Efficient and Robust Protocols for Privacy-Preserving Semi-Decentralized Machine Learning

César Sabater

École Doctorale Mathematiques, Sciences du Numérique et de leurs Interactions (MADIS)

Thèse de Doctorat

Protocoles Efficaces et Robustes pour l'Apprentissage Automatique Semi-Décentralisé Préservant la Confidentialité

Préparée et soutenue publiquement par

César Sabater

à Villeneuve d'Ascq, le 20 Juin 2022, pour obtenir le grade de

Docteur en Informatique

Directeur: Jan Ramon

Soutenue devant le jury composé de:

Sonia Ben Mokhtar	Directrice de Recherche, CNRS - LIRIS	Rapporteuse et Présidente
Emiliano De Cristofaro	Full Professor, University College London	Rapporteur
Christian Weinert	Lecturer, University of London	Examinateur
Jan Ramon	Directeur de Recherche, INRIA Lille	Directeur

Université de Lille

Inria

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Doctoral school Mathematics and Digital Sciences (MADIS)
Doctoral Dissertation

Efficient and Robust Protocols for Privacy-Preserving Semi-Decentralized Machine Learning

---

Prepared and publicly defended by

César Sabater

in Villeneuve d'Ascq, on June 20th, 2022, to obtain the degree of

Doctor in Computer Science

Supervisor: Jan Ramon

Defended before the jury composed of:

Sonia Ben Mokhtar	Directrice de Recherche, CNRS - LIRIS	Reviewer and President
Emiliano De Cristofaro	Full Professor, University College London	Reviewer
Christian Weinert	Lecturer, University of London	Examiner
Jan Ramon	Directeur de Recherche, INRIA Lille	Supervisor

To my father Carlos.
To my grandmother María, in loving memory.

Acknowledgments

I am grateful to Jan Ramon, my advisor, who guided me through this project. I appreciate his dedication in reviewing my daily work and providing detailed observations. I want to take this opportunity to thank him for challenging me to give the best of myself in the production of qualitative results. Through his supervision, I have grown to become more independent as a researcher.

I would like to thank all the researchers for their contributions to improve this work. Aurélien Bellet, Andreas Peter and Florian Hahn, in addition to their collaboration in the presented contributions, improved my insights with fruitful discussions and helped me discover key literature for the project. I also want to express my deepest gratitude to Sonia Ben Mokhtar, Emiliano De Cristofaro and Christian Weinert, who evaluated this dissertation and gave important feedback to improve it.

I am grateful to all the professors and friends who motivated me in my studies. I am deeply thankful to Eric Biagioli, my coach for the olympiads in informatics, who was the first person to show me the beauty of computer science and encouraged me to pursue university studies in this area. I also want to thank professors Guido Macchi, Pablo Granitto, Guillermo Grinblat, Gabriela Argiroffo and Ana Casali, who inspired me and wrote me recommendation letters when I applied to this great PhD topic that completely fitted my interests.

I express my gratitude to the members of the Magnet team who created a friendly research environment. I particularly enjoyed the shared midday meals, discussions, "Magnet au vert" trips and foosball games. I thank Marc Tommasi and William De Vazelhes who helped me to find an apartment and settle in Lille.

I would like to thank all my friends for their support. Arijus, Pauline, Brij, Mahsa, Carlos, Onkar, Mariana and Mathieu, whom I met in Lille, made my everyday life much more interesting by filling it with conversations and trips within France and overseas. I am very grateful with Júlia, my girlfriend, who was an amazing listener and always gave me her support. My Argentine friends Mariano, Javier, Maximiliano, Damian, Yueh-Wei, Fepi, Juan and Ariel made me feel like home with their visits and phone calls. In particular, I thank them for their company during the Covid lock-down.

I cannot express with words how thankful I am with my parents Raquel and Carlos and my sisters Paola, Carla and María José. They gave me their unconditional love and support from the distance. I am grateful to Adrián, my brother-in-law, who was always attentive to my concerns and provided me with his great advice.

In this world where hostility is abundant, kindness has been one my biggest motivational forces. I thank all those people who spread it and that have helped me in different ways through these years. They have given purpose to this project.

Abstract

In recent years, the concern for privacy has significantly grown. This is a consequence of the regular use of data-intensive services that require the massive outsourcing and processing of individuals data, which is often sensitive. For that reason, measures to regulate the manipulation of individual's data and to prevent its exposure gained notable relevance.

Two important limitations of existing algorithms as used in the field of machine learning are that they often are not robust against colluding adversaries, and that a trusted party is needed to among others perform differential privacy perturbation. This PhD dissertation aims to address these problems. In particular it contains two major contributions:

The first contribution of this work is a decentralized and secure protocol that performs differentially private aggregation. In this setting, each party in a group has its own private data, and they want to collaboratively compute a statistic, e.g., an average, without disclosing the sensitive information. Our protocol is robust against inference attacks by colluding parties and allows for verification of the correctness of computations. It only requires every party to communicate with a logarithmic number of other parties, and achieves differential privacy with a utility nearly as good as in the trusted central curator setting.

The second contribution proposes a protocol to draw random numbers in a multi-party computing setting in such a way that all parties can verify that the generated number follows a prescribed probability distribution and is effectively pseudo-random, i.e., no group of colluding parties can bias the randomness. In particular, we consider drawing random numbers publicly (so all parties can see them), privately (so exactly one party can see them) or in a hidden way (so they are output as secret shares and hence are not known by any of the parties). We instantiate our methods for drawing random numbers from the Laplace and Gaussian distributions. As a byproduct of independent interest, we propose algorithms to verify transcendental computations such as logarithmic, trigonometric and exponential functions in zero knowledge.

Résumé

Ces dernières années, la préoccupation pour la protection de la vie privée s'est considérablement accrue. Cela s'explique par l'utilisation régulière de services qui nécessitent l'externalisation et le traitement massif de données personnelles, souvent sensibles. Pour cette raison, les mesures visant à réglementer la manipulation des données personnelles et à empêcher leur divulgation ont gagné en importance.

Deux limitations importantes des algorithmes existants utilisés dans le domaine de l'apprentissage automatique sont qu'ils ne sont souvent pas robustes contre les attaques par collusion, et qu'un tiers de confiance est nécessaire pour (entre autres) effectuer une perturbation aléatoire permettant d'obtenir des garanties de confidentialité différentielle (differential privacy). Cette thèse vise à résoudre ces problèmes. Elle contient en particulier deux contributions majeures.

La première contribution est un protocole décentralisé et sécurisé qui effectue une agrégation satisfaisant la confidentialité différentielle. Dans ce contexte, chaque partie possède ses propres données privées et souhaite calculer de manière collaborative une statistique, par exemple une moyenne, sans divulguer ses informations sensibles. Notre protocole est robuste aux attaques d'inférence par des parties en collusion et permet de vérifier l'exactitude des calculs. Il nécessite que chaque partie ne communique qu'avec un nombre logarithmique d'autres parties et permet d'obtenir des garanties de confidentialité différentielle avec une utilité presque équivalente au cas où l'on aurait recourt à un tiers de confiance.

