleneck of any MPC-based solution. While we purposefully designed our MPC protocols to reduce the number of multiplications as much as possible, many multiplications remain. These secure multiplications are implemented using Beaver triples. A well known improvement is to pre-compute these triples during an "offline phase" (an option not available in MP-SPDZ at the time of this research) so that they are immediately available for consumption by the MPC protocols for postprocessing during the "online phase". The runtimes reported include both the pre-computations that can be done during an offline phase (before the computing servers become active) and an online phase (that starts when the clients send encrypted shares of their information to the computing servers). Performing the offline computations prior to model debiasing (e.g. during model training), can make our `PrivFairFL-Post` more efficient for practical use.

6.5 Summary

We introduced `PrivFairFL`, a framework that leverages DP-in-MPC to train group-fair models in FL, particularly in cross-device settings. We developed MPC protocols for both pre- and post-processing bias mitigation algorithms. We implemented MPC protocols for: (a) collecting ROC curves and computing statistics over label and sensitive attribute distributions across clients, and (b) releasing these statistics under formal differential privacy guarantees.

Through empirical evaluation, we demonstrated that `PrivFairFL-Pre` and `PrivFairFl-Post` can effectively balance data and generate fair model predictions, without degrading overall model utility. An interesting future direction of research is incorporating in-processing bias mitigation techniques into `PrivFairFL`. While many bias mitigation techniques rely on aggregated statistics or components of the confusion matrix, not all fairness definitions do. In particular, a few individual and causal fairness notions do not naturally fit into this framework. One can extend `PrivFairFL` to such fairness notions.

Chapter 7

ACHIEVING PRODUCER FAIRNESS IN PRIVACY-PRESERVING RECOMMENDATIONS

Information systems such as music recommendation platforms (e.g. Spotify), job recommendations (e.g. LinkedIn), search engines (e.g. Google), or delivery services (e.g. Uber Eats) often involve multiple stakeholders. There are *users* who consume the information, and there are content producers or service providers who generate or supply that information. In the previous chapter, we addressed how to simultaneously ensure privacy and fairness for the same stakeholder: the user. In this chapter, we take a different perspective. We consider settings where privacy is important for one stakeholder, the user (e.g. someone who listens to music but does not want their preferences to be revealed), and fairness is a requirement from another stakeholder (e.g. musicians who want fair exposure or representation in recommendation systems).

As in the previous chapter, we consider cross-device federated learning (FL) setups, that naturally arise in many real-world information systems, such as music streaming platforms, personalized news feeds, or web search engines. We introduce DP-in-MPC in existing federated information systems to jointly achieve both privacy and fairness.

We detail the problem of fair ranking in Section 7.1. Section 7.2 provides preliminaries on fair ranking metrics, followed by a discussion of related work on privacy in this context in Section 7.3. We then present our solution, which leverages DP-in-MPC, in Section 7.4, along with its evaluation and discussion in Section 7.5. Finally, we summarize the chapter in Section 7.6.

7.1 Introduction

Information systems, such as those used for search retrieval and recommendation, have become a ubiquitous part of our daily lives. These systems provide users with curated information such as content produced in social media or web pages, or organize the information in a way that is most relevant to users to aid their decision-making on what to buy, what to watch, or who to hire. These results are often output in the form of ranked lists. To maximize the utility of the system for each user, the ranked list is ordered by decreasing relevance (based on some score or rating) to the user. Users are then susceptible to paying most of their attention to the highest-ranked items in the ranked list, causing *position bias* [255, 256]. For a single ranked list (generated for a single user or a single query), the attention given to each item would decrease at a faster rate than relevance, i.e., lower-ranked items become disadvantaged by receiving disproportionately less attention relative to their relevance. When a sequence of ranked lists is generated in this manner, higher-ranked items reap increasingly disproportionately more attention relative to their relevance. This results in economic and social impacts on major stakeholders–the item providers, a.k.a. producers–of the information systems, leading to economic disparity, bias toward underrepresented producers, and unhealthy markets [257, 258].

Methods for mitigating unfairness in rankings have been proposed in the literature [259, 260, 261, 262, 263]. In this thesis, we focus on post-processing techniques that reorder ranked lists to distribute exposure fairly [264, 265]. These methods assume that the rankings or preferences of the users are available to a central entity that can then apply post-processing techniques to achieve fairness for the items. But, as we have discussed in Chapter 1, centralizing user data can lead to privacy leakage or even intentional privacy violations, such as when the data is routinely sold when companies undergo bankruptcy [266]. The need for more stringent user privacy protections, as well as the requirement to comply with regulations such as the GDPR has prompted the increased use of privacy-enhancing technologies (PETs) in recommender systems [267, 268, 269, 270, 271]. To the best of our current

knowledge, no research address the challenge of achieving fairness for producers (a.k.a. items) while preserving the privacy of consumers (a.k.a. users) in ranking systems. In this chapter, we explore this under-researched area in the literature.

We address the problem where, in a FL setting, each user (client) u_l has a list ρ_l of items $d_1^l, d_2^l, \ldots, d_i^l, \ldots, d_n^l$ that was generated in a privacy-preserving manner and ranked according to relevance. During a post-processing phase for bias mitigation, we wish to alter every user's ranking ρ_l to optimize for equity of amortized attention [265] without requiring any user disclosure of ρ_l to a centralized entity.

One could achieve this goal with MPC, i.e. by having each client encrypt their data to send to a set of MPC servers, having the MPC servers perform all computations needed for reranking over the encrypted data, and then return the results to the client to decrypt. While this approach would preserve end-to-end privacy and produce the same rankings as the bias mitigation method from Biega et al. [265] yields in the clear (i.e. without any encryption), the computational and communication overhead would be prohibitively large for practical applications. We therefore propose a much more scalable approach by adopting DP-in-MPC. In our solution, MPC servers store intermediate results and then perturb them with DP noise, which clients subsequently use to perform local computations. We demonstrate the applicability of our proposed solution by experimenting on three real-world datasets. We empirically show that fairness for items can be improved while ensuring the privacy of users, without making additional sacrifices to the ranking quality.

7.2 Fairness in Ranking Systems

To maximize utility, ranking algorithms generate lists of items sorted by their predicted level of relevance to a query or user. The most relevant items are positioned at the top of the list and receive the most exposure. Most work in fairness in rankings has been studied in the context of fairly distributing exposure to the elements of the ranked list. The elements of the ranked list could be people or items (e.g., content, products, places). Much of the work involving exposure has centered around group fairness, where exposure should

ideally be distributed equally among different groups defined by their protected attributes (such as gender or race). For example, Yang and Stoyanovich [259] proposed extending the traditional fairness concept of statistical parity to the ranking system, where being a member of a protected group does not influence a person's position in a ranking. Zehlike and Castillo [262] proposed a fair top-k ranking algorithm following the fairness notion of affirmative action, where a minimum number of protected group members are guaranteed in every top-k ranking (top 10, top 20, etc.). Celis et al. [272] proposed an algorithm addressing the constrained ranking maximization problem, where there is a limit to the number of sensitive items per protected group in the ranking, and no one group dominates. Sapiezynski et al. [273] also used statistical parity, but used a geometric distribution to model user attention as an analogue to exposure. Singh and Joachims [264] introduced fairness of exposure in rankings and proposed to use linear programming to optimize for the maximum utility in a ranking, subject to group fairness constraints. However, limited studies have been done on individual item fairness in rankings.

One such work is Biega et al.'s [265] proposal of equity of attention, which aims to achieve amortized fairness over a sequence of rankings by distributing attention proportional to item relevance. Their fairness notion, which we describe below, is based on the idea that attention, driven by item exposure, should be fairly allocated across items over time.

7.2.1 Equity of Amortized Attention (EOAA)

Biega et al. introduced the fairness notion of *equity of amortized attention* to achieve *amortized individual fairness* for a set of items $\mathcal{D} = \{d_1, d_2, \ldots, d_i, \ldots, d_n\}$ appearing in a sequence of relevance-based rankings $\rho_1, \rho_2, \ldots, \rho_l, \ldots, \rho_L$ for users $u_1, u_2, \ldots, u_l, \ldots, u_L$, respectively [265]. The position of item d_i in ranking ρ_l influences the amount of attention that d_i receives. EOAA is achieved if each item d_i receives cumulative attention A_i proportional to its cumulative relevance R_i, when amortized over a sequence of rankings. Biega et al. define a measure for this notion of fairness by taking the sum of the absolute differences between A_i

and R_i for $i = 1 \dots n$ as shown in Equation 7.1:

$$unfairness(\rho_1, ..., \rho_L) = \sum_{i=1}^{n} |A_i - R_i| = \sum_{i=1}^{n} \left| \sum_{l=1}^{L} a_i^l - \sum_{l=1}^{L} r_i^l \right|. \tag{7.1}$$

For each item d_i, r_i^l is its relevance score for user u_l, and a_i^l is the amount of attention it receives in ranking ρ_l. Biega et al. proposed to improve the EOAA of a sequence of relevance-based rankings $\rho_1, \rho_2, \dots, \rho_l, \dots, \rho_L$ by reranking each of the rankings sequentially to produce $\rho_1^*, \rho_2^*, \dots, \rho_l^*, \dots, \rho_L^*$. To rerank ρ_l, taking into account the previously computed rerankings $\rho_1^*, \rho_2^*, \dots, \rho_{l-1}^*$, the following post-processing linear program (ILP) is solved:

$$\text{Minimize } \sum_{i=1}^{n} \sum_{j=1}^{n} \left| A_i^{l-1} + \hat{w}_j - (R_i^{l-1} + \hat{r}_i^l) \right| \cdot X_{i,j} \tag{7.2}$$

$$\text{Subject to } \sum_{j=1}^{k} \sum_{i=1}^{n} \frac{2^{\hat{r}_i^l} - 1}{\log_2(j+1)} X_{i,j} \geq \theta \cdot DCG(\rho_l)@k \tag{7.3}$$

$$X_{i,j} \in \{0, 1\}, \forall_{i,j} \text{ and } \sum_{i} X_{i,j} = 1, \forall_j \text{ and } \sum_{j} X_{i,j} = 1, \forall_i. \tag{7.4}$$

This ILP solves for n^2 decision variables $X_{i,j}$ that represent the reranking ρ_l^* of the items in ranking ρ_l. A_i^{l-1} and R_i^{l-1} are the accumulated attention and relevance of item d_i, respectively, over rerankings $\rho_1^*, \rho_2^*, \dots, \rho_{l-1}^*$. $\hat{w}_j$ is the normalized attention weight of placing an item at position j, calculated as $\hat{w}_j = w_j / \sum_{t=1}^{n} w_t$, based on the geometric attention model $w_j = 0.5(0.5)^{j-1}$ assumed in [265]. $\hat{r}_i^l$ is the normalized relevance score of r_i^l, calculated as $\hat{r}_i^l = \frac{r_i^l - r_{\min}}{r_{\max} - r_{\min}} \Big/ \sum_{t=1}^{n} \frac{r_t^l - r_{\min}}{r_{\max} - r_{\min}}$. $r_{\max}$ and $r_{\min}$ are the maximum and minimum relevance scores, respectively, that a user can have for an item. The coefficient $\left| A_i^{l-1} + w_j - (R_i^{l-1} + r_i^l) \right|$ is the unfairness measure of placing an item d_i at position j in the current l^{th} reranking. The constraint in Equation 7.3 bounds the quality of the first k items in the reranking in terms of its discounted cumulative gain (DCG), such that it is no lower than $0 \leq \theta \leq 1$ times the DCG of the top-k items of the original relevance-based ranking. The constraints in Equation 7.4 specify that the n^2 decision variables $X_{i,j}$ are binary, and that there is a single 1 per j for all i's, and a single 1 per i for all j's. $X_{i,j} = 1$ indicates item d_i is placed in position j, and $X_{i,j} = 0$ otherwise.

The quality of reranking ρ_l^* is measured in terms of its divergence from the original relevance-based ranking ρ_l. This is quantified as $NDCG(\rho_l, \rho_l^*)$ in Equation 7.5, where $DCG(\rho_l^*)$ is normalized by $DCG(\rho_l)$.

$$NDCG(\rho_l, \rho_l^*) = \frac{DCG(\rho_l^*)}{DCG(\rho_l)} \tag{7.5}$$

The maximum NDCG value is 1, and occurs when either $\rho_l = \rho_l^*$, or if items of equal relevance scores are shuffled with each other.

In this chapter, we extend Biega et al.'s [265] reranking framework by incorporating privacy-preserving mechanisms into the fair ranking process. To the best of our knowledge, no existing work supports such fair reranking in a privacy-preserving manner.

7.3 Related Work

7.3.1 Privacy-Preserving Ranking Systems

Various PETs have been used in previous works on privacy-preserving learning-to-rank (LTR) systems. Dehghani et al. [274] used mimic learning, where only a model trained on the sensitive data is shared, and not the data itself. Furthermore, Laplace noise is used during aggregation as part of the DP guarantee. Yang et al. [268] used the information-theoretic privacy approach, which involves obfuscating data in accordance with a data distortion budget. Kharitonov [275] used a federated learning setup with evolution strategies optimization, and then incorporated local DP. Wang et al. [276] extended Kharitonov's work to larger datasets and found a substantial loss in utility compared to other non-private online learning-to-rank systems. Wang et al. [277] then used a federated learning setup and local DP, similar to Kharitonov [275], but incorporated a pairwise differentiable gradient descent (PDGD) optimization approach instead. Ge et al. [267] incorporated the Paillier homomorphic encryption algorithm in their PrivItem2Vec model. Among these previous works in privacy-preserving ranking systems, none have exploited the use of MPC together with DP.

7.3.2 Fair and Privacy-Preserving Ranking Systems

It is evident that both fairness in rankings and privacy-preserving methods in ranking systems have been well-studied separately; however, there is a dearth of research at the intersection of fairness and privacy in ranking systems. Resheff et al. [278] used privacy-adversarial training in their recommender system to obtain user representations that obfuscate users' sensitive attributes. This approach focuses on improving group fairness and preserves some user privacy by way of preventing implicit private information attacks. Sato [279] proposed a local ranking system framework that is independent from the centralized recommender system, where users can post-process the rankings they receive by themselves by setting their own fairness constraints to their preferences. Their privacy-preserving method is in each user developing their own recommender system. However, this is very computationally expensive and many users' devices do not have the processing power to maintain these systems. Both of these works address privacy and fairness with respect to the users' protected attributes, yet to the best of our knowledge, there has been no work so far that addresses privacy and fairness with respect to the amount of attention items receive in relation to their relevance.

7.4 Privacy-Preserving Fair Recommendations

7.4.1 Problem Description

We consider a regression-style recommendation model $\mathcal{M}$ that is trained in a privacy-preserving manner (such as [271, 275]), i.e., the model does not leak any information about the training data. $\mathcal{M}$ is deployed at each user u_l to predict their relevance scores $\boldsymbol{r}^l = (r^l_1, r^l_2, \ldots, r^l_i, \ldots, r^l_n)$ on a global set of items $\mathcal{D}$, where $r^l_i = \mathcal{M}(u_l, d_i)$. In this process, the users' raw data – such as their preferences, demographics, embeddings, etc. – are not disclosed to anyone. User u_l's relevance-based ranking ρ_l is its list of items sorted in decreasing order of their relevance scores.

The post-processing technique described in Section 7.2 assumes that a central server S accesses the relevance-based rankings $\rho_1, \rho_2, \ldots, \rho_l, \ldots, \rho_L$ of all users and reranks them

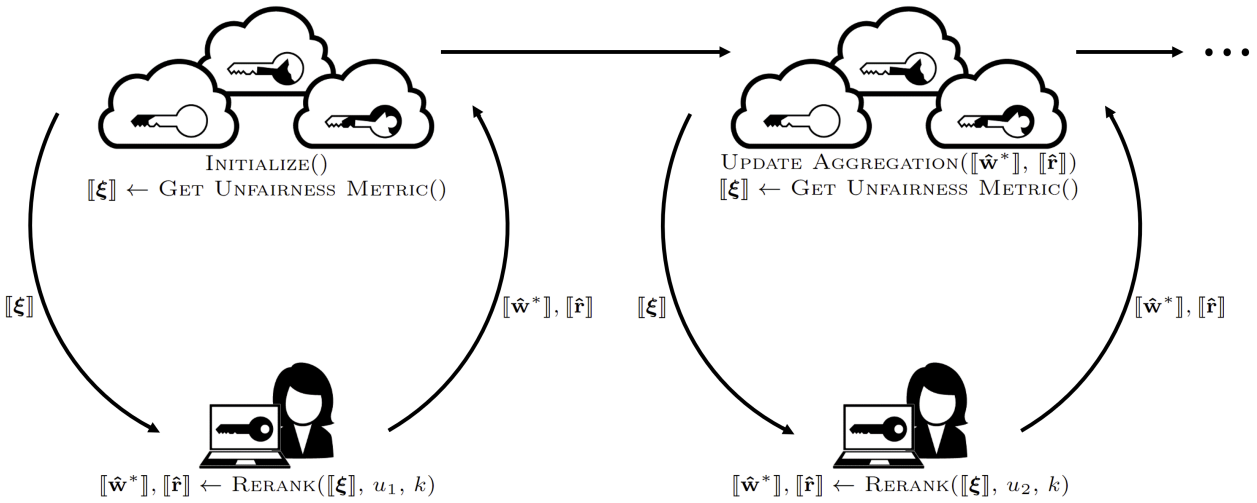


Figure 7.1: Flow diagram of privacy-preserving fair item ranking algorithm[1]

into $\rho_1^*, \rho_2^*, \ldots, \rho_l^*, \ldots, \rho_L^*$ to achieve *equity of amortized attention* without losing the ranking utility beyond the set threshold. This setup, which we refer to as the *centralized setup*, causes leakage of sensitive user information to the central server S, including:

(P_1) the preference of the user for all items in the form of relevance scores,

(P_2) the top-k items that the user is most likely to be interested in, and

(P_3) the order of the top-k items the user is most likely to be interested in.

Below we describe how we adapt the above post-processing method using DP-in-MPC to address privacy issues (P_1)–(P_3) in order to achieve individual fairness for the items and preserve the privacy of the users.

7.4.2 Proposed Method

The key observation underlying our method is that each user u_l has all the information needed to solve the ILP to rerank their original ranking ρ_l into ρ_l^*, except for the values of A_i^{l-1} and R_i^{l-1} in Equation 7.2. A_i^{l-1} and R_i^{l-1} are the accumulated attention and relevance

[1]Figure adapted from `https://sepior.com/mpc-blog`

of item d_i, respectively, over rerankings $\rho_1^*, \rho_2^*, \ldots, \rho_{l-1}^*$, i.e., the values of A_i^{l-1} and R_i^{l-1} depend on sensitive information from users $u_1, \ldots, u_{l-1}$ which we neither want to disclose to user u_l nor to a central server (as is the case in the centralized setup). In our solution, we therefore maintain encrypted versions of A_i^{l-1} and R_i^{l-1} which initially are 0 (see Procedure INITIALIZE in Algorithm 11), and are updated in a privacy-preserving manner each time a user completes their ILP computation. More precisely, no entity knows by itself at any point what the current values of A_i^{l-1} and R_i^{l-1} are. Instead, these values are split into secret shares and distributed over MPC servers who can jointly perform operations to update the shares, without ever learning the true values of the inputs or the results of the computations. Figure 7.1 illustrates the high-level flow of our solution, and Algorithm 11 presents the pseudo-code, which we explain in more detail below. We use $\boldsymbol{A}^{l-1}$ and $\boldsymbol{R}^{l-1}$ to denote the vectors $[A_1^{l-1}, \ldots, A_n^{l-1}]$ and $[R_1^{l-1}, \ldots, R_n^{l-1}]$, respectively.

The users update their rankings in sequence $l = 1 \ldots L$. When the sequence reaches user u_l, u_l first requests the current vector of differences $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}$ from the MPC servers, prompting the MPC servers to compute a secret sharing $[\![\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}]\!]$ from their local secret shares of $\boldsymbol{A}^{l-1}$ and $\boldsymbol{R}^{l-1}$ (Procedure GET UNFAIRNESS METRIC). In principle, each MPC server could send their secret shares of $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}$ to user u_l, which u_l could combine to construct the value of $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}$. However, although $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}$ consists of aggregated information only, this value could still leak information about the previous users $u_1, \ldots, u_{l-1}$ to user u_l, especially if u_l is one of the first to rerank. In our solution, this privacy loss is mitigated by having the MPC servers perturb the value at each index of $[\![\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}]\!]$ with Laplace noise *before* sending it to user u_l (Procedure GET UNFAIRNESS METRIC). We denote this perturbed secret sharing as $[\![\boldsymbol{\xi}]\!]$. The DP guarantee that the MPC servers provide in this manner is that the probability of returning any specific value of $\boldsymbol{\xi}$ is very similar to the probability of returning that value if the data of a previous user u_i ($i = 1 \ldots l-1$) would have been left out of the computation of $[\![\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}]\!]$ (see Equation 2.1). This entails that the value of $\boldsymbol{\xi}$ returned to user u_l does not leak information about the users who computed their rerankings before it was u_l's turn. To this end, the MPC servers generate secret shares of

Laplace noise using the MPC-protocol π_{LAP} for secure sampling from a Laplace distribution as described in Section 6.3.1 of Chapter 6 and add these secret shares to $[\![\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}]\!]$, effectively emulating the global DP paradigm without having to rely on a central trusted aggregator. We provide more information about the scale parameter b for the Laplace noise below.

Algorithm 11: Privacy-Preserving Fair Item Ranking

Achieving EOAA privately over L users

$n \leftarrow$ number of items per ranking
$L \leftarrow$ number of rankings to rerank
$k \leftarrow$ number of items in quality constraint Equation 7.3
$\boldsymbol{w} \leftarrow$ attention weights
$\hat{\boldsymbol{w}} \leftarrow$ NORMALIZE($\boldsymbol{w}$)

INITIALIZE()

for each u_l, **where** $l = 1 \dots L$ **do**
$[\![\boldsymbol{\xi}]\!] \leftarrow$ GET UNFAIRNESS METRIC()
$[\![\hat{\boldsymbol{w}}^*]\!], [\![\hat{\boldsymbol{r}}]\!] \leftarrow$ RERANK($[\![\boldsymbol{\xi}]\!]$, u_l, k)
UPDATE AGGREGATION($[\![\hat{\boldsymbol{w}}^*]\!]$, $[\![\hat{\boldsymbol{r}}]\!]$)
end for

User u_l subroutine

procedure RERANK($[\![\boldsymbol{\xi}]\!], u_l, k$)
$\boldsymbol{\xi} \leftarrow (\epsilon/L) \sum [\![\boldsymbol{\xi}]\!]$
$\boldsymbol{r} \leftarrow \mathcal{M}(u_l, \mathcal{D})$
$\hat{\boldsymbol{r}} \leftarrow$ NORMALIZE($\boldsymbol{r}$)
$\boldsymbol{s} \leftarrow$ ILP($\boldsymbol{\xi}, \hat{\boldsymbol{r}}, k$)
$\hat{\boldsymbol{w}}^* \leftarrow \hat{\boldsymbol{w}}[\boldsymbol{s}]$
return $[\![\hat{\boldsymbol{w}}^*]\!], [\![\hat{\boldsymbol{r}}]\!]$
end procedure

MPC servers' subroutines

procedure INITIALIZE()
$\Delta f \leftarrow$ sensitivity calculated from Equation 7.10
$\epsilon \leftarrow$ privacy budget
$[\![\boldsymbol{A}]\!] \leftarrow [\![0]\!]$
$[\![\boldsymbol{R}]\!] \leftarrow [\![0]\!]$
end procedure

procedure GET UNFAIRNESS METRIC()
//MPC protocol for global DP
$b \leftarrow \Delta f/(\epsilon/(n \cdot L))$
$[\![\boldsymbol{\xi}]\!] \leftarrow [\![\boldsymbol{A} - \boldsymbol{R}]\!] + \pi_{\mathsf{LAP}}(b)$
return $[\![\boldsymbol{\xi}]\!]$
end procedure

procedure UPDATE AGGREGATION($[\![\hat{\boldsymbol{w}}^*]\!], [\![\hat{\boldsymbol{r}}]\!]$)
//MPC protocol to perform aggregation
$[\![\boldsymbol{A}]\!] \leftarrow [\![\boldsymbol{A}]\!] + [\![\hat{\boldsymbol{w}}^*]\!]$
$[\![\boldsymbol{R}]\!] \leftarrow [\![\boldsymbol{R}]\!] + [\![\hat{\boldsymbol{r}}]\!]$
end procedure

Once user u_l has received secret shares of $\boldsymbol{\xi}$ from each MPC server, u_l can sum up the secret shares to get $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1} + \pi_{\mathsf{LAP}}(b)$, using this as a proxy for $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1}$, and then proceed to solve the ILP program in Section 7.2 (Procedure RERANK) for $X_{i,j}$ ($i = 1 \dots n$ and $j = 1 \dots n$). In our implementation, we scaled $\boldsymbol{A}^{l-1} - \boldsymbol{R}^{l-1} + \pi_{\mathsf{LAP}}(b)$ by a positive factor ϵ/L so that the solution of the ILP is easier to compute. We note that scaling by a positive

factor does not affect the outcome in $X_{i,j}$. $X_{i,j}$ can then translate to a vector $\boldsymbol{s}$ of indices to reorder the normalized attention weights $\hat{\boldsymbol{w}}$. $\hat{\boldsymbol{w}}^*$ is the vector of attention weights distributed to each item at each index in the order of $\boldsymbol{s}$. Thus, $\hat{\boldsymbol{w}}^*$ and $\hat{\boldsymbol{r}}$ make up the reranking ρ_l^*. User u_l subsequently encrypts the values in these vectors by splitting them into secret shares $[\![\hat{\boldsymbol{w}}^*]\!]$ and $[\![\hat{\boldsymbol{r}}]\!]$, and distributes the shares among the MPC servers. This enables the MPC servers to update their secret shares of the aggregated values to $[\![\boldsymbol{A}^l]\!]$ and $[\![\boldsymbol{R}^l]\!]$ (Procedure UPDATE AGGREGATIONS), which they will need when the next user, u_{l+1}, makes a request. The whole process is repeated until all users have completed their reranking.

Scale parameter b. To make our algorithm ϵ-DP, the MPC servers answer each query by adding noise to the true aggregate. To this end, the MPC servers sample noise from a Laplace distribution with mean 0 and scale $b = \Delta f/\epsilon'$ in which Δf denotes the sensitivity and ϵ' the privacy budget per query. Appropriate values for these parameters are described next. In Algorithm 11, each user u_l ($l = 1 \ldots L$) queries the MPC servers for aggregated information $A_i^{l-1} - R_i^{l-1}$ about each item d_i ($i = 1 \ldots n$) through the Procedure GET UNFAIRNESS METRIC. The total number of queries to be answered by the MPC servers is in other words $n \cdot L$. These queries are executed against overlapping datasets, as u_l queries the aggregated information of users $u_1, \ldots, u_{l-1}$; u_{l+1} queries the aggregated information of users $u_1, \ldots, u_l$, etc. Given a total privacy budget ϵ, we therefore allocate $\epsilon' = \epsilon/(n \cdot L)$ per query. Δf is the sensitivity computed for the aggregate ${A_i}^{l-1} - {R_i}^{l-1}$, and is given by Equation 7.10 that computes the maximum value that can be contributed to this aggregate by any single user.

$$\hat{r}_{\min} = \frac{r_{\min} - r_{\min}}{r_{\max} - r_{\min}} \Bigg/ \left(\frac{r_{\min} - r_{\min}}{r_{\max} - r_{\min}} + (n-1) \left(\frac{r_{\max} - r_{\min}}{r_{\max} - r_{\min}} \right) \right) = 0 \tag{7.6}$$

$$\hat{r}_{\max} = \frac{r_{\max} - r_{\min}}{r_{\max} - r_{\min}} \Bigg/ \left(\frac{r_{\max} - r_{\min}}{r_{\max} - r_{\min}} + (n-1) \left(\frac{r_{\min} - r_{\min}}{r_{\max} - r_{\min}} \right) \right) = 1 \tag{7.7}$$

$$\hat{w}_{\min} = \frac{w_n}{\sum_{j=1}^{n} w_j} \tag{7.8}$$

$$\hat{w}_{\max} = \frac{w_1}{\sum_{j=1}^{n} w_j} \tag{7.9}$$

$$\Delta f = \max(|\hat{w}_{\max} - \hat{r}_{\min}|, |\hat{w}_{\min} - \hat{r}_{\max}|) \tag{7.10}$$

Equations 7.8, 7.9 are based on the normalized geometric attention model and Equations 7.6, 7.7 are based on the range of the normalized relevance scores that the MPC servers may receive. Computation of Δf and b is independent of the users' data and can be precomputed by one of the MPC servers in the clear, i.e., without encryption (see Procedure INITIALIZE).

7.5 Results

7.5.1 Datasets

We use three recommender system datasets in our experiments: Amazon Digital Music,[2] Book Crossing,[3] and MovieLens-1M.[4] Each dataset contains information about each user and item, and ratings that users gave to items. We detail the number of users, items, ratings, and the range of possible ratings of each dataset in Table 7.1.

7.5.2 Experimental Setup

We trained a singular value decomposition (SVD) model[5] for each dataset and predicted relevance scores for every user-item pair. We assume that these models were trained in a

[2] `http://jmcauley.ucsd.edu/data/amazon/links.html`

[3] `http://www2.informatik.uni-freiburg.de/~cziegler/BX/`

[4] `https://grouplens.org/datasets/movielens/1m/`

[5] `https://surpriselib.com/`

Table 7.1: Statistics of datasets used to train each SVD model

Dataset	# Users	# Items	# Ratings	Rating Levels
Amazon Digital Music	478,235	266,414	836,006	$1, 2, ..., 5$
Book Crossing	77,805	185,973	433,671	$1, 2, ..., 10$
MovieLens 1M	6,040	3,706	1,000,209	$1, 2, ..., 5$

privacy-preserving manner. For our experiments, we select $n = 100$ items to rerank for all datasets, and rerank $L = 3000$ users' rankings for the Amazon Digital Music and Book Crossing datasets, and all $L = 6040$ users' rankings for the MovieLens-1M dataset. We use Gurobi[6] to solve the ILP in Equation 7.2–7.4. We set $k = 100$, the quality loss constraint to $\theta = 0.8$, and calculate the sensitivity based on Equation 7.10 to be $\Delta f = 1$ for all datasets. We perform an empirical analysis for privacy budget $\epsilon \in \{0.5, 1, 10, 100, 1000, 10000, 100000\}$.

All MPC computations are done in the MP-SPDZ framework [110] and performed over a ring $\mathbb{Z}_q$ with $q = 2^{64}$. We perform experiments in a dishonest majority security setting of 2 computing parties (2PC) with passive adversaries and 1 corruption threshold.

We evaluate our approach in terms of unfairness (Equation 7.1) and utility (Equation 7.5) and study the privacy-fairness-utility trade-offs.

7.5.3 Discussion

Fairness vs. Privacy Trade-offs. We demonstrate the cost on item fairness when preserving the privacy of users in Figure 7.2 (left column; Figures 7.2a, 7.2c, and 7.2e). 'Centralized setup (no fairness)' are the unfairness measures without any bias mitigation, and 'Centralized setup (w/fairness)' are the unfairness measures when applying the post-processing technique from Section 7.2 in a centralized setup without any privacy protection. We ideally need privacy-preserving techniques that result in the unfairness metrics close to the 'Central-

[6] `https://www.gurobi.com/`

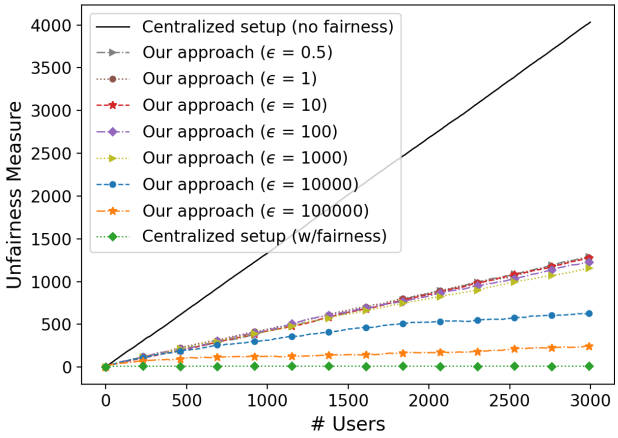
ized setup (w/fairness)'. Our results show that our method can still improve fairness even with the addition of our privacy-preserving techniques. Specifically, our method preserves users' privacy at every step of the process, both when users transfer $[\![\hat{\boldsymbol{w}}^*]\!]$ and $[\![\hat{\boldsymbol{r}}]\!]$ to the servers, and when the servers transfer $[\![\boldsymbol{\xi}]\!]$ to the users. We observe a trade-off between privacy and unfairness in our results, where a decrease in the privacy budget (i.e., more privacy) imposes a higher cost to the fairness, which is in line with the literature. This is because the amount of noise added perturbs the values of the unfairness measure, which consequently affects how well the ILP can compute rerankings when compared to the centralized setup without differential privacy. We note that our solution is able to preserve input privacy even with higher ϵ.