La deuxième contribution propose un protocole pour générer des nombres aléatoires dans un cadre de calcul multipartite de telle sorte que toutes les parties puissent vérifier que le nombre généré suit la distribution de probabilité souhaitée et est effectivement pseudo-aléatoire, c'est-à-dire qu'aucun groupe de parties en collusion ne peut en fausser le caractère aléatoire. En particulier, nous considérons le tirage de nombres aléatoires publics (de sorte que toutes les parties puissent les voir), privés (de sorte qu'une seule partie puisse les voir) ou de manière cachée (de sorte qu'ils soient émis sous forme de parts secrètes et ne soient donc connus d'aucune des parties). Nous instancions nos méthodes de tirage de nombres aléatoires pour la distribution de Laplace et la distribution gaussienne. Comme sous-produit de notre approche pouvant avoir un intérêt en soi, nous proposons des algorithmes à divulgation nulle de connaissance pour vérifier les calculs transcendantaux tels que les fonctions logarithmique, trigonométrique et exponentielle.

Contents

Chapter 1

Introduction

In this chapter, we introduce the problems treated in this thesis. We start by describing the context in Section 1.1 and present the problems in Section 1.2. Next, we describe our contributions in Section 1.3 and conclude by presenting the structure of the thesis in Section 1.4.

1.1 Context

In the last three decades, the domain of Machine Learning experienced a huge growth. It is applied to a huge variety of domains such as consumer service, mobility and logistics in cities, research in health, chemistry, market prediction, cognitive processes, etc. among many others. To obtain improvements, large amounts of data are required. One of the reasons of the popularization of machine learning is the growing amount of interconnected digital information, e.g., due to automated data generation such as video streams, automated patient monitoring in hospitals, news and social media, etc. Many other sources of information improve automated tasks and services.

Much of this data belongs to individuals or organizations and is clearly sensitive: e-mails and instant messages contain all kinds of private information, medical records stored in hospital databases have individuals' conditions and databases of companies might contain data of business strategy. To generate machine learning models, the predominant approach due to its organizational simplicity is the *centralized setting*: vast collections of data of individuals, called *data subjects*, or other entities are accumulated and processed in one or a few storage centers, typically owned by large companies or institutions, called *data holders*.

However, the centralized setting has important issues: first, data subjects have no or little control over their data and second, data concentration creates a single and valuable target of external attacks. A leak in a data center is a much bigger threat than in a few mobile phones. This risk increases if security measures are inadequate and lead to system vulnerabilities. There have been many cases of sensitive data compromised by attacks in production systems [114, 94, 1, 103]. Even if data is not directly exposed, it has been shown that models obtained from its exploitation reveal sensitive information [106, 116, 67, 31]. Another issue concerns the linkage of information of a data subject from different sources. For example, one can learn that a person regularly eats fast food from its credit card bill. A health insurance company with access to this information could conclude that this person has unhealthy eating habits, therefore a risk of cardiovascular and other related problems which is above the average, and raise its insurance fee. Data about the lifestyle, political

orientation, religious and personal preferences is not something that people are willing to share [30, 42, 107, 118].

Data subjects not only are exposed to such risks, but additionally they have an opaque view of the flow that will take or has taken their data, and the specific risks it entangles (see for example the Cambridge Analytica privacy scandal [44]). With the growing concerns, privacy has become a priority for many legislators and citizens. Different type of measures have been taken to protect individuals and provide them safer services. We identify two types of measures: legislative and technical.

Legislative measures These measures define legal boundaries and regulations on the collection and processing of data. A first measure of this kind is the creation of more transparent privacy policies by organizations, where users could clearly be informed on how their data will be used. However, even if this provides more information to data subjects, data holders are still the ones who decide the exploitation boundaries. Stronger regulations are now being imposed by governments, such as the General Data Protection Regulation (GDPR) in the European Union or the Personal Information Protection and Electronic Documents Act (PIPEDA) in Canada. In general lines, they establish a legal framework that forces organizations to ask individuals for consent when data is collected, to inform them in more level of detail whether this data is going to be shared with other parties, how is it going to be processed and gives the option to withdraw the consent at any point in the future, which leads to the mandatory removal of collected data.

While regulations are a significant advance into data sovereignty, data holders have to be trusted on following the law, and sometimes it is not possible to track or audit how data is handled. If infractions are made, it is difficult to prove. Therefore, data is safe only if their holders are trusted. In addition, privacy practices that are considered "good enough" change over time and specifying correctly the details of their legal applicability is challenging.

Technical Measures Technical measures often aim at prevention, avoiding that data subjects need to trust external parties. They provide a stronger type of guarantee in which sensitive data is seen as little as possible by untrusted parties. They mostly rely on the implementation of algorithms that statistically obfuscate and/or cryptographically encrypt sensitive information.

A natural way to implement this type of guarantees is to adopt a *decentralized setting*, where data is kept local to the data subjects, which then engage with each other in protocols to access services and process their data in a privacy preserving way. This setting is attractive not only for privacy preserving algorithms, but also for learning from excessively large amounts of data that are prohibitive to centralize, e.g., when processing video data from many cars in autonomous driving applications. For that reason, research work on decentralized (or more often called federated) learning has increased exponentially in the last few years [84].

Preventive privacy is not only applicable in current domains, but would also allow other interesting machine learning scenarios. For example, in the case where small or mid-sized organizations such as companies or hospitals would be willing to collaboratively generate more qualitative models. For most of these organizations, this is not possible or encouraging in the centralized setting, as sharing data is not allowed by law or would lead to competitive disadvantage. If preventive measures provide sufficiently strong guarantees, another interesting possibility is to perform machine learning in a privacy-preserving way

using sensitive data that owners would be unwilling to share in a centralized setting with a trusted curator.

Notions of Privacy Tools that provide privacy require precise definitions of what is a privacy preserving procedure. In cryptography, a widely used notion of privacy states that a computation is private if the only information revealed was the outcome, and no party learned anything about the input or intermediate computations. Sometimes this does not satisfy current needs because, as we said previously, the output can reveal a substantial amount of sensitive data.
Another notion is k-anonymity, which defines a computation as private if the revealed data of each participant cannot be distinguished from that of $k-1$ other participants. However, this still reveals sensitive information in certain types of computation.
A more empirical notion is to measure the resilience of an algorithm to popular attacks, for example, an attack that aims to identify a participant of the computation. One line of work is formed by adversarial networks [74]. This provides privacy against known attacks, but it does not ensure any kind of protection against newly designed attacks that may come in the future, or even to variations of current attacks.
Currently, the most popular framework for privacy in statistics is Differential Privacy (DP) [58], which precisely quantifies the sensitive information leaked in a computation. It is considered a gold standard for privacy in the context of exploitation of data.

DP was originally considered to prevent leakage of sensitive information in the outcome of a computation in the centralized setting, where a trusted party is in charge of the computation. Later, it was adopted in the decentralized setting, where no party is trusted, for example using the Local Differential Privacy (LDP) framework [56, 87]. Algorithms that achieve DP rely on adding noise to publicly visible information in a way that allows to hide sensitive data while still being able to perform computations. The principle follows the line of common practice in privacy preserving surveys, where participants randomize their responses to sensitive questions to achieve plausible deniability. However, LDP requires significantly more noise than in the centralized setting. This leads to models with unacceptable accuracy except in protocols with a massive amount of participants. As we will see, encryption strategies hiding inputs and intermediate results can reduce the amount of needed noise to about what is needed to achieve DP in the centralized setting.

1.2 The Problem

This thesis focuses on the study of protocols that provide differential privacy in the decentralized setting, where no trusted central party is assumed. We consider scenarios with a realistic behavior of the participants, and aim at protocols whose cost does not make them prohibitive to implement in practice.

As mentioned in Section 1.1, LDP results in a significant accuracy deterioration with respect to the centralized setting. This can be overcome using cryptographic primitives such as Secure Aggregation [59, 34, 122, 22, 80] but, except for recent contributions [12, 124], they have a communication cost that does not scale well to a large number of parties. This *tension between privacy, accuracy and communication cost* in the decentralized setting is the first problem we focus on.

Another approach to deal with the above problem is by positioning in between the centralized and decentralized settings, for example, using Secure Multi-party Computation

(MPC) and trusting the computation to a small set of parties assumed not to collude [45, 78] or in the *shuffle model*, where there is a trusted shuffler that randomly mixes the updates of participants [63, 38, 9, 68]. While we provide protocols for the setting with a few non-colluding parties, we note that these alternatives have less applicability in real scenarios.

Protocols in the decentralized setting also face new challenges. A first challenge is that participants might be interested to bias the outcome of a computation for many reasons. For example, a company that wants to make its competitor to perform sub-optimally, infiltrating the computation with malicious participants that collude with each other and thereby biasing its business analysis. Malicious participants are already a problem in non-private decentralized machine learning and data mining [123, 130]. When protocols are private, preventing this becomes more difficult as it is not possible to directly detect inconsistent contributions. Privacy preserving approaches mostly assume a honest-but-curious behavior of participants, which might try to infer data from others but will follow the protocol. While there exist some approaches that are resilient to some malicious types of behavior, they do not consider a broad spectrum of attacks. A second challenge is that the more parties are involved, the more vulnerable is the computation to suffer from unexpected events such as participants going offline due to connectivity issues. This could significantly impact on the accuracy or directly lead to the abortion of the computation. The problem of *robustness to malicious parties and dropouts* is the second focus of this work. In this dissertation, we aim to address these challenges.

1.3 Contributions

We present two main contributions which are described below.

First Contribution The first contribution of this work is a decentralized protocol called GOPA, which performs differentially private averaging. While the latter is a simple statistic, its secure computation is challenging and it has important applications. It is a key primitive to aggregate updates in federated learning algorithms and it allows to train models that can be computed from averages, as linear models and decision trees. We propose a protocol to compute private averages that only requires logarithmic communication cost in the number of participants and can compute private models with an amount of noise that is significantly lower than in LDP. In fact, it matches the amount of noise required for DP when the central party is trusted. To prevent malicious behavior, the protocol provides in addition a verification procedure using cryptographic zero knowledge proofs. This enables users to prove the correctness of their computations without compromising their privacy. Finally, we show measures to make the protocol resilient to dropouts.

This work, which will be described in Chapter 3, has been accepted for the *ECML / PKDD Journal Track*[1]. It is also available in arXiv by

- César Sabater, Aurélien Bellet, and Jan Ramon. "An Accurate, Scalable and Verifiable Protocol for Federated Differentially Private Averaging." *arXiv preprint arXiv:2006.07218 (2020)*.

[1] `https://ecmlpkdd.org/`

and parts of it have been presented in *NeurIPS 2020 Workshop on Privacy Preserving Machine Learning* [2] and *Conférence sur l'Apprentissage automatique (CAp) 2020* [3].

All the authors made approximately equal contributions to this work. My contribution involves, in joint work with the other authors, the proofs of privacy guarantees, the development of techniques to detect malicious behavior and the implementation of experiments. My contributions in privacy guarantees were produced with the guidance and expertise in the field of the other authors. The result on random graphs presented in Section 3.5.4 was contributed primarily by another author. The study of cryptographic tools which were used for the robustness results presented in Section 3.6 was primarily done by me.

Second Contribution The second contribution focuses on private sampling algorithms in the presence of malicious parties. We propose algorithms for unbiased sampling from uniform, Gaussian, Laplacian and arbitrary distributions in 3 settings: (1) verifiably sampling numbers that are public, (2) verifiably sampling numbers that are private to one party and (3) verifiably sampling numbers which are a secret shared by all parties, and therefore hidden to all of them. Such algorithms can be used in the context of differentially private federated learning to prevent some types of data poisoning. In decentralized machine learning, sampling unbiased noise to ensure that the participants' information exchanges are privacy-preserving is crucial to provide theoretical privacy guarantees. While doing so, we propose algorithms to verify private transcendental computations such as logarithmic, trigonometric and exponential functions. These techniques can be used as building blocks to construct larger systems in which machine learning computations are verifiable in zero-knowledge.

This contribution is co-authored with Jan Ramon, Andreas Peter from the University of Oldenburg and Florian Hahn from the University of Twente. It has been accepted under major revisions for *Proceedings on Privacy Enhancing Technologies (PoPETs) 2023* [4] and parts of it have been presented in

- César Sabater and Jan Ramon. "Zero Knowledge Arguments for Verifiable Sampling." *NeurIPS 2021 Workshop Privacy in Machine Learning*. 2021.

We will present it in Chapter 4. This work was mainly produced and written by me with the guidance of the other authors.

All the content presented in Chapters 3 and 4 has been additionally peer-reviewed in their submission to the venues mentioned above.

1.4 Structure of the Thesis

In Chapter 2, we introduce general notations and the necessary background in privacy and cryptography for the following chapters. In Chapter 3 we present the first contribution, the GOPA protocol. In Chapter 4 we present how to perform private sampling with malicious samplers, our second contribution. We conclude and discuss future work in Chapter 5.

[2] https://ppml-workshop.github.io/ppml20/
[3] https://cap-rfiap2020.sciencesconf.org/
[4] https://petsymposium.org/cfp23.php

Chapter 2

Background

In this chapter, we present concepts on which our contributions rely. We describe common elements of the setting we consider in Section 2.1. Then, in Section 2.2 we present several notions of privacy and in Section 2.3 we further develop on how to achieve Differential Privacy, the notion used in this work. After that, we introduce the required cryptographic concepts: in Section 2.4, we describe basic cryptographic concepts present in the contributions, and in Section 2.5 we describe Zero Knowledge Proofs, a key cryptographic primitive for the robustness of our protocols.