Utility vs. Privacy Trade-offs. We show the impact on reranking quality when introducing privacy for the users in Figure 7.2 (right column; Figures 7.2b, 7.2d, and 7.2f). $NDCG = 1$ represents the upper boundary of NDCG and indicates no change in the ordering of the relevance scores of the items compared to the original ranking. The dotted line at $NDCG = 0.8$ represents the quality constraint θ set in the ILP. Our results show that the quality of the rerankings are always maintained in the set boundary, $0.8 \leq NDCG \leq 1$, irrespective of the amount of noise added to preserve privacy. Our study shows that using our approach, the privacy of users can be preserved without losing utility beyond the threshold θ initially set in the ILP.

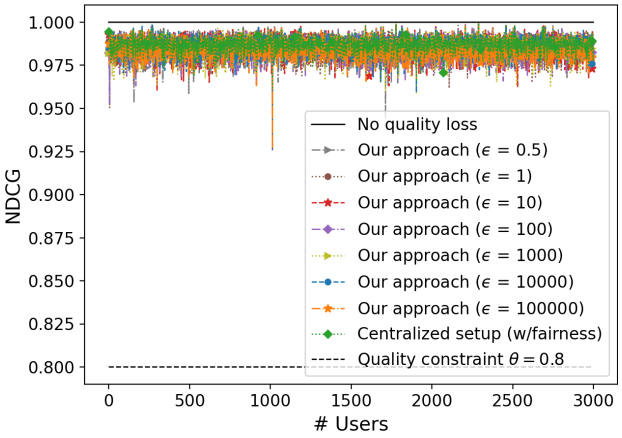
The resulting privacy-fairness trade-offs stem from preserving the output privacy. The utility-fairness trade-offs are due to the bias mitigation techniques and with and without privacy.

Runtime. All experiments were performed on a 2.6 GHz 6-Core Intel Core i7 with 16 GB RAM. It takes about 0.67 seconds (averaged over $L = 6,040$ users) to rerank a user's ranking for $n = 100$ in a centralized setup. The additional cost in runtime due to added privacy is less than 5 seconds per client for a 2PC passive security setting with mixed circuits [172]. These runtimes vary across different security settings. With a 3PC passive security setting

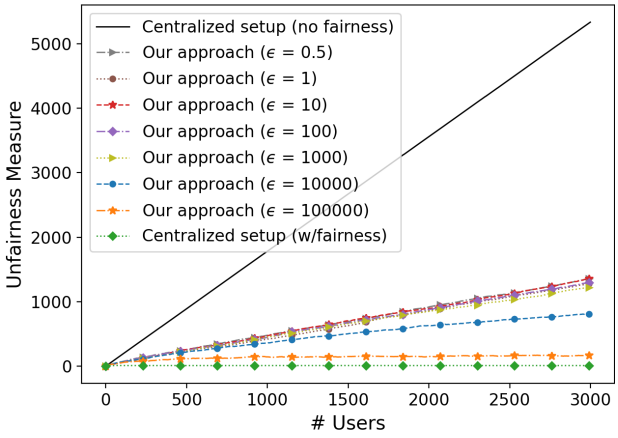
[3] with corruption threshold of 1, this additional runtime can be reduced to less than 1 second. We believe that this increased cost in runtime to rerank a user's ranking is worth the gain in privacy. We note that the MPC schemes are normally divided into two phases: the offline phase and the online phase, and we have reported runtimes for both. The offline phase performs computations independent of the data and thus can be carried out prior to the users' rerankings. By doing so, the responsiveness of our approach can be further improved to make our approach feasible in practice and near real-time.



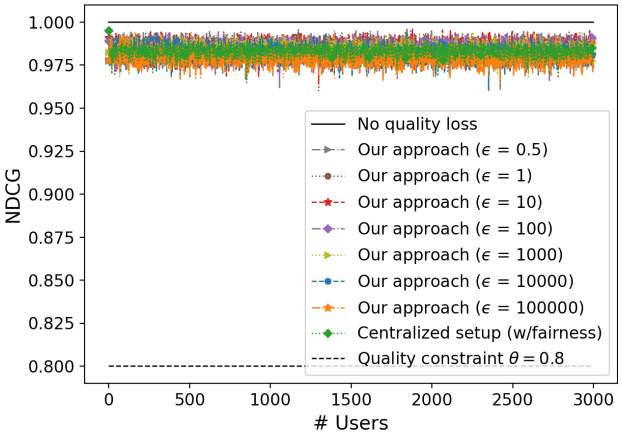
(a) Unfairness Measure on Amazon Digital Music



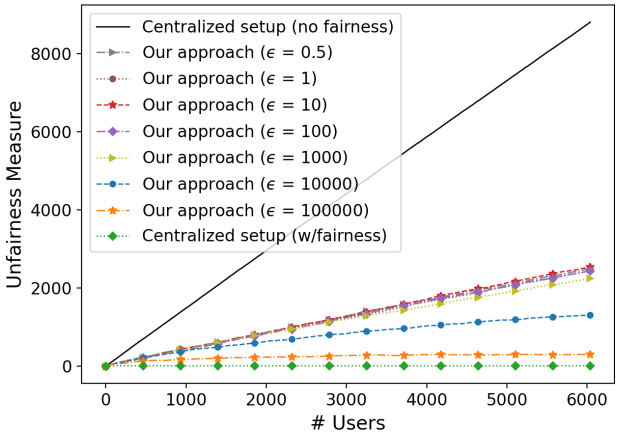
(b) Ranking Quality on Amazon Digital Music



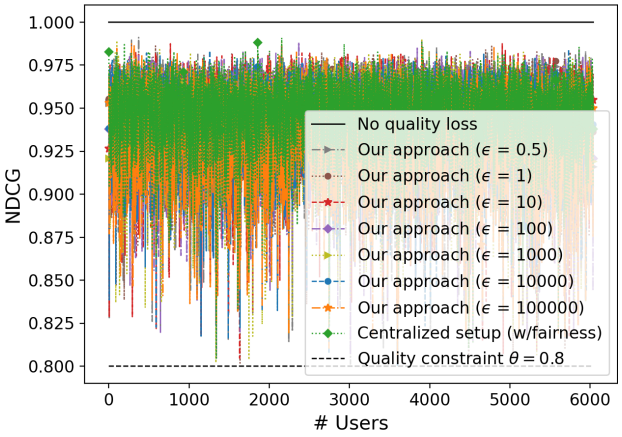
(c) Unfairness Measure on Book Crossing



(d) Ranking Quality on Book Crossing



(e) Unfairness Measure on MovieLens-1M



(f) Ranking Quality on MovieLens-1M

Figure 7.2: Model performance on each dataset with different values of ϵ

7.6 Summary

We presented the novel idea of promoting producer (item) fairness while preserving the privacy of the consumers (users) in a recommendation ecosystem using post-processing techniques. We leveraged DP-in-MPC and proposed an approach in which users work in tandem with MPC servers to rerank items, taking into account both relevance scores and attention weights. The MPC servers receive secret-shares of the user data, and perform all computations over the secret-shares. Furthermore, whenever the MPC servers need to release aggregated information to a user, they perturb it with noise to provide DP guarantees, thereby avoiding leakage of the data of any user to any other user.

We demonstrated that the incorporation of DP-in-MPC results in unfairness mitigation without additional cost to utility, through comparison to the centralized approach in which all users disclose their data to a central server. This approach can be extended to other bias mitigation techniques and various other notions of fairness in rankings. We believe our research promotes research possibilities at the intersection of privacy and fairness in recommender systems, while also encouraging development of techniques for end-to-end privacy-preserving and fairness promoting pipelines for both producers and consumers in multi-stakeholder recommender systems.

Part III

PRIVACY-PRESERVING SYNTHETIC DATA GENERATION

"The most dangerous phrase in the language is, 'We've always done it this way.'"

Grace Hopper

Chapter 8
SYNTHETIC DATA

In the first two parts of this thesis, we leveraged DP-in-MPC to enable the training of ML models while preserving privacy and/or improving fairness. This required developing multiple MPC protocols for specific algorithms.

AI is evolving rapidly, with specialized algorithms emerging across diverse tasks and domains. Designing new MPC protocols for every algorithm or component in an AI pipeline is not scalable. As AI continues to be adopted by researchers and practitioners from a wide range of domains, the burden of adapting Privacy Enhancing Technologies (PETs) – such as the development of MPC protocols – becomes increasingly challenging. It is an arduous process that demands expertise and can slow down the pace of innovation. Not every researcher and practitioner will have the technical capacity to design or integrate such solutions. This highlights the need for a new perspective, an approach that can ease the adoption and use of existing AI algorithms without worrying about privacy, thereby supporting broader and faster adoption of AI.

In this part of the thesis, we explore the possibility of making data available that inherently preserves privacy – data that can be shared or used for ML modeling and data analytics without the complexity of adopting PETs for every AI task. This leads us to *synthetic data* as a privacy-preserving alternative.

8.1 Introduction

As mentioned in Chapter 1, AI's success is largely driven by the availability of large volumes of data. AI's progress in many critical domains, such as in healthcare, has been hindered by the phenomenon of data silos. In many cases, although vast amounts of data exist, they

remain locked up in silos, controlled by entities such as hospitals or banks, guarded by privacy regulations. Prominent examples include the N3C enclave of the NIH's National Center for Advancing Translational Sciences (NCATS), which contains COVID-19 patient data [280], and the "All of Us" program by the NIH, which is collecting genetic and health data from one million participants across the United States [281].

AI researchers are discouraged and held back by the onerous application and onboarding processes for each silo they wish to access, limiting their ability to conduct impactful and reproducible research [282, 283]. All these are due to obvious privacy reasons as mentioned in Chapter 1.

The solutions we have developed so far in this thesis offer a promising alternative by enabling privacy-preserving computation on data silos without sharing it. *But it does not make the data itself available.* As such, these solutions cannot be easily integrated into existing AI pipelines or quickly adapted to new algorithms.

Early efforts to publish privacy-preserving data primarily relied on de-identification techniques, which involved removing personally identifiable information (PII) such as names, addresses, or social security numbers from datasets. Linkage attacks, where adversaries combine de-identified data with auxiliary information, have shown that individuals can still be re-identified in such anonymized datasets [284, 285]. Moreover, de-identification in this manner often compromises the utility of the data, making it less effective for training AI models or conducting meaningful data analysis.

One approach to enabling secure and privacy-preserving data access that preserves the utility of data is through the use of secure data enclaves (e.g. N3C Enclave, All of Us). While secure data enclaves do have value in enabling privacy-aware research, granting limited access to a select group of researchers is not sufficient to democratize data access or foster a data-rich environment that accelerates AI advancements in privacy-sensitive domains. Furthermore, researchers are typically not allowed to combine data from different silos because each silo's data needs to be processed within its dedicated enclave. Even importing publicly available data into a silo is often not allowed.

This motivates us to explore *synthetic data* as an alternative mechanism for enabling data sharing and analysis without compromising privacy. Synthetic data is shareable, flexible, and applicable across domains and use cases. We briefly explain the concept of synthetic data next.

8.2 Synthetic Data Generation

Synthetic data (SD) offers a promising alternative to publish privacy-preserving data. Synthetic data is "artificial" data that mimics the characteristics of the real data (i.e. original data) but, crucially, without replicating any individual's personal information. This means that synthetic data retains the distribution and relationships of real data, but does not contain actual individuals' information. Rather than removing identifiers from real data, synthetic data generation (SDG) creates entirely new artificial data points. This makes it suitable for open publication and for fostering AI research, therefore enabling broader adoption of AI across domains.

Synthetic data can be generated using a range of techniques. The most effective approaches are those that learn from real data to capture its underlying structure and statistical properties. These data-driven generators can be broadly classified into two categories. The first category is comprised of statistical methods that rely on estimated probabilistic models of real-world distributions. Examples include multivariate Gaussian distribution modeling [286], and graphical models such as PrivBayes [287] or probabilistic graphical models (PGMs) [288]. The other category are AI-based methods that leverage deep-learning models such as generative adversarial networks (GANs) [289, 290], variational autoencoders (VAEs) [291], and diffusion models [292] to learn and reproduce complex data patterns. Once the models have learned the underlying patterns and distributions, they can be used to generate new, artificial data points (similar to an inference phase in an AI pipeline).

An important characteristic of synthetic data is that the generated data fits any data processing workflow designed for the original data, allowing a seamless switch between original and synthetic data. This means that synthetic data can be used by anyone with basic

data science skills in the comfort of their preferred AI pipelines and software, even if they have no technical knowledge about PETs.

As a result, synthetic data has become a common technique to fuel data science competitions. In a recent data challenge organized by the Food and Drug Administration (FDA)[1], for instance, participants were asked to develop AI models to predict cardiovascular health-related outcomes in Veterans [293]. Phase I participants were evaluated on synthetic data, with the top contenders invited to deploy their models in the Veterans Affairs (VA) enclave. Models that performed well on the synthetic data also performed well in the VA enclave during Phase II.

Data holders with sufficient data can often train their own synthetic data generators and publish synthetic datasets[2]. In many real-world scenarios, however, individual data holders often lack enough representative data to generate high-quality synthetic datasets on their own [294]. The 2023 U.S-U.K. PETs Prize Challenge, for instance, used synthetic data for financial fraud detection that was generated based on real data from the global financial institutions BNY Mellon, Deutsche Bank, and SWIFT[3]. Another common use case are rare diseases, with each data holder (clinical site) having data for only a few patients [295].

Many companies now offer "SDG-as-a-Service" (e.g. MostlyAI, Gretel, ...) where data holder(s) upload their private datasets to the service provider's platform, which then generates synthetic data on their behalf. For instance, in the PETs Prize Challenge mentioned above, the original real data from multiple sources was centralized and disclosed to Mostly AI for synthetic data generation. Such reliance on a trusted third party is not always desirable and may not even be legally permissible in many privacy-sensitive domains (see Chapter 1).

There is a need for techniques to generate synthetic data based on real, distributed data held by multiple data holders, without requiring these data holders to disclose their raw data. This means training generative models in distributed data settings – a setting very similar

[1]https://precision.fda.gov/challenges/31/results

[2]This is assuming that data holders have enough resources and/or expertise to do so.

[3]`https://petsprizechallenges.com/`

to that in Chapter 3, but now in the context of generative models rather than discriminative models.

This motivates Research Question 3 (RQ3) – "How can we generate synthetic data with input and output privacy guarantees when the real data is held by multiple data holders in an arbitrary way?" that we will address in the final part of this thesis. We note that, here, input privacy refers to protecting the privacy of the real data held by each data holder through MPC, while output privacy through DP ensures that the synthetic data does not leak sensitive individual information.

To address this, we can again leverage DP-in-MPC. Although developing MPC protocols for different generative algorithms is still necessary, the effort, number of protocols, and PET adoption burden are significantly reduced compared to what would be required if synthetic data were not available.

A wide variety of SDGs have been designed for various data modalities (tabular, images, time-series) and applications (healthcare, finance, mobility). In this thesis, we focus on tabular data. Tabular data is ubiquitous in many domains ranging from healthcare, to humanitarian action, education, and socioeconomic studies. DP synthetic tabular data can be used for ML tasks and data analysis, as well as enable answering an arbitrary number of statistical queries. Our specific objective is then to generate DP synthetic *tabular* data in a distributed setting, where the real data is partitioned across multiple data holders.

Among existing SDGs, statistical models, particularly those that follow the select-measure-generate template [296, 287, 297, 298], have shown optimal privacy-utility trade-offs in the centralized paradigm [299]. These SDGs provide DP guarantees under the global DP model. We briefly describe this template in Section 8.2.1.

8.2.1 Select-Measure-Generate

Consider a dataset D with n instances. D has a set of d attributes (features) denoted by $x = \{x_1, x_2, \ldots, x_d\}$. The domain for the attribute x_i is a finite, discrete set given by Ω_i, i.e., $|\Omega_i| = \omega_i$. The domain for x is the cartesian product of the domains of the individual

attributes, i.e. $\Omega = \prod_{i=1}^{d} \Omega_i$. Let $\mathcal{Q}$ represent a collection of measurement sets where each set q in $\mathcal{Q}$ is a set of attributes to measure, i.e. $q \subseteq x$.

A marginal on a set of attributes is denoted by $q \subseteq x$. It refers to a histogram computed over D for the attributes in q, i.e. it counts the number of occurrences in D for each $t \in \Omega_q$ where $\Omega_q = \prod_{x_i \in q} \Omega_i$. Example: all 2-way marginals will consist of all q where $|q| = 2$, i.e. all 2-combinations of features such as $\{x_i, x_j\}$ where $i, j \in \{1, \ldots, d\}$ and $i < j$. Informally, one can for instance think of the 2-way marginal on $\{Gender, Age\}$ as a histogram over all possible combinations of gender and age values in D. A k-way marginal is a marginal computed on k attributes, i.e., $|q| = k$. We denote the result of the computation of a marginal on q over D as $\mu_q(D)$, which is basically a vector of counts.[4] In what follows, we refer to a set of attributes as a *query*.

"Select-measure-generate" SDG algorithms aim to create synthetic tabular data that preserves key statistical properties, particularly the marginals, of the original dataset in centralized paradigm [298]. These methods focus on maintaining the accuracy of marginals for the predefined set of queries (i.e.$\mathcal{Q}$ also known as a workload) such that their values on the synthetic data are as close as possible to those on the real data.

The algorithm typically proceeds iteratively in three steps: select, measure, and generate synthetic data $\hat{D}$, as shown in Algorithm 12. The algorithm usually takes the real dataset D and a workload of queries $\mathcal{Q}$ as inputs, and runs for T iterations. It ensures output privacy through DP (global DP in the centralized setting), using the privacy parameters (ϵ, δ). The algorithm typically begins with setting up and initializing a generative model/synthesizer $\mathcal{G}$. During iterations, the algorithm learns a synthesizer $\mathcal{G}$ that best represents the data distribution of D. The learning proceeds in 3 main steps.

1. **Select.** This step chooses a query q_s from the set $\mathcal{Q}$ to evaluate on the real dataset. Since repeatedly using all queries may violate privacy, a differentially private selection mechanism (e.g. exponential mechanism) is generally used to choose the "most infor-

[4]We consider $\mu_q(D)$ to be a flattened vector for any k-way marginal.

Algorithm 12: Select-Measure-Generate Based SDG

Input: Dataset D, set of linear queries $\mathcal{Q}$, number of iterations T, and privacy parameter (ϵ, δ)

Output: $\hat{D}, \mathcal{G}$

Initialize synthesizer $\mathcal{G}$

for $i \in \{1, \ldots, T\}$ **do**

  Select: Privately select a query $q_s \in \mathcal{Q}$ using a DP mechanism based on a given selection criterion

  Measure: Compute $\hat{\mu}_{q_s} = \mu_{q_s}(D) + \text{Noise}(\epsilon, \delta)$

  Generate: Update $\mathcal{G}$ using $\hat{\mu}_{q_s}$ and generate synthetic data $\hat{D}$

end

mative" query. The goal is to select queries that will most improve the quality of the synthetic data.

2. **Measure.** Once a query is selected, its answer (i.e., marginal $\mu_{q_s}(D)$) is computed on real data. To preserve privacy, appropriate noise (usually a Gaussian or Laplace noise) is added to the computed marginal that results in a noisy answer $\hat{\mu}_{q_s}$.

3. **Generate.** This step now uses the noisy answer(s) (previous and current) to update the model $\mathcal{G}$ (e.g. a graphical model like PGM) that learns a joint distribution over the data.

Once $\mathcal{G}$ is learned it can be used to generate synthetic data $\hat{D}$. This means the synthesizer samples "artificial" data points from the learned distribution.

In this part of the thesis, we build upon the above descriptions and notations to develop MPC protocols for this family of generators (a.k.a. synthesizers). In Chapter 9, we develop MPC protocols to build a generalized framework that enables DP-in-MPC training of this family of synthesizers in a distributed data setting. Chapter 10 extends this and develops MPC protocols to build privacy-preserving end-to-end SDG pipeline, which includes secure pre-processing and secure hyperparameter tuning. Together, these two chapters present a cohesive framework for private, distributed synthetic data generation using DP-in-MPC.

Chapter 9

CaPS: COLLABORATIVE AND PRIVATE SYNTHETIC DATA GENERATION

As discussed in Chapter 8, synthetic data generation (SDG) emerges as a compelling approach to privacy-preserving data sharing. In particular, SDG with DP has received considerable attention from the research community (see e.g. [287, 289, 298, 299] and references therein). However, most of the existing SDG approaches operate in *the centralized paradigm*, i.e., they assume that the real data needed to train the synthesizer resides with one data holder, or, if the data originates from different data holders, that the latter are able to send their data to a trusted aggregator who in turn will use it as input for SDG algorithms. This raises privacy concerns as motivated in Chapter 1.

There is a substantial gap in the literature on solutions for SDG from distributed data sources while providing input privacy guarantees. While approaches such as [300, 301], which are based on FL [59], seem to provide a viable solution for this problem, they work either for horizontal (HFL) or for vertical (VFL) data partitioning, not both. Moreover, they rely on a single honest-but-curious aggregator, hence a single point of failure (See Chapter 2). Ramesh et al. [302] proposed to use MPC combined with specialized hardware (a *trusted* execution environment) to generate synthetic data from distributed sources; their approach still relies on the presence of a trusted entity.

In this chapter, we address the challenge of generating DP synthetic tabular data from multiple data holders without requiring centralized access to the real tabular data. Specifically, we propose a framework called `CaPS` that extends select-measure-generate based SDG methods (described in Chapter 8) to provide input privacy through the use of the DP-in-MPC paradigm.

We begin by reviewing related work in Section 9.1. Section 9.2 introduces our proposed solution and presents the corresponding MPC protocols. In Section 9.3, we evaluate the performance and effectiveness of our approach and finally summarize it in Section 9.4.

9.1 Related Work

To the best of our knowledge, there are only a handful of works on generating synthetic data from multiple private data sources [303, 301, 304, 302, 305, 160]. The research closest to ours is [156] that focuses on loosely-coordinated federated settings rather than synchronized horizontal distributed setting. All of these methods either work only for a given data distribution scenario or rely on specialized hardware. In contrast with existing work, we focus on generating synthetic data with the 'DP-in-MPC' paradigm for SDG algorithms that follow the select-measure-generate template, in a manner that works for any arbitrary distribution and does not require specialized hardware. In the general modular framework that we will introduce next, one can plug in other protocols from the MPC literature, including protocols for sampling noise, see e.g. [306, 307, 308].

9.2 Description of `CaPS`

9.2.1 Key Idea

We observe that SDG techniques based on the 'select-measure-generate' template only need to perform computations on the private data in the 'select' and 'measure' steps. The outputs of these steps are protected with DP guarantees. Computations thereafter, such as the 'generate' step, are *post-processing* steps[1].

Following this observation, it is sensible to (1) secret-share the data, (2) carry out the 'select' and 'measure' steps with DP-in-MPC over the secret-shared data, and (3) publish the perturbed marginals for consumption by the 'generate' step. We capture this idea in a framework called `CaPS` (Figure 9.1) to generate DP synthetic tabular data from distributed

[1]Outputs of differentially private mechanisms are immune to post-processing, see Section 2.3 in Chapter 2.

data holders that works for any kind of data partitioning (horizontal, vertical, or mixed).

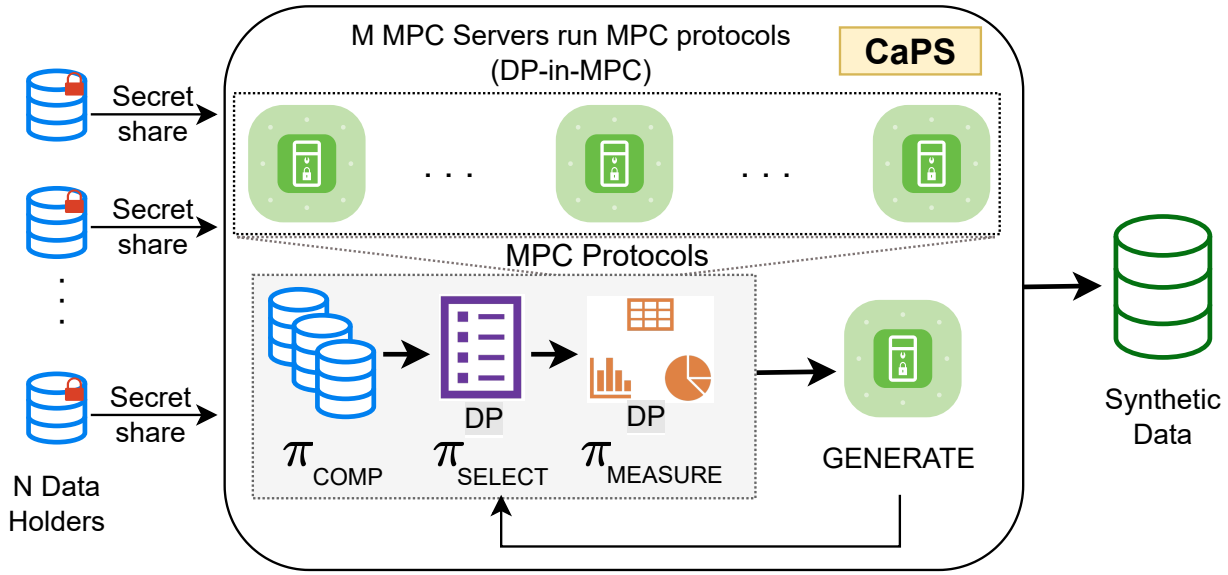


Figure 9.1: `CaPS`: A framework that leverages 'DP-in-MPC' to collaboratively and privately generate tabular synthetic data using marginal-based SDG techniques with the 'select-measure-generate' template. Servers run MPC protocols for 'select' and 'measure'. The 'generate' step is performed over DP measurements.

We designed `CaPS` to be modular. The individual algorithms for each of the select, measure, and generate steps can be replaced. Thus, the MPC protocols for the select and measure steps can be replaced or switched. This means that `CaPS` can be easily adapted to any SDG algorithm by simply replacing or designing the MPC protocols for the given select and measure steps. To illustrate this, we apply `CaPS` to create privacy-preserving collaborative counterparts of the state-of-the-art SDG algorithms MWEM+PGM [309] and AIM [297]. We design MPC protocols for the select and measure steps of these algorithms and develop pipelines to generate tabular synthetic data.

Algorithm 13: `CaPS`: Generating tabular synthetic data with DP-in-MPC in the select-measure-generate template

Input: Domain Ω, Queries $\mathcal{Q}$

Output: Synthetic Data $\hat{D}$

Parameters: Privacy parameters ϵ, δ

Hyper-Parameters: Hyper-parameters for $\mathcal{G}$

H: $[\![\mu_q^i]\!], [\![D_i]\!] \leftarrow$ LOCAL_COMPUTE( )

Initialization by S_1: $\hat{D} \leftarrow$ INIT( )

COMPUTE ANSWERS: $[\![\mu_q]\!] \leftarrow \pi_{\mathsf{COMP}}([\![\mu_q^i]\!], [\![D_i]\!])$

repeat

  SELECT: $q_s \leftarrow \pi_{\mathsf{SELECT}}([\![\mu_q]\!], \hat{\mu}_q)$ // S_1: $\hat{\mu}_q$ on $\hat{D}$

  MEASURE: $\hat{\mu}_s \leftarrow \pi_{\mathsf{MEASURE}}([\![\mu_{q_s}]\!], q_s)$

  GENERATE by S_1: $\hat{D} \leftarrow F(\hat{\mu}_s, q_s)$

until terminate

Problem Formulation. We consider a set of N honest data holders $\mathbf{H} = \{H_1, H_2, \ldots, H_N\}$. Each data holder H_i holds a dataset D_i such that[2] $D = \bigcup_{i=1}^{N} D_i$. A marginal-based synthesizer $\mathcal{G}$ takes a workload $\mathcal{Q}$ and Ω as input and outputs DP tabular synthetic data $\hat{D}$. $\mathcal{G}$ is as defined for the centralized paradigm. Our goal is to generate $\hat{D}$ in the distributed setting with $\mathbf{H}$. To do so, we consider a set of M independent MPC servers $\mathbf{S} = \{S_1, S_2, \ldots, S_M\}$ that perform computations over distributed private data.

Algorithm 13 outlines the flow of our proposed framework `CaPS`. The key idea is to identify the steps of the SDG algorithm $\mathcal{G}$ that require computations across data from multiple data holders and invoking the MPC servers to perform such computations. If the output of these computations is differentially private, the outputs can be revealed and the other steps can be done in-the-clear to avoid the overhead due to MPC.

[2]We do not make any assumptions on how exactly the data is partitioned among the data holders, i.e. each data holder may have (partial) rows and/or (partial) columns of D.

9.2.2 Setup Phase

The framework begins with a setup phase in which the MPC servers **S** load and compile the MPC protocols for the chosen $\mathcal{G}$, the dimensions of the dataset D, and the MPC scheme. On Line 1 in Algorithm 13, the data holders **H** perform local computations as defined by `LOCAL_COMPUTE()`; see below for a brief description. We consider S_1 to be the initiating server that is responsible for generating $\hat{D}$ and publishing the generated synthetic data to the required stakeholder. S_1 begins with randomly initializing $\hat{D}$ as defined by `INIT()` on Line 2. We note that this initialization is the same as in the centralized version of $\mathcal{G}$.

Description of `LOCAL_COMPUTE()`. Each data holder locally computes answers to the queries in the workload $\mathcal{Q}$ to the best of its ability. We denote the answers computed by H_i as $\mu_q^i, \forall q \in Q$. If data holder H_i does not have all the attributes contained in q, then H_i is unable to compute a local answer for q, so μ_q^i is assigned a vector of 0s. All the data holders then convert their local answers into secret-shares as per the given secret-sharing scheme as $[\![\mu_q^i]\!]$. In a non-horizontal distribution scenario, each H_i additionally secret-shares their private local dataset D_i as $[\![D_i]\!]$.

The setup phase is followed by the training phase where the MPC servers **S** run the MPC protocols for $\mathcal{G}$ to generate synthetic data $\hat{D}$ (Lines 3 – 8 in Algorithm 13). This phase in general does not require the N data holders to stay online or participate in the generation process. However, to reduce MPC overhead, one can optimize `CaPS` depending on the availability of any of the data holders (See Section 9.2.5). The training phase proceeds with invoking MPC protocols for computing query answers μ_q over the (secret-shared) real data (π_{COMP}), and then repeatedly selecting a query q_s (π_{SELECT}) that needs attention and measuring/preparing a noisy answer $\hat{\mu}_s$ (π_{MEASURE}) that can be disclosed.

9.2.3 Computation of Answers on Distributed Data

The MPC servers run π_{COMP} to compute secret-shares of $[\![\mu_q]\!]$ (Line 3 in Algorithm 13). In an arbitrary setting, π_{COMP} can compute $[\![\mu_q]\!]$ by directly performing computations on $[\![D_i]\!], \forall i$.

Such computations do not require data holders to perform local computations of answers, but does increase the MPC overhead.

To optimize MPC overhead, we propose for each query q that all the data holders who have all attributes in q perform one-time local computations to compute partial query answers $[\![\mu_q^i]\!]$ (see Line 1 in Algorithm 13); note that for queries involving only one attribute, i.e. one-way marginals, many such local computations can be done. The first step in π_{COMP} is then to simply add the secret-shares of the locally computed answers from each data holder $[\![\mu_q^i]\!], \forall q \in Q$ (Lines 1–5 in Protocol 14). This is then followed by computing secret-shares of workload answers $[\![\mu_{q^*}]\!]$ on $[\![D_i]\!]$ for queries q^* that can not be computed locally, i.e. queries containing attributes that are distributed across different data holders. We call the set of these queries Q^*. After the execution of π_{COMP}, the MPC servers $\mathbf{S}$ hold the secret-shares of $[\![\mu_q]\!], \forall q \in Q$.

Note that when the data is horizontally partitioned, $Q^* = \emptyset$, and the workload answers $[\![\mu_q]\!]$ can be fully computed by adding secret-shares of locally computed results, i.e. Lines 1–5 in Protocol 14. This results in a total of $(N-1) \cdot |Q| \cdot \max(\omega_q)$ additions in MPC implying that π_{COMP} is linear in the number of data holders for a given $\mathcal{Q}$ and Ω for horizontally distributed data. When $Q^* \neq \emptyset$, secret-sharings of the complete query answers can be obtained by letting the servers $\mathbf{S}$ perform more computation over the secret-shared data of the data holders, as explained below.

MPC protocol π_{COMP}. Protocol 14 computes $[\![\mu_q]\!]$ and is optimized for AIM and MWEM+ PGM in an arbitrary distributed setting. The workload $\mathcal{Q}$ in these algorithms consists of 1-way and 2-way marginals. Lines 1–5 aggregate the workload answers from all the data holders by performing addition of secret-shares of $[\![\mu_q^i]\!]$. This aggregation accounts for computation of 1-way marginals for any arbitrary setting and 2-way marginals for the horizontal setting.