2.1 Setting and General notation

General Notation We will denote the set of the first k positive integers by $[k] = \{i \in \mathbb{N} \mid 1 \leq i \leq k\}$. $a \leftarrow_R S$ means that a is sampled uniformly at random from elements of S. For vectors $\bar{a} = (a_1, \ldots, a_k)$ and $\bar{b} = (b_1, \ldots, b_k)$, $\bar{a} + \bar{b}$ and $\bar{a} * \bar{b}$ are the element-wise addition and product. $\bar{a}^{\bar{b}}$ is the multi-exponentiation $\prod_{i=1}^{k} a_i^{b_i}$. For a scalar s, $s + \bar{a} = (s, \ldots, s) + \bar{a}$, $s * \bar{a} = (s, \ldots, s) * \bar{a}$ and $\bar{a}^s = \bar{a}^{(s,\ldots,s)}$. We denote by $\|\bar{a}\|_1 = \sum_{i=1}^{k} |a_i|$ and $\|\bar{a}\|_2 = \sqrt{\sum_{i=1}^{k} a_i^2}$ to the ℓ_1 and ℓ_2 norms respectively. The function $\mathrm{sign}(x)$ is equal to 1 if $x \geq 0$ and to -1 otherwise. $a =_? b$ is true if $a = b$ and false otherwise.

In our cryptographic primitives, the security parameter will be denoted by λ. We say that a function is negligible in λ if, for each positive polynomial f, it is smaller than $\frac{1}{f(\lambda)}$ for sufficiently big λ. For an algorithm $\mathcal{A}$, $\mathcal{A}(X)$ is the output of $\mathcal{A}$ on input X. We denote by 1^λ to the string $1 \cdots 1$ of length λ and by $\mathcal{A}(1^\lambda)$ to an algorithm $\mathcal{A}$ that takes an input of size $O(1^\lambda)$, that is, exponential in λ. We sometimes omit λ in 'negligible function' when it is clear we mean 'a function negligible in λ'.

Multiparty Protocols We consider a scenario with n parties that have private data and want to compute a model over it by exchanging messages. We assume that all parties have an identifier associated with the computation which is securely generated. We will not consider the problem of anonymization of identities of parties. For simplicity, we just enumerate them from 1 to n. Messages between parties are sent over a secure channel and are cryptographically signed, so that the receiver is ensured that the message comes from the claimed sender. We describe the cryptographic elements in more detail in Section 2.4. For a *multiparty protocol* or just *protocol* Π, we denote by $\Pi(X)$ to its output on input X.

Threat model In our model, parties are either *honest-but-curious* or *malicious*. These are common adversary models formalized by [70] and used in the design of many secure protocols. A *honest-but-curious* (*honest* for short) user will follow the protocol specification, but may use all the information obtained during the execution to infer information about other users. A honest user may accidentally drop out at any point of the execution (in a way that is independent of the private values X). On the other hand, a *malicious user* may deviate from the protocol execution (e.g, sending incorrect values or dropping out on purpose). Malicious users can collude, and thus will be seen as a single malicious party (the *adversary*) who has access to all information collected by malicious users. However, we assume that there is sufficient deterrence (in the form of punishment or banning) so that malicious users cannot risk that their deviations from the protocol are detected.

2.2 Notions of Privacy

We now discuss common notions of privacy in the context of the exploitation of data.

Cryptography-oriented notion First, we present a notion of privacy that is common in cryptographically secure protocols. Let $X_1, \ldots, X_n$ be sensitive data points such that for all $i \in [n]$, X_i is private to party i, and let Π be a protocol that takes as input $(X_i)_{i=1}^n$ and outputs Y. Then Π is private if it reveals no or negligible information about the input other than Y.

Although this notion might be sufficient for certain kinds of computations, it might not be enough in some cases. Consider for example that X_i is the risk of cancer of a patient i obtained by some inferences over its medical data, and suppose we want to compute the statistic $Y = \min_i X_i$. Even if the computation is private, it is revealed that one of the patient has a risk exactly equal to Y. If the risk is measured in the range $[0, 1]$ and the outcome Y is equal to 0.98, we immediately learn that all patients have a risk between 0.98 and 1.

Sensitive data can be similarly exposed if Y is the result of an aggregation. Consider that Y is the average salary of the employees of a company where the salaries are known to range between 2400€ and 10000€ per month. If Y is sufficiently close to 2400€ or to 10000€, the income of all employees can be estimated accurately. Now consider that this aggregate is computed again after one year in the same company, which hired one extra employee, but all others employees have the same contract as in the previous year. The result of this aggregate reveals the exact income of the newcomer. It has been shown that exact outcome of statistics and machine learning models can reveal sensitive data [106, 116, 67, 31].

k-anonymity A common notion when revealing a set of entries, each associated with an individual, is k-anonymity [126]. A computation satisfies k-anonymity if each entry is indistinguishable from $k - 1$ other entries. Databases can be k-anonymized by sufficiently generalizing the attributes of individuals or suppressing identity fields. However, this notion might not be enough. For example, when all individuals have an identical sensitive attribute, then the entries satisfy k-anonymity but the exact sensitive information is still revealed.

Randomized Response When performing a survey over sensitive data, a common method to protect individuals is randomized response. When a question is sensitive, the technique consists on answering it non-deterministically, giving individuals *plausible deniability* while the aggregation of all responses still remains a valid estimation of the true result. Consider a survey where it is wanted to know the number of people that committed a certain crime. To answer to this question, an individual

1. tosses a coin, if *heads* then it answers truthfully, and if *tails* then
2. it tosses a second coin and answers "Yes" if *heads* and "No" if *tails*.

Then if they answered "Yes", they can claim that the the first coin was *tails* and the second *heads*. While this method does not precisely define a notion of privacy, it is the principle in which the notion of Differential Privacy, that we describe below, relies on.

Differential Privacy Differential Privacy [58] quantifies the amount of information leaked in a protocol. To do that, it measures the difference in the output when a party $i \in [n]$ participates in the protocol with its private value X_i to that when i does not participate or is replaced by another individual. Here, the output is all the information publicly visible by untrusted parties.

Consider two possible input vectors (or also called *databases*) $X^A = (X_i^A)_{i=1}^n$ and $X^B = (X_i^B)_{i=1}^n$. They are called *neighboring* or *adjacent* inputs if they only differ on the input of one individual, i.e.

$$X^B = (X_1^A \dots X_{i-1}^A, X_i^B, X_{i+1}^A \dots X_n^A)$$

for some $i \in [n]$. The definition of neighboring inputs may change to suit a particular problem, but essentially it reflects the fact that some individual i is present in the computation or not. We will use the definition as described above, where an individual is replaced by another one with a different value.

Now we define Differential Privacy. Let Π be a probabilistic algorithm or, as usually called, *mechanism*. For $\varepsilon > 0$ and $\delta \in [0, 1)$, a mechanism Π satisfies (ε, δ)-Differential Privacy if for every pair of neighboring inputs X^A and X^B and for all sets of possible outputs $\mathcal{O}$, we have that

$$\Pr(\Pi(X^A) \in \mathcal{O}) \leq e^{\varepsilon} \Pr(\Pi(X^B) \in \mathcal{O}) + \delta. \tag{2.1}$$

Here, ε determines the upper bound on the relative change in probability of the output, while δ gives more flexibility by capturing some cases where the strict bound $e^{\varepsilon} \Pr(\Pi(X^B) \in \mathcal{O})$ might not hold. δ must be sufficiently small to ensure that the probability of leakage is negligible. It is usually in $O(1/n^2)$ or, for more safety, in $O(1/2^n)$.