If $Q^* \neq \emptyset$, then $\mathbf{S}$ proceed to assemble a secret-sharing of the joint dataset using the protocol π_{JOIN} on Line 7. π_{JOIN} initializes a matrix $[\![D]\!]$ of dimensions $n \times d$. We assume that

Protocol 14: π_{COMP}:MPC Protocol for **COMPUTE ANSWERS**

Input : secret-shares of locally computed answers $[\![\mu_q^i]\!]$, secret-shares of data $[\![D_i]\!]$ (needed if not horizontal), Queries $\mathcal{Q}$, including queries Q^* with attributes that are distributed among data holders, Domain Ω

Output: $[\![\mu_q]\!], \forall q \in Q$

for $i = 1$ *to* N **do**
  for *each* $q \in Q$ **do**
    $[\![\mu_q]\!] \leftarrow [\![\mu_q]\!] + [\![\mu_q^i]\!]$
  end
end
if $Q^* \neq \emptyset$ **then**
  $[\![D]\!] \leftarrow \pi_{\mathsf{JOIN}}([\![D_i]\!] | i = 1 \dots N)$
end
for *each* $q^* = \{a_1, a_2\}$ *in* Q^* **do**
  for $i = 1$ *to* n **do**
    $[\![x]\!] \leftarrow [\![D_{a_1}[i]]\!]$
    $[\![y]\!] \leftarrow [\![D_{a_2}[i]]\!]$
    for $j \in \Omega_{a_1}$ *and* $k \in \Omega_{a_2}$ **do**
      $[\![m]\!] \leftarrow \pi_{\mathsf{MUL}}(\pi_{\mathsf{EQ}}([\![x]\!], j), \pi_{\mathsf{EQ}}([\![y]\!], k))$
      $[\![\mu_{q^*}[j * |\Omega_{a_2}| + k]]\!] \leftarrow [\![\mu_{q^*}]\!] + [\![m]\!]$
    end
  end
end
return $[\![\mu_q]\!]$, $\forall q \in Q$

all the data samples are aligned before π_{JOIN}.[3] π_{JOIN} then combines all the secret-sharings $[\![D_i]\!]$ into a secret-sharing $[\![D]\!]$ of the overall dataset D using simple assignment statements.

[3]This can be achieved using protocols for Private Set Intersection (PSI) available in literature.

In Protocol 14, π_{JOIN} requires $n.d$ assignment operations and works for any number of data holders.

S then run Lines 9–18 to compute secret-sharings of 2-way marginals. Consider a 2-way marginal $q^* \in Q^*$ which is represented as a pair $q^* = \{a_1, a_2\}$. $[\![D_{a_1}]\!]$ denotes the secret-shares of $[\![D]\!]$ for the a_1 attribute and $[\![D_{a_2}]\!]$ for a_2. On Lines 10–17, the MPC servers iterate over all the n data samples in $[\![D_{a_1}]\!]$ and $[\![D_{a_2}]\!]$ to compute the number of occurrences of all the combinations of values in the domains Ω_{a_1} and Ω_{a_2}. Line 14 in Protocol 14 relies on MPC primitives for multiplication π_{MUL} and equality testing π_{EQ} to check if a combination of attribute values occurs in the data. On Line 14, the MPC servers compute $[\![m]\!]$ which is a secret-sharing of 0 or 1 that adds to the the number of occurrences of the combination of attributes values held in $[\![\mu_{q^*}]\!]$.

We note that a major MPC overhead is due to Line 14 performed in the loop. This indicates that for an arbitrary data setting, there are $(N-1)\cdot|Q|\cdot\max(\omega_q)+n\cdot\max(\omega_q)^2\cdot|Q^*|$ additions, $2\cdot n\cdot\max(\omega_q)^2\cdot|Q^*|$ equality checks, and $n\cdot\max(\omega_q)^2\cdot|Q^*|$ multiplications. Given a fixed set of queries $\mathcal{Q}$ and fixed domain Ω, Protocol 14 grows linearly in the total number of samples n and the total number of data holders N.

Extending Protocol 2 to compute p-way marginals. MPC protocols are typically designed as specific circuits, composed of a sequence of addition and multiplication gates. Therefore, changing the functionality to be implemented in MPC usually requires the design of a new circuit. Protocol 14 is specifically designed to suit algorithms that consider 1-way and 2-way marginals. One can extend Protocol 14 to compute p-way marginals. Below we describe extending π_{COMP} to compute 3-way marginals in the following ways:

- A direct extension of Protocol 14 to compute p-way marginals is possible. Say the p marginals are represented by a set $\mathcal{M}$. This requires p conditionals instead of 2 on Line 13 of Protocol 14. Instead of 2 equality checks on Line 14, this requires p equality checks. This means a total of $p\cdot n\cdot\prod_{i\in\mathcal{M}}\omega_i$ equality checks and $p\cdot\prod_{i\in\mathcal{M}}\omega_i$ multiplications. As mentioned earlier, π_{COMP} can be replaced by other protocols such

as [310].

For $p = 3$, change $q^* = \{a_1, a_2, a_3\}$ and add $[\![z]\!] \leftarrow [\![D_{a_3}[i]]\!]$ after Line 12 on Protocol 14. Change Line 13 to include $l \in \Omega_{a_3}$ and add a multiplication operation with $\pi_{\mathsf{EQ}}([\![z]\!], l$ on Line 14. Line 15 should be modified to index accordingly. Please see Protocol 15.

- Another possible way to compute 3-way marginals is by repeated computations of 2-way marginals. We compute for $\{a_1, a_2\}$ first and then get a column with the concatenated values for the feature $a_1||a_2$ and then compute 2-way marginal for $\{a_1||a_2, a_3\}$. Depending the domain size of each column, this might perform better or worse than the above method.

Protocol 15: MPC Protocol to compute p-way marginals

Input : secret-shares of data $[\![D_i]\!]$ (needed if not horizontal), Queries $\mathcal{Q}$, including queries Q^* with attributes that are distributed among data holders, Domain Ω

Output: $[\![\mu_q]\!], \forall q \in Q$

1 **if** $Q^* \neq \emptyset$ **then**
2 $[\![D]\!] \leftarrow \pi_{\mathsf{JOIN}}([\![D_i]\!] | i = 1 \dots N)$
3 **end**
4 **for** *each* $q^* = \{a_1, a_2, \dots, a_p\}$ *in* Q^* **do**
5 **for** $i = 1$ *to* n **do**
6 $[\![x]\!] \leftarrow [\![D_{a_1}[i]]\!]$; $[\![y]\!] \leftarrow [\![D_{a_2}[i]]\!]$; $\dots$ $[\![z]\!] \leftarrow [\![D_{a_p}[i]]\!]$
7 **for** $j \in \Omega_{a_1}$ *and* $k \in \Omega_{a_2} \cdots$ *and* $l \in \Omega_{a_p}$ **do**
8 $[\![m]\!] \leftarrow \pi_{\mathsf{MUL}}(\pi_{\mathsf{EQ}}([\![x]\!], j), \pi_{\mathsf{EQ}}([\![y]\!], k), \cdots, \pi_{\mathsf{EQ}}([\![z]\!], l))$
9 index $\leftarrow$ compute the index for marginal vector
10 $[\![\mu_{q^*}[\text{index}]\!] \leftarrow [\![\mu_{q^*}]\!] + [\![m]\!]$
11 **end**
12 **end**
13 **end**
14 **return** $[\![\mu_q]\!], \forall q \in Q$

Design and optimization of every possible MPC protocols for all scenarios of select-measure-generate paradigm is task-specific. This chapter provides an important baseline

framework that can be adapted to particular algorithms. In Chapter 10, we optimize protocol to compute marginals for a specific use-case.

Discussion on common datasets. CaPS, in principle, works for all the distributed settings due to the modularity offered. This includes scenarios where the data holders hold common data as shown in Table 9.1-9.2. To illustrate how MPC can be leveraged, Protocol 14 for π_{COMP} considers disjoint dataset.

π_{COMP} also works with very little modifications when the data is common across some of the data holders as shown in Table 9.1-9.2 for two data holders. In such case, the computations in π_{COMP} begin with executing π_{JOIN} that results in a union of all datasets. This is followed by computation of all the marginals in MPC (such as computations done on Lines 9–18 of Protocol 14 to compute 2-way marginals). This requires removing Lines 1–6 in Protocol 14 and having $Q^* = Q$. This will of course result in change in the number of computations performed in MPC.

Table 9.1: Data held by H_1

ID	a	b
Alice	0	1
Bob	1	1

Table 9.2: Data held by H_2

ID	a	b	c
Alice	0	1	2
Charlie	1	0	2

9.2.4 Selection of the Query

In the select step, **S** run the MPC protocol π_{SELECT} to select the query q_s based on which the estimate of $\hat{D}$ should be improved (Line 5 in Algorithm 13). We note that π_{SELECT} is independent of the data distribution setting as the set of MPC servers **S** already have secret-shares of the computed answers $[\![\mu_q]\!]$ from all the data holders.

Protocol 16 provides a general template for selecting q_s. S_1 who holds $\hat{D}$ computes $\hat{\mu}_q$ and

provides it as input to Protocol 16. On Line 1 the MPC servers compute a secret-sharing of the error $[\![err]\!]$ between $[\![\mu_q]\!]$ and $\hat{\mu}_q$ by taking the sum of absolute differences of the answers using π_{ERR}[4]. This is followed by computing secret-shares of the normalized errors $[\![err]\!]$ using π_{NORM}[5] on Line 2. A secret-shared probability vector over the queries is then constructed using π_{PROB} on Line 3 which takes the secret-shared normalized errors $[\![err]\!]$, the privacy parameters – the scale b and sensitivity w – as input.

Protocol 16: π_{SELECT}: MPC Protocol for **SELECT**

Input : secret-shares of $[\![\mu_q]\!]$, Estimated answer $\hat{\mu_q}$, length of domains ω_q for each q, privacy parameter b, sensitivity w

Output: Selected query q_s

1 $[\![err]\!] \leftarrow \pi_{\mathsf{ERR}}([\![\mu_q]\!], \hat{\mu_q})$

2 $[\![err]\!] \leftarrow \pi_{\mathsf{NORM}}([\![err]\!])$

3 $[\![prob]\!] \leftarrow \pi_{\mathsf{PROB}}([\![err]\!], b, w, \omega_q)$

4 $[\![s]\!] \leftarrow \pi_{\mathsf{RC}}([\![prob]\!])$

5 Reveal s **return** q_s

What happens within π_{ERR}, π_{NORM} and π_{PROB} may differ from one SDG algorithm to the next. We present π_{SELECT} with these subprotocols specifically for AIM and MWEM+PGM in Protocols 18 and 20 respectively.

Finally on Line 4, **S** run subprotocol π_{RC} to randomly select the query using exponential mechanism, thus implementing the DP-in-MPC paradigm. π_{RC} outputs the secret-shares of the randomly chosen index $[\![s]\!]$, which is revealed to S_1. The query with index s is selected as q_s. The MPC overhead due to π_{SELECT} is in the order of $|Q| \times \max(\omega_q)$ of the computations involved. This means that π_{SELECT} does not depend on the number of data holders N or the total number of samples n but on he given number of queries $|Q|$ and the domain of the

[4] π_{ERR} computes the secret-shares of error as defined by the given equation between secrets shares of two vectors.

[5] π_{NORM} normalizes, either by scaling or computing L1 norm, the secret-shares of the error vector.

dataset Ω.

MPC protocol for π_{RC}. Protocol 17 takes as input the secret-shares of the computed probabilities, $[\![prob]\!]$ and chooses the first index of the probability vector for which the probability value satisfies a condition based on random threshold. Lines 1–8 compute the random threshold $[\![\mathrm{t}]\!]$ using the MPC primitives for multiplication (π_{MUL}), random number generation ($\pi_{\mathsf{GR-RANDOM}}(0,1)$). Lines 10–15 select the index conditioned on the threshold without exiting the loop, as it could reveal the value of returned index to the adversary. Note that π_{RC} can be considered as an implementation of the exponential mechanism mentioned in Chapter 2.

We design Lines 10–15 to prevent such side-channel attacks. To understand this part of the code, note that we have a list $p[1..|Q|]$ of non-decreasing values, i.e. the cumulative probability sums, and the MPC servers have to find the first index i in $p[1..|Q|]$ for which $p[i] > \mathrm{t}$. In a mock example with $|Q| = 10$, and assuming that the first such $p[i]$ value is at position 7, the tests on Line 11 will generate the results 0,0,0,0,0,0,1,1,1,1. On Line 12, these results are accumulated in k, which eventually becomes 4, and the desired index is computed as $N - (k-1) = 10 - 3 = 7$. Lines 14–15 take care of the edge case when $p[i] \leq \mathrm{t}$ for all i (i.e. k is 0). We protect the value of k by employing MPC primitives for multiplication to simulate a conditional statement.

Description of π_{SELECT} for AIM. Protocol 18 is the straight forward implementation of the select step from the centralized algorithm of AIM. Lines 2–7 compute the secrets shares of errors between the answers from real and synthetic data and form the subprotocol π_{ERR}. This protocol relies on $\pi_{\mathsf{L1-NORM}}1$ to compute the L1-norm of the errors. Protocol 19 computes secret-shares of L1-norm of the error vector. This protocol takes the input size m which is the length of the answer for for query q_i, i.e. $\mathrm{m} = \omega_{q_i}$ (this is an implementation details where we consider the unpadded vector answer). Lines 9–12 normalize the secret-shares of errors by scaling it with secret-share of maximum error computed using primitive π_{MAX}. Lines 9–12 form the subprotocol for normalize π_{NORM}. Line 14 computes secret-shares of

Protocol 17: π_{RC}: MPC Protocol for random selection

Input : secret-shares of probability vector $[\![prob]\!]$, length of probability vector $|Q|$

Output: secret-shares of selected index $[\![s]\!]$

1 sum $\leftarrow 0$
2 Initialize a vector **p** of length N
3 **for** $i = 1$ *to* $|Q|$ **do**
4 $\quad [\![\text{sum}]\!] \leftarrow [\![\text{sum}]\!] + [\![prob[i]]\!]$
5 $\quad [\![p[i]]\!] \leftarrow [\![\text{sum}]\!]$
6 **end**
7 $[\![\text{r}]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$ // with protocol for random number generation $\pi_{\mathsf{GR-RANDOM}}$
8 $[\![\text{t}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{sum}]\!], [\![\text{r}]\!])$
9 $k \leftarrow 0$ **for** $i = 1$ *to* $|Q|$ **do**
10 $\quad [\![\text{c}]\!] \leftarrow \pi_{\mathsf{GT}}([\![p[i]]\!], [\![\text{t}]\!])$
11 $\quad [\![k]\!] \leftarrow [\![k]\!] + [\![\text{c}]\!]$
12 **end**
13 $[\![\text{c}]\!] \leftarrow \pi_{\mathsf{EQ}}([\![k]\!], 0)$
14 $[\![s]\!] \leftarrow N - \pi_{\mathsf{MUL}}([\![k]\!] - 1, 1 - [\![\text{c}]\!])$
15 **return** $[\![s]\!]$

the probabilities by relying on the subprotocol π_{SOFTMAX}. This is defined as π_{PROB} for AIM. Finally on Line 16, MPC protocol for exponential mechanism is called π_{RC} that outputs the selected query based on computed secret-shares of the probabilities.

Description of π_{SELECT} for MWEM+PGM. Protocol 20 is the straight forward implementation of the select step from the centralized algorithm of MWEM+PGM. Lines 2–9 compute the secrets shares of errors between the answers from real and synthetic data and form the subprotocol π_{ERR}. Lines 11–14 normalize the secret-shares of errors by scaling it with secret-share of maximum error computed using primitive π_{MAX}. Lines 11–14 form the

Protocol 18: π_{SELECT}: MPC Protocol for **SELECT** for AIM

Input : Secrets shares of $[\![\mu_q]\!]$, Estimated answer $\hat{\mu}_q$, length of domains ω_q for each q, privacy parameters ϵ, a vector bias b for each q, weight for queries w , max_sensitivity s

Output: Selected query q_s

1 ********* Compute errors - π_{ERR} *********

2 Initialize a vector err of length $max(\omega_q)$

3 **for** $i = 1$ *to* $|Q|$ **do**

4 $\quad [\![\text{diff}]\!] \leftarrow [\![\mu_{q_i}]\!] - \hat{\mu}_{q_i}$

5 $\quad [\![err[i]]\!] \leftarrow \pi_{\mathsf{L1-NORM}} 1([\![\text{diff}]\!], \omega_{q_i})$// computes L1 norm of the errors

6 $\quad [\![err[i]]\!] \leftarrow \pi_{\mathsf{MUL}}(w[i], [\![err[i]]\!] - b[i])$

7 **end**

8 ********* Normalize errors - π_{NORM} *********

9 $[\![\text{max\_err}]\!] \leftarrow \pi_{\mathsf{MAX}}([\![err]\!])$ **for** $i = 1$ *to* $|Q|$ **do**

10 $\quad [\![err[i]]\!] \leftarrow [\![err[i]]\!] - [\![\text{max\_err}]\!]$

11 **end**

12 ********* Compute probabilities - π_{PROB} *********

13 $[\![prob]\!] \leftarrow \pi_{\mathsf{SOFTMAX}}(0.5 \cdot \epsilon \cdot (1/s) \cdot [\![err]\!])$

14 ********* Select random index - π_{RC} *********

15 $[\![s]\!] \leftarrow \pi_{\mathsf{RC}}([\![prob]\!])$

16 Reveal s

17 **return** q_s

subprotocol for normalize π_{NORM}. Line 16 computes secret-shares of the probabilities by relying on the subprotocol π_{SOFTMAX}. This is defined as π_{PROB} for MWEM+PGM. Finally on Line 18, MPC protocol for exponential mechanism is called π_{RC} that outputs the selected query based on computed secret-shares of the probabilities.

9.2.5 Measuring Answer to the Selected Query

In the measure step, **S** run the MPC protocol π_{MEASURE} to compute the noisy answer to selected query q_s (Line 6 in Algorithm 13). As **S** holds the secrets shares $[\![\mu_s]\!]$ computed for any arbitrary data distribution, π_{MEASURE} computes $[\![\hat{\mu}_s]\!]$ by generating the secret-shares of

Protocol 19: $\pi_{\mathsf{L1-NORM}}$: MPC Protocol to compute L1-norm

Input : Input vector $[\![\text{diff}]\!]$, length of vector m

Output: Normalized vector $[\![err[i]]\!]$

1 sum $\leftarrow 0$

2 **for** $i = 1$ *to* m **do**

3 $\quad [\![\text{sign}]\!] \leftarrow \pi_{\mathsf{LT}}([\![\text{diff[i]}]\!], 0)$

4 $\quad [\![\text{abs\_diff}]\!] \leftarrow \pi_{\mathsf{MUL}}(1 - 2 \cdot [\![\text{sign}]\!], [\![\text{diff[i]}]\!])$

5 $\quad [\![\text{sum}]\!] \leftarrow [\![\text{sum}]\!] + [\![\text{abs\_diff}]\!]$

6 **end**

7 **return** $[\![sum]\!]$

noise vector $[\![\gamma]\!]$ and adding it to the $[\![\mu_s]\!]$. Once the **S** compute $[\![\hat{\mu}_s]\!]$, it is revealed to S_1 for generation step.

The MPC overhead is mainly due to generation of $[\![\gamma]\!]$. To reduce MPC overhead in an arbitrary distributed setting, one can consider the availability of the data holder who holds the attributes defined in q_s and request for generation of the noisy vector (S_1 plays the role of sending request and receiving response). This is a direct optimization in a vertical distributed setting when the selected query is a 1-way marginal. However, the advantage of generating noise in MPC is that the noise is added in a correct and private manner, such that private inputs cannot be reconstructed back (Recall discussion in Chapter 2). One can also optimize the MPC primitives to generate the secret-shares of noise as we consider in Protocol 21.

Protocol 21 computes Gaussian noise as required by AIM and MWEM+PGM for arbitrary distributed setting. To do so, we consider Irwin-Hall approximation to generate Gaussian samples on Lines 2 – 7 as it reduces the MPC overhead considerably. The protocol for generating noise requires $(13.\omega_s + \max(\omega_q))$ additions and $12.\omega_s$ random bit generations. These lines can be replaced by other MPC protocols for other sampling techniques such as the Box-Muller method for which we developed the protocol π_{GSS} in Chapter 3.

A few algorithms such as MWEM [296] consider adding Laplace noise instead. In such

Protocol 20: π_{SELECT}: MPC Protocol for **SELECT** for MWEM+PGM

Input : Secrets shares of $[\![\mu_q]\!]$, Estimated answer $\hat{\mu}_q$, length of domains ω_q for each q, privacy parameters ϵ

Output: Selected query q_s

1 ********* Compute errors - π_{ERR} *********
2 Initialize a vector err of length $max(\omega_q)$
3 **for** $i = 1$ *to* $|Q|$ **do**
4 $\quad [\![\text{diff}]\!] \leftarrow [\![\mu_{q_i}]\!] - \hat{\mu}_{q_i}$
5 $\quad [\![\text{sign}]\!] \leftarrow \pi_{\mathsf{LT}}([\![\text{diff}]\!], 0)$
6 $\quad [\![\text{abs\_diff}]\!] \leftarrow \pi_{\mathsf{MUL}}(1 - 2 \cdot [\![\text{sign}]\!], [\![\text{diff}]\!])$ // computes absolute difference of error between answers on real data and synthetic data
7 $\quad [\![\text{sum}]\!] \leftarrow \pi_{\mathsf{SUM}}(1 - 2 \cdot [\![\text{abs\_diff}]\!])$
8 $\quad [\![err[i]]\!] \leftarrow [\![\text{sum}]\!] - \omega_{q_i}$
9 **end**
10 ********* Normalize errors - π_{NORM} *********
11 $[\![\text{max\_err}]\!] \leftarrow \pi_{\mathsf{MAX}}([\![err]\!])$
12 **for** $i = 1$ *to* $|Q|$ **do**
13 $\quad [\![err[i]]\!] \leftarrow [\![err[i]]\!] - [\![\text{max\_err}]\!]$
14 **end**
15 ********* Compute probabilities - π_{PROB} *********
16 $[\![prob]\!] \leftarrow \pi_{\mathsf{SOFTMAX}}(0.5 \cdot \epsilon \cdot [\![err]\!])$
17 ********* Select random index - π_{RC} *********
18 $[\![s]\!] \leftarrow \pi_{\mathsf{RC}}([\![prob]\!])$
19 Reveal s
20 **return** q_s

cases, Lines 2 – 7 can be replaced by Protocol 22. We breifly describ ethis protocol below.

In Protocol 22, the noise is sampled as $b \cdot \ln \mathrm{l} \cdot$ c where b is the scale, l is a random value drawn from the uniform distribution in [0,1] and c is a random value selected from $\{-1, 1\}$. On Lines 3–4 of Protocol 22, the MPC servers straightforwardly compute l and its natural log. To compute c, the servers, on line 5, generate secret-shares of a random bit $[\![\mathrm{r}]\!]$, i.e. a

Protocol 21: π_{MEASURE}: MPC Protocol for **MEASURE** using Gaussian noise

Input : Secrets shares of $[\![\mu_{q_s}]\!]$, length of domain ω_s for q_s, scale b

Output: Noisy measurement $\hat{\mu}_s$

1 Initialize vector $[\![\gamma]\!]$ of length $\max(\omega_r)$ with 0s
2 **for** *i = 0 to* ω_{q_s} **do**
3 $[\![sum]\!] \leftarrow 0$
4 **for** *j = 0 to 12* **do**
5 $[\![sum]\!] \leftarrow [\![sum]\!] + \pi_{\mathsf{GR-RANDOM}}(0, 1)$
6 **end**
7 $[\![\gamma[i]]\!] \leftarrow [\![sum]\!] - 6$
8 **end**
9 $[\![\hat{\mu}_s]\!] \leftarrow [\![\mu_s]\!] + b.[\![\gamma]\!]$
10 **return** $\hat{\mu}_s$

value $\in \{0, 1\}$ is chosen, where each value has a chance of 50% to be chosen. On line 6, the servers transform r to a value $\in \{-1, 1\}$ using the logic c = 2· r −1. Lines 7–10 is straightforward computation of the noise vector $[\![\gamma[i]]\!]$ and noisy measurement $[\![\hat{\mu}_s]\!]$.

One can also replace the Lines 3 – 6 by the MPC protocol π_{LAP} that we developed in Chapter 6 in Part II of the thesis. Lines 5 – 6 of π_{LAP} will then need to replaced by the pseudocode below

$[\![l]\!] \leftarrow -1 \cdot \pi_{\mathsf{MUL}}([\![\ln_u]\!], [\![\mathrm{sgn}_u]\!])$

$[\![\gamma[i]]\!] \leftarrow [\![l]\!]$

9.2.6 Generation of Synthetic Data

S_1 takes the noisy measurement $\hat{\mu}_s$ and runs the estimate algorithm of $\mathcal{G}$ to generate $\hat{D}$. S_1 can run this step without the need of MPC protocols as it takes DP input which means that the privacy of D is preserved due to the post-processing property of DP.

Protocol 22: π_{MEASURE}: MPC Protocol for **MEASURE** using Laplacian noise

Input : Secrets shares of $[\![\mu_{q_s}]\!]$, length of domain ω_s for q_s, scale b

Output: Noisy measurement $\hat{\mu}_s$

Initialize vector $[\![\gamma]\!]$ of length $\max(\omega_r)$ with 0s

for *i = 0 to* ω_{q_s} **do**

  $[\![l]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$ // with protocol for random number generation $\pi_{\mathsf{GR-RANDOM}}$

  $[\![\text{ln\_l}]\!] \leftarrow \pi_{\mathsf{LN}}([\![\text{l}]\!])$ // with secure logarithm protocol π_{LN}

  $[\![\text{r}]\!] \leftarrow \pi_{\mathsf{GR-RNDM-BIT}}()$ // with protocol for random bit generation $\pi_{\mathsf{GR-RANDOM}}$

  $[\![\text{c}]\!] \leftarrow 2 \cdot [\![\text{r}]\!] - 1$ $[\![\gamma[i]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{ln\_l}]\!], [\![\text{c}]\!])$ // with secure multiplication protocol π_{MUL}

end

$[\![\hat{\mu}_s]\!] \leftarrow [\![\mu_s]\!] + b.[\![\gamma]\!]$

return $\hat{\mu}_s$

9.2.7 Note on Modularity

The protocols π_{SELECT} and π_{MEASURE} can be replaced by custom MPC protocols for the select and measure steps in one's SDG algorithm of choice. The modularity of these protocols is evident from the variety of protocols we presented in previous sections. In this chapter, we specifically design MPC protocols for AIM and MWEM+PGM (also MWEM). Our choice of these algorithms is based on the literature showcasing that these are the state-of-the-art synthetic tabular data generation techniques [297, 311]

`CaPS` is applicable to workload-based synthetic generation algorithms too such as RAP [312] that follow the select-measure-generate template. Our π_{MEASURE} protocol can be used as is for other algorithms such as RAP and MST [298]. If the considered distribution setting is only horizontal, π_{MEASURE} can benefit in terms of MPC overhead by employing distributed DP in case of Gaussian noise addition. We also notice that π_{COMP} is equivalent to computing joint histograms from all the data holders. This protocol can be replaced in our framework with other efficient protocols, if available, for the given scheme and datasets distributions (e.g. [310, 313]).

The implementations of MPC protocols for differentially private mechanisms can also be replaced to satisfy different project requirements. For example, the Gaussian sampling protocol π_{GSS}, which is based on the Box-Muller transform, can be replaced by other methods [314].

9.3 Evaluation of CaPS

We evaluated CaPS with three centralized algorithms: MWEM, MWEM+PGM, and AIM. For MWEM, we follow the implementation provided in SmartNoise [5], while for MWEM+ PGM and AIM, we use the implementations from the Private-PGM framework[6]. Since these algorithms originate from different software libraries[7], we integrated our DP-in-MPC protocols into each library independently. Consequently, we present the evaluations for MWEM and for MWEM+PGM/AIM in separate sections, using different datasets and metrics. By preserving these original setups, we ensure fidelity to their experimental protocols while demonstrating the adaptability of CaPS across different software.

Note: MPC protocols runs in two phases. The offline phase is data-independent; it runs beforehand to generate "correlated randomness" like multiplication triples, which are essential cryptographic building blocks. This computationally intensive setup avoids processing actual inputs. The online phase then efficiently uses this pre-computed material to perform the computation on the parties' private inputs, making it much faster since the heavy cryptographic lifting is already done.

9.3.1 Evaluation with MWEM

Datasets. We evaluate CaPS using two publicly available datasets, namely the Car and Adult datasets, which have been featured in previous DP synthetic dataset generation analyses [296, 5].

---

[6] https://github.com/ryan112358/private-pgm/

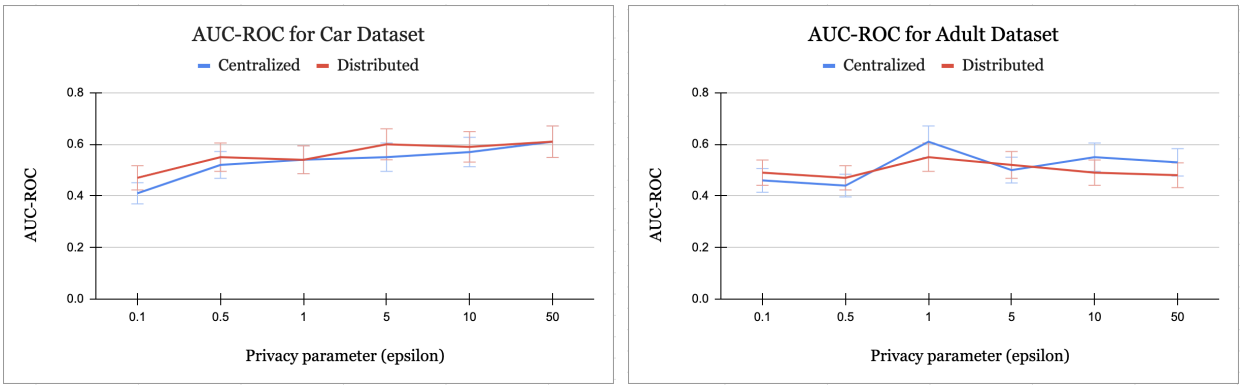
[7] Each software has a different generate step and different data structures that need to be integrated into CaPS.

Metrics. We empirically validate the utility of the produced synthetic data and measure performance by training LR models using synthetic data and testing the models on real data, as in [5]. We evaluate model performance using AUC-ROC. We use a maximum number of iterations of 1000 (T in Algorithm 12 in Chapter 8) and other default parameters of LR available in Scikit-learn [315] to train the models. We compare the performance of models trained on synthetic data generated in the centralized mode, and synthetic data generated in the distributed mode using MPC where the data is split horizontally across data holders. We repeat the comparison process for different privacy parameter values.

Evaluation Setup. In all the results below, **centralized** refers to the setting in which all data holders disclose their data to a central, trusted curator who runs the MWEM algorithm over all the data combined, while **distributed** refers to the setting in which the data holders secret-share their data with MPC servers. The results for the centralized setting are obtained with an implementation of MWEM in SmartNoise [5]. For the distributed setting, we implemented our MPC protocols π_{RC} and π_{LAP} in the MPC framework MP-SPDZ [110].

Utility Analysis. In Figure 9.2, we investigate the trade-off between the privacy parameter ϵ and utility of models trained with synthetic data generated by the two different modes (centralized and distributed). The models perform similarly in terms of AUC-ROC, for the different values of ϵ. Additionally, the trends are also consistent for both datasets. In the experiments using the Car dataset we see a upward trend for both modes, whereas for the adult dataset we see a small spike for $\epsilon = 1$ for both modes. Based on similar trendlines for both settings, we conclude that DP-in-MPC emulates the centralized mode of operation. The small differences observed in the plots are a result of the noise introduced by the DP mechanisms. The results in Figure 9.2 are averaged over 10 runs.

Performance Analysis. We measure runtime for different values of the number of iterations T, which is a hyperparameter of MWEM. Previous works have demonstrated the



(a) (b)

Figure 9.2: AUC-ROC of LR models trained on synthetic data generated by two different modes (centralized and distributed) with varying privacy budget. The results presented are averaged over 10 runs.

trade-off between the number of iterations and quality of the synthetic data [296]. Table 9.3 shows the runtime for different choices of T averaged over 3 runs for the centralized setting and for the distributed setting with 2, 3, and 4 computing servers. All MPC based computations were done in ring $\mathbb{Z}_q$ with $q = 2^{64}$. As observed, the runtimes increase with T. We note that the runtimes further depend on the dimensions of the datasets and the number of queries, as shown in [296]. The increased runtimes for the distributed setting when compared to their corresponding centralized setting are due to the runtimes of the MPC protocols. For example, in a 3PC passive security setting for $|T| = 1$, each call to π_{RC} adds $\sim$0.74 secs for $|Q| = 400$ and π_{LAP} adds $\sim$0.006 secs to the synthetic generation process. The differences in runtime observed across different security settings are in line with existing literature [4]. All the experiments were run on Azure D8ads_v5 8 vCPUs, 32Gib RAM. The runtimes of our proposed method depend only on the computing servers and the threat model. Our proposed MPC protocols can be made scalable and efficient by replacing the appropriate and efficient underlying MPC schemes.

We measure runtimes of our method for different numbers of MWEM iterations T and compare with the centralized setting, while keeping other parameters constant[8].

We note that our approach works for any partitioning of data – horizontal, vertical or mixed – with appropriate MPC protocols for any required pre-processing steps. We demonstrate this in our next evaluations.

Table 9.3: Runtime for different values of T (MWEM iterations) and $\epsilon = 1$. Central: Centralized setting runs the MWEM algorithm [5]; Other columns: Distributed setting with 2 data holders and MPC protocols run on different number of computing servers with different security settings: 2PC [1], 3PC [3, 4], 4PC [4]; all with corruption threshold of 1. The runtimes inlcude online and offline phases of MPC. $|\mathcal{Q}|$ is the number of queries, (a x b) denotes the dataset size (dimension).

Dataset	T	Central	2PC passive	3PC passive	3PC active	4PC active
Car (1,728 x 7) $\lvert\mathcal{Q} = 400\rvert$	10	0.33 sec	12.14 sec	10.09 sec	20.52 sec	11.81 sec
	20	0.71 sec	23.50 sec	20.26 sec	43.01 sec	23.98 sec
	30	1.30 sec	37.86 sec	31.91 sec	66.40 sec	36.22 sec
	40	2.13 sec	51.20 sec	43.60 sec	87.53 sec	51.85 sec
Adult (12,499 x 12) $\lvert\mathcal{Q} = 500\rvert$	10	4.03 sec	62.95 sec	56.928 sec	109.56 sec	63.26 sec
	20	4.89 sec	152.70 sec	147.42 sec	229.05 sec	127.26 sec
	30	6.38 sec	244.80 sec	236.10 sec	428.87 sec	263.54 sec
	40	8.43 sec	272.35 sec	232.50 sec	456.50 sec	272.63 sec

[8]We use the same parameters (such as number of queries, etc.) as in the SmartNoise tutorial notebooks. Similarly, for the Adult dataset, we use only the categorical columns as per the notebook [5].

9.3.2 Evaluation with AIM and MWEM-PGM

Datasets. We evaluate `CaPS` on three privacy sensitive datasets: breast-cancer [316], prison recidivism (COMPAS)[9] [317], and diabetes [318]. We randomly split all the datasets into train and test in an 80% to 20% ratio. The breast-cancer dataset has 10 categorical attributes and 285 samples. The COMPAS data consists of categorical data. We utilize the same version as in [198], which consists of 7 categorical features and 7,214 samples. The diabetes dataset has 9 continuous attributes and 768 samples. We use the train sets to generate synthetic data and the test sets to evaluate the quality of the synthetic data.

Metrics. To assess utility, we train classifiers on the generated synthetic data and test the models on the real test data. For breast-cancer, the task is to predict if the cancer will recur. For COMPAS, the task is to predict whether a criminal defendant will re-offend. For the diabetes dataset, the task is to classify a patient as diabetic. We train logistic regression and random forest models on the generated synthetic datasets and report the AUC-ROC and F1 score. We also evaluate `CaPS` for statistical utility of the generated data as the workload error Δ [297]:

$$\Delta(D, \hat{D}) = \frac{1}{|Q|} \sum_{q \in Q} \|\mu_q(D) - \mu_q(\hat{D})\|$$

Evaluation Setup. We implemented the MPC protocols of `CaPS` in MP-SPDZ [110]. We evaluate `CaPS` for horizontal (`CaPS`(H)) and vertical (`CaPS`(V)) data distribution scenarios, and compare against the centralized paradigm (CDP) in which all data holders give their data to a trusted aggregator (i.e. no input privacy). For the horizontal setup, we distribute the samples randomly and evenly among the data holders. For the vertical setup, we distribute the attributes randomly and evenly among the data holders.

---

[9] `https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis`

Utility Analysis. Table 9.4 shows that CaPS can generate synthetic data whose utility is at par with the CDP in terms of ML utility and statistical utility for both horizontal and vertical partitions. For the COMPAS and diabetes datasets, CDP and CaPS give similar ML and statistical utility. We attribute the higher variability observed in the cancer data experiments (AUC and F1) to the small data size. The variation observed in results in Table 9.4 are due to to randomness resulting from DP.

Performance Analysis. To measure the average time to generate synthetic data in CDP and CaPS, we run experiments in a simulated environment on Azure D8ads_v5 8 vCPUs, 32Gib RAM. We leverage MP-SPDZ to create the simulated environment in a single machine. In Table 9.5, we observe that, as expected, CaPS takes longer than CDP due to MPC overhead. For MWEM+PGM, it takes longer to generate synthetic data in a vertical setup due to overhead of π_{COMP}. For AIM, the vertical setup takes comparatively less time. This is because AIM requires computation of one-way marginals before the selection step, which can be done by the data holders in a vertical distributed scenario.

In the horizontal setup, computation of one-way marginals has to be done in MPC, generating additional overhead in computation of marginals compared to the vertical setup. We also note that $|Q| \times \max(\omega_q)$ impacts the runtime for (CaPS(H)), whereas $|Q| \times \max(\omega_q)$ and n impact the runtime for (CaPS(V)). We believe these runtimes are acceptable since fast response time is not crucial for SDG. Moreover, the benefits of generating synthetic data while preserving both input and output privacy surpass this additional cost.

In Table 9.6, we evaluate individual MPC subprotocols for different threat models when they are run on independent instances of Azure Standard F16s v2 (16 vcpus, 32 GiB memory) and network bandwidth of 12.5Gbps. The results are in-line with the literature [170]. In table 9.6, we report online times, as the offline phase can be run in the setup phase of CaPS. We observe that 3PC passive provides the least MPC overhead. For t iterations of the loop in Algorithm 13, the additional time due to MPC in a 3PC setting is (2.882 + t·1.1913) sec

Table 9.4: **Utility evaluation**. Synthetic data was generated with $\epsilon = 1.0$. For the distributed scenario with `CaPS`, $N = 2$ and $M = 3$ (3PC passive [3], corruption threshold of 1). All values are averaged across 3 runs. Abbreviations: CDP = Central Differential Privacy with trusted aggregator (no input privacy); `CaPS` = Our proposed approach for arbitrary distribution; H = Horizontal distribution; V = Vertical distribution; LR = Logistic Regression; RF = Random Forest; Cat. = Categorical; Cont. = Continuous

Dataset	Algorithm	Setup	LR-AUC	RF-AUC	LR-F1	RF-F1	Δ
Breast-Cancer(Cat.)	AIM	CDP	0.49	0.58	0.47	0.54	0.30
		`CaPS`(H)	0.49	0.45	0.43	0.44	0.23
		`CaPS`(V)	0.51	0.53	0.48	0.51	0.23
	MWEM+PGM	CDP	0.50	0.51	0.48	0.44	0.24
		`CaPS`(H)	0.55	0.54	0.49	0.46	0.21
		`CaPS`(V)	0.44	0.49	0.43	0.50	0.21
COMPAS(Cat.)	AIM	CDP	0.67	0.65	0.62	0.61	0.017
		`CaPS`(H)	0.66	0.65	0.62	0.61	0.019
		`CaPS`(V)	0.68	0.67	0.63	0.62	0.015
	MWEM+PGM	CDP	0.66	0.62	0.62	0.59	0.022
		`CaPS`(H)	0.66	0.61	0.60	0.58	0.022
		`CaPS`(V)	0.66	0.64	0.61	0.60	0.022
Diabetes (Cont.)	AIM	CDP	0.76	0.74	0.68	0.66	0.14
		`CaPS`(H)	0.77	0.72	0.65	0.463	0.13
		`CaPS`(V)	0.73	0.71	0.66	0.63	0.13
	MWEM+PGM	CDP	0.62	0.57	0.55	0.55	0.15
		`CaPS`(H)	0.64	0.60	0.52	0.56	0.14
		`CaPS`(V)	0.67	0.63	0.60	0.58	0.14

for AIM and $(2.882 + t \cdot 0.0463)$ sec for MWEM+PGM for the chosen parameters.[10] Given

[10]We note that the number of servers in MPC corresponds to M servers in our framework and not N data

Table 9.5: **Runtime evaluation**. Synthetic data was generated with $\epsilon = 1.0$. For the distributed scenario with CaPS, $N = 2$ and $M = 3$ (3PC passive [3], corruption threshold of 1). For CaPS we report runtimes for experiments done in a ***simulated environment***. All values are in seconds and averaged across 3 runs. Abbreviations: CDP = Central Differential Privacy with trusted aggregator (no input privacy); CaPS = Our proposed approach for arbitrary distribution; H = Horizontal distribution; V = Vertical distribution.

	$\lvert Q\rvert \times \max(\omega_q)$	CDP	CaPS(H)	CaPS(V)
	Breast-Cancer ($n = 228, d = 10$)			
AIM	57×77	98.065	99.789	98.992
MWEM+PGM	45×77	81.187	82.905	100.295
	COMPAS ($n = 4120, d = 9$)			
AIM	45×9	153.383	155.559	301.390
MWEM+PGM	36×9	61.477	61.656	152.153
	Diabetes ($n = 614, d = 9$)			
AIM	45×25	97.222	113.909	97.403
MWEM+PGM	36×25	60.926	63.727	91.760

that the one-time additional cost is comparatively much lower than the time it takes to generate synthetic data itself in-the-clear, CaPS achieves near-practical performance. With further optimizations of MPC primitives in the future, CaPS has the capability to be deployed in real-world scenarios. Our results demonstrate a clear path for future research in this direction and adapting various synthetic data generation techniques smartly in the "DP-in-MPC" paradigm. The major MPC overhead due to the arbitrary distributed setting in CaPS is attributed to the computations in π_{COMP}. The performance of this protocol is impacted

holders.

Table 9.6: **Performance evaluation of MPC protocols for different threat models**. MPC protocols are run with $N = 2, n = 614, d = 9, |Q| = 45, \max(\omega_q) = 25$. We run experiments with $M = 2, 3, 4$ and corruption threshold of 1 in a LAN setup and the mentioned threat models. We report runtimes in seconds and total communication cost (Comm.) in MB for the online phase of the MPC protocols. π_{SELECT}(A) refers to the MPC protocol for 'select' for the AIM algorithm and π_{SELECT}(M) for the MWEM+PGM algorithm. π_{COMP} refers to the computation of 1- and 2-way marginals in a vertical distribution.

Protocol	2PC passive		3PC passive	
	time(s)	Comm.(MB)	time(s)	Comm.(MB)
π_{MEASURE}	0.020	2.214	0.0043	0.261
π_{SELECT}(A)	2.972	119.053	1.187	0.554
π_{SELECT}(M)	0.783	167.658	0.042	5.341
π_{COMP}	135.482	30137.40	2.882	696.644
Protocol	3PC active		4PC active	
	time(s)	Comm.(MB)	time(s)	Comm.(MB)
π_{MEASURE}	0.030	2.140	0.0157	0.714
π_{SELECT}(A)	1.773	11.971	2.169	13.707
π_{SELECT}(M)	0.306	33.946	0.135	4.700
π_{COMP}	21.612	3830.53	6.00	1615.61

by the total number of samples n in D. We empirically evaluate the scalability for n and N.

Scalability with total training samples n. We run experiments in a 3PC passive setting and evaluate the scalability of π_{COMP} in Figure 9.3. This is the only protocol that depends on the value of total number of samples over all the distributed datasets. We note that we run experiments for π_{COMP} which computes 1-way and 2-way marginals in a distributed setting. The overhead due to the number of data holders (requires only additions) in this case is negligible when compared to computation of 2-way marginal for large n. This means

that our findings are independent of the number of data holders N.

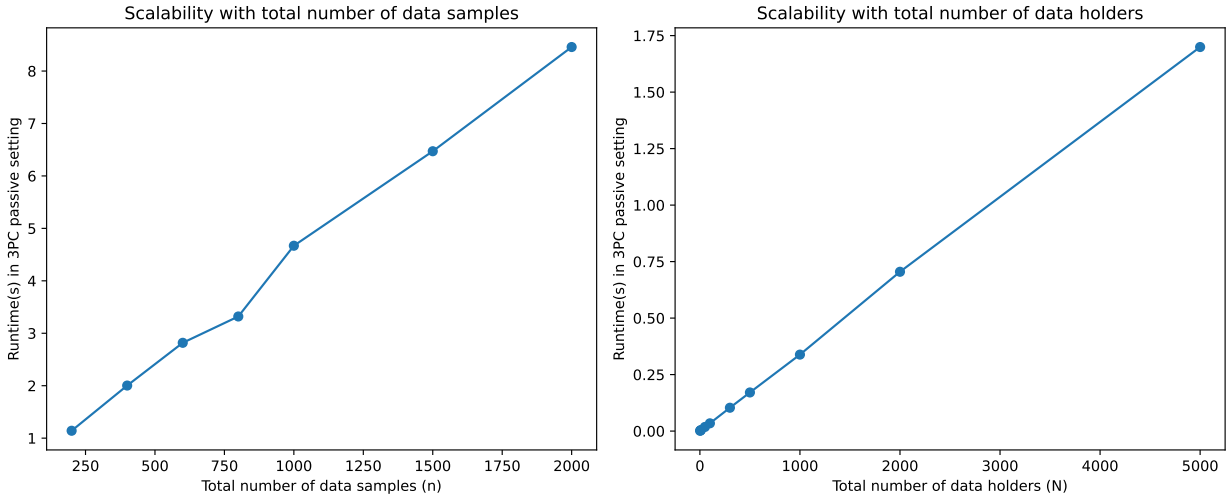


Figure 9.3: **Scalability of π_{COMP} in a 3PC passive setting**. MPC protocols are run with $M = 3$ (corruption threshold of 1) , $d = 10, |Q| = 36, max(\omega_q) = 25$. On left: Scalability of π_{COMP} for different number of total dataset size n. On right: Scalability of π_{COMP} for different number of data holders N.

Scalability with data holders N. To illustrate the affect of increase in number of data holders, we consider a horizontal distributed scenario where each data holder holds equal number of samples. The overhead in this case is due to only addition of workload answers from N data holders for a given $|Q|$ and ω.

We note that the scalability of other protocols π_{SELECT} and π_{MEASURE} depend on $|Q|$ and ω. We think the impact due to $|Q|$ and ω is similar to the impact on centralized algorithms. Based on literature, runtimes for MPC protocols grow with large M, depending on the MPC scheme available for M. `CaPS` adapts MPC-as-a-service scenario, where we can choose the optimal number of MPC server, M. We think based on the current literature in MPC $M = 3$ is a better option for `CaPS`.

9.4 Summary

We introduced a general framework `CaPS` that leverages DP-in-MPC for collaborative and private generation of synthetic tabular data based on real tabular data from multiple data holders. `CaPS` is designed for the state-of-the-art family of DP synthetic data generators that follow the select-measure-generate template and rely on marginal based distributions. We developed MPC protocols for computing marginals (which is relevant in non-horizontal data partitioning scenarios), as well as for the select and measure steps. Since the generate step operates on already privatized data, it can be performed outside of MPC. This chapter is the first in the thesis where we used or developed MPC protocols for all the DP mechanisms introduced in Section 2.3 in Chapter 2.

We demonstrated the applicability of `CaPS` for the marginal based synthetic data generation algorithms AIM, MWEM and MWEM+PGM. We consider generating synthetic data using DP-in-MPC for other data modalities and their state-of-the art generation algorithms as a future research direction.

Chapter 10

E2E-CAPS: END TO END COLLABORATIVE AND PRIVATE SYNTHETIC DATA GENERATION

In the preceding chapters, we focused on building privacy-preserving machine learning (ML) models trained on data distributed across multiple data holders, which included discriminative models (Chapters 3, 6, and 7) and a synthetic data generator (Chapter 9). However, model training is only one component of the ML pipeline, which typically includes stages such as data pre-processing, parameter tuning, and post-processing. These stages often require direct access to private data, introducing additional privacy risks. Therefore, each step in the ML pipeline must be designed to preserve privacy, as vulnerabilities at any stage can compromise the overall privacy guarantees of the system. In Chapter 6, we addressed this challenge by applying DP-in-MPC to the pre- and post-processing stages for bias mitigation, while in Chapter 4 we focused on privacy-preserving evaluation for bias auditing. Both chapters address privacy in different stages of ML pipeline but in the specific context of AI biases rather than general model training.

In this final chapter, we extend that idea further by building end-to-end privacy-preserving pipelines to train Differentially Private (DP) synthetic tabular data generators (SDGs) using real tabular data distributed across multiple data holders. Our choice to focus on SDGs, rather than other tasks, is motivated by the advantages discussed in Chapter 8, most notably, that once privacy-preserving synthetic data is generated and published, it can be reused across multiple tasks and easily integrated into existing workflows.

To this end, we extend `CaPS` introduced in Chapter 9 into a more comprehensive system called `E2E-CaPS`. As noted earlier in Chapter 8, designing effective MPC protocols requires tailoring them to each algorithm and task. We, therefore, build upon and optimize the MPC

protocols introduced in Chapter 9, applying them to the sensitive domain of healthcare by generating synthetic genomic data for leukemia patients using data held by multiple hospitals.

In Section 10.1, we briefly outline the privacy challenges in end-to-end SDG pipelines and review relevant existing work. Section 10.2 presents our proposed solution: a general framework composed of modular and generic MPC protocols. In Section 10.3, we apply this framework to a specific use case for generating genomic leukemia data, where we develop the necessary MPC protocols and evaluate their performance. We end the chapter by summarizing it in Section 10.4.

10.1 Introduction

A common limitation across existing solutions for collaboratively generating privacy-preserving synthetic data from multiple data holders, including `CaPS`, is that they focus exclusively on synthesizer training. They assume that the training data has already been pre-processed and that the optimal training configuration is known. As a result, these approaches are limited to one-shot synthesizer training and data publication. Generating high-quality synthetic data, however, often involves a multi-stage process, including data preparation, evaluation of the synthetic data against real data, and hyperparameter tuning for the SDG algorithm as we discuss below. In this chapter, we consider a scenario where a fixed privacy budget is allotted for generating DP synthetic data of a desired quality from multiple data holders.

- *Pre-processing.* Many SDG algorithms require specific pre-processing steps to generate synthetic data. For instance, SDGs that follow the select-measure-generate template often rely on categorical data, hence requiring discretization of continuous features. Existing solutions such as [156] discretize the entire dataset before distributing it across silos. This is unrealistic in practical scenarios where the data originates from different sources and cannot be brought together – the very scenario that federated SDG is supposed to address. A few methods such as in `CaPS` uses techniques like equi-width binning, which assumes knowledge of the range (i.e. potential minimum and maximum values known in advance) of a continuous feature and divides it into a fixed number of equal-length intervals. Depending

on the data distribution, this can lead to inferior results compared to equi-depth binning, such as quantile binning, in which interval boundaries are chosen dynamically based on the data such that each interval (bin) contains approximately the same number of instances.

Pre-processing techniques, such as normalization or missing data imputation, yield better results for downstream tasks like training synthetic data generators. These techniques tend to yield better results when applied to the *combined* data from multiple data holders [319], rather than having each holder pre-process their dataset locally, as is typically done in federated learning (FL). We need a mechanism that enables pre-processing over the combined data while still preserving input privacy.

- *Evaluation.* Evaluating synthetic datasets against real data is a crucial step in assessing the quality of the generated synthetic data. The evaluation results, i.e. evaluation metrics, help in refining the SDG process by guiding the hyperparameter tuning or retraining of the SDG model (i.e. the synthesizer). Furthermore, evaluation metrics are often used to decide whether the generated synthetic data meets the quality standards necessary for publishing the data. In many cases, evaluation metrics are published alongside the synthetic datasets. A few FL frameworks even support the local evaluation of synthetic data against real data during each FL round and allow publishing of the resulting metrics to the aggregator [320].

  However, revealing evaluation metrics based on real data has been shown to be vulnerable to privacy attacks [321]. A straightforward solution to this is to ensure that the evaluation metrics are DP. But, as discussed in Section 2.3 of Chapter 2 (see Sequential Composition), each access to real data consumes additional privacy budget. In this case, computing DP evaluation metrics would consume part of the overall fixed budget that could otherwise be allocated to improving the SDG itself.

  To address this, we need a mechanism that allows computation of evaluation metrics on real data to guide the SDG process without ever revealing the metrics themselves. This approach is, of course, applicable in use cases where it is sufficient to publish synthetic

data that meets the desired quality standards, and where publishing the evaluation metrics is not necessary.

- *Hyperparameter tuning.* Hyperparameter tuning for SDG involves systematically searching for optimal model configurations to maximize the utility of generated synthetic data. This typically requires multiple iterations of synthetic data generation, evaluation and potentially pre-processing (if the dataset changes with each iteration). Existing solutions for generating synthetic data based on data from multiple data holders do not focus on this hyperparameter tuning of the SDG model. Their primary focus is on publishing either the trained SDG model or the synthetic data in a single shot, without iterations, implicitly assuming that optimal model configurations for a dataset are known beforehand.

  In practice, this is rarely the case. Hyperparameter tuning often needs to be done specific to the dataset and typically involves repeated access to real data throughout the process. Each such access (whether for pre-processing, SDG training, or evaluation) consumes part of the privacy budget due to the DP property of sequential composition[1]. Naively repeating the pipeline can lead to rapid budget exhaustion before the synthesizer is properly tuned, ultimately degrading the quality of the final synthetic data.

  To address this, we need a mechanism that supports hyperparameter tuning while minimizing unnecessary privacy budget expenditure. Only the final, high-quality synthetic dataset should be published, while all intermediate steps remain private, thereby preserving the privacy budget and enabling more effective model development.

The above considerations call for a framework capable of *performing privacy-preserving pre-processing across data silos, and publishing the synthetic data only after parameter tuning and ensuring that desired quality standards are met.* To the best of our knowledge, there is

[1]This is typically how hyperparameter tuning is done in the centralized setting, where all data resides in one place. The intermediate steps though they access real data are not published and need not be made DP and only the final output DP synthetic data is published. Whereas we operate in a distributed setting, where such an approach is not feasible; every step needs to be made private.

no work in the open literature that performs *the entire pipeline of SDG on data from multiple data holders while preserving input privacy and efficient use of allotted privacy budget.*

10.1.1 Related Work

The primary focus of existing research on privacy-preserving synthetic data generation with input and (or) output privacy guarantees has been on one shot generation of synthetic data [322, 304, 323, 324]. There is some literature on efficient parameter tuning with DP guarantees in the centralized setting [325, 326, 327]. In cross-silo federated settings, recent work by Mitic et al. introduces PrivTuna, a framework for privacy-preserving hyperparameter tuning [328]. PrivTuna leverages multi-party homomorphic encryption to share the locally tuned parameters and performance metrics and focuses only on parameter tuning phase instead of end-to-end privacy-preserving pipeline with DP guarantees. There is no literature, to the best of our knowledge, that performs privacy-preserving hyperparameter tuning and runs the entire SDG pipeline, including pre-processing and evaluation, while providing both input and output privacy.

10.2 Description of `E2E-CaPS`

We assume that a fixed privacy budget, say ϵ, is allotted for generating synthetic data of the desired quality. To effectively use this budget, we adopt an iterative approach where the full SDG pipeline comprising of pre-processing, synthesizer training, and evaluation is executed multiple times, each with a different set of hyperparameters.

To streamline this process, we use K-fold evaluation within each iteration to assess the performance of the synthetic data against the real data. If the average evaluation metrics across folds do not meet predefined thresholds of quality, we run the next iteration with a different set of hyperparameters and repeat the process. Since no intermediate outputs (e.g. models, data, or evaluation metrics) are published during these iterations, the privacy budget can be "reset" with each iteration. This means that it allows us to reuse the same budget ϵ across iterations without accumulating privacy loss.

Once the best set of hyperparameters is identified through K-fold validation, we execute the final run of the SDG pipeline using the same privacy budget ϵ as if only a single iteration had occurred, thus enabling efficient hyperparameter tuning without additional privacy cost beyond what would be required for a single run. While this approach is standard in centralized settings, adapting it to the cross-silo context poses significant challenges.

We propose using DP-in-MPC to run the above process in a cross-silo setting (see Algorithm 23). To this end, we propose modular framework `E2E-CaPS` for tabular data that we describe below.

10.2.1 Framework Setup

Consider n data holders $\mathbf{C} = \{C_1, C_2, \ldots, C_n\}$, who each hold a private dataset D_i. For MPC-as-a-Service scenario, we consider m non-colluding and independent MPC servers $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$. These servers are equipped with MPC protocols to generate synthetic data $\hat{D}$ of the desired quality by running the full SDG pipeline, including hyperparameter tuning described above, over the combined data $D = \bigcup_{i=1}^{n} D_i$. The pipeline includes MPC protocols for privacy-preserving pre-processing ($\pi_{\mathsf{pre-proceSS}}$), training of the SDG model (π_{SDG}), and evaluation of the generated synthetic data against real data (π_{EVAL}).

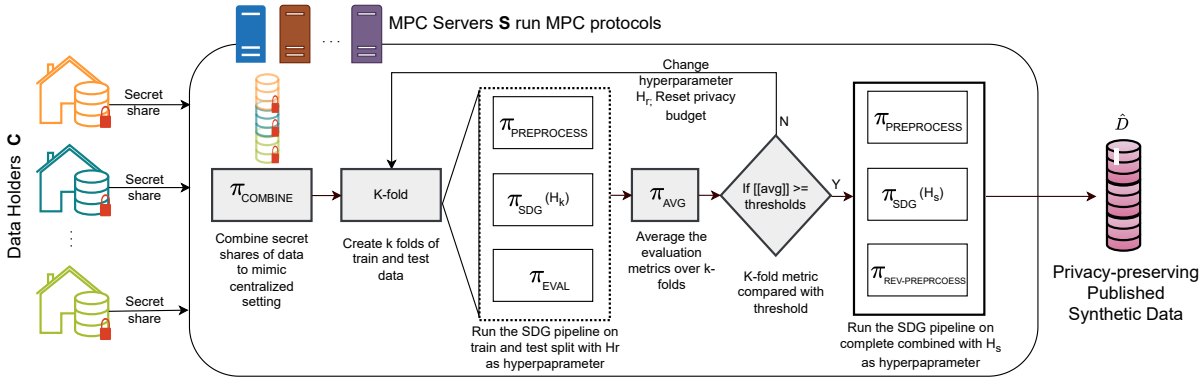


Figure 10.1: `E2E-CaPS`. Extending `CaPS` for privacy-preserving publishing of synthetic data with hyperparameter tuning

10.2.2 Overview of the Framework.

Algorithm 23 provides an overview of our proposed framework, which follows the iterative process described above for hyperparameter tuning of the SDG. The framework is modular and composed of MPC sub-protocols that we describe below.

The data holders **C** secret share their private data $[\![D_i]\!]$ and the desired quality of synthetic data as thresholds $[\![T_i]\!]$ to MPC servers **S** on Line 1. On Line 2, the servers set up according to the underlying MPC scheme, initialize using secret-shared inputs from the data holders, along with public inputs such as the data dimensions and the set of hyperparameters to be tuned.

On Line 3, **S** runs π_{CONCAT} to concatenate the secret shares of data received from all **C**. As a result the **S** now holds the secret shares of the dataset $[\![D]\!]$. This mimics a centralized setting, as if all the data resided in a single location.

Parameter tuning is performed iteratively from Lines 5 – 24 until one of the following conditions is met: the desired quality of synthetic data is achieved (publish $\leftarrow$ 1), the maximum number of iterations is reached (max_loops $\geq L$), or the hyperparameter set is exhausted[2]. Lines 6 – 9 initialize the iterative process, while Lines 10 – 17 execute the K-fold cross-validation. On Line 11, training and test data splits are obtained for each fold using getData(), based on the indices computed in Line 7[3].

The first step in each fold is to pre-process the training data, if any. On Line 12, the servers **S** execute the MPC protocol π_{RW} to perform this pre-processing on the secret-shared training data. Whether or not this step is done with DP guarantees depends on the nature of the pre-processing and whether its results will be indirectly exposed through the final synthetic data.

---

[2]This can be easily modified to exhaustively search for all parameters, and then choose the best hyperparameter for which we obtained the best evaluation metrics. This would require a large number of secure comparisons – comparing with thresholds and comparing with all the sets of averaged evaluation metrics.

[3]Here, $[\![D_{test}]\!]$ is required only if the evaluation metrics involve metrics such as utility scores of ML models on real test data.

Algorithm 23: `E2E-CaPS`: Publishing privacy-preserving synthetic tabular data

Input: Set of MPC servers $\mathbf{S}$, Data holders $\mathbf{C}$, number of folds K, fixed privacy budget (ϵ, δ) for published synthetic data $\hat{D}$, Maximum number of loops L, Set $\mathbf{H}$ of hyperparameter values to perform search over for fine-tuning

Output: Synthetic Data $\hat{D}$

Data holders, $\mathbf{C}$, secret share their respective datasets with MPC servers, $\mathbf{S}$, as $[\![D_i]\!]$ and thresholds of data quality $[\![T_i]\!]$

$\mathbf{S}$ setup based on MPC schemes

$[\![D]\!] \leftarrow \pi_{\mathsf{CONCAT}}([\![D_i]\!] \,|\, \forall i \in |\mathbf{S}|)$

publish $\leftarrow$ false; max_loops $\leftarrow 0$

repeat

  max_loops $\leftarrow$ max_loops + 1

  Get random indices for K-fold; Get hyperparameters $\mathbf{H_l}$ for this loop

  Initialize $\mathbf{R}$ of length K

  Set privacy budgets (ϵ_s, δ_s) for SDG and (ϵ_p, δ_p) for pre-processing // see description

  for

    $[\![D_{\text{train}}]\!], [\![D_{\text{test}}]\!] \leftarrow \text{getData}([\![D]\!])$

    $[\![D_{\text{train}}]\!] \leftarrow \pi_{\mathsf{pre-proceSS}}([\![D_{\text{train}}]\!], (\epsilon_p, \delta_p))$ or $[\![D_{\text{train}}]\!] \leftarrow \pi_{\mathsf{pre-proceSS}}([\![D_{\text{train}}]\!])$

    $[\![\hat{D}_{\text{train}}]\!] \leftarrow \pi_{\mathsf{SDG}}([\![D_{\text{train}}]\!], (\epsilon_s, \delta_s), \mathbf{H_l})$

    $[\![r]\!] \leftarrow \pi_{\mathsf{EVAL}}([\![D_{\text{train}}]\!], [\![D_{\text{test}}]\!], [\![\hat{D}_{\text{train}}]\!])$ // see description

    $[\![R[k]]\!] \leftarrow [\![r]\!]$

  end for

  $[\![M]\!] \leftarrow \pi_{\mathsf{AVG}}([\![R[k]]\!] | \forall k)$

  votes $= |\mathbf{C}|$

  for every data holder c get threshold $[\![T_c]\!]$

    votes $\leftarrow$ votes $-\pi_{\mathsf{LT}}([\![M]\!], [\![T_c]\!]))$

  end for

  publish $\leftarrow \pi_{\mathsf{EQ}}$ (votes, $|\mathbf{C}|$)

  $\mathbf{H_s} \leftarrow \mathbf{H_l}$

until publish or (max_loops $\geq L$) or no $\mathbf{H}$ left to explore

If publish

  $[\![D]\!] \leftarrow \pi_{\mathsf{pre-proceSS}}([\![D]\!], (\epsilon_p, \delta_p))$ or $[\![D]\!] \leftarrow \pi_{\mathsf{pre-proceSS}}([\![D]\!])$

  $[\![\hat{D}]\!] \leftarrow \pi_{\mathsf{SDG}}([\![D]\!], (\epsilon_s, \delta_s), \mathbf{H_s})$

  $[\![\hat{D}]\!] \leftarrow \pi_{\mathsf{REV-pre-proceSS}}([\![\hat{D}]\!])$

  Reveal $\hat{D}$ to $\mathbf{C}$

end if

For instance, if the pre-processing involves tasks that do not need to be reversed or exposed later, such as missing data imputation that only influence the training phase and are not visible in the final output, then π_{RW} does not need to be DP. This is typical in cases where the SDG learns to generate fully complete data in the same format as real data. However, some pre-processing steps should be made DP if their outputs will influence the published synthetic data. Consider the case of discretizing continuous values using binning, and then later applying reverse binning by replacing bin identifiers with the mean of values in each bin. If these means are computed from real data and appear in the final published synthetic dataset, they must be protected with DP since they represent additional access to real data. Whether DP is required for pre-processing is determined during the setup phase (Line 9), and appropriate portions of the overall privacy budget are allocated accordingly. If no DP pre-processing is needed, then the full privacy budget $\epsilon_s = \epsilon$ can be used for the synthetic data generation, and π_{RW} is called without a privacy parameter. Otherwise, if DP is required, π_{RW} is invoked with the assigned privacy budget ϵ_p on Line 9.

On Line 13, MPC protocol for π_{SDG} is invoked to train an SDG model on secret-shares of pre-processed training data $[\![D_{train}]\!]$ with the set of hyperparameter $\mathbf{H_l}$ chosen for this loop. Note that generated synthetic data $[\![\hat{D}_{train}]\!]$ is in same format as $[\![D_{train}]\!]$ i.e. it is in pre-processed format. Here the entire SDG process (training and generation) is carried out within MPC.