The above definition defines a framework to precisely measure privacy leakage. Consider the example given for randomized response. The sensitive data X is whether the individual in question committed a crime or not and Y is the answer. The two possible inputs are $X =$ "Yes" and $X =$ "No", which are neighboring. We have that

$$\frac{\Pr(Y = \text{“Yes”} | X = \text{“Yes”})}{\Pr(Y = \text{“Yes”} | X = \text{“No”})} = \frac{\Pr(Y = \text{“No”} | X = \text{“No”})}{\Pr(Y = \text{“No”} | X = \text{“Yes”})} = \frac{3/4}{1/4} = 3.$$

Then, for the two possible outcomes R and R', we have that

$$\Pr(Y = R | X = R) = 3 \Pr(Y = R | X = R')$$

and the mechanism is $(\ln 3, 0)$-Differentially Private.

DP Flavors In order to satisfy different needs, many variations of classic DP have been studied. Some of them are *Concentrated* [61], *Renyi* [109], *Shuffle* [38] and *Network* [49] *DP*. In a similar direction, *Pufferfish Privacy* [90] is a customizable framework which allows to create and analyze definitions of privacy.

Empirical Privacy DP provides strong guarantees of privacy. However, it is hard to define which values of ε that provide a good level of privacy in practice. This also depends on the type of computation. In addition, some machine learning applications such as speech processing do complex transformations of the input, and it is hard to prove that they satisfy DP. In these applications, a common notion of privacy is to empirically prove that they are resistant to existent attacks, such as linkage [95] or membership of a group [74]. However, this notion has the important issue that it does not prove privacy for future attacks or variations of existent attacks.

This dissertation focuses on the theoretical guarantees provided by the classical definition of Differential Privacy.

2.3 Algorithms for Differential Privacy

Now we describe common mechanisms to achieve DP. They rely on the randomization of the outcome of protocols. This is obtained by adding noise according to some probability distribution. The amount of noise depends on how sensitive is the outcome to the input of one individual. Hence, for $i \in \{1, 2\}$ we define the ℓ_i-*sensitivity* of protocol Π by

$$\Delta_i \Pi = \max_{X^A \sim X^B} \|\Pi(X^A) - \Pi(X^B)\|_i$$

where $X^A \sim X^B$ denotes that X^A and X^B are neighboring inputs and $\|\cdot\|_i$ is either the ℓ_1 or ℓ_2 norm. The sensitivity is an upper bound of the change of the output when one individual is replaced by another. We outline below two mechanisms to obtain differentially private protocols.

Laplace Mechanism The Laplace distribution, denoted $Lap(b)$ is defined by

$$P_{Lap(b)}(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$$

For a protocol Π with outcome $Y \in \mathbb{R}^k$, the Laplace Mechanism [60] applied to Π, defines a Protocol Π_L that takes the input of Π, computes Y and outputs

$$\hat{Y} = Y + (\eta_1, \ldots, \eta_k).$$

where the η_i's are i.i.d. samples of $Lap(\Delta_1 \Pi/\varepsilon)$ for some $\varepsilon > 0$.

Theorem 1 (Laplace Mechanism)**.** *The Laplace Mechanism is* $(\varepsilon, 0)$*-Differentially Private.*

Proof. See Theorem 3.6 of [60]. □

Gaussian Mechanism The Gaussian Mechanism [60] adds noise according to the normal distribution, denoted by $\mathcal{N}(\mu, \sigma^2)$ and defined by

$$P_{\mathcal{N}(\mu,\sigma^2)}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right).$$

Let Π be as defined for the Laplace Mechanism. The Gaussian Mechanism applied to Π and derives a new protocol Π_G with output

$$\hat{Y} = Y + (\eta_1, \ldots, \eta_k).$$

where the η_i's are i.i.d. samples of $\mathcal{N}(0, \sigma^2)$ for some $\sigma > 0$.

Theorem 2. *Recall that $\Delta_2\Pi$ is the ℓ_2-sensitivity of Π as previously defined. Let $\varepsilon \in (0, 1)$ and let $c^2 > 2\ln(1.25/\delta)$. Then, the Gaussian Mechanism with $\sigma \geq c\Delta_2\Pi/\varepsilon$ is (ε, δ)-Differentially Private.*

Proof. See Theorem A.1 in the Appendix of [60]. □

Differential privacy has other practical properties. For example, upper bounds (even if possibly loose) of the level of privacy of a composition of mechanisms can be directly derived from the DP parameters of the sub-mechanisms. Also, it is clear that any post-processing of the outcome of a mechanism, satisfies the same privacy as this mechanism. Elementary properties can be found in [60].

Central and Local Privacy Above, we described two common mechanisms to make protocols differentially private by applying noise. We now distinguish between classic DP (also called centralized DP) and local DP (LDP). Classic DP adds to an aggregate sufficient noise to make the aggregate differentially private. LDP adds to the data owned by a single data owner sufficient noise to make this data differentially private by itself. The latter requires more noise, but once data is LDP one can publish it and do any operation on it without worrying about privacy.

Consider the task of computing a (ε, δ)-differentially private average of the input values $(X_1, \ldots, X_n) \in [a, b]^n$ for some range $[a, b] \subset \mathbb{R}$. We use the Gaussian Mechanism to obtain DP. In the central DP setting, all private values are averaged without revealing them, e.g., by sending them to a trusted curator that computes $X^{avg} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and reveals the perturbed average $\hat{X}^{avg} = X^{avg} + \eta$ after adding noise $\eta \sim \mathcal{N}(0, \sigma^2)$. The only exposed value is $\hat{X}^{avg}$. As the ℓ_2-sensitivity of X^{avg} is $(b-a)/n$, the variance σ^2 of the noise must be at least

$$2\ln\left(\frac{1.25}{\delta}\right)\frac{(b-a)^2}{n^2\varepsilon^2}.$$

Now consider the LDP setting. The average is computed between untrusted parties. Each party i adds noise $\eta_i \sim \mathcal{N}(0, \sigma^2_{local})$ to its private value X_i before revealing $\hat{X}_i = X_i + \eta_i$. The perturbed average is equal to $\hat{X}^{avg} = \frac{1}{n}\sum_{i=1}^{n} \hat{X}_i$. Here, all $\hat{X}_i$'s are exposed. The sensitivity is $b - a$ and therefore it is required that $\sigma^2_{local} \geq 2\ln(1.25/\delta)(b-a)^2/\varepsilon^2$. The variance σ^2 of the estimate $\hat{X}^{avg}$ is

$$Var\left(\frac{1}{n}\sum_{i=1}^{n}\eta_i\right) = \frac{1}{n^2}\sum_{i=1}^{n}\sigma^2_{local} > 2\ln\left(\frac{1.25}{\delta}\right)\frac{(b-a)^2}{n\varepsilon^2}.$$

This is a factor n bigger than in the classic DP setting. This may make LDP prohibitive unless a massive amount of parties participate in the average.