On Line 14, $\mathbf{S}$ run π_{EVAL} to compute secret shares of the predefined evaluation metrics $[\![r]\!]$ ($[\![r]\!]$ can be a vector of multiple metrics). Depending on the pre-processing and reverse processing techniques and the defined pipelines, π_{EVAL} takes as input (a) the secret shares of the pre-processed real training data $[\![D_{train}]\!]$ or the reverse processed $\pi_{\mathsf{REV-pre-proceSS}}([\![D_{train}]\!])$[4], (b) pre-processed test data $\pi_{\mathsf{pre-proceSS}}([\![D_{test}]\!])$ or test data $[\![D_{test}]\!]$, and (c) synthetic data $[\![\hat{D}_{train}]\!]$ or the reverse processed synthetic data $\pi_{\mathsf{REV-pre-proceSS}}([\![D_{train}]\!])$. The value of $[\![r]\!]$ is never published and hence need not be DP. Note that all of the above steps perform

[4]Reverse pre-processing is done here as it is more appropriate to run MPC protocols for evaluation on the same format of data that would be published.

computations on combined real data and hence are done in MPC. On Line 17, **S** compute secret-shares of K-fold evaluation $[\![M]\!]$ using π_{AVG}.

Lines 18 – 22 determine whether the evaluation metrics $[![M]!]$ meet the quality thresholds set by all data holders. The process begins on Line 18 by assuming that the thresholds are met for all data holders, i.e. setting votes = $|\mathbf{C}|$. Then, on Line 20, votes are decremented for each data holder whose threshold is not met. Finally, Line 22 checks if the number of remaining votes still equals $|\mathbf{C}|$ and if so, the synthetic data is marked for publication, and the hyperparameters of the loop are selected as the final set $\mathbf{H_s}$ on Line 23. The voting is done in a privacy-preserving way – (a) it does not require release of any evaluation metrics; and (b) the thresholds of each data holder and the number of votes are kept confidential.

Finally, on Line 26 – 29, if the synthetic data of desired quality could be generated, then the combined data $[\![D]\!]$ is first pre-processed, and a synthesizer is trained using the allocated privacy budgets. Depending on the type of pre-processing applied, reverse pre-processing may be required on Line 28 to ensure the output synthetic data matches the format of the original real data that was secret-shared. Line 29 then publishes the synthetic dataset, which by default has the same length as the combined input data, though this can be adjusted in the framework (e.g. on Line 27).

A key observation is that fixed privacy budget is spent only on Lines 26 – 29, specifically on Line 29, where the final synthetic dataset $\hat{D}$ is published. This is the only point in the framework where any output derived from the real data D is released, and is protected by DP on Lines 26 – 28. All prior computations, including pre-processing, training, and evaluation on Lines 12 – 14, though based on privacy parameters (ϵ_p, δ_p) and (ϵ_s, δ_s), do not consume any part privacy budget, as their results are never revealed or published, thereby minimizing budget usage to a single, final publication of synthetic data.

Note on modularity of `E2E-CaPS`. The framework is modular and can be adapted to fit any specific SDG pipeline. The order of operations can be modified, and individual steps can be skipped based on the use case. It is flexible enough to support different pre-

processing methods, SDG algorithms, and evaluation metrics. With suitable MPC protocols in place, the framework can also switch from K-fold validation to other hyperparameter tuning strategies. While our focus has been on tabular synthetic data, the framework can be extended to other data modalities as well. For example, in the case of images, $\pi_{\mathsf{pre-proceSS}}$ may not be DP (e.g. for cropping), and reverse pre-processing may not be necessary since many transformations do not need to be inverted.

10.3 `E2E-CaPS` *for Privacy-Preserving Publishing of Synthetic Genomic Data for Leukemia*

Sharing genomic data is essential for advancing biomedical research with AI; however, such data is highly privacy-sensitive and hard to access [329]. Concerns regarding privacy and confidentially of clinical data have recently resulted in stricter NIH controlled-access genomic data requirements [330], and can expected to become even more important with the collection of single-cell data. To address the data access bottleneck, researchers are exploring the potential of sharing synthetic datasets that protect patient privacy [331, 332, 333]. This existing work is based on the premise that a single data holder has sufficient data to train a synthetic data generator. In rare disease settings however, existing genomic data is often spread across multiple clinical sites. While each clinical site by itself may not have sufficient data to train a synthesizer, together they often do. Our proposed framework is designed for federated SDG in scenarios like these. Below, we demonstrate the use of our framework to publish synthetic genomic data for leukemia.

Description of Data. We consider bulk RNA-seq data compiled as a matrix, where each row represents the specimen of a patient and each column represents a gene expression level [332][5]. It includes samples from 5 disease categories, 4 of which are types of leukemia AML, ALL, CML, CLL, while the fifth is classified as "Other." The dataset has 1,181 samples and

[5]This dataset is openly available on https://github.com/MarieOestreich/PRO-GENE-GEN and is used purely to demonstrate our solution.

12K genes, which is reduced to 958 genes based on the L1000 list.

Generating Synthetic Genomic Data with `E2E-CaPS`. Generating synthetic data for genomic applications is particularly challenging. We use a select-measure-generate style SDG method for tabular data, namely Private-PGM [288][6]. While Private-PGM does not explicitly implement the 'select' step, we categorize it within the select-measure-generate family due to its structural similarities.

Private-PGM constructs undirected graphical models from DP noisy measurements of low-dimensional marginals, enabling synthetic data generation through sampling from the learned graphical model. It works on records with discrete attributes i.e. it takes categorical data as input and benefits from a small domain size for each feature. Simply put, for each q, Private-PGM firstly computes DP marginals $\mu_q(D) + \mathcal{N}(0, \sigma_q^2 I)$, where $\mathcal{N}(.)$ is Gaussian noise with scale σ_q^2 determined based on (ϵ_s, δ_s). This is the measurement step. Then it estimates the joint marginal distribution that best explains all the noisy measurements. In parallel, it estimates the parameters of the graphical model using graph inference and learning algorithms such as belief propagation on a junction tree. The graphical model is then used to sample marginals and reconstruct the synthetic data set. This is the generate step, which is also used in the SDGs AIM and MWEM-PGM discussed in Chapter 9. We refer to McKenna et al. [288] for more details.

In the context of our use case, a sample record is denoted by $g = \{g_1, g_2, \ldots, g_d, y\}$ where g_i is the level of gene expression of the ith gene and y is the label that denotes the type of leukemia. For the measure step, we compute all 1-way marginals as well as the 2-way marginals associated with the label attribute (i.e. a total of 959 measurements with $|q| = 1$ and 958 measurements with $|q| = 2$, resulting in 1,917 noisy marginals). The privacy budget is allocated uniformly across each measurement i.e. $\epsilon_q = \epsilon_s/1,917$ following

[6]Our initial experiments with other SDG methods of the same select-measure-generate template, such as AIM [297] used in Chapter 9, indicated poor performance. Our initial results with recent SDG techniques based on diffusion models with DP [292] were not promising.

sequential composition. For the generate step, a graphical model is constructed based on $\mathcal{Q}$ (see Figure 10.2), and learning is performed to obtain the maximum entropy distribution of the dataset using mirror descent-based optimization. To evaluate the quality of the synthetic data generated, we train a logistic regression model to infer the type of leukemia based on gene expression values, and compute the workload error for 1-way marginals.

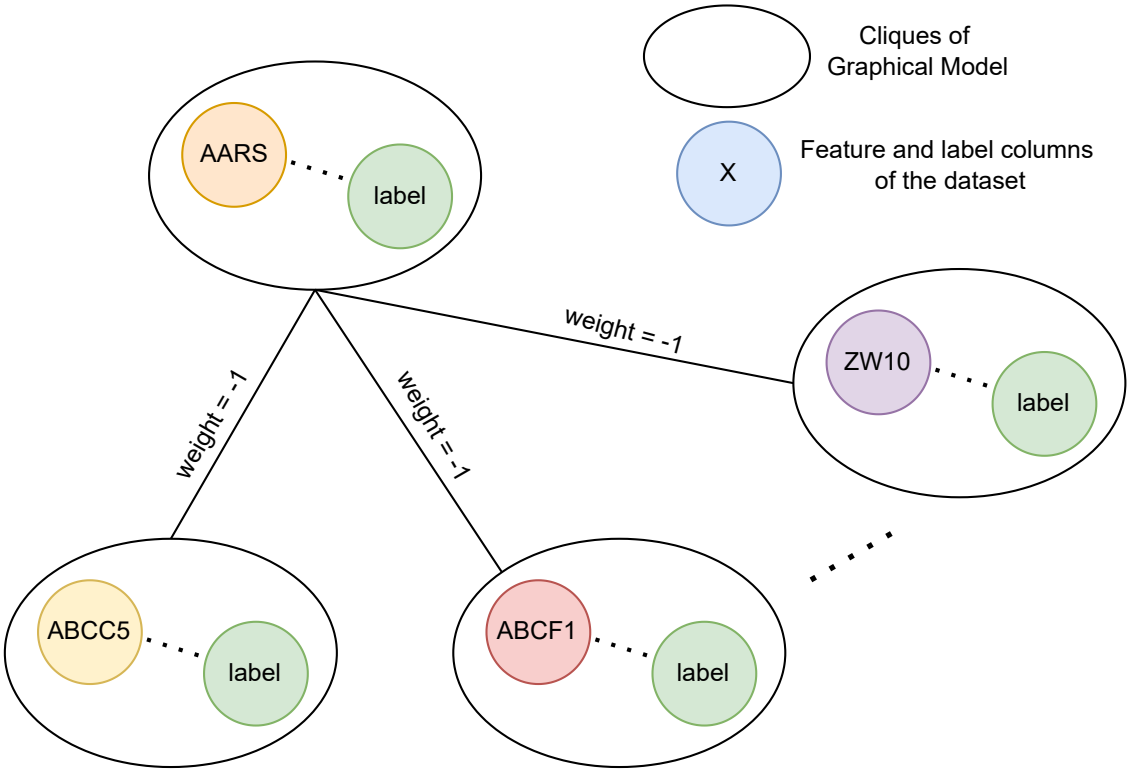


Figure 10.2: **Input graph structure.** We assume that metadata about the dataset such as feature columns – gene and label names, graphical model, domain, and dataset dimensions are known. This assumption does not leak any information about the data itself. The graph contains 958 nodes, each representing a clique derived from a single gene and the label column. A total of 1,917 marginals are measured with DP.

10.3.1 Secure Pre-processing

Chen et al. [332] generate synthetic data based on the leukemia dataset, assuming that it is available in its entirety with one data holder. They pre-process the data by binning each gene feature into intervals based on quantiles (0.25, 0.5, and 0.75). Each gene is then assigned one of the four values in $\{0, 1, 2, 3\}$, thereby reducing the domain size to $\omega_i = 4$, $\forall i \in [1, d]$, while the label domain size ω_y remains 5. Once the final synthetic data is generated, it is de-binned (reverse processed). De-binning replaces each bin with the computed mean of the original values within that bin.

We extend this approach to the cross-silo setting by proposing an MPC protocol π_{BIN}, which is a concrete instantion of π_{RW} from Algorithm 23. We improve upon the de-binning process proposed in [332] by computing the DP means of each bin. This is necessary to ensure that the published synthetic data is indeed DP.

Description of π_{BIN}. π_{BIN} bins each column i.e. each gene. It takes as input secret-shares of a single gene feature g as $[\![D^g]\!]$ and outputs the binned gene column along with the corresponding mean values $[\![m^g]\!]$ to be used during inverse pre-processing. Lines 4 – 8 in Protocol 24 compute the secret-shares of the quantiles (the cut points) in a straightforward manner by executing an MPC protocol π_{SORT} to sort the secret shares of the N samples of one gene (see Bogdanov et al. [334] for an overview of oblivious sorting algorithms). A secret-shared vector of cut-off points $[\![Q]\!]$ is computed as following on Lines 5 – 8. For each r in $[0.25, 0.50, 0.75]$: $pos = (N-1)\cdot \mathrm{r}$ and $[\![Q[\mathrm{r}]]\!] = [\![\widetilde{D^g}[\lfloor \mathrm{pos} \rfloor]]\!] + f\cdot([\![\widetilde{D^g}[\lfloor \mathrm{pos} \rfloor + 1]]\!] - [\![\widetilde{D^g}[\lfloor \mathrm{pos} \rfloor]]\!])$.

Lines 9 – 12, then, bin data into 4 bins. On Line 10, the bin is determined by comparing the secret-shared value of the gene expression for the ith sample $[\![D^g[i]]\!]$ with the secret-shared cut-off point $[\![Q[\mathrm{r}]]\!]$ for each quantile r, using an MPC protocol π_{LT}. On Line 11, we compute the bin index by leveraging the small finite size of bins to minimize computationally heavy operations $g_i = 3 - (g_i < Q[0]) - (g_i < Q[1]) - (g_i < Q[2])$. At this point, $\mathbf{S}$ hold the secret-shares of the discretized data.

Protocol 24: π_{BIN}: Protocol to bin the genomic data (π_{RW})

Input : Secret shares of one gene feature column of $[\![D^g]\!]$ for gene g, the number of samples N, privacy parameter ϵ_p

Output: Secret shares of one gene feature binned column $[\![D^g]\!]$ for gene g, secret shares of mean $[\![m^g]\!]$ for gene g for inverse discretization done later

1 Compute $\epsilon_{bin} = \frac{\epsilon_p}{d \cdot B}$

2 Set quantiles = [0.25,0.5,0.75] and initialize Q of size 3

3 Create a copy of $[\![D^g]\!]$ as $[\![D^g_{copy}]\!]$; Initialize $[\![Q]\!]$ of length 3

4 $[\![\widetilde{D^g}]\!] \leftarrow \pi_{\mathsf{SORT}}([\![D^g]\!])$ `// Protocol for sorting based on radix sort; see MP-SPDZ [110]`

5 **for** *each quantile r in quantiles* **do**

6 pos $\leftarrow (N-1) \cdot$ r ; $f \leftarrow$ pos $-$ $\lfloor$pos$\rfloor$

7 $[\![Q[\text{r}]]\!] \leftarrow [\![\widetilde{D^g}[\lfloor \text{pos} \rfloor]]\!] + f \cdot ([\![\widetilde{D^g}[\lfloor \text{pos} \rfloor + 1]]\!] - [\![\widetilde{D^g}[\lfloor \text{pos} \rfloor]]\!])$

8 **end**

9 **for** $i \leftarrow 1$ *to* N **do**

10 $[\![c_0]\!] \leftarrow \pi_{\mathsf{LT}}([\![D^g[i]]\!], [\![Q[0]]\!])$; $[\![c_1]\!] \leftarrow \pi_{\mathsf{LT}}([\![D^g[i]]\!], [\![Q[1]]\!])$; $[\![c_2]\!] \leftarrow \pi_{\mathsf{LT}}([\![D^g[i]]\!], [\![Q[2]]\!])$

11 $[\![D^g[i]]\!] \leftarrow 3 - [\![c_0]\!] - [\![c_1]\!] - [\![c_2]\!]$

12 **end**

13 Initialize $[\![m^g]\!]$ and $[\![\text{ctr}^g]\!]$ each of length 4

14 **for** $i \leftarrow 1$ *to* N **do**

15 **for** $b \leftarrow 0$ *to* 3 **do**

16 $[\![c]\!] \leftarrow \pi_{\mathsf{EQ}}([\![D^g[i]]\!], b)$

17 $[\![m^g[b]]\!] \leftarrow [\![m^g[b]]\!] + \pi_{\mathsf{MUL}}([\![c]\!], [\![D^g_{copy}[i]]\!])$

18 $[\![m^g[b]]\!] \leftarrow [\![m^g[b]]\!] + \pi_{\mathsf{LAP}}(2 \cdot \pi_{\mathsf{MAX}}([\![D^g]\!])/\epsilon_{bin})$ `// `π_{LAP}` to compute DP mean`

19 $[\![\text{ctr}^g[b]]\!] \leftarrow [\![\text{ctr}^g[b]]\!] + [\![c]\!]$

20 $[\![\text{ctr}^g[b]]\!] \leftarrow [\![\text{ctr}^g[b]]\!] + \pi_{\mathsf{LAP}}(2/B \cdot \epsilon_{bin})$

21 **end**

22 **end**

23 **for** $b \leftarrow 0$ *to* 3 **do**

24 $[\![m^g[b]]\!] \leftarrow \pi_{\mathsf{DIV}}([\![m^g[b]]\!], [\![\text{ctr}^g[b]]\!])$

25 **end**

26 **return** $[\![D^g]\!], [\![m^g]\!]$

Lines 13–25 compute the DP mean values for each bin (required for de-binning process) in a straightforward manner, using MPC protocols for equality testing π_{EQ} and multiplication

π_{MUL} of secret-shared values [7]. We use π_{LAP} to compute secret-shares of noise to be added to the sum and count of the mean. Note that $[\![\text{ctr}^g[b]]\!]$ is used as a secret-shared accumulator variable for the number of values that fall into bin b.

Note on DP for mean of bins. The privacy budget ϵ_p is divided equally among all d genes. Since gene values from the same patient may be correlated, we apply sequential composition and allocate ϵ_p/d per gene. Within each gene's data, values are discretized into $B = 4$ quantile bins, with each bin requiring a separate DP mean calculation. This further divides the privacy budget to $\epsilon_{bin} = \frac{\epsilon_p}{d \cdot B}$ per bin.

The sensitivity represents the maximum possible change in a bin's mean when a single data point is added or removed from the dataset. It depends on the value range and the number of contributing records. With quantile binning, adding or removing a single record causes cascading effects: (1) bin boundaries shift, (2) records may be reassigned between bins, and (3) multiple bins' counts and sums change. This affects the calculation of both bin sums $[\![m^g[b]]\!]$ on Line 17 and counts $[\![\text{ctr}^g[b]]\!]$ on Line 19.

The bin sums are, in the worst case bounded by the maximum value in the dataset, accounting for both direct value changes and boundary shifts. The sensitivity of bin counts reflects the $1/B$ fractional change in bin size plus boundary adjustment effects. We compute the scale of the Laplace mechanism on Lines 18 and 20 based on this sensitivity. We equally divide each bin's privacy budget between bin's sum and bin's count [8]. Further research could establish tighter sensitivity bounds based on limited bin edge movement as the the maximum value typically produces overly conservative estimates. This is because the bin's sum is bounded by bin boundaries which are less likely to change to the maximum value.

Description of $\pi_{\mathsf{INV-BIN}}$. $\pi_{\mathsf{INV-BIN}}$ takes as input secret shares of binned $[\![D]\!]$ and secret

[7] In the production deployments the multiple π_{MUL}s can be invoked in parallel to further improve efficiency.

[8] When $B = 1$, this analysis simplifies to standard mean calculation where numerator sensitivity is max value M that can be added to the dataset, denominator sensitivity is 1, and overall sensitivity approximates M/N for large datasets, validating our approach while highlighting binning's additional complexities.

shares of all DP mean values $[\![m^g]\!]$ and outputs the de-binned data. We replace the value g_i with the mean following $\sum_{i=0}^{3} m^g[b] \cdot (g_i == b)$ on Lines 4–5. We note that Line 4 can be further optimized by using the indicator polynomials described in Section 10.3.2.

Protocol 25: $\pi_{\mathsf{INV-BIN}}$: Protocol to inverse discretize genomic data ($\pi_{\mathsf{REV-pre-proceSS}}$)

Input : Secret shares of binned data of $[\![D]\!]$, the number of samples N, secret shares of mean $[\![m^g]\!]$ for all genes

Output: Secret shares of de-binned data $[\![D]\!]$

1 **for** $i \leftarrow 1$ *to* N **do**

2 **for** *every gene* g **do**

3 $[\![x]\!] \leftarrow [\![D[i][g]]\!]$

4 $[\![c_0]\!] \leftarrow \pi_{\mathsf{EQ}}([\![x]\!], 0)$; $[\![c_1]\!] \leftarrow \pi_{\mathsf{EQ}}([\![x]\!], 1)$; $[\![c_2]\!] \leftarrow \pi_{\mathsf{EQ}}([\![x]\!], 2)$; $[\![c_3]\!] \leftarrow \pi_{\mathsf{EQ}}([\![x]\!], 3)$

5 $[\![D[i][g]]\!] \leftarrow$ $\pi_{\mathsf{MUL}}([\![c_0]\!], [\![m^g[0]]\!]) + \pi_{\mathsf{MUL}}([\![c_1]\!], [\![m^g[1]]\!]) + \pi_{\mathsf{MUL}}([\![c_2]\!], [\![m^g[2]]\!]) + \pi_{\mathsf{MUL}}([\![c_3]\!], [\![m^g[3]]\!])$

6 **end**

7 **end**

8 **return** $[\![D]\!]$

10.3.2 Secure SDG

To enable efficient MPC protocols for the use-case, we assume that data-independent metadata is publicly available, including the graph structure (Figure 10.2), cliques, feature indices, dataset attributes (feature and label names, and their domains). This information can be inferred from domain knowledge and remains independent of patients' private data.

Secure Measure

While we proposed the protocols for the measurement step in marginals-based SDG algorithms in Chapter 9, here we propose an efficient protocol that leverages the predefined small domain size. We denote 1-way marginals for gene g_i as μ_{g_i} and the 1-way marginal for label y as μ_y. Given $\Omega_{g_i} = \{0, 1, 2, 3\}$ (after discretization) and $\Omega_y = \{0, 1, 2, 3, 4\}$, to compute a marginal, we compute the number of occurrences for all the values of Ω_q in D where $q \in \{g_1, \ldots, g_d, y\}$. To do so, for example, we need to compare each value in the data

column D^{g_i} of gene g_i to each value in the discretized domain Ω_{g_i}. To avoid equality test operations,[9] we define indicator polynomials, denoted by I as follows:

$$I_0(x) = \tfrac{(1-x)(2-x)(3-x)}{6} \qquad I_1(x) = \tfrac{x(2-x)(3-x)}{2}$$

$$I_2(x) = \tfrac{x(x-1)(3-x)}{2} \qquad I_3(x) = \tfrac{x(x-1)(x-2)}{6}$$

x	0	1	2	3
$I_0(x)$	1	0	0	0
$I_1(x)$	0	1	0	0
$I_2(x)$	0	0	1	0
$I_3(x)$	0	0	0	1

Using the above equations, we compute $\mu^b_{g_i} = \sum_{j=1}^{N} I_b(D[j][g_i])$, for $b \in \{0,1,2,3\}$.

The technique with indicator polynomials also works for the label feature with discretized domain $\Omega_y = \{0,1,2,3,4\}$:

$$I'_0(y) = \tfrac{(y-1)(y-2)(y-3)(y-4)}{24} \qquad I'_1(y) = \tfrac{y(y-2)(y-3)(y-4)}{6}$$

$$I'_2(y) = \tfrac{y(y-1)(y-3)(y-4)}{2} \qquad I'_3(y) = \tfrac{y(y-1)(y-2)(y-4)}{6}$$

$$I'_4(y) = \tfrac{y(y-1)(y-2)(y-3)}{24}$$

y	0	1	2	3	4
$I'_0(y)$	1	0	0	0	0
$I'_1(y)$	0	1	0	0	0
$I'_2(y)$	0	0	1	0	0
$I'_3(y)$	0	0	0	1	0
$I'_4(y)$	0	0	0	0	1

Then, $\mu^b_y = \sum_{j=1}^{N} I'_b(D[j][y])$ for $b \in \{0,1,2,3,4\}$. We leverage the precomputed polynomials above to compute 2-way marginals $\mu_{g_i,y}$. We further optimize this by considering each 2-way marginal to be a flattened array of fixed size $20 = 4 \times 5$. Then

$$\mu^{b_g,b_y}_{g_i,y} = \sum_{j=1}^{N} I^{g_i}_{b_g}(D[j][g_i]) \cdot I^{l}_{b_y}(D[j][y])$$

for $b_g \in \{0,1,2,3\}$ and $b_y \in \{0,1,2,3,4\}$.

Protocol 26 computes the secret-shared noisy marginals. Lines 2–26 for computing secret shares of 1-way and 2-way marginals are straightforward, as they directly follow the equations described above. Once the marginals are computed, we utilize MPC protocols from Chapter 9 to generate and add Gaussian noise on Lines 27 – 29.

[9]Comparison operations such as "equals to" are computationally heavy operations in MPC.

Protocol 26: $\pi_{\mathsf{NOISY-MARG}}$: Protocol to compute noisy marginals for binned genomic data

Input : Secret shares of binned data $[\![D]\!]$, scale σ, number of patients N, Domain of gene features $\Omega_g = \{0,1,2,3\}$ and classes $\Omega_y = \{0,1,2,3,4\}$, d number of genes

Output: Secret shares of noisy marginals $[\![\mu]\!]$ for 1-way and selected 2-way marginals

Initialize matrix $[\![\mu_g]\!]$ of size $[d,4]$, array $[\![\mu_y]\!]$ of size $[5]$ and matrix $[\![\mu_{g,y}]\!]$ of size $[d,20]$
for $i \leftarrow 1$ *to* N **do**
    $[\![x]\!] \leftarrow [\![D[i][y]]\!]$; Initialize array $[\![L]\!]$ of size 5 `// Compute 1-way marginal for the label`
    $[\![s_0]\!] \leftarrow [\![x]\!]$; $[\![s_1]\!] \leftarrow [\![x]\!] - 1$; $[\![s_2]\!] \leftarrow [\![x]\!] - 2$; $[\![s_3]\!] \leftarrow [\![x]\!] - 3$; $[\![s_4]\!] \leftarrow [\![x]\!] - 4$
    $[\![L[0]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_1]\!],[\![s_2]\!],[\![s_3]\!],[\![s_4]\!],(1/24))$; $[\![L[1]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_0]\!],[\![s_2]\!],[\![s_3]\!],[\![s_4]\!],(1/6))$
    $[\![L[2]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_0]\!],[\![s_1]\!],[\![s_3]\!],[\![s_4]\!],(1/2))$; $[\![L[3]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_0]\!],[\![s_1]\!],[\![s_2]\!],[\![s_4]\!],(1/6))$
    $[\![L[4]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_0]\!],[\![s_1]\!],[\![s_2]\!],[\![s_3]\!],(1/24))$
    for $r \in \Omega_y$ **do**
        $[\![\mu_y[\mathrm{r}]]\!] \leftarrow [\![\mu_y[\mathrm{r}]]\!] + [\![L[\mathrm{r}]]\!]$
    end
    for *every gene* g **do**
        $[\![x]\!] \leftarrow [\![D[i][g]]\!]$; Initialize array $[\![G]\!]$ of size 5 `// Compute 1-way marginals for genes`
        $[\![s_1]\!] \leftarrow 1 - [\![x]\!]$; $[\![s_{11}]\!] \leftarrow [\![x]\!] - 1$; $[\![s_2]\!] \leftarrow 2 - [\![x]\!]$; $[\![s_{21}]\!] \leftarrow [\![x]\!] - 2$; $[\![s_3]\!] \leftarrow 3 - [\![x]\!]$;
        $[\![G[0]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_1]\!],[\![s_2]\!],[\![s_3]\!],(1/6))$; $[\![G[1]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x]\!],[\![s_2]\!],[\![s_3]\!],(1/2))$
        $[\![G[2]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x]\!],[\![s_{11}]\!],[\![s_3]\!],(1/2))$; $[\![G[3]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x]\!],[\![s_{11}]\!],[\![s_{21}]\!],(1/6))$
        for $r \in \Omega_g$ **do**
            $[\![\mu_g[\mathrm{r}]]\!] \leftarrow [\![\mu_g[\mathrm{r}]]\!] + [\![G[\mathrm{r}]]\!]$
        end
        for $r \in \Omega_g$ **do**
            idx $\leftarrow$ r $*$ 5 `// Compute 2-way selected marginals`
            for $f \in \Omega_y$ **do**
                $[\![\mu_{g,y}[g][idx+f]]\!] \leftarrow [\![\mu_{g,y}[g][idx+f]]\!] + \pi_{\mathsf{MUL}}([\![G[\mathrm{r}]]\!],[\![L[f]]\!])$
            end
        end
    end
end
for *every value* $[\![x]\!]$ *in* $[\![\mu_g]\!]$, $[\![\mu_y]\!]$, $[\![\mu_{g,y}]\!]$ **do**
    $[\![x]\!] \leftarrow [\![x]\!] + \sigma \cdot \pi_{\mathsf{GAUSS}}(0,1)$ `// Adding noise to guarantee DP`
end
return $[\![\mu]\!]$

Secure Learning of Probabilistic Graphical Model

Unlike Chapter 9, the generate step here is done in MPC as discussed in Section 10.2. After computing secret-shares of the noisy marginals, the MPC servers compute secret-shares of the parameters of a probabilistic graphical model (PGM) representing the joint distribution of all variables involved, i.e. the genes and the label. To this end, we follow the Proximal Estimation Algorithm from [288], which uses mirror descent optimization to find model parameters that minimize the loss function between the marginals of the probability distribution represented by the graphical model and the measured noisy marginals as computed in Section 10.3.2. We refer to McKenna et al. [288] for more details on the algorithm.

MPC protocol for Mirror Descent. Protocol 27 presents the MPC protocol for mirror descent used to train PGMs. For mirror descent we closely follow the implementation in [332]. The algorithm iteratively updates model parameters using carefully chosen step sizes and performs belief propagation at each step to maintain consistent marginal distributions. Protocol 27 implements this approach with line search to ensure convergence.

Protocol 27 takes as input the graph structure (see Figure 10.2), assumed to be publicly known, and the secret shares of the computed noisy marginals $[\![\mu]\!]$ from $\pi_{\mathsf{NOISY-MARG}}$. During execution, the MPC servers initialize potential functions over cliques with random values[10], and iteratively update these potentials and the corresponding belief functions. This process involves multiple sequential invocations of the MPC protocols π_{BP} for belief propagation and π_{LOSS} to compute the loss between the measured noisy marginals and the marginals of the current estimated distribution. At the end of the mirror descent protocol, the MPC servers hold secret shares of the learned graphical model parameters as $([\![\boldsymbol{\mu}_{\boldsymbol{\Theta}}]\!], [\![\Theta]\!])$.

Description of π_{BP}. Belief propagation computes marginal posterior distributions through iterative message passing on the posterior junction tree [335]. In a junction tree, each node

[10]Potential functions are factor functions of the joint probability distribution. Cliques are fully connected subsets of nodes.

Protocol 27: $\pi_{\mathsf{MIRROR-DESCENT}}$: Protocol to securely train PGM

Input : Metadata of the dataset D, secret shares of noisy marginals $[\![\mu]\!]$, number of iterations H, total samples N

Output: Secret shares of optimal marginals ($[\![\boldsymbol{\mu}_{\boldsymbol{\Theta}}]\!]$ and potentials $[\![\Theta]\!]$)

Setup the graphical model based on metadata – cliques $\mathcal{C}$ and Groups $\mathbf{G}$, message order M, inverse of scaled noise ς computed using budget accounting

Initialize potentials $[\![\Phi]\!]$

$[\![\mathbf{B}]\!] \leftarrow \pi_{\mathsf{BP}}([\![\Phi]\!], \mathcal{C}, \mathrm{M}, \Omega, N)$ // Belief Propagation

$[\![\mathcal{L}]\!], [\![\Delta]\!] \leftarrow \pi_{\mathsf{LOSS}}([\![\mathbf{B}]\!], [\![\mu]\!], \mathcal{C}, \mathbf{G}, \varsigma)$ // Marginal Loss

$\alpha \leftarrow 1/N^2$

for $i \leftarrow 1$ **to** H **do**

  $[\![\phi]\!], [\![\nu]\!] \leftarrow [\![\Phi]\!], [\![\mathbf{B}]\!]$

  $[\![l]\!], [\![\delta l]\!] \leftarrow [\![\mathcal{L}]\!], [\![\Delta]\!]$

  $\alpha_i \leftarrow 2\alpha/i$ // Decreasing step size schedule

  $j \leftarrow 1$ // Line Search

  while $j \leq 25$ **do**

    $[\![\theta_{new}]\!] \leftarrow [\![\phi]\!] - \pi_{\mathsf{MUL}}(\alpha_i, [\![\delta l]\!])$

    $[\![\nu_{new}]\!] \leftarrow \pi_{\mathsf{BP}}([\![\theta_{new}]\!], \mathcal{C}, \mathrm{M}, \Omega, N)$

    $[\![l_{new}]\!], [\![\delta l_{new}]\!] \leftarrow \pi_{\mathsf{LOSS}}([\![\nu_{new}]\!], [\![\mu]\!], \mathcal{C}, \mathbf{G}, \varsigma)$

    $[\![diff]\!] \leftarrow [\![l]\!] - [\![l_{new}]\!]$

    $[\![prod]\!] \leftarrow \pi_{\mathsf{DOT}}([\![\delta l]\!], [\![\nu]\!] - [\![\nu_{new}]\!])$

    $[\![cond]\!] \leftarrow ([\![diff]\!], 0.5 \cdot \alpha_i \cdot [\![prod]\!])$ // Check Armijo condition

    $[\![b]\!] \leftarrow [\![cond]\!] \geq 0$

    $[\![\Phi]\!] \leftarrow [\![b]\!] \cdot [\![\theta_{new}]\!] + (1 - [\![b]\!]) \cdot [\![\Phi]\!]$

    $[\![\mathbf{B}]\!] \leftarrow [\![b]\!] \cdot [\![\nu_{new}]\!] + (1 - [\![b]\!]) \cdot [\![\mathbf{B}]\!]$

    $[\![\mathcal{L}]\!] \leftarrow [\![b]\!] \cdot [\![l_{new}]\!] + (1 - [\![b]\!]) \cdot [\![\mathcal{L}]\!]$

    $[\![\Delta]\!] \leftarrow [\![b]\!] \cdot [\![\delta l_{new}]\!] + (1 - [\![b]\!]) \cdot [\![\Delta]\!]$

    $[\![continue]\!] \leftarrow 1 - [\![b]\!]$

    $\alpha_i \leftarrow \alpha_i \cdot (1 - [\![b]\!]) \cdot 0.5 + \alpha_i \cdot [\![b]\!]$

    $j \leftarrow j + [\![continue]\!]$

  end

  $[\![\Phi]\!], [\![\mathbf{B}]\!] \leftarrow [\![\theta_{new}]\!], [\![\nu_{new}]\!]$ // Fallback update if line search exhausted

  $[\![\mathcal{L}]\!], [\![\Delta]\!] \leftarrow [\![l_{new}]\!], [\![\delta l_{new}]\!]$

end

$[\![\Theta]\!], [\![\boldsymbol{\mu}_{\boldsymbol{\Theta}}]\!] \leftarrow [\![\Phi]\!], [\![\mathbf{B}]\!]$

represents a clique (maximal fully connected subgraph) with associated potential function $\Phi_c(x_c)$ encoding local probability information. Message passing involves each clique node c sending messages $\mu_{c \to s}$ to its neighboring separator sets s, where each message represents

the local belief about variables in the separator. Mathematically, a message from clique c to separator s is computed as:

$$\mu_{c \to s}(x_s) = \sum_{x_c \setminus x_s} \Phi_c(x_c) \prod_{n \in N(c) \setminus s} \mu_{n \to c}(x_{n \cap c})$$

where $N(c)$ denotes neighbors of clique c, and $x_c \setminus x_s$ represents variables in clique c but not in separator s. These messages are iteratively updated until convergence, allowing beliefs to propagate through the junction tree.

To ensure numerical stability, each clique node applies the `logsumexp` function to the received message matrix τ as follows:

$$\text{logsumexp}(\tau)_j = m_j + \log \sum_i \exp(\tau_{i,j} - m_j); \quad m_j = \max_i \tau_{i,j}$$

Each message matrix (with dimensions $\omega_i \times \omega_y$; 4×5 in our use case) requires 20 exponential and 5 logarithmic operations, making these computations expensive in MPC. We approximate the exponential function with its Taylor series expansion and evaluate the polynomial with Horner's method:

$$\text{poly\_exp} = 1.0 + x \cdot \left(1.0 + x \cdot \left(0.5 + x \cdot \left(\frac{1}{6} + x \cdot \left(\frac{1}{24} + x \cdot \frac{1}{120}\right)\right)\right)\right)$$

Protocol 28: $\pi_{\mathsf{EXP-H}}$: Protocol for secure exponential computation using Horner's method

Input : Secret share $[\![x]\!]$

Output: Secret share $[\![p]\!]$ of the computed exponential

$[\![p]\!] \leftarrow 1.0 + \pi_{\mathsf{MUL}}([\![x]\!], (1.0 + \pi_{\mathsf{MUL}}([\![x]\!], (0.5 + \pi_{\mathsf{MUL}}([\![x]\!], (\frac{1}{6} + \pi_{\mathsf{MUL}}([\![x]\!], (\frac{1}{24} + \pi_{\mathsf{MUL}}([\![x]\!], \frac{1}{120}))))))))))$

return $[\![p]\!]$

We implemented MPC protocol $\pi_{\mathsf{EXP-H}}$ (Protocol 28) for poly_exp. Our empirical evaluation shows that using this polynomial for the exponential function introduces minimal utility loss. The maximum error observed in a single belief computation over 5 runs was

approximately 2.9×10^{-5}. However, applying a similar polynomial approximation for the logarithm resulted in a higher error (∼172 per belief computation), likely due to a broader input range.

Protocol 29: π_{BP}: Protocol for secure belief propagation

Input : Secret shares of potentials $[\![\Phi_c]\!]$ for each clique $c \in \mathcal{C}$ of the input graph, message order $\mathrm{M} = \{[c_i, c_j]\}$ for clique pair c_i, c_j, domain ω_g for gene features and ω_y for label, total number N of samples

Output: Secret shared beliefs $[\![\mathbf{B}]\!]$ for all cliques in $\mathcal{C}$

Initialize $[\![\mathbf{B}]\!]$ with $[\![\Phi]\!]$

`// Message Passing`

Initialize dictionary to hold messages passed $\mathsf{M} \leftarrow \{\}$

for *(i,j) in M* **do**

  if *(j,i) in M* **then**

    $[\![\tau]\!] \leftarrow [\![\mathbf{B}_i]\!] - [\![\mathsf{M}_{(j,i)}]\!]$ `// Element wise subtraction`

  else

    $[\![\tau]\!] \leftarrow [\![\mathbf{B}_i]\!]$

  end

  $[\![\tau]\!] \leftarrow \pi_{\mathsf{LNSEXP-VEC}}([\![\tau]\!], \omega_y)$ `// Shapes based on input Fig. 10.2`

  $[\![\mathsf{M}_{(j,i)}]\!] \leftarrow [\![\tau]\!]$

  $[\![\mathbf{B}_i]\!] \leftarrow [\![\tau]\!]$

end

`// Scale beliefs`

$[\![\mathrm{logZ}]\!] \leftarrow \pi_{\mathsf{LNSEXP}}([\![\mathbf{B}_0]\!])$ `// See Protocol 31`

$[\![\mathrm{logZ}]\!] \leftarrow \log(N) - [\![\mathrm{logZ}]\!]$

for *all cliques* $c \in \mathcal{C}$, $i \in \{1, \ldots, \omega_g\}$, $j \in \{1, \ldots, \omega_y\}$ **do**

  $[\![\mathbf{B}_c[i][j]]\!] \leftarrow \pi_{\mathsf{EXP-H}}([\![\mathbf{B}_c[i][j]]\!] + [\![\mathrm{logZ}]\!])$ `// Element-wise`

end

return $[\![\mathbf{B}]\!]$

Protocol 29 runs the belief propagation algorithm on secret shares of the potentials, denoted as $[\![\Phi_c]\!]$, for each clique in the graph and results in secret shares of the computed

beliefs, $[\![\mathbf{B}]\!]$, for each clique node.

Lines 3 – 12 in Protocol 29 implement message passing. Line 9 invokes the MPC protocol for `logsumexp` (Protocol 30), which provides the option to use either $\pi_{\mathsf{EXP-H}}$ or π_{EXP}, the latter being an MPC sub-protocol available in MP-SPDZ [110]. Based on our analysis, a single call to exp(x) with $\pi_{\mathsf{EXP-H}}$ reduced runtime to 0.005 seconds in the 3PC passive setting, compared to 0.011 seconds for the default implementation in MP-SPDZ, achieving a ∼55% reduction in runtime[11].

After message passing, Lines 13 – 17 scale the computed secret shares of beliefs using the following equation in straightforward manner

$$\mathbf{B}_c \leftarrow \exp\left(\mathbf{B}_c + \log(N) - logZ\right); \quad logZ \leftarrow \text{logsumexp}(\mathbf{B}_c)$$

Note that `logsumexp` operates on a matrix, and is computed on Line 13 via Protocol 31. *Remark*: Using `logsumexp` is a standard technique to ensure numerical stability and scale beliefs appropriately with dataset size, enabling accurate marginal loss computation in PGMs. One of the alternative ways for MPC-friendly scaling is to use following equations instead

$$\mathbf{B}_c = \exp\left(\mathbf{B}_c - \max\left(\mathbf{B}_c\right)\right); \quad \mathbf{B}_c = \mathbf{B}_c \cdot \left(\frac{N}{\sum \mathbf{B}_c}\right)$$

Using alternative scaling resulted in a statistical utility loss of 0.002 (in terms of workload error) and an ML utility loss of 0.15, while reducing the MPC runtime by 0.02 seconds per belief scaling. For 958 beliefs, this yields a total runtime reduction of approximately 19 seconds (measured on a local machine)[12]. Protocol 32 implements the above mentioned alternative scaling. Depending on the acceptable trade-off between error tolerance and runtime, an appropriate MPC protocol can be selected. .

---

[11]For datasets with different ranges where polynomial approximation might be unsuitable, one can use π_{EXP} from MP-SPDZ. We use π_{LOG} from MP-SPDZ for logarithm computations. All performance analyses were conducted on a local machine.

[12]There are several ways to implement MPC-friendly scaling. We experimented with various techniques, including removing `logsumexp` and using softmax, but all of them led to higher observed errors, even in the clear setting.

Protocol 30: $\pi_{\mathsf{LNSEXP-VEC}}$: Protocol to compute logsumexp of a vector

Input : Secret shares of a matrix $[\![\tau]\!]$, domain size ω_y, polynomial approximation flag poly_approx

Output: Secret shares $[\![\mathrm{s}]\!]$ of the computed logsumexp

Initialize $[\![a_{max}]\!]$ of size ω_y
$[\![\tau_T]\!] \leftarrow [\![\tau^T]\!]$ // Transpose
for $i \leftarrow 1$ **to** $|[\![\tau_T]\!]|$ **do**
    $[\![a_{max}]\!] \leftarrow \pi_{\mathsf{MAX}}([\![\tau_T[i]]\!])$
end
$[\![\tau[i]]\!] \leftarrow [\![\tau[i]]\!] - [\![a_{max}]\!] \quad \forall \quad i \quad$ rows of$[\![\tau]\!]$
if *poly_approx* **then**
    $[\![\tau[i][j]]\!] \leftarrow \pi_{\mathsf{EXP-H}}([\![\tau[i][j]]\!]) \quad \forall$ elements of $[\![\tau]\!]$
else
    $[\![\tau[i][j]]\!] \leftarrow \pi_{\mathsf{EXP}}([\![\tau[i][j]]\!]) \quad \forall$ elements of $[\![\tau]\!]$ // π_{EXP} from MP-SPDZ
end
Initialize $[\![\mathrm{s}]\!]$ of size ω_y
for $j \leftarrow 1$ **to** ω_y **do**
    for $i \leftarrow 1$ **to** $|[\![\tau]\!]|$ **do**
        $[\![\mathrm{s}[j]]\!] \leftarrow [\![\mathrm{s}[j]]\!] + [\![\tau[i][j]]\!]$
    end
    $[\![\mathrm{s}[j]]\!] \leftarrow \pi_{\mathsf{LOG}}([\![\mathrm{s}[j]]\!])$ // π_{LOG} from MP-SPDZ
end
$[\![\mathrm{s}]\!] \leftarrow [\![a_{\max}]\!] + [\![\mathrm{s}]\!]$
return $[\![s]\!]$

Description of π_{LOSS}. Protocol 34, π_{LOSS}, computes the secret shares of the L2 marginal loss, as defined by the equation below, along with the corresponding gradients for the graphical model. It measures the error between the secret shares of the noisy input marginals $[\![\mu]\!]$ obtained from $\pi_{\mathsf{NOISY-MARG}}$ and the beliefs $[\![\mathbf{B}]\!]$ computed by π_{BP}.

$$\mathcal{L} = \sum_{c} \sum_{\forall \mu, \mu_{marg} \in \mathbf{B}} \frac{\varsigma}{2} \left(\mu_{\text{marg}} - \mu\right)^2; \quad \varsigma \text{ is constant scaling factor}$$

Protocol 31: π_{LNSEXP}: Protocol to compute logsumexp of a matrix

Input : Secret shares of a matrix $[\![\mathbf{B}_0]\!]$

Output: Secret shares $[\![x]\!]$ of the computed logsumexp of the matrix

Initialize $[\![a_{max}]\!]$ of size ω_y
$[\![\tau_T]\!] \leftarrow [\![\mathbf{B}_0^T]\!]$ // Transpose
for $i \leftarrow 1$ **to** $|[\![\tau_T]\!]|$ **do**
    $[\![a_{max}]\!] \leftarrow \pi_{\mathsf{MAX}}([\![\tau_T[i]]\!])$
end
$[\![\tau[i]]\!] \leftarrow [\![\mathbf{B}_0[i]]\!] - [\![a_{max}]\!] \quad \forall \quad i \quad$ rows of$[\![\mathbf{B}_0]\!]$
if *poly_approx* **then**
    $[\![\tau[i][j]]\!] \leftarrow \pi_{\mathsf{EXP-H}}([\![\tau[i][j]]\!]) \quad \forall$ elements of $[\![\tau]\!]$
else
    $[\![\tau[i][j]]\!] \leftarrow \pi_{\mathsf{EXP}}([\![\tau[i][j]]\!]) \quad \forall$ elements of $[\![\tau]\!]$ // π_{EXP} from MP-SPDZ
end
Initialize $[\![\mathrm{s}]\!]$ of size ω_y
for $j \leftarrow 1$ **to** ω_y **do**
    for $i \leftarrow 1$ **to** $|[\![\tau]\!]|$ **do**
        $[\![\mathrm{s}[j]]\!] \leftarrow [\![\mathrm{s}[j]]\!] + [\![\tau[i][j]]\!]$
    end
    $[\![\mathrm{s}[j]]\!] \leftarrow \pi_{\mathsf{LOG}}([\![\mathrm{s}[j]]\!])$ // π_{LOG} from MP-SPDZ
end
$[\![\mathrm{s}]\!] \leftarrow [\![a_{\max}]\!] + [\![\mathrm{s}]\!]$
return $[\![s]\!]$

This loss is used to guide the optimization process for learning the graphical model by minimizing this difference.

Apart from the cliques $\mathcal{C}$ of the graphical model, π_{LOSS} also takes $\mathbf{G}$ as input which groups the input marginals according to the structure of the graph. Each group associates the input marginals $[\![\mu]\!]$ with the clique c and its corresponding belief $[\![\mathbf{B}_c]\!]$. Each group is represented as a tuple containing the input marginal and the type of computed marginal. The marginal type is enumerated as follows: type 0 represents a 1-way marginal for the gene

Protocol 32: π_{SCALEB}: Protocol to scale beliefs

Input : Secret shares of computed beliefs $[\![\mathbf{B}]\!]$ for all cliques $\in \mathcal{C}$, N number of samples

Output: Secret shares of scaled beliefs $[\![\mathbf{B}]\!]$ for all cliques $\in \mathcal{C}$

`// Scale beliefs`

1 **for** *all cliques* $c \in \mathcal{C}$ **do**

2 Initialize $[\![\text{max}]\!]$ and $[\![\text{sum}]\!]$ each of size ω_y

3 **for** *all rows* $i \leftarrow 1$ **to** ω_y **do**

4 $[\![\text{max}[i]]\!] \leftarrow \pi_{\mathsf{MAX}}([\![\mathbf{B}_c[i]]\!])$

5 **end**

6 $[\![\text{max\_val}]\!] \leftarrow \pi_{\mathsf{MAX}}([\![\text{max}]\!])$

7 **for** *all rows* $i \leftarrow 1$ **to** ω_y **do**

8 $[\![\text{sum}[i]]\!] \leftarrow \pi_{\mathsf{SUM}}([\![\mathbf{B}_c[i]]\!])$

9 **end**

10 $[\![\text{sum\_val}]\!] \leftarrow \pi_{\mathsf{SUM}}([\![\text{sum}]\!])$

11 **for** *each element* $i \leftarrow 1$ **to** ω_g, $j \leftarrow 1$ **to** ω_y **do**

12 $[\![\mathbf{B}_c[i][j]]\!] \leftarrow \pi_{\mathsf{EXP-H}}([\![\mathbf{B}_c[i][j]]\!] - [\![\text{max\_val}]\!])$ `// Element-wise`

13 $[\![\mathbf{B}_c[i][j]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\mathbf{B}_c[i][j]]\!], \pi_{\mathsf{DIV}}(N, \text{sum\_val}))$ `// Element-wise`

14 **end**

15 **end**

$[\![\mu_g]\!]$, type 1 represents a 1-way marginal for the label $[\![\mu_y]\!]$, and type 2 represents a 2-way marginal for the gene-label pair $[\![\mu_{g,y}]\!]$. This grouping is independent of the data and can be performed in the clear and provided as metadata. Example: The root node clique `('AARS', 'label')` will have one corresponding belief and three marginals, namely, `('AARS')` (type = 0), `('label')` (type = 2), and `('AARS', 'label')` (type = 2)[13].

Following this, π_{LOSS} computes the secret shares of the loss $[\![\mathcal{L}]\!]$ and the secret shares of the gradients $[\![\Delta]\!]$ for each clique. It iterates over each clique c on Line 2, fetching the

[13]Note that since the input graph is a star network, each of the remaining cliques has only two associated input marginals: the 1-way marginal for the gene and the corresponding 2-way marginal with the root clique. The 1-way marginal for the 'label' is already assigned to the root clique.

Protocol 33: π_{MARG}: Protocol to extract 1-way beliefs from 2-way beliefs

Input: Secret shares $[\![\tau]\!]$, type

Output: Secret shares of marginal $[\![\mu_{marg}]\!]$

1 **if** *type == 1* **then**

2 $\quad [\![\tau]\!] \leftarrow [\![\tau^T]\!]$ `// Transpose`

3 **end**

4 $n \leftarrow |[\![\tau]\!]|$ `// Get the dimension`

5 Initialize 1 dimensional vector $[\![\mu_{marg}]\!]$ of length n

6 **for** $i = 1$ **to** n **do**

7 $\quad [\![\mu_{marg}[i]]\!] \leftarrow [\![\mu_{marg}[i]]\!] + \pi_{\mathsf{SUM}}([\![\tau[i]]\!])$ `// `π_{SUM}` for vector sum`

8 **end**

9 **return** $[\![\mu_{marg}]\!]$

corresponding belief $[\![\mathbf{B}_c]\!]$ of dimension $\omega_g \times \omega_y$ and group $[\![\mathbf{G}_c]\!]$. On Line 4, we iterate over each marginal in the group. If it is a 1-way marginal, we extract the corresponding 1-way belief on Line 6 using Protocol 33 (π_{MARG}). π_{MARG} simply sums up the secret shares of elements along the rows or columns, resulting in a 1-dimensional output of size ω_g or ω_y. If it is a 2-way marginal, we use π_{FLATTEN} to convert it into a 1-dimensional array on Lines 9 – 10. The secret share of the loss is computed on Line 13 using π_{DOT} that computes the dot product. The secret share of the gradient is computed on Line 14. Secret shares of the gradients are then reshaped and accumulated for each clique on Lines 15 – 16.

Secure Generate through Model Inference

At this point, the MPC servers hold the secret shares of the learned graphical model $[\![\boldsymbol{\mu}_{\boldsymbol{\Theta}}]\!]$, which is used to generate the secret shares of synthetic data $[\![\hat{D}]\!]$ using π_{GEN}.

Description of π_{GEN}. Protocol 36 takes as input $[\![\boldsymbol{\mu}_{\boldsymbol{\Theta}}]\!]$, which represents the optimal 2-way marginals for column pairs (g, y) with dimensions $\omega_g \times \omega_y$, along with the ordered cliques. To generate synthetic data with N' samples, π_{GEN} initializes the secret shares of synthetic

Protocol 34: π_{LOSS}: Protocol for secure loss computation

Input: Secret shares of beliefs $[\![\mathbf{B}]\!]$ for all cliques $c \in \mathcal{C}$ of input graph, noisy marginals $[\![\mu]\!]$ grouped into groups $\mathbf{G}$ based on input graph, ς inversed noise scale

Output: Secret shares of marginal loss $[\![\mathcal{L}]\!]$, gradients $[\![\Delta]\!]$

`// Assumption: All inputs are orderly structured based on cliques`

1 Initialize dictionary to hold computed gradients for each clique $[\![\Delta]\!] \leftarrow \{\}$ and $[\![\mathcal{L}]\!] \leftarrow 0$
2 **for** *each clique* $c \in \mathcal{C}$ **do**
3 Initialize $[\![\Delta_c]\!]$ to a matrix of size $\omega_g \times \omega_y$
4 **for** *every* $([\![\mu]\!]$*, type)* $\in \mathbf{G_c}$ **do**
5 **if** *type != 2* **then**
6 $[\![\mu_{marg}]\!] \leftarrow \pi_{\mathsf{MARG}}([\![\mathbf{B_c}]\!], \text{type})$
7 **end**
8 **else**
9 $[\![\mu_{marg}]\!] \leftarrow \pi_{\mathsf{FLATTEN}}([\![\mathbf{B_c}]\!])$ `// `π_{FLATTEN}` to flatten matrix in MP-SPDZ`
10 $[\![\mu]\!] \leftarrow \pi_{\mathsf{FLATTEN}}([\![\mu]\!])$
11 **end**
12 $[\![d]\!] = \varsigma * ([\![\mu_{marg}]\!] - [\![\mu]\!])$ `// Element-wise subtraction`
13 $[\![\mathcal{L}]\!] \leftarrow [\![\mathcal{L}]\!] + 0.5 \cdot \pi_{\mathsf{DOT}}(d, d)$ `// `π_{DOT}` for dot product`
14 $[\![\delta]\!] \leftarrow \varsigma \cdot [\![d]\!]$
15 $[\![\delta]\!] \leftarrow \text{Reshape}([\![\delta]\!], \omega_g \times \omega_y)$ `// Reshapes the matrix`
16 $[\![\Delta_c]\!] \leftarrow [\![\Delta_c]\!] + [\![\delta]\!]$
17 **end**
18 **end**
19 **return** $[\![\mathcal{L}]\!], [\![\Delta]\!]$

data $[\![\hat{D}]\!]$ of size $N' \times (d+1)$ on Line 12.

The key idea is to first extract the marginal distribution for each column individually, starting with the label column. The number of occurrences for each value in the label's domain is determined from the marginal, ensuring that each label value appears the appropriate number of times. Next, for each label value, the corresponding marginal counts for the associated feature are extracted, and feature values are generated accordingly. This

		Marginal			
	label/feature	0	1	2	
	0	2	1	1	
	1	1	2	2	
					Generated g col
label	vals_l=0.g	vals_l=1,g	vals_l=2,g	vals	vals + domain_g
0	0	-1	-1	-2	0
0	0	-1	-1	-2	0
0	1	-1	-1	-1	1
1	-1	0	-1	-2	0
1	-1	1	-1	-1	1
1	-1	1	-1	-1	1
2	-1	-1	0	-2	0
2	-1	-1	1	-1	1
2	-1	-1	1	-1	1

Figure 10.3: Example to generate a gene column with domain {0,1} after the label column with domain {0,1,2} is generated

ensures that the generated feature values align with the given marginals while preserving the dependencies between columns.

To this end, π_{GEN} begins by generating the label column first[14], obtaining counts by aggregating column-wise using π_{MARG} on Line 2. The first clique, indexed at 0, represents the root node and contains all accumulated beliefs for the label column. Any extracted marginal is optionally normalized on Line 3 and 11, typically by converting real-valued marginals to integers to represent occurrence counts accurately (See Protocol 37 in Appendix). Normalization by N' scales marginals optimized for N, but since our framework assumes $N = N'$, this step can be skipped. The computed $[\![\mu_{\Theta,y}]\!]$ now holds the frequency of each label domain value.

[14]All marginals are conditioned on the label.

Protocol 35: $\pi_{\mathsf{GR-COL}}$: Protocol for generating a column conditioned on other column

Input: Secret shares $[\![c]\!]$, Secret shares $[\![\text{p}]\!]$

Output: Secret shares of generated column values $[\![val]\!]$

1 Initialize $[\![val]\!]$ of size N' with values of -1
2 Initialize $[\![sp]\!]$ of size same as $[\![c]\!]$; $[\![sp[0]]\!] \leftarrow [\![\text{p}]\!]$
3 **for** $i \leftarrow 2$ **to** $|[\![c]\!]|$ **do**
4 $[\![sp[i]]\!] \leftarrow [\![sp[i]]\!] + [\![c[i-1]]\!]$
5 **end**
6 **for** $k \leftarrow 0$ **to** $N' - 1$ **do**
7 **for** $j \leftarrow 0$ **to** $|[\![c]\!]| - 1$ **do**
8 $[\![\text{st}]\!] \leftarrow [\![sp[j]]\!]$; $[\![\text{end}]\!] \leftarrow [\![c[j]]\!]$
9 $[\![\text{ge}]\!] \leftarrow \pi_{\mathsf{GTE}}((k - [\![\text{st}]\!]), 0)$
10 $[\![\text{lt}]\!] \leftarrow \pi_{\mathsf{LT}}((k - [\![\text{end}]\!]), 0)$
11 $[\![\text{rg}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\text{ge}]\!], [\![\text{lt}]\!])$
12 $[\![val[pos]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![val[pos]]\!], (1 - [\![\text{rg}]\!])) + \pi_{\mathsf{MUL}}(j, [\![\text{rg}]\!])$
13 **end**
14 **end**
15 **return** $[\![val]\!]$

For example, $\mu_{\Theta,y} = [3, 3, 3]$ indicates that each label value (indexed from 0; assuming the domain of the label is $\{0, 1, 2\}$, hence the size of $\mu_{\Theta,y}$ is 3) should appear in three rows, resulting in a total of 9 rows. The label column is initialized with -1 to represent unassigned values: $[-1, -1, -1, -1, -1, -1, -1, -1, -1]$. The process starts by assigning the first domain value 0 using its count in $\mu_{\Theta,y}$, producing: $[0, 0, 0, -1, -1, -1, -1, -1, -1]$. To determine where to begin inserting the next domain value 1, we need the cumulative count of previous values. However, since these counts are secret-shared, they cannot be accessed directly. $\pi_{\mathsf{GR-COL}}$ in Protocol 35 handles this logic securely, and the resulting secret-shared label column is produced on Line 4 of Protocol 36 and appended column-wise to $[\![\hat{D}]\!]$ on Line 5.

Protocol 36: π_{GEN}: Protocol for secure generation of data

Input: Secret shares of optimized marginals $[\![\boldsymbol{\mu_\Theta}]\!]$ for all cliques $\mathcal{C}$, domain dimensions ω_g, ω_y, Number of rows to generate N', number of genes d

Output: Secret shares of generated synthetic data $[\![\hat{D}]\!]$

`// Assumption: All cliques are ordered with first clique being the root node and ordered in the sequence of generation`

Initialize $[\![\hat{D}]\!]$ with $N' \times (d+1)$;

`// Generate label`

$[\![\mu_{\Theta,y}]\!] \leftarrow \pi_{\mathsf{MARG}}([\![\mu_\Theta[0]]\!])$;

Convert $[\![\mu_{\Theta,y}]\!]$ to integers and normalize ; `// Protocol 37`

$[\![\text{vals}]\!] \leftarrow \pi_{\mathsf{GR-COL}}([\![\mu_{\Theta,y}]\!], [\![0]\!])$;

$[\![\hat{D}[-1]]\!] \leftarrow \pi_{\mathsf{ASSGN}}([\![\text{vals}]\!])$; `// Columnwise assignment`

for *g in* $\mathcal{C}$ **do**

  $[\![\mu_g]\!] \leftarrow [\![\mu_\Theta[g]]\!]^T$; `// Each gene is associated with a unique clique`

  Initialize array $[\![\text{vals}]\!]$ of size N';

  for $r \in \Omega_y$ **do**

    $[\![\mu_{rg}]]\!] \leftarrow [\![\mu_g[r]]\!]$;

    Convert $[\![\mu_{rg}]]\!]$ to integers and normalize ; `// Protocol 37`

    if *r==0* **then**

      $[\![\text{st\_pos}]]\!] \leftarrow [\![0]]\!]$; $[\![\text{pr\_st\_pos}]]\!] \leftarrow [\![0]]\!]$;

    end

    else

      $[\![\text{st\_pos}]]\!] \leftarrow [\![\mu_g[r-1]]]\!]$; $[\![\text{pr\_st\_pos}]]\!] \leftarrow [\![\text{st\_pos}]]\!]$;

    end

    $[\![\text{vals}_{lg}]\!] \leftarrow \pi_{\mathsf{GR-COL}}([\![\mu_{rg}]]\!], [\![\text{st\_pos}]]\!])$;

    $[\![\text{vals}]\!] \leftarrow [\![\text{vals}]\!] + [\![\text{vals}_{lg}]\!]$;

  end

  $[\![\text{vals}]\!] \leftarrow [\![\text{vals}]\!] + (\omega_y - 1)$;

  $[\![\hat{D}[g]]\!] \leftarrow \pi_{\mathsf{ASSGN}}([\![\text{vals}]\!])$;

end

return $[\![\hat{D}]\!]$;

Protocol 37: Protocol to convert into integers and normalize

Input: Secret shares of marginal for a single clique $[\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!]$, N', n rows to generate

Output: Secret shares of normalized integer marginal for a single clique $[\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!]$

1 $[\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!], \pi_{\mathsf{DIV}}(N', \pi_{\mathsf{SUM}}([\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!]))$;
2 $[\![\boldsymbol{\mu}^{\boldsymbol{Z}}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!] \leftarrow \pi_{\mathsf{FLOOR}}([\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!])$;
3 $[\![\mathrm{diff}]\!] \leftarrow [\![\boldsymbol{\mu}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!] - [\![\boldsymbol{\mu}^{\boldsymbol{Z}}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!]$;
4 $[\![\mathrm{ex}]\!] \leftarrow N' - \pi_{\mathsf{SUM}}([\![\boldsymbol{\mu}^{\boldsymbol{Z}}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!])$;
5 Initialize $[\![\mathrm{I}]\!]$ and $[\![\mathrm{m}]\!]$ of size N';
6 **for** $k \leftarrow 1$ **to** N' **do**
7 $[\![\mathrm{I}_k]\!] \leftarrow \pi_{\mathsf{RC}}[\![\mathrm{diff}]\!]$;
8 $[\![\mathrm{I}[k]]\!] \leftarrow \pi_{\mathsf{MUL}}([\![\mathrm{I}_k]\!], \pi_{\mathsf{LT}}(k, [\![\mathrm{ex}]\!]))$;
9 **end**
10 **for** $i \leftarrow 1$ **to** n **do**
11 $[\![\mathrm{sum}]\!] \leftarrow 0$;
12 **for** $k \leftarrow 1$ **to** N' **do**
13 $[\![\mathrm{sum}]\!] \leftarrow [\![\mathrm{sum}]\!] + \pi_{\mathsf{MUL}}(\pi_{\mathsf{EQ}}([\![\mathrm{I}[k]]\!], k), \pi_{\mathsf{LT}}(k, [\![\mathrm{ex}]\!]))$;
14 **end**
15 $[\![\mathrm{m}[i]]\!] \leftarrow [\![\mathrm{sum}]\!]$;
16 **end**
17 **for** $i \leftarrow 1$ **to** n **do**
18 $[\![\boldsymbol{\mu}^{\boldsymbol{Z}}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!] \leftarrow [\![\boldsymbol{\mu}^{\boldsymbol{Z}}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!] + [\![\mathrm{m}[i]]\!]$;
19 **end**
20 **return** $[\![\boldsymbol{\mu}^{\boldsymbol{Z}}_{\boldsymbol{\Theta},\boldsymbol{col}}]\!]$;

The logic behind $\pi_{\mathsf{GR-COL}}$ is to maintain the secret shares of the start and end positions for each domain value, as computed on Lines 3–6 and Line 9 of Protocol 35. To avoid leaking index positions and to defend against side-channel attacks, the protocol does not use simple conditional branching. Instead, it relies on arithmetic expressions on Lines 10–12 to securely check whether the current index (traversed on Line 7) falls within the valid range for a given domain value. Only when this condition is satisfied is the column value updated on Line 13. This approach ensures that position checks are data-oblivious and do not reveal any information about index positions or marginal counts[15].

[15] Although these marginals are already differentially private (DP), the key idea of our approach is to perform all operations within MPC, avoiding intermediate disclosures and thus preserving the privacy budget until the final stage.

This procedure is repeated for each gene column on Lines 6–23 to generate the secret shares of the gene values corresponding to each label. For every unique label value on Line 9, the corresponding marginals are extracted on Lines 7 and 10 to generate the associated gene column. Lines 12–17 determine the secret shares of the starting positions in the label column for each label value, which are then used by $\pi_{\mathsf{GR-COL}}$ on Line 18. Lines 19 – 21 then generate the secret shares of the gene column.

Figure 10.3 illustrates this process for generating a gene column with domain $\{0, 1\}$. For instance, for label value 0, the extracted counts are $[2, 1]$, and the corresponding column `vals=0,g` is generated. Similar procedures are followed for other label values. Note that invalid ranges are assigned the value -1, which appears $(\omega_y - 1)$ times in each row. To eliminate these, all subset columns are summed on Line 35, and $(\omega_y - 1)$ is added on Line 37 to obtain the final gene column. This final column is then appended column-wise to the secret shares of $[\![\hat{D}]\!]$ on Line 38.