Privacy in Machine Learning When computing a machine learning model $\mathcal{M}$ with parameters θ, achieving DP in the central setting requires to compute $\hat{\theta} = \theta + \eta$ where η is defined according to some privacy mechanism and then releasing $\hat{\theta}$.

Achieving accurate models with LDP is more challenging. Techniques such as *federated learning* [84] and distributed empirical risk minimization can be used. In these settings, each party i holds a private dataset $\mathcal{D}_i$ and the goal is to find θ^* such that $\theta^* \in \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} f(\theta; \mathcal{D}_i)$ where f is some loss function. Popular algorithms [97, 125, 104, 80, 3] all follow the same high-level procedure: at round t, each party i computes a local update θ_i^t based on $\mathcal{D}_i$ and the current global model θ^{t-1}, and the updated global model is computed as $\theta^t = \frac{1}{n}\sum_u \theta_u^t$. To achieve DP, each party adds noise to each update θ_i^t before sharing it. As said, the amount of noise is substantially higher with LDP mechanisms. These settings have the disadvantage that, as many partial updates are exposed, they increase the attack surface for external parties.

In a decentralized setting where there is no central trusted party, LDP is one possible strategy. To reduce the amount of noise needed, one can use encryption strategies to hide inputs and intermediate computations, such as *secure aggregation* [59, 34, 122, 22, 80, 79], other kinds of *multiparty computation* (MPC) [45, 78] or *fully homomorphic encryption* [120], can be used to reduce the noise. However, with the exception of very recent contributions [12, 124], they relax their assumptions, for example by trusting the computation to a few honest-but-curious parties assumed not to collude, or do not scale well to a large number of parties.

2.4 Basic Cryptographic Tools

Below, we outline basic cryptographic primitives and concepts that are related to our following chapters.

Signatures A *signature scheme* allows parties to authenticate their messages. It is a triplet of algorithms that consist on:

- a probabilistic *key generation* algorithm G that takes as input a string of length $O(1^\lambda)$ and produces a pair (K_p, K_s) of matching public and secret keys,
- a probabilistic *signing* algorithm S that takes a message m and a secret key K_s and produces a signature s
- a *verification* algorithm V which given a signature s, a message m and a public key K_p returns **accept** if s is a valid signature of m with respect to K_p, or **reject** otherwise

The scheme must satisfy $\Pr[(K_p, K_s) \leftarrow G(1^\lambda), V(S(m, K_s), m, K_p) = \textbf{accept}] = 1$ for correctness. For security, it is required that no probabilistic adversary that runs in polynomial time must be able to generate valid signatures with more than negligible probability in λ. The latter must hold even if the adversary can query the signing algorithm with arbitrary messages as input (which are not equal to the message targeted for forgery) and see its output. A practical instance of a signature scheme is ECDSA[81].

Public-Key Encryption A public key encryption scheme is a triplet of algorithms composed of

- a probabilistic *key generation* algorithm G that takes as input a string of length $O(1^\lambda)$ and generates a pair (K_p, K_s) of a public key and a secret key
- a probabilistic *encryption* algorithm E that takes a message m and a public key K_p and generates a *ciphertext* c
- a *decryption* algorithm D that takes a ciphertext c and a private key K_s and generates a message m

Let (K_p, K_s) be the output of $G(1^\lambda)$. For correctness, a scheme must satisfy

$$D(E(m, K_s), K_p) = m$$

for all messages m. A scheme is secure (or *semantically secure*) if for all possible messages m, no probabilistic polynomial time attacker who only has access to K_p (which can use to generate ciphertexts for any message $m' \neq m$) can distinguish the ciphertext $E(m, K_s)$ from a random number except with negligible probability in λ.

Let M be the domain of messages and C that of ciphertexts. An encryption scheme is *partially homomorphic* if, for some operations $+$ and $+'$, $(M, +)$ and $(C, +')$ are groups, and for every pair of messages $m_1, m_2 \in M$, it holds that $D(E(m_1)+'E(m_2)) = m_1+m_2$. This property allows external parties to perform computations involving the operator $+$ over private values without learning any information about the messages. A semantically secure and partially homomorphic encryption scheme is ElGamal [62, 92].

Hash Functions For a positive integer T, a (cryptographic) *hash function* is a deterministic function $H : \mathbb{Z} \to \{0,1\}^T$ that is fast to evaluate, but inverting its outputs or finding two inputs with the same output can be done only with negligible probability in T for a polynomial time algorithm. The output of a hash function is indistinguishable from a random number drawn from $\{0,1\}^T$. Practical instances of H can be found in [52].

Pseudo-Random Generators A (secure) *pseudo-random number generator* [132] (PRG) is a deterministic function $G : \{0,1\}^k \to \{0,1\}^{p(k)}$ for some polynomial p with $p(k) > k$ such that for any randomized polynomial time algorithm $A : \{0,1\}^{p(k)} \to \{0,1\}$ there holds

$$|P_{x \leftarrow_R \{0,1\}^k}(A(G(x)) = 1) - P_{x \leftarrow_R \{0,1\}^{p(k)}}(A(x) = 1)| \leq \mu(k)$$

for some function μ negligible in k. In other words, a PRG is a function which takes a string x as input and outputs a longer string $G(x)$ which cannot be distinguished from a random sequence by a polynomial time algorithm. It can be instantiated with AES-CTR in practice.

Commitments Commitment schemes, first introduced in [20], allow for committing to values while keeping them hidden. It consists of a pair of (computationally efficient) probabilistic algorithms $(Setup, Com)$. Algorithm $Setup$ is executed once, with randomness $t \in O(1^\lambda)$ as input, and outputs a tuple $\Theta \leftarrow Setup(t)$ of parameters of the scheme. The algorithm Com with parameters Θ, denoted Com_Θ, is a function $Com_\Theta : \mathcal{B}_\Theta \times \mathcal{R}_\Theta \to \mathcal{C}_\Theta$,

where $\mathcal{B}_\Theta$ is called the message space, $\mathcal{R}_\Theta$ the randomness space, and $\mathcal{C}_\Theta$ the commitment space. For a message $x \in \mathcal{B}_\Theta$, the algorithm draws $r \in \mathcal{R}_\Theta$ uniformly at random and computes a commitment $P \leftarrow Com_\Theta(x, r)$. We say that (x, r) is an *opening* of P. For simplification and when r is not relevant, we relax the notation $Com_\Theta(x, r)$ to $Com_\Theta(x)$ and assume r is drawn appropriately. We outline three properties of commitments below.