10.3.3 Secure Evaluation of Synthetic Data

We evaluate the quality of synthetic data based on two metrics – (a) the accuracy of a logistic regression model trained for multiclass classification to identify the disease type, and (b) workload error. As instantiations of π_{EVAL} on Line 14 in Algorithm 23, we develop MPC protocols to perform these evaluations within MPC. In this case, $[\![r]\!]$ and $[\![M]\!]$ in Algorithm 23 are vectors of size 2.

For (a) we use an MPC protocol for training and inference with logistic regression, building upon the MPC primitives available for Dense and Softmax layers in MP-SPDZ [110]. For (b), we designed an MPC protocol π_{WLE} that computes secret shares of the workload error following the equation

$$e = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \sum_{b \in \Omega_q} |\mu_q^b(D) - \mu_q^b(\hat{D})|$$

Protocol 38: π_{WLE}: Protocol to compute normalized marginal error (π_{EVAL})

Input : Secret shares of binned data $[\![D]\!]$, Secret shares of binned synthetic data $[\![\hat{D}]\!]$, number of patients N, Domain of gene features $\Omega_g = 0, 1, 2, 3$ and classes $\Omega_y = 0, 1, 2, 3, 4$

Output: Secret shares of workload error $[\![e]\!]$

1 Initialize $[\![e]\!]$

2 $[\![\mu_g]\!]$, $[\![\mu_y]\!]$, $[\![\mu_{g,y}]\!] \leftarrow$ Run Lines 1–26 from Protocol 26 on $[\![D]\!]$

3 $[\![\hat{\mu}_g]\!]$, $[\![\hat{\mu}_y]\!]$, $[\![\hat{\mu}_{g,y}]\!] \leftarrow$ Run Lines 1–26 from Protocol 26 on $[\![\hat{D}]\!]$

4 **for** *every value* $[\![x]\!]$ *in* $[\![\mu_g]\!]$, $[\![\mu_y]\!]$, $[\![\mu_{g,y}]\!]$ **do**

5 $\quad [\![x]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x]\!], (1/N))$

6 **end**

7 **for** *every value* $[\![x]\!]$ *in* $[\![\hat{\mu}_g]\!]$, $[\![\hat{\mu}_y]\!]$, $[\![\hat{\mu}_{g,y}]\!]$ **do**

8 $\quad [\![x]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x]\!], (1/N))$

9 **end**

10 **for** *every value* ($[\![\hat{x}]\!]$ *in* $[\![\hat{\mu}_g]\!]$, $[\![\hat{\mu}_l]\!]$, $[\![\hat{\mu}_{g,y}]\!]$) *and* ($[\![x]\!]$ *in* $[\![\mu_g]\!]$, $[\![\mu_l]\!]$, $[\![\mu_{g,y}]\!]$) **do**

11 $\quad [\![e]\!] \leftarrow [\![e]\!] + \pi_{\mathsf{ABS}}([\![x]\!] - [\![\hat{x}]\!])$

12 **end**

13 **return** $[\![e]\!]$

Protocol 38 computes μ following the logic in Protocol 26 and the rest of the computations are straightforward.[16] We note that Line 21 in Algorithm 23 decrements $[\![c]\!]$ for a data holder only if both the metrics meet the given thresholds. The data holders secret-share an array of thresholds.

10.3.4 Empirical Evaluation

We implemented our framework in MP-SPDZ [110] and leverage the MPC primitives available therein. We consider the number of iterations in the generate step of the Private-PGM as the hyperparameter to be tuned. For simplicity we consider $\mathbf{H} = \{10, 15, 25, 30\}$.

Utility evaluation. We evaluate the framework in Algorithm 23 on the genomic data for

[16] Our current implementation does not yet optimize to leverage the precomputed marginals for computation of error in the pipeline; this relatively straightforward step can be done when building a software for this pipeline. Our focus here is development of individual protocols.

leukemia[17]. Table 10.1 reports the quality of synthetic data that is published on line 29 in Algorithm 23 when compared with the real data. While throughout the execution of Algorithm 23, we use MPC protocols to compute the workload error and to train and test logistic regression models for π_{EVAL} on line 14, in Table 10.1 we report the workload error (WLE) as defined in Section 10.3.3 and the accuracy of a logistic regression and a decision tree model computed in the clear, i.e. without MPC. This allows to assess the fidelity and the utility of the generated synthetic data without the influence of MPC.

For this experiment, we split the leukemia dataset into train and test datasets. We split the train dataset further into 2 equal parts, simulating a scenario with 2 data holders $|\mathbf{C}| = 2$, where each data holder holds a part of the train dataset. Algorithm 23 takes the split train datasets as input, while the test dataset is kept aside for evaluation. We run our framework with $|S| = 3$. Once the synthetic data is published at the end of Algorithm 23, we train a logistic regression model and a decision tree on the real train data and the generated synthetic data independently. We then evaluate each model on the test data, and report the absolute difference between accuracy and F1 score in Table 10.1. To demonstrate the effectiveness of pre-processing across data silos, we conduct the experiment where data holders perform binning (pre-processing) independently on their respective portions of the split data and then secret share, rather than pre-processing the combined dataset (row 1 of Table 10.1). Our experiments show that this dataset benefits from combined pre-processing for ML tasks, specifically the ML model and metric for which it was tuned (LR). The evaluation suggests that we need better SDG algorithms for genomic data. We further note that we considered a relatively simple format of genomic data.

Performance evaluation. We run our framework for different threat models [3, 4, 1] and report the benchmarking runtimes for running the MPC protocols for leukemia data in Table 10.2. We run for the similar setup as mentioned above, and therefore the input data size $N = 945$. For 3PC passive, the parts of the framework with MPC protocol run for a total of ~ 5.1

[17]For the results in this section, we include the "generate" step in-the-clear.

hours to generate synthetic data with parameter tuning (3 search spaces, 5 folds for every search with protocols in Table 10.2[18])Table 10.2 reports performance evalaution for different threat models. The observed runtimes are in-line with the literature and demonstrate that an MPC based framework is feasible for end-to-end synthetic data generation where privacy is of utmost importance compared to the computational time required to generate valuable synthetic data. Future work will focus on optimizing these protocols and proposing new optimized MPC protocols for the proposed framework. In our experiments, we observed that if the data is binned without computing means required for debinning, then only the test data in each fold needs to be binned (example for $|d| = 5$, the runtime of π_{BIN} improves from 0.40 to 0.34 on local compute). This improves the performance of individual protocols and may improve the overall framework's performance, particularly for high-dimensional data. For π_{LR}, we report performance with 150 and 300 epochs illustrating that the MPC protocol scales linearly(applicable to all MPC protocols and threat models).

Table 10.1: **Quality evaluation of the generated synthetic genomic data**. We run the framework with $|S| = 3$. We set $k = 5$ and tune the number of iterations of the Private-PGM model with $\epsilon = 5, \delta = 1e-5$. Once the final synthetic data is published, we compute WLE between synthetic and combined real data and report difference of machine learning utility scores (accuracy and F1 scores with logistic regression ΔLR and decision tree ΔDT) between synthetic and combined real data (note: this evaluation is not part of the framework). We report averages taken over 3 runs.

	WLE	|ΔLR Accuracy|	|ΔLR F1|	|ΔDT Accuracy|	|ΔDT F1|
Local pre-processing	**0.015**	0.026	0.141	0.346	0.170
Ours	0.019	**0.003**	**0.135**	**0.079**	**0.125**

[18]We estimate it to be at max 6 hours including other parts of the framework. Faster hardware and parallelization can further reduce the runtime.

Table 10.2: **Performance evaluation**. We run experiments with $|S| = 2, 3, 4$ for different threat models (with corruption threshold of 1) with $\epsilon = 5, \delta = 1e-5$ and $N = 945$. We report runtimes in seconds and total communication cost (Comm.) in GB for one run of the major MPC protocols of our framework. The runtimes include both online and offline phases and are averaged over 5 runs. The experiments were run on TACC Frontera nodes with CPUs.

	Protocol	2PC Passive		3PC passive		3PC active		4PC active	
		Time(s)	Comm(GB)	Time(s)	Comm(GB)	Time(s)	Comm(GB)	Time(s)	Comm(GB)
π_{RW}	π_{BIN}	15669.36	2573.41	105.73	5.63	604.06	25.33	18219.53	87.77
	$\pi_{\mathsf{INV-BIN}}$	348.08	12.37	55.39	4.06	524.11	31.57	84.09	12.01
π_{SDG}	$\pi_{\mathsf{NOISY-MARG}}$	2644.13	92.51	285.58	45.48	2420.30	361.61	814.23	123.42
	π_{BP}	95.21	16.18	9.27	0.82	100.83	6.69	27.64	2.20
	π_{LOSS}	4.85	0.12	0.51	0.08	3.51	0.63	1.18	0.22
	π_{GEN}	12066.34	4242.45	201.77	77.11	1527.31	377.27	573.53	172.21
π_{EVAL}	π_{LR} (150 ep)	946.54	34.81	62.68	14.58	610.02	109.97	132.66	36.12
	π_{LR} (300 ep)	1372.73	66.64	115.81	28.00	1176.55	223.79	272.75	76.93
	π_{WLE}	1903.74	72.38	243.75	47.62	2043.83	371.57	524.83	127.41

10.4 Summary

We introduced a general framework, `E2E-CaPS`, that leverages DP-in-MPC to publish the desired quality of DP synthetic data from multiple data holders. The framework executes the full SDG pipeline, including pre-processing, hyperparameter tuning via cross-validation.

To achieve this, `E2E-CaPS` extends the `CaPS` framework by incorporating privacy-preserving hyperparameter tuning, secure pre-processing, and secure evaluation – all while making efficient use of allotted fixed privacy budget for the entire process. The idea is to not publish any computations, except for the final output.

We evaluate `E2E-CaPS` by generating synthetic genomic data for leukemia, and we develop the necessary MPC protocols to instantiate the framework for this use case. Specifically, we implement MPC protocols for the end-to-end SDG pipeline using Private-PGM (a measure-generate SDG approach) for leukemia tabular data. This includes custom protocols for secure

pre-processing, optimized implementations of the measurement protocols from Chapter 9, and a MPC protocol for the generate step. Our evaluation demonstrates the feasibility of the approach in practice. Nonetheless, generating high-quality synthetic genomic data remains a challenging task, even in non-private settings, and future work is needed to further optimize the underlying MPC protocols.

Chapter 11

CONCLUSION

"The power to question is the basis of all human progress."

Indira Gandhi

The meteoric rise of Artificial Intelligence (AI) has dramatically transformed how automated systems and data-driven algorithms have become central to decision-making processes across healthcare, finance, governance, and digital services. Much of what fuels these modern AI systems is deeply personal – our online searches, shopping behavior, health records, financial transactions, and location histories.

AI models are often data-hungry – the more data they have, the better they learn, and the more accurate and effective they become. But the data they rely on is seldom available as an aggregated whole in a single, centralized location (i.e., with a trusted central entity). Instead, it is typically distributed across multiple sources in different ways – referred to as data holders – such as hospitals, banks, or personal devices, with each holding only a portion of the overall dataset.

This dissertation advances the state-of-the-art of privacy-preserving AI systems in distributed data settings, often referred to as federated settings.

When training AI models in federated settings, one must address (a) input privacy: neither other data holders nor any external entity, including the AI model developer, should be able to learn anything about a data holder's private input to the AI pipeline; and (b) output privacy: outputs of a trained AI model or computations done on the distributed data should not reveal sensitive information about a data holder's training data.

Among various Privacy Enhancing Technologies (PETs) proposed in the literature, Dif-

ferential Privacy (DP) has emerged as the most widely adopted method for output privacy. In distributed settings, Federated Learning (FL) combined with DP is often used in attempts to offer some form of both input and output privacy. However, as discussed in Chapter 2, these approaches have weaknesses, particularly in relation to the accuracy of the AI models, data distribution constraints, and the reliance on a trusted aggregator.

To overcome these limitations, this dissertation proposed the DP-in-MPC paradigm combining Secure Multiparty Computation (MPC) for input privacy with DP for output privacy. This paradigm replaces the trusted aggregator of the centralized paradigm with an MPC-as-a-Service infrastructure enabling: (a) privacy-preserving model training and data analysis with stronger input and output privacy guarantees; (b) utility comparable to models trained in centralized settings; (c) elimination of the need for private data disclosure or centralized trust; (d) support for arbitrary data partitions, including horizontal, vertical, and mixed; and (e) scalability to a large number of data holders. DP-in-MPC is the central idea of this thesis, and while it enables strong privacy guarantees across a wide range of tasks, it also comes with limitations, most notably increased computational cost. A significant effort was therefore spent on the selection and development of MPC-friendly AI techniques, the optimization of MPC protocols, and careful consideration of which computation tasks can be done outside of MPC without loss of privacy.

We leveraged the DP-in-MPC paradigm to address three core research questions outlined in Chapter 1, organizing the dissertation into three corresponding parts. In the first part we addressed the privacy-preserving training of AI models in distributed settings. In particular, we developed MPC protocols for training linear models with output perturbation in Chapter 3, demonstrating how input and output privacy can be preserved without significantly compromising utility. This approach won the iDASH 2021 Genome Privacy competition, demonstrating its effectiveness on real-world health data.

The second part explored the intersection of privacy and algorithmic fairness.After recalling core fairness concepts in Chapter 4, we proposed novel MPC-based solutions across Chapters 5 – 7 to audit and mitigate bias in AI systems. These include protocols for privacy-

preserving fairness auditing (Chapter 5), group fairness in cross-device FL (Chapter 6), and fair item ranking while preserving user data in cross-device FL (Chapter 7).

Recognizing the challenges and benefits of synthetic data, as discussed in Chapter 8, the third part of the thesis addressed the training of synthetic data generation (SDG) models in distributed settings. Chapters 9 and 10 presented frameworks and developed novel MPC protocols for the select–measure–generate paradigm, enabling collaborative and end-to-end privacy-preserving SDG pipelines.

To demonstrate their utility and scalability, all proposed DP-in-MPC protocols were implemented and thoroughly evaluated in an empirical manner across a variety of datasets and real-world use cases – including privacy-preserving emotion recognition, book recommendations, and cancer diagnosis based on genomics data.

Opportunities, Challenges, and Future Work

The contributions in this dissertation represent significant progress in privacy-preserving AI, while also illuminating promising directions for future research and shedding light on remaining challenges in the field.

A long-term perspective for deploying the DP-in-MPC paradigm lies in enabling large-scale, privacy-preserving collaboration across organizations, regions, and even nations. In such scenarios, institutions could contribute sensitive data such as healthcare, financial, or demographic records using DP-in-MPC to collectively train models or generate synthetic datasets, without ever exposing private data. Beyond cross-silo collaborations, this paradigm can be extended to cross-device environments involving millions of individuals. For example, patients could contribute their private data toward generating a shared synthetic dataset or enabling distributed AI tasks, with strong cryptographic and DP guarantees. This would pave the way for a practical implementation of "AI-as-a-Service" or "SDG-as-a-Service", where sensitive individual data never appears in-the-clear to any entity.

The integration of DP-in-MPC with decentralized technologies such as blockchains opens up transformative possibilities for privacy-preserving data marketplaces [336]. Such systems

would allow untrusted parties to share, audit, and monetize data in a controlled and transparent manner. By leveraging verifiable smart contracts and incentive mechanisms, these platforms could ensure accountability, enforce privacy policies, and enable collaborative analytics across sectors like healthcare, finance, and public research.

These future applications represent not just technical possibilities for secure data collaboration but potentially transformative shifts in how society approaches the fundamental tension between utilizing the potential of data and privacy protection. However, realizing this vision requires addressing several critical limitations of current approaches.

A key limitation of DP-in-MPC is its computational overhead, particularly the runtime and communication cost associated with MPC protocols. While DP-in-MPC enables strong privacy guarantees without centralizing data, it demands significantly more resources than centralized computation or in some cases even FL. This restricts its use for time-sensitive or resource-constrained settings. Accelerating MPC protocols (and MPC primitives) using hardware accelerators such as GPUs represents a promising direction for improving performance and scalability[1]. Another direction of future research involves developing MPC protocols to adapt DP-in-MPC for different use-cases and algorithms.

A broader direction for future work is to develop efficient MPC protocols for a variety of DP mechanisms, potentially assembling them into a reusable suite within the DP-in-MPC framework. This includes exploring discrete variants of DP mechanisms and analyzing their robustness against new classes of attacks.

The discriminative AI training protocols developed in this dissertation focus primarily on linear models with output perturbation. A natural extension would be to adapt DP-in-MPC to support more complex training mechanisms such as DP-SGD building over [154] and teacher-student approaches like PATE. These methods offer different privacy-utility tradeoffs and could expand DP-in-MPC to more complex model architectures, including deep neural networks. Moreover, as discussed in Chapter 10, the training pipelines could benefit from

[1]There has been progress in this direction [337, 338, 339], but not all frameworks support all MPC primitives and adversarial models.

custom MPC protocols for pre-processing to improve the model utility.

We focused on privacy-preserving auditing in the context of fairness, but this approach can be extended to other forms of auditing, such as membership inference attacks (MIA) for privacy auditing, model quality evaluation, and more. In the context of fairness, expanding DP-in-MPC to other fairness notions such as counterfactual fairness, casual fairness and others represents an important avenue for future research.

The interplay between privacy and fairness remains a complex and evolving area of research, as DP can have disparate impacts on different demographic groups, potentially exacerbating algorithmic bias rather than mitigating it. An interesting direction is to study this interplay systematically and develop mitigation strategies spanning from statistical techniques to more complex reinforcement learning-based approaches. Developing frameworks that can optimally balance utility, privacy, fairness, and computational cost continues to be a significant open challenge.

SDG is still an emerging field with numerous opportunities for advancement. Our current framework addresses tabular data generation, but extending it to other modalities (including images, text, and time-series data) represents a promising direction, particularly with the rise of Generative AI (GenAI). Open questions remain in handling multimodal data, longitudinal records, data imputation, and ensuring the robustness (such as data and concept shifts) while preserving privacy and utility of SDG. Adapting Large Language Models (LLMs) and diffusion models to work within our DP-in-MPC framework could significantly enhance the quality and utility of the generated data for specific cases while maintaining privacy guarantees. Additionally, developing novel metrics to assess fairness, realism, and task relevance of synthetic data is underexplored.

While this dissertation addressed two pillars of trustworthy AI – privacy and fairness – future privacy-preserving systems must be part of a broader vision of trustworthy AI, one that includes interpretability, robustness, safety, and accountability. For instance, while `PrivFairFL` allows for fair and private decision-making, it is still based on black-box AI models that may lack transparency and are vulnerable to adversarial inputs. Such limitations

must be considered before deploying these systems in environments where model decisions directly impact individuals.

Finally, defining privacy alongside other pillars of trustworthy AI, especially in the era of GenAI such as LLMs and agentic AI, is an increasingly important interdisciplinary challenge. As foundational models are being adopted by sectors like healthcare, it remains an open question whether DP-in-MPC can be effectively applied, is necessary, or must be redefined in these contexts. With exponential trends in AI, privacy will always remain a concern, and DP-in-MPC offers a strong option for preserving it. However, the contextual nature of privacy in such systems may require a fundamental rethinking of existing approaches. Addressing these challenges will demand continued research at the intersection of computer science, ethics, law, and the social sciences.

In Conclusion

This dissertation advances the field of privacy-preserving AI in federated settings by proposing the DP-in-MPC paradigm, which combines Secure Multiparty Computation (MPC) and Differential Privacy (DP) to ensure strong input and output privacy guarantees. We demonstrated the effectiveness of this approach across three applications: training AI models in cross-silo environments, promoting algorithmic fairness in cross-device setups, and generating high-utility synthetic data in federated environments, all while providing privacy guarantees. To this end, we developed several MPC protocols and empirically evaluated them across real-world datasets. While these contributions mark a significant step toward enabling privacy-preserving AI, building a comprehensive, responsible and trustworthy AI – one that integrates fairness, privacy, robustness, interpretability, and efficiency – remains an open and vital direction for future research.

BIBLIOGRAPHY

[1] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *Annual International Cryptology Conference*, pages 769–798. Springer, 2018.

[2] Ivan. Damgård, Daniel Escudero, Tore. Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning. In *IEEE Symposium on Security and Privacy (SP)*, pages 1102–1120, 2019.

[3] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 805–817, 2016.

[4] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In *USENIX 2021*, pages 2183–2200, 2021.

[5] Lucas Rosenblatt, Xiaoyan Liu, Samira Pouyanfar, Eduardo de Leon, Anuj Desai, and Joshua Allen. Differentially private synthetic data: Applied evaluations and enhancements. *arXiv preprint arXiv:2011.05537*, 2020.

[6] Office Connecticut General Assembly. The Evolution of Privacy: A Look at the Past, Present, and Future. *An Informational Brief Prepared for Members of the Connecticut General Assembly by the Office Of Legislative Research*. `https://www.cga.ct.gov/PS98/rpt%5Colr%5Chtm/98-R-1455.htm`, 1998.

[7] Katia Hetter. Doctor explains how artificial intelligence is already being deployed in medical care. *CNN*, March 2025.

[8] Article PYMNTS News. Apple reportedly developing ai agent 'doctors' in latest health push. *PYMNTS*, March 2025.

[9] Rohit S Vilhekar and Alka Rawekar. Artificial intelligence in genetics. *Cureus*, 16(1), 2024.

[10] Sameer Quazi. Artificial intelligence and machine learning in precision and genomic medicine. *Medical Oncology*, 39(8):120, 2022.

[11] Kevin B Johnson, Wei-Qi Wei, Dilhan Weeraratne, Mark E Frisse, Karl Misulis, Kyu Rhee, Juan Zhao, and Jane L Snowdon. Precision medicine, ai, and the future of personalized health care. *Clinical and translational science*, 14(1):86–93, 2021.

[12] Wilhelmina Afua Addy, Adeola Olusola Ajayi-Nifise, Binaebi Gloria Bello, Sunday Tubokirifuruar Tula, Olubusola Odeyemi, and Titilola Falaiye. Ai in credit scoring: A comprehensive review of models and predictive analytics. *Global Journal of Engineering and Technology Advances*, 18(02):118–129, 2024.

[13] K Tulsi, Arpan Dutta, Navneet Singh, and Deepansh Jain. Transforming financial services: The impact of ai on jp morgan chase's operational efficiency and decision-making. *International Journal of Scientific Research & Engineering Trends*, 10(1):207–2013, 2024.

[14] Paul Tierno. Artificial intelligence and machine learning in financial services. Technical Report R47997, Congressional Research Service, April 2024.

[15] Richard A Berk. Artificial intelligence, predictive policing, and risk assessment for law enforcement. *Annual Review of Criminology*, 4(1):209–237, 2021.

[16] Shaina Raza and Chen Ding. News recommender system: a review of recent progress, challenges, and opportunities. *Artificial Intelligence Review*, pages 1–52, 2022.

[17] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)*, 54(7):1–38, 2021.

[18] Qian Zhang, Jie Lu, and Yaochu Jin. Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, 7(1):439–457, 2021.

[19] Michael Palmer. Data is the new oil. `https://ana.blogs.com/maestros/2006/11/data_is_the_new.html`, November 2006.

[20] George Firican. How data is (and isn't) like oil. `https://tdwi.org/articles/2019/04/22/data-all-how-data-is-like-oil.aspx`, April 2019.

[21] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *California Law Review*, 104:671, 2016.

[22] Heather Chen and Kathleen Magramo. Finance worker pays out $25 million after video call with deepfake 'Chief Financial Officer'. *CNN*, February 2024.

[23] U.S. Department of Homeland Security. Increasing threat of deepfake identities. Technical report, U.S. Department of Homeland Security.

[24] Kayte Spector-Bagdady. 23andme is bankrupt. here's what that means for your genetic data. *The Conversation*, April 2025.

[25] Bonnie Kaplan. Selling health data: de-identification, privacy, and speech. *Cambridge Quarterly of Healthcare Ethics*, 24(3):256–271, 2015.

[26] Akshat Pandey and Aylin Caliskan. Disparate impact of artificial intelligence bias in ridehailing economy's price discrimination algorithms. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 822–833, 2021.

[27] Kashmir. Hill. The secretive company that might end privacy as we know it. *The New York Times*, Jan 18, 2020.

[28] Lena Kempe. The price of emotion: Privacy, Manipulation, and Bias in Emotional AI. *Internet Law & Cybersecurity*, September 2024.

[29] Till Speicher, Muhammad Ali, Giridhari Venkatadri, Filipe Nunes Ribeiro, George Arvanitakis, Fabrício Benevenuto, Krishna P Gummadi, Patrick Loiseau, and Alan Mislove. Potential for discrimination in online targeted advertising. In *Conference on fairness, accountability and transparency*, pages 5–19. PMLR, 2018.

[30] Muhammad Ali, Piotr Sapiezynski, Miranda Bogen, Aleksandra Korolova, Alan Mislove, and Aaron Rieke. Discrimination through optimization: How Facebook's ad delivery can lead to biased outcomes. In *Proceedings of the ACM on Human-Computer Interaction*, volume 3, pages 1–30, 2019.

[31] Latanya Sweeney. Discrimination in online ad delivery: Google ads, black names and white names, racial discrimination, and click advertising. *Queue*, 11(3):10–29, 2013.

[32] Neil Vigdor. Apple card investigated after gender discrimination complaints. *The New York Times*, Nov 10, 2019.

[33] Sheridan Wall and Hilke Schellmann. LinkedIn's job-matching AI was biased. The company's solution? More AI. *MIT Technology Review*, June 23, 2021.

[34] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the Conference on Fairness, Accountability and Transparency*, pages 77–91. PMLR, 2018.

[35] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, May 23, 2016.

[36] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9(1):3–3, 2016.

[37] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. 2019. fairmlbook.org.

[38] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

[39] Carole Cadwalladr and L Glendinning. Exposing Cambridge Analytica:'It's been exhausting, exhilarating, and slightly terrifying'. *The Guardian*, 28, 2018.

[40] Michael Kan. OpenAI: Sorry, ChatGPT bug leaked payment info to other users. *PCMag*, March 2023.

[41] Siladitya Ray. Samsung bans ChatGPT among employees after sensitive code leak. *Forbes*, May 2023.

[42] General Data Protection Regulation (GDPR), 2016.

[43] California Consumer Privacy Act, 2020.

[44] Aloni Cohen and Kobbi Nissim. Towards formalizing the gdpr's notion of singling out. *Proceedings of the National Academy of Sciences*, 117(15):8344–8352, 2020.

[45] AIM-AHEAD, NIH. AIM-AHEAD Federated Network Program. 2024.

[46] U.S. Census Bureau. What are synthetic data? `https://www.census.gov/library/fact-sheets/2021/what-are-synthetic-data.html`, 2021.

[47] L. Kalman. New European data privacy and cyber security laws – one year later. *Communications of the ACM*, 62:38–39, 2019.

[48] Alessandro Acquisti, Laura Brandimarte, and Jeff Hancock. How privacy's past may shape its future. *Science*, 375(6578):270–272, 2022.

[49] Edmund T Hall and Edward T Hall. The hidden dimension. *Doubleday & Company, Inc.*, 1966.

[50] Irwin Altman. The environment and social behavior: privacy, personal space, territory, and crowding. *ERIC*, 1975.

[51] Samuel D. Warren and Louis D. Brandeis. The right to privacy. *Harvard Law Review*, 4(5):193–220, 1890.

[52] Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[53] Helen Nissenbaum. Privacy in context: Technology, policy, and the integrity of social life. *Journal of Information Policy*, 1:149–151, 2011.

[54] Lavender Yao Jiang, Xujin Chris Liu, Nima Pour Nejatian, Mustafa Nasir-Moin, Duo Wang, Anas Abidin, Kevin Eaton, Howard Antony Riina, Ilya Laufer, Paawan Punjabi, et al. Health system-scale language models are all-purpose prediction engines. *Nature*, 619(7969):357–362, 2023.

[55] Latanya Sweeney, Akua Abu, and Julia Winn. Identifying participants in the personal genome project by name (a re-identification experiment). *arXiv preprint arXiv:1304.7605*, 2013.

[56] Latanya Sweeney. Only you, your doctor, and many others may know. *Technology Science*, 2015092903(9):29, 2015.

[57] C Christine Porter et al. De-identified data and third party data mining: The risk of re-identification of personal information. *Seattle: Shidler Journal of Law, Commerce and Technology*, 2008.

[58] Jelle Vos, Sikha Pentyala, Steven Golob, Ricardo Maia, Dean Kelley, Zekeriya Erkin, Martine De Cock, and Anderson Nascimento. Privacy-preserving membership queries for federated anomaly detection. *Proceedings on Privacy Enhancing Technologies (PoPETS)*, 3:186–201, 2024.

[59] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[60] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[61] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I know what you trained last summer: A survey on stealing machine learning models and defences. *ACM Computing Surveys*, 55(14s):1–41, 2023.

[62] Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 1–9, 2019.

[63] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021.

[64] Oded Goldreich. Foundations of cryptography: Basic applications vol 2. 2, 2001.

[65] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[66] A Cuthbertson. Amazon admits employees listen to alexa conversations. *The Independent*, 2019.

[67] Stuart A. Thompson and Charlie Warzel. Twelve million phones, one dataset, zero privacy. *NY Times*, 2019.

[68] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[69] Sheng Liu, Zihan Wang, Yuxiao Chen, and Qi Lei. Data reconstruction attacks and defenses: A systematic evaluation. *arXiv preprint arXiv:2402.09478*, 2024.

[70] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1287–1305. IEEE, 2024.

[71] Cynthia Dwork. Differential privacy: A survey of results. In *Proc. of 5th International Conference on Theory and Applications of Models of Computation*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008.

[72] O. Catrina and S. De Hoogh. Secure multiparty linear programming using fixed-point arithmetic. In *European Symposium on Research in Computer Security*, pages 134–150. Springer, 2010.

[73] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[74] Matthew Jagielski, Michael Kearns, Jieming Mao, Alina Oprea, Aaron Roth, Saeed Sharifi-Malvajerdi, and Jonathan Ullman. Differentially private fair learning. In *International Conference on Machine Learning*, pages 3000–3008. PMLR, 2019.

[75] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[76] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[77] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[78] Joshua C Zhao, Saurabh Bagchi, Salman Avestimehr, Kevin S Chan, Somali Chaterji, Dimitris Dimitriadis, Jiacheng Li, Ninghui Li, Arash Nourian, and Holger R Roth. Federated learning privacy: Attacks, defenses, applications, and policy landscape-a survey. *arXiv preprint arXiv:2405.03636*, 2024.

[79] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning. *arXiv preprint arXiv:2211.12814*, 2022.

[80] Zilong Zhao, Han Wu, Aad Van Moorsel, and Lydia Y Chen. VT-GAN: Cooperative tabular data synthesis using vertical federated learning. *arXiv preprint arXiv:2302.01706*, 2023.

[81] Dario Pasquini, Mathilde Raynal, and Carmela Troncoso. On the (in) security of peer-to-peer decentralized machine learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 418–436. IEEE Computer Society, 2023.

[82] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.

[83] Mohamed Alloghani, Mohammed M Alani, Dhiya Al-Jumeily, Thar Baker, Jamila Mustafina, Abir Hussain, and Ahmed J Aljaaf. A systematic review on the status and progress of homomorphic encryption technologies. *Journal of Information Security and Applications*, 48:102362, 2019.

[84] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/Ispa*, volume 1, pages 57–64. IEEE, 2015.

[85] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[86] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete Gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pages 5201–5212. PMLR, 2021.

[87] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[88] Tanushree Shenwai. Google AI implements machine learning model that employs federated learning with differential privacy guarantees. MarkTechPost, 2022.

[89] An Ji, Bortik Bandyopadhyay, Congzheng Song, Natarajan Krishnaswami, Prabal Vashisht, Rigel Smiroldo, Isabel Litton, Sayantan Mahinder, Mona Chitnis, and Andrew W Hill. Private federated learning in real world application – a case study, 2025.

[90] Extracting value from siloed healthcare data using federated learning with Azure Machine Learning. December 2022.

[91] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[92] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8485–8493, 2022.

[93] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.

[94] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[95] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International conference on machine learning*, pages 5650–5659. PMLR, 2018.

[96] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2429–2443, New York, NY, USA, 2022. Association for Computing Machinery.

[97] Yichuan Shi, Olivera Kotevska, Viktor Reshniak, Abhishek Singh, and Ramesh Raskar. Dealing doubt: Unveiling threat models in gradient inversion attacks under federated learning, a survey and taxonomy. *arXiv preprint arXiv:2405.10376*, 2024.

[98] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[99] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

[100] Li Bai, Haibo Hu, Qingqing Ye, Haoyang Li, Leixia Wang, and Jianliang Xu. Membership inference attacks and defenses in federated learning: A survey. *ACM Computing Surveys*, 57(4):1–35, 2024.

[101] Shiqiang Wang, Nathalie Baracaldo Angel, Olivia Choudhury, Gauri Joshi, Peter Richtárik, Praneeth Vepakomma, and Han Yu. Federated learning: Recent advances and new challenges. In *Annual Conference on Neural Information Processing Systems*, 2022.

[102] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[103] Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201, 2023.

[104] Ronald Cramer, Ivan Damgard, and Jesper Nielsen. *Secure Multiparty Computation Secret Sharing*. Cambridge University Press Print, New York, 2015.

[105] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[106] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 316–334. Springer, 2000.

[107] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2010.