- A commitment is *binding* if there exists no polynomial time algorithm $\mathcal{A}$ that can find $x, y \in \mathcal{B}_\Theta$, $r, s \in \mathcal{R}_\Theta$ such that $x \neq y$ and $Com_\Theta(x, r) = Com_\Theta(y, s)$. The binding property typically depends on the input t of $Setup$ not being biased, in particular, it must be hard to guess non-trivial information about t.
- A commitment scheme is *hiding* if, for all secrets $x \in \mathcal{B}_\Theta$ and given that r is chosen uniformly at random from $\mathcal{R}_\Theta$, the commitment $Com_\Theta(x, r)$ does not reveal any information about x.
- A commitment scheme is *homomorphic* if $\mathcal{B}_\Theta$, $\mathcal{R}_\Theta$ and $\mathcal{C}_\Theta$ are abelian groups, and for all $x, y \in \mathcal{B}_\Theta$, $r, s \in \mathcal{R}_\Theta$ we have

$$Com_\Theta(x, r) + Com_\Theta(y, s) = Com_\Theta(x + y, r + s).$$

Please note that the three occurrences of the '$+$' sign in the above definition are operations in three different spaces, and hence may have different definitions and do not necessarily correspond to the regular addition of numbers. Note that the authenticity of commitments can always be trivially proven by revealing the committed value. This is sometimes known as *revealing*.

Discrete Logarithm Assumption (DLA) The security of our protocols relies on the Discrete Logarithm Assumption. Let $\mathbb{G}$ be a cyclic multiplicative group of large prime order $q \in O(1^\lambda)$, and g and h two elements chosen independently and uniformly at random from $\mathbb{G}$. Then, no probabilistic polynomial time algorithm $\mathcal{A}$ that takes as input the tuple $(\mathbb{G}, q, g, h)$ outputs a value b such that $P(g^b = h)$ is non-negligible in λ. For more details on the DLA, see Chapter 7 of [89].

Pedersen Commitments We use the Pedersen commitment scheme [117]. Let p and q be two large primes such that q divides $p - 1$, and a let $\mathbb{G}$ be the cyclic subgroup of order q of the multiplicative group $\mathbb{Z}_p^*$. Primes p and q are such that $\mathbb{G}$ is suitable for the DLA (see [89]). We have that $\mathbb{G} = \{a^i \mod p \mid 0 \leq i < q\}$ for any $a \in \mathbb{G}$ distinct to 1, which means that any member of $\mathbb{G}$ except 1 can generate it. Pedersen's $Setup$ function samples at random parameters $\Theta = (g, h)$ where g, h are two generators of $\mathbb{G}$. We will also refer to g and h as *bases*. Recalling the definition of commitments, we have $\mathcal{B}_\Theta = \mathcal{R}_\Theta = \mathbb{Z}_q$, and $\mathcal{C}_\Theta = \mathbb{G}$. The commitment function Com_Θ is defined as

$$\begin{aligned} Com_\Theta &: \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{G} \\ Com_\Theta(x, r) &= g^x \cdot h^r, \end{aligned} \tag{2.2}$$

where $(\cdot)$ is the product modulo p of group $\mathbb{G}$, x is the secret and r is the randomness. Pedersen commitments are *homomorphic* (in particular, $Com_\Theta(x + y, r + s) = Com_\Theta(x, r) \cdot Com_\Theta(y, s)$), hiding and binding under the DLA.

The $Setup$ algorithm only requires public randomness to sample random bases of $\mathbb{G}$. This can be done without a trusted party. We show how to generate such unbiased randomness in the presence of malicious participants in Section 3.6.3.

Vector Commitments For efficiency in communication, we also consider a generalization for vectors of Pedersen commitments. Here, the $Setup$ algorithm outputs a vector $\bar{g} = (g_1, \ldots, g_k)$ of bases sampled at random from $\mathbb{G}$. A commitment $P \in \mathbb{G}$ of a vector $\bar{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$ satisfies $P = \bar{g}^{\bar{x}}$. We have that $\bar{x} = (\bar{x}', r)$, where $\bar{x}'$ is the data and r is sampled uniformly at random from $\mathbb{Z}_q$, therefore P is uniformly distributed in $\mathbb{G}$ and the scheme is hiding. Binding and homomorphic properties also hold. For the latter, given commitments P and Q of $\bar{x}$ and $\bar{y}$ respectively, PQ is a commitment of $\bar{x} + \bar{y}$.

Arithmetic Circuits We define arithmetic circuits, which will be used to relate statements over committed values. An *arithmetic circuit* (or just *circuit*) $C : \mathbb{Z}_q^k \to \mathbb{Z}_q^s$ is a function that only contains additions and multiplications modulo q. In the following of the dissertation, we will define circuits using the notation

$$C(a; i_1, \ldots, i_k) := \begin{bmatrix} o_1 \\ \vdots \\ o_s \end{bmatrix},$$

where $(i_1, \ldots, i_k)$ is the input, $(o_1, \ldots, o_s)$ is the output, and a are constants that may change the circuit structure, for example, in the case we are defining a family of similar circuits.

2.5 Zero Knowledge Proofs

We use *Zero Knowledge Proofs* (ZKP) [73] to prove statements about private committed values. Basic concepts about ZKP are explained in Section 2.5.1. We use two ZKP techniques: Σ-protocols [46] and a compressed version of them by [5] with reduced communication cost. They can be used to prove arithmetic relations, range membership and certain types of logical formulas involving private values. We explain the classic (non-compressed) Σ-protocols and show some examples of them in Section 2.5.2. In Section 2.5.3, we show a key protocol used to understand compressed Σ-protocols. After that, in Section 2.5.4 we show the compression technique of [5]. In Section 2.5.5, we present compressed protocols to prove arithmetic circuits. Finally, Section 2.5.6 has an application for compressed range proofs and Section 2.5.7 discusses the costs of the compressed techniques.

2.5.1 Zero Knowledge Proofs and Arguments

ZKP are a special case of *interactive Proofs of Knowledge* (PoK). We first define PoK. For a NP relation $\mathcal{R}$, a PoK is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which $\mathcal{P}$ tries to prove to $\mathcal{V}$ that he knows a *witness* w such that $(a, w) \in \mathcal{R}$ for a public statement a. At the end of the protocol, $\mathcal{V}$ either *accepts* or *rejects* the proof. We denote by $(a; w)$ to a member of a relation or an input of a protocol, using a semicolon to separate the public statement a from the private witness w. The tuple of all messages in a proof is called the *conversation* or *transcript*. A ZKP is a PoK that satisfy the following properties:

- **Completeness:** a proof is *complete* if $\mathcal{V}$ always accepts the proof when $(a; w) \in \mathcal{R}$ and $\mathcal{P}$ knows w.

- **Soundness:** a proof is *sound* if any prover whose proof for statement a is accepted by the verifier knows a valid witness w such that $(a; w) \in \mathcal{R}$ with overwhelming probability. The notion of soundness we use is called *witness extended emulation* [98]. Proofs that are sound only if the prover is computationally bounded are also called *arguments*.

- **Zero Knowledge:** a proof is *zero knowledge* if its transcript reveals no or negligible information about the witness other than its validity.