[108] Anders Dalskov, Daniel Escudero, and Marcel Keller. Secure evaluation of quantized neural networks. arXiv preprint arXiv:1910.12435, 2019.

[109] O. Catrina and S. De Hoogh. Improved primitives for secure multiparty integer computation. In *International Conference on Security and Cryptography for Networks*, pages 182–199. Springer, 2010.

[110] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1575–1590, 2020.

[111] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[112] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. *Technical Report, SRI International*, 1998.

[113] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[114] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2006.

[115] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273, 2008.

[116] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1), 2010.

[117] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9, 2014.

[118] Joseph P. Near and Chiké Abuah. Programming differential privacy. 1, 2021.

[119] Ethan Cowan, Michael Shoemate, and Mayana Pereira. Hands-on differential privacy. *O'Reilly Media, Inc.*, 2024.

[120] Krishnamurty Muralidhar and Steven Ruggles. Escalation of commitment: A case study of the united states census bureau efforts to implement differential privacy for the 2020 decennial census. In *International Conference on Privacy in Statistical Databases*, pages 393–402. Springer, 2024.

[121] Mike Schneider. Census Bureau valiantly conducted 2020 census, but privacy method degraded quality, report says. *AP News*, 2023.

[122] Steven Golob, Sikha Pentyala, Anuar Maratkhan, and Martine De Cock. Privacy vulnerabilities in marginals-based synthetic data. *IEEE Secure and Trustworthy Machine Learning Conference (SaTML)*, 2025.

[123] Jiankai Jin, Eleanor McMurtry, Benjamin IP Rubinstein, and Olga Ohrimenko. Are we there yet? Timing and floating-point attacks on differential privacy systems. In *2022 IEEE Symposium on security and privacy (SP)*, pages 473–488. IEEE, 2022.

[124] Tao Zhang, Tianqing Zhu, Renping Liu, and Wanlei Zhou. Correlated data in differential privacy: definition and analysis. *Concurrency and Computation: Practice and Experience*, 34(16):e6015, 2022.

[125] Sara Saeidian, Tobias J Oechtering, and Mikael Skoglund. Evaluating differential privacy on correlated datasets using pointwise maximal leakage. In *Annual Privacy Forum*, pages 73–86. Springer, 2024.

[126] Simson L Garfinkel, John M Abowd, and Sarah Powazek. Issues encountered deploying differential privacy. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pages 133–137, 2018.

[127] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. LDP-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[128] Florian Hartmann and Peter Kairouz. Distributed differential privacy for federated learning, 2023.

[129] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661, 2012.

[130] Joshua Zhao, Saurabh Bagchi, Salman Avestimehr, Kevin Chan, Somali Chaterji, Dimitris Dimitriadis, Jiacheng Li, Ninghui Li, Arash Nourian, and Holger Roth. The federation strikes back: A survey of federated learning privacy attacks, defenses, applications, and policy landscape. *ACM Computing Surveys*, 57(9):1–37, 2025.

[131] Genomics PPFL Platform 2025 Red Teaming Event. *NIST*, 2025.

[132] Kristin Lauter. Private AI: Machine learning on encrypted data. In *Recent advances in industrial and applied mathematics*, pages 97–113. Springer, 2022.

[133] Andrés F Castro Torres and Aliakbar Akbaritabar. The use of linear models in quantitative research. *Quantitative Science Studies*, 5(2):426–446, 2024.

[134] Christopher M Bishop. Machine learning. *Mach. Learn*, 128(10.1117):1–2819119, 2006.

[135] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2008.

[136] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506, 2020.

[137] Jinhyun So, Chaoyang He, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E Ali, Basak Guler, and Salman Avestimehr. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.

[138] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Technologies*, 2017(4):345 – 364, 2017.

[139] Anisha Agarwal, Rafael Dowsley, Nicholas D. McKinney, Dongrui Wu, Chin-Teng Lin, Martine De Cock, and Anderson C. A. Nascimento. Protecting privacy of users in brain-computer interface applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(8):1546–1555, 2019.

[140] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 36–54, 2000.

[141] Sebastiaan de Hoogh, Berry Schoenmakers, Ping Chen, and Harm op den Akker. Practical secure decision tree learning in a teletreatment application. In *International Conference on Financial Cryptography and Data Security*, pages 179–194. Springer, 2014.

[142] Mark Abspoel, Daniel Escudero, and Nicolaj Volgushev. Secure training of decision trees with continuous attributes. *Proceedings on Privacy Enhancing Technologies*, (1):167–187, 2021.

[143] Samuel Adams, Chaitali Choudhary, Martine De Cock, Rafael Dowsley, David Melanson, Anderson CA Nascimento, Davis Railsback, and Jianwei Shen. Privacy-preserving training of tree ensembles over continuous data. *Proceedings on Privacy Enhancing Technologies*, (2):205–226, 2022.

[144] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, 2017.

[145] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 3:26–49, 2019.

[146] Chuan Guo, Awni Hannun, Brian Knott, Laurens van der Maaten, Mark Tygert, and Ruiyu Zhu. Secure multiparty computations in floating-point arithmetic. *Information and Inference: A Journal of the IMA*, 11(1):103–135, 2022.

[147] Martine De Cock, Rafael Dowsley, Anderson C. A. Nascimento, Davis Railsback, Jianwei Shen, and Ariel Todoki. High performance logistic regression for privacy-preserving genome analysis. *BMC Medical Genomics*, 14(23), 2021.

[148] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[149] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium*, pages 601–618, 2016.

[150] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 587–601, 2017.

[151] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX*, pages 267–284, 2019.

[152] Junxu Liu, Jian Lou, Li Xiong, Jinfei Liu, and Xiaofeng Meng. Cross-silo federated learning with record-level personalized differential privacy. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 303–317, 2024.

[153] Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. DP-cryptography: marrying differential privacy and cryptography in emerging applications. *Communications of the ACM*, 64(2):84–93, 2021.

[154] Sankha Das, Sayak Ray Chowdhury, Nishanth Chandran, Divya Gupta, Satya Lokam, and Rahul Sharma. Communication efficient secure and private multi-party deep learning. *Proceedings on Privacy Enhancing Technologies*, 2025.

[155] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private heavy hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2361–2377, 2021.

[156] Samuel Maddock, Graham Cormode, and Carsten Maple. FLAIM: AIM-based synthetic data generation in the federated setting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2165–2176, 2024.

[157] Abbas Acar, Z Berkay Celik, Hidayet Aksu, A Selcuk Uluagac, and Patrick McDaniel. Achieving secure and differentially private computations in multiparty settings. In *IEEE Symposium on Privacy-Aware Computing*, pages 49–59, 2017.

[158] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: privacy-preserving empirical risk minimization. In *Advances in Neural Information Processing Systems 31*, pages 6346–6357, 2018.

[159] Manas A Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems 23*, pages 1876–1884, 2010.

[160] Razane Tajeddine, Joonas Jälkö, Samuel Kaski, and Antti Honkela. Privacy-preserving data sharing on vertically partitioned data. *arXiv preprint arXiv:2010.09293*, 2020.

[161] Melissa Chase, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, and Peter Rindal. Private collaborative neural network learning. *Cryptology ePrint Archive*, 2017.

[162] David Byrd and Antigoni Polychroniadou. Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–9, 2020.

[163] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.

[164] Xiaolan Gu, Ming Li, and Li Xiong. PRECAD: Privacy-preserving and robust federated learning via crypto-aided differential privacy. *arXiv preprint arXiv:2110.11578*, 2021.

[165] Christopher A Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. CaPC learning: Confidential and private collaborative learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[166] Jiabo Wang, Elmo Xuyun Huang, Pu Duan, Huaxiong Wang, and Kwok-Yan Lam. PSA: private set alignment for secure and collaborative analytics on large-scale data. *arXiv preprint arXiv:2410.04746*, 2024.

[167] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[168] Eusebio Gómez Sánchez-Manzano, Miguel Angel Gomez-Villegas, and Juan-Miguel Marín-Diazaraque. A matrix variate generalization of the power exponential family of distributions. *Communications in Statistics-Theory and Methods*, 31(12):2167–2182, 2002.

[169] George E.P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[170] Marcel Keller and Ke Sun. Secure quantized training for deep learning. In *International Conference on Machine Learning*, pages 10912–10938. PMLR, 2022.

[171] Ahsan Huda, Adam Castaño, Anindita Niyogi, Jennifer Schumacher, Michelle Stewart, Marianna Bruno, Mo Hu, Faraz S Ahmad, Rahul C Deo, and Sanjiv J Shah. A machine learning model for identifying patients at risk for wild-type transthyretin amyloid cardiomyopathy. *Nature communications*, 12(1):1–12, 2021.

[172] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In *Annual International Cryptology Conference*, pages 823–852. Springer, 2020.

[173] Payman Mohassel, Peter Rindal, and Mike Rosulek. Fast database joins and PSI for secret shared data. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1271–1287, 2020.

[174] Alexandre Bailly, Corentin Blanc, Élie Francis, Thierry Guillotin, Fadi Jamal, Béchara Wakim, and Pascal Roy. Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models. *Computer Methods and Programs in Biomedicine*, 213:106504, 2022.

[175] Vipul Goyal, Hanjun Li, Rafail Ostrovsky, Antigoni Polychroniadou, and Yifan Song. ATLAS: Efficient and scalable mpc in the honest majority setting. In *Annual International Cryptology Conference*, pages 244–274. Springer, 2021.

[176] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In *Annual International Cryptology Conference*, pages 34–64. Springer, 2018.

[177] Haozhe Xie, Jie Li, Qiaosheng Zhang, and Yadong Wang. Comparison among dimensionality reduction techniques based on random projection for cancer classification. *Computational Biology and Chemistry*, 65:165–172, 2016.

[178] Xiling Li, Rafael Dowsley, and Martine De Cock. Privacy-preserving feature selection with secure multiparty computation. In *International Conference on Machine Learning*, pages 6326–6336. PMLR, 2021.

[179] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.

[180] Jeffrey Dastin. Amazon scraps secret AI recruiting tool that showed bias against women. In *Ethics of data and analytics*, pages 296–299. Auerbach Publications, 2022.

[181] Jana Kasperkevic. Google says sorry for racist auto-tag in photo app. *The Guardian*, 1:2015, 2015.

[182] Tom Simonite. When it comes to gorillas, google photos remains blind. *Wired, January*, 13, 2018.

[183] Abeba Birhane, Sepehr Dehdashtian, Vinay Prabhu, and Vishnu Boddeti. The dark side of dataset scaling: Evaluating racial classification in multimodal models. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 1229–1244, 2024.

[184] Naoki Ito, Sakina Kadomatsu, Mineto Fujisawa, Kiyomitsu Fukaguchi, Ryo Ishizawa, Naoki Kanda, Daisuke Kasugai, Mikio Nakajima, Tadahiro Goto, and Yusuke Tsugawa. The accuracy and potential racial and ethnic biases of GPT-4 in the diagnosis and triage of health conditions: evaluation study. *JMIR Medical Education*, 9:e47532, 2023.

[185] Federico Bianchi, Pratyusha Kalluri, Esin Durmus, Faisal Ladhak, Myra Cheng, Debora Nozza, Tatsunori Hashimoto, Dan Jurafsky, James Zou, and Aylin Caliskan. Easily accessible text-to-image generation amplifies demographic stereotypes at large scale. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 1493–1504, 2023.

[186] Karen Hao. This is how AI bias really happens—and why it's so hard to fix. *MIT Technology Review*, 4, 2019.

[187] Harini Suresh and John Guttag. A framework for understanding sources of harm throughout the machine learning life cycle. In *Proceedings of the 1st ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, pages 1–9, 2021.

[188] Eirini Ntoutsi. Bias in AI-systems: a multi-step approach. In *2nd Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence*, pages 3–4, 2020.

[189] Agostina J Larrazabal, Nicolás Nieto, Victoria Peterson, Diego H Milone, and Enzo Ferrante. Gender imbalance in medical imaging datasets produces biased classifiers for computer-aided diagnosis. *Proceedings of the National Academy of Sciences*, 117(23):12592–12594, 2020.

[190] Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A Smith. The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678, 2019.

[191] Sahil Verma and Julia Rubin. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, 2018.

[192] Arvind Narayanan. Tutorial: 21 fairness definitions and their politics. *ACM Fairness, Accountability and Transparency Conference*, 2018.

[193] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proc. of the 3rd Innovations in Theoretical Comp. Science Conference*, pages 214–226, 2012.

[194] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

[195] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information systems*, 33(1):1–33, 2012.

[196] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333. PMLR, 2013.

[197] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2015.

[198] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Optimized pre-processing for discrimination prevention. *Advances in Neural Information Processing Systems*, 30:3995–4004, 2017.

[199] Shen Yan, Hsien-te Kao, and Emilio Ferrara. Fair class balancing: enhancing model fairness without observing sensitive attributes. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1715–1724, 2020.

[200] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

[201] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012.

[202] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340, 2018.

[203] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International conference on machine learning*, pages 60–69. PMLR, 2018.

[204] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *Proc. of ICML*, pages 2564–2572, 2018.

[205] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 319–328, 2019.

[206] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. *Advances in neural information processing systems*, 32, 2019.

[207] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *12th International Conference on Data Mining*, pages 924–929. IEEE, 2012.

[208] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3315–3323, 2016.

[209] Toon Calders and Sicco Verwer. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010.

[210] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: testing software for discrimination. In *Proc. of the 11th Joint Meeting on Foundations of Software Engineering*, pages 498–510, 2017.

[211] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, pages 5680–5689, 2017.

[212] Michael Veale and Reuben Binns. Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data & Society*, 4(2), 2017.

[213] Wil Michiels. How do you protect your machine learning investment. *EE Times, Technical Report*, 2020.

[214] Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. Fairlearn: A toolkit for assessing and improving fairness in AI. *Microsoft*, (MSR-TR-2020-32), May 2020.

[215] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, A Mojsilović, et al. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4/5):4–1, 2019.

[216] Michael D Ekstrand, Rezvan Joshaghani, and Hoda Mehrpouyan. Privacy for all: Ensuring fair and equitable privacy protections. In *Conf. on Fairness, Accountability and Transparency*, 2018.

[217] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 50(1):3–44, 2021.

[218] N Kilbertus, A Gascón, MJ Kusner, M Veale, KP Gummadi, and A Weller. Blind justice: Fairness with encrypted sensitive attributes. In *Proc. of ICML*, pages 2630–2639, 2018.

[219] Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. Fairness in the eyes of the data: Certifying machine-learning models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 926–935, 2021.

[220] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson Nascimento, Wing-Sea Poon, and Stacey Truex. Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. *IEEE Trans Dependable Secure Comput*, 16(2):217–230, 2019.

[221] Kyle Fritchman, Keerthanaa Saminathan, Rafael Dowsley, Tyler Hughes, Martine De Cock, Anderson Nascimento, and Ankur Teredesai. Privacy-preserving scoring of tree ensembles: A novel framework for AI in healthcare. In *Proc. of 2018 IEEE BigData*, pages 2412–2421, 2018.

[222] N. Agrawal, A. Shahin Shamsabadi, M.J. Kusner, and A. Gascón. QUOTIENT: two-party secure neural network training and prediction. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1231–1247, 2019.

[223] Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, and Anderson Nascimento. Privacy-preserving classification of personal text messages with secure multi-party computation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3752–3764, 2019.

[224] Sikha Pentyala, Rafael Dowsley, and Martine De Cock. Privacy-preserving video classification with convolutional neural networks. In *38th International Conference on Machine Learning*, volume 139 of *PMLR*, 2021.

[225] Payman Mohassel and Peter Rindal. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 35–52, 2018.

[226] T. Araki, A. Barak, J. Furukawa, M. Keller, Y. Lindell, K. Ohara, and H. Tsuchida. Generalizing the SPDZ compiler for other protocols. Cryptology ePrint Archive, Report 2018/762, 2018. `https://eprint.iacr.org/2018/762`.

[227] D. Demmler, T. Schneider, and M. Zohner. ABY – a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[228] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In *Annual International Cryptology Conference*, pages 823–852. Springer, 2020.

[229] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[230] Steven R Livingstone and Frank A Russo. The Ryerson audio-visual database of emotional speech and song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PloS one*, 13(5), 2018.

[231] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, pages 797–806, 2017.

[232] e-CFR. Adverse impact and four-fifths rule. § 1607.4 Information on impact. `https://www.ecfr.gov/current/title-29/subtitle-B/chapter-XIV/part-1607/subject-group-ECFRdb347e844acdea6/section-1607.4`, 1981.

[233] Dana Pessach and Erez Shmueli. Algorithmic fairness. *arXiv preprint arXiv:2001.09784*, 2020.

[234] Zirui Zhou, Lingyang Chu, Changxin Liu, Lanjun Wang, Jian Pei, and Yong Zhang. Towards fair federated learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4100–4101, 2021.

[235] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.

[236] Annie Abay, Yi Zhou, Nathalie Baracaldo, Shashank Rajamoni, Ebube Chuba, and Heiko Ludwig. Mitigating bias in federated learning. *arXiv preprint arXiv:2012.02447*, 2020.

[237] Wei Du, Depeng Xu, Xintao Wu, and Hanghang Tong. Fairness-aware agnostic federated learning. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 181–189. SIAM, 2021.

[238] Sen Cui, Weishen Pan, Jian Liang, Changshui Zhang, and Fei Wang. Addressing algorithmic disparity and performance inconsistency in federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[239] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. *Advances in Neural Information Processing Systems*, 31:2796–2806, 2018.

[240] Zhe Yu, Joymallya Chakraborty, and Tim Menzies. Fairbalance: How to achieve equalized odds with data pre-processing. *IEEE Transactions on Software Engineering*, 2024.

[241] Fengda Zhang, Kun Kuang, Yuxuan Liu, Chao Wu, Fei Wu, Jiaxun Lu, Yunfeng Shao, and Jun Xiao. Unified group fairness on federated learning. *arXiv preprint arXiv:2111.04986*, 2021.

[242] Afroditi Papadaki, Natalia Martinez, Martin Bertran, Guillermo Sapiro, and Miguel Rodrigues. Federating for learning group fair models. *1st NeurIPS Workshop on New Frontiers in Federated Learning*, 2021.

[243] Xubo Yue, Maher Nouiehed, and Raed Al Kontar. GIFAIR-FL: A framework for group and individual fairness in federated learning. *INFORMS Joural on Data Science*, 2(1):10–23, 2023.

[244] Samhita Kanaparthy, Manisha Padala, Sankarshan Damle, and Sujit Gujar. Fair federated learning for heterogeneous data. In *Proceedings of the 5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*, pages 298–299, 2022.

[245] Junyuan Hong, Zhuangdi Zhu, Shuyang Yu, Zhangyang Wang, Hiroko H Dodge, and Jiayu Zhou. Federated adversarial debiasing for fair and transferable representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 617–627, 2021.

[246] Yahya H Ezzeldin, Shen Yan, Chaoyang He, Emilio Ferrara, and A Salman Avestimehr. FairFed: Enabling group fairness in federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7494–7502, 2023.

[247] Borja Rodríguez Gálvez, Filip Granqvist, Rogier van Dalen, and Matt Seigel. Enforcing fairness in private federated learning via the modified method of differential multipliers. In *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

[248] Daniel Yue Zhang, Ziyi Kou, and Dong Wang. FairFL a fair federated learning approach to reducing demographic bias in privacy-sensitive classification models. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1051–1060. IEEE, 2020.

[249] Manisha Padala, Sankarshan Damle, and Sujit Gujar. Federated learning meets fairness and differential privacy. In *International Conference on Neural Information Processing*, pages 692–699. Springer, 2021.

[250] Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In *37th International Conference on Computer Design (ICCD)*, pages 246–254. IEEE, 2019.

[251] Gergely Ács and Claude Castelluccia. I have a dream! (DiffeRentially privatE smArt Metering). In *International Workshop on Information Hiding*, pages 118–132. Springer, 2011.

[252] Vincent Bindschaedler, Shantanu Rane, Alejandro E. Brito, Vanishree Rao, and Ersin Uzun. Achieving differential privacy in secure multiparty data aggregation protocols on star networks. CODASPY '17, page 115–125, 2017.

[253] Giorgio Roffo and Alessandro Vinciarelli. Personality in computational advertising: A benchmark. In *4 th Workshop on Emotions and Personality in Personalized Systems (EMPIRE) 2016*, page 18, 2016.

[254] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[255] Thorsten Joachims and Filip Radlinski. Search engines that learn from implicit feedback. *Computer*, 40(8):34–40, 2007.

[256] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2):7–es, 2007.

[257] Matthew Kay, Cynthia Matuszek, and Sean A Munson. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3819–3828, 2015.

[258] Yudan Liu, Kaikai Ge, Xu Zhang, and Leyu Lin. Real-time attention based look-alike model for recommender system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2765–2773, 2019.

[259] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 1–6, 2017.

[260] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2243–2251, 2018.

[261] Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking. *Advances in Neural Information Processing Systems*, 32, 2019.

[262] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578, 2017.

[263] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd international ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 429–438, 2020.

[264] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228, 2018.

[265] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 405–414, 2018.

[266] John Canny. Collaborative filtering with privacy. In *Proc. of IEEE S&P*, pages 45–57, 2002.

[267] Zhengqiang Ge, Xinyu Liu, Qiang Li, Yu Li, and Dong Guo. PrivItem2Vec: a privacy-preserving algorithm for top-N recommendation. *International Journal of Distributed Sensor Networks*, 17(12), 2021.

[268] Dingqi Yang, Bingqing Qu, and Philippe Cudré-Mauroux. Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(3):507–520, 2018.

[269] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

[270] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. Secure federated matrix factorization. *IEEE Intelligent Systems*, 36(5):11–20, 2020.

[271] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. FedGNN: federated graph neural network for privacy-preserving recommendation. *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML*, 2021.

[272] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with fairness constraints. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:15, 2018.

[273] Piotr Sapiezynski, Wesley Zeng, Ronald E Robertson, Alan Mislove, and Christo Wilson. Quantifying the impact of user attention on fair group representation in ranked lists. In *Companion Proceedings of the 2019 World Wide Web Conference*, pages 553–562, 2019.

[274] Mostafa Dehghani, Hosein Azarbonyad, Jaap Kamps, and Maarten de Rijke. Share your model instead of your data: Privacy preserving mimic learning for ranking. *arXiv preprint arXiv:1707.07605*, 2017.

[275] Eugene Kharitonov. Federated online learning to rank with evolution strategies. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 249–257, 2019.

[276] Shuyi Wang, Shengyao Zhuang, and Guido Zuccon. Federated online learning to rank with evolution strategies: a reproducibility study. In *European Conference on Information Retrieval*, pages 134–149. Springer, 2021.

[277] Shuyi Wang, Bing Liu, Shengyao Zhuang, and Guido Zuccon. Effective and privacy-preserving federated online learning to rank. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 3–12, 2021.

[278] Yehezkel S Resheff, Yanai Elazar, Moni Shahar, and Oren Sar Shalom. Privacy and fairness in recommender systems via adversarial training of user representations. *arXiv preprint arXiv:1807.03521*, 2018.

[279] Ryoma Sato. Private recommender systems: How can users build their own fair recommender systems without log data? In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 549–557. SIAM, 2022.

[280] Melissa A Haendel, Christopher G Chute, Tellen D Bennett, David A Eichmann, Justin Guinney, Warren A Kibbe, Philip RO Payne, Emily R Pfaff, Peter N Robinson, Joel H Saltz, et al. The national COVID cohort collaborative (N3C): rationale, design, infrastructure, and deployment. *Journal of the American Medical Informatics Association*, 28(3):427–443, 2021.

[281] All of Us Research Program Investigators. The "all of us" research program. *New England Journal of Medicine*, 381(7):668–676, 2019.

[282] Timothy M Errington, Alexandria Denis, Nicole Perfito, Elizabeth Iorns, and Brian A Nosek. Challenges for assessing replicability in preclinical cancer biology. *eLife Sciences Publications, Ltd*, 2021.

[283] Hope Watson, Jack Gallifant, Yuan Lai, Alexander P Radunsky, Cleva Villanueva, Nicole Martinez, Judy Gichoya, Uyen Kim Huynh, and Leo Anthony Celi. Delivering on NIH data sharing requirements: avoiding open data in appearance only. *BMJ Health & Care Informatics*, 30(1), 2023.

[284] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the Netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[285] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.

[286] Thee Chanyaswad, Changchang Liu, and Prateek Mittal. Ron-gauss: Enhancing utility in non-interactive private data release. *Proceedings on Privacy Enhancing Technologies*, 2019.

[287] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.

[288] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444. PMLR, 2019.

[289] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

[290] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. *Advances in Neural Information Processing Systems*, 32, 2019.

[291] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[292] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pages 17564–17579. PMLR, 2023.

[293] V-CHAMPS. The veterans cardiac health and AI model predictions challenge. `https://precision.fda.gov/challenges/31/results`, 2023.

[294] Lukas Prediger, Joonas Jälkö, Antti Honkela, and Samuel Kaski. Collaborative learning from distributed data with differentially private synthetic data. *BMC Medical Informatics and Decision Making*, 24(1):167, 2024.

[295] Trae Claar, Steven Golob, Sikha Pentyala, Geetha Sitaraman, Martine De Cock, Jineta Banerjee, and Luca Foschini. Securely generating synthetic genomic data from distributed data silos. In *11th International Workshop on Genome Privacy and Security (GenoPri'24)*, 2024.

[296] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. *Advances in Neural Information Processing Systems*, 25, 2012.

[297] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. AIM: an adaptive and iterative mechanism for differentially private synthetic data. *Proceedings of the VLDB Endowment*, 15(11):2599–2612, 2022.

[298] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. Winning the NIST contest: A scalable and general approach to differentially private synthetic data. *Journal of Privacy and Confidentiality*, 11(3), 2021.

[299] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms. In *AAAI-22 Workshop on Privacy-Preserving Artificial Intelligence (PPAI)*, 2022.

[300] Bangzhou Xin, Wei Yang, Yangyang Geng, Sheng Chen, Shaowei Wang, and Liusheng Huang. Private FL-GAN: Differential privacy synthetic data generation based on federated learning. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2927–2931. IEEE, 2020.

[301] Monik Raj Behera, Sudhir Upadhyay, Suresh Shetty, Sudha Priyadarshini, Palka Patel, and Ker Farn Lee. Fedsyn: Synthetic data generation using federated learning. *arXiv preprint arXiv:2203.05931*, 2022.

[302] Vishal Ramesh, Rui Zhao, and Naman Goel. Decentralised, scalable and privacy-preserving synthetic data generation. *arXiv preprint arXiv:2310.20062*, 2023.

[303] Bangzhou Xin, Wei Yang, Yangyang Geng, Sheng Chen, Shaowei Wang, and Liusheng Huang. Private FL-GAN: Differential privacy synthetic data generation based on federated learning. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2927–2931, 2020.

[304] Rui Zhao, Naman Goel, Nitin Agrawal, Jun Zhao, Jake Stein, Ruben Verborgh, Reuben Binns, Tim Berners-Lee, and Nigel Shadbolt. Libertas: Privacy-preserving computation for decentralised personal data stores. *arXiv preprint arXiv:2309.16365*, 2023.

[305] Ali Reza Ghavamipour, Fatih Turkmen, Rui Wang, and Kaitai Liang. Federated synthetic data generation with stronger security guarantees. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies*, pages 31–42, 2023.

[306] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. Securely sampling discrete Gaussian noise for multi-party differential privacy. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2262–2276, 2023.

[307] Martin Pettai and Peeter Laud. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 421–430, 2015.

[308] Raymond K Zhao, Ron Steinfeld, and Amin Sakzad. FACCT: fast, compact, and constant-time discrete Gaussian sampler over integers. *IEEE Transactions on Computers*, 69(1):126–137, 2019.

[309] Ryan Mckenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4435–4444. PMLR, 2019.

[310] James Bell, Adria Gascon, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. Distributed, private, sparse histograms in the two-server model. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 307–321, 2022.

[311] Mayana Pereira, Meghana Kshirsagar, Sumit Mukherjee, Rahul Dodhia, Juan Lavista Ferres, and Rafael de Sousa. Assessment of differentially private synthetic data for utility and fairness in end-to-end machine learning pipelines for tabular data. *Plos one*, 19(2):e0297271, 2024.

[312] Giuseppe Vietri, Cedric Archambeau, Sergul Aydore, William Brown, Michael Kearns, Aaron Roth, Ankit Siva, Shuai Tang, and Steven Z Wu. Private synthetic data for

multitask learning and marginal queries. *Advances in Neural Information Processing Systems*, 35:18282–18295, 2022.

[313] Gilad Asharov, Koki Hamada, Ryo Kikuchi, Ariel Nof, Benny Pinkas, and Junichi Tomida. Secure statistical analysis on multiple datasets: Join and group-by. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 3298–3312, 2023.

[314] Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

[315] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[316] Matjaz Zwitter and Milan Soklic. Breast Cancer. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C51P4M.

[317] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica*, May 23, 2016.

[318] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, page 261. American Medical Informatics Association, 1988.

[319] Swier Garst, Julian Dekker, and Marcel Reinders. A comprehensive experimental comparison between federated and centralized learning. *Database*, 2025.

[320] Pascal Riedel, Lukas Schick, Reinhold von Schwerin, Manfred Reichert, Daniel Schaudt, and Alexander Hafner. Comparative analysis of open-source federated learning frameworks-a literature-based survey and review. *International Journal of Machine Learning and Cybernetics*, 15(11):5257–5278, 2024.

[321] Georgi Ganev and Emiliano De Cristofaro. On the inadequacy of similarity-based privacy metrics: Reconstruction attacks against "truly anonymous synthetic data". *arXiv preprint arXiv:2312.05114*, 2023.

[322] Aditya Shankar, Hans Brouwer, Rihan Hai, and Lydia Chen. Silofuse: Cross-silo synthetic data generation with latent tabular diffusion models. *arXiv preprint arXiv:2404.03299*, 2024.

[323] Bangzhou Xin, Yangyang Geng, Teng Hu, Sheng Chen, Wei Yang, Shaowei Wang, and Liusheng Huang. Federated synthetic data generation with differential privacy. *Neurocomputing*, 468:1–10, 2022.

[324] Eugenio Lomurno, Alberto Archetti, Lorenzo Cazzella, Stefano Samele, Leonardo Di Perna, and Matteo Matteucci. SGDE: Secure generative data exchange for cross-silo federated learning. In *Proceedings of the 2022 5th International Conference on Artificial Intelligence and Pattern Recognition*, pages 205–214, 2022.

[325] Antti Koskela and Tejas D Kulkarni. Practical differentially private hyperparameter tuning with subsampling. *Advances in Neural Information Processing Systems*, 36, 2024.

[326] Zihang Xiang, Chenglong Wang, and Di Wang. How does selection leak privacy: Revisiting private selection and improved results for hyper-parameter tuning. *arXiv preprint arXiv:2402.13087*, 2024.

[327] Youlong Ding and Xueyang Wu. Revisiting hyperparameter tuning with differential privacy. *arXiv preprint arXiv:2211.01852*, 2022.

[328] Natalija Mitic, Apostolos Pyrgelis, and Sinem Sav. How to privately tune hyperparameters in federated learning? Insights from a benchmark study. *arXiv*, 2024.

[329] Luca Bonomi, Yingxiang Huang, and Lucila Ohno-Machado. Privacy challenges and research opportunities for genomic data sharing. *Nature genetics*, 52(7):646–654, 2020.

[330] NIH. Implementation update for data management and access practices under the genomic data sharing policy. https://grants.nih.gov/grants/guide/notice-files/NOT-OD-24-157.html, 2024.

[331] Marie Oestreich, Dingfan Chen, Joachim L Schultze, Mario Fritz, and Matthias Becker. Privacy considerations for sharing genomics data. *Experimental and Clinical Sciences Journal*, 20:1243, 2021.

[332] Dingfan Chen, Marie Oestreich, Tejumade Afonja, Raouf Kerkouche, Matthias Becker, and Mario Fritz. Towards biologically plausible and private gene expression data generation. In *Proceedings in Privacy-Enhancing Technologies*, volume 2, pages 531–554, 2024.

[333] European Lighthouse on Secure and Safe AI (ELSA). Health privacy challenge: Advancing AI privacy in computational biology at CAMDA 2025. https://benchmarks.elsa-ai.eu/?ch=4, 2025.

[334] Dan Bogdanov, Sven Laur, and Riivo Talviste. Oblivious sorting of secret-shared data. Technical Report T-4-19, Cybernetica, `http://research.cyber.ee`, 2013.

[335] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2002.

[336] Steven Golob, Sikha Pentyala, Rafael Dowsley, Bernardo David, Mario Larangeira, Martine De Cock, and Anderson Nascimento. A decentralized information marketplace preserving input and output privacy. In *Proceedings of the Second ACM Data Economy Workshop*, pages 1–6, 2023.

[337] Wuxuan Jiang, Xiangjun Song, Shenbai Hong, Haijun Zhang, Wenxin Liu, Bo Zhao, Wei Xu, and Yi Li. Spin: An efficient secure computation framework with gpu acceleration. *arXiv preprint arXiv:2402.02320*, 2024.

[338] Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. Piranha: A GPU platform for secure computation. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 827–844, 2022.

[339] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.