2.5.2 Basics of Σ-protocols

Here we explain the basics to understand classic Σ-protocols proposed by [46, 47]. While the proofs defined therein are applicable for a wide family of commitments, we instantiate them for the Pedersen scheme.

A Σ-protocol is a 3-message PoK with a transcript of the form (m_1, t, m_2), where m_1 and m_2 are messages of $\mathcal{P}$, and t is a message of $\mathcal{V}$ called *challenge*, which is uniformly distributed in some challenge space S, in our case, of size $O(1^\lambda)$.

The security of these protocols rely on $\mathcal{V}$ behaving honestly in the draw of challenges. To circumvent this in settings where no trusted verifier is available, Σ-protocols can be transformed to non-interactive using the *Fiat-Shamir heuristic* [66, 16]. This consists on replacing the challenge of $\mathcal{V}$ by a hash function that computes t from m_1 and the statement of the proof. Then, a prover can generate a transcript by itself and send it to an untrusted verifier. This transformation is proven secure when the hash function is modeled as a random oracle [13]. We will use this transformation for both classic and compressed Σ-protocols described in the following sections. However, for simplicity, we will describe protocols in their interactive form.

Below, we describe some existent protocols, explain their complexity and refer to the literature for other aspects such as security proofs. The protocols are secure under the DLA and instantiated for Pedersen commitments, described in Section 2.4. Recall parameters $\mathbb{Z}_p^*$, $\mathbb{Z}_q$, $\mathbb{G}$, g and h defined therein. For some building blocks, we use extra pairs of elements of $\mathbb{G}$ as bases of for Θ, where bases belonging to the same pair are always assumed to be chosen independently at random from $\mathbb{G}$. Recall that p is the order of $\mathbb{Z}_p^*$, of which $\mathbb{G}$ is subgroup of. Products and exponentiations of members of $\mathbb{G}$ are group operations, which means that they are implicitly modulo p. To analyze the computational cost of the building blocks, we count group exponentiations (GEX), which are the dominant operations. For a ZKP, we call the *size of a proof* to the size of the transcript. To describe sizes of proofs, we overload the notation and use $\mathbb{G}$ for the size in bits of an element of this group. In that context, $\mathbb{G}$ is equal to $\lfloor \log_2 p \rfloor + 1$, but in some implementations the size can be different so we abstract from details.

Basic proof of knowledge This proof is also known as the basic protocol. It allows $\mathcal{P}$ to prove that he knows an opening of a commitment P. That is, a value x and randomness r such that $P = g^x h^r$. This is a proof for the relation

$$\{(P \in \mathbb{G}; x, r \in \mathbb{Z}_q) : P = g^x h^r\}.$$

The protocol goes as follows:

1. $\mathcal{P}$ draws $a, b \leftarrow_R \mathbb{Z}_q$, computes $A \leftarrow g^a h^b$ and sends A to $\mathcal{V}$.

2. $\mathcal{V}$ draws a challenge $t \leftarrow_R \mathbb{Z}_q$ and sends it to $\mathcal{P}$.

3. $\mathcal{P}$ computes $d \leftarrow a + xt \pmod q$ and $e \leftarrow b + rt \pmod q$ and sends (d, e) to $\mathcal{V}$.

4. $\mathcal{V}$: if $g^d h^e = AP^t$ then **accept**, else **reject**

The computational cost of the proof is 2 GEX for $\mathcal{P}$, and 3 for $\mathcal{V}$. The proof size is of $3\mathbb{G}$ bits. This protocol was first proposed by [121] for a single base, and adapted to many bases (in our case, g and h) in [35].

We provide an intuition of how ZKP properties are obtained for the basic proof of knowledge. Completeness follows directly from the homomorphic property. For soundness, consider a prover $\mathcal{P}^*$ that, by following the protocol, can produce an accepting transcript $(A, t_1, (d_1, e_1))$ with significant probability. Then it is shown that also with non-negligible probability, by using $\mathcal{P}^*$'s strategy many times, another accepting transcript of the form $(A, t_2, (d_2, e_2))$ can be produced, where the first message is equal in both transcripts and such that $t_1 \neq t_2$. Now, since both transcripts are accepting, we have $g^{d_1} h^{e_1} = AP^{t_1}$ and $g^{d_2} h^{e_2} = AP^{t_2}$ so $\mathcal{P}^*$ can efficiently compute the witness $x = \frac{d_1 - d_2}{t_1 - t_2}$ and $r = \frac{e_1 - e_2}{t_1 - t_2}$ which is valid as $g^x h^r = P$. Either $\mathcal{P}^*$ already knew it or he can efficiently compute the discrete logarithms base g of h, which is a contradiction due to the DLA.

Zero knowledge is obtained by showing that all information seen in the proof is randomness that does not depend on the secrets. By only knowing the statement P one can sample $d' \leftarrow_R \mathbb{Z}_q$, $e' \leftarrow_R \mathbb{Z}_q$ and $t' \leftarrow_R \mathbb{Z}_q$, compute $A' = P^{-t'} g^{d'} h^{e'}$ and generate the transcript $(A', t', (d', e'))$ has the same distribution as a conversation between an honest prover and an honest verifier. Note that, as it is computed in reverse, $(A', t', (d', e'))$ cannot be efficiently produced by a dishonest prover in the actual protocol, as the order of messages cannot be changed.

In the following, classic Σ-protocols use similar arguments as the ones described for the basic proof of knowledge to prove their properties. In particular, they can all generate accepting transcripts "in reverse" even if witnesses are not known, as shown to prove the zero knowledge property above. We will not describe their security proofs, but we will point to the literature where these can be found.

Proof of equality This proof allows $\mathcal{P}$ to prove that he committed to the same private value x in two different commitments $P = g_1^x h_1^r$ and $P' = g_2^x h_2^{r'}$. That is, the relation

$$\{(P, P' \in \mathbb{G}; x, r, r' \in \mathbb{Z}_q) : P = g_1^x h_1^r \wedge P' = g_2^x h_2^{r'}\}.$$

The pairs of bases (g_1, h_1) and (g_2, h_2) might be different. This proof is described by [36] for a single base and a generalization for many bases can be found in [35]. The protocol goes as follows:

1. $\mathcal{P}$ generates $a, b, c \leftarrow_R \mathbb{Z}_q$, computes $A \leftarrow g_1^a h_1^b$ and $B \leftarrow g_2^a h_2^c$ and sends (A, B) to $\mathcal{V}$.

2. $\mathcal{V}$ draws a challenge $t \leftarrow_R \mathbb{Z}_q$ and sends it to $\mathcal{P}$.

3. $\mathcal{P}$ computes $d \leftarrow a + xt \pmod q$, $e \leftarrow b + rt \pmod q$ and $f \leftarrow c + r't \pmod q$ and sends (d, e, f) to $\mathcal{V}$.

4. $\mathcal{V}$: if $g_1^d h_1^e = AP^t$ and $g_2^d h_2^f = B(P')^t$, then **accept** else **reject**.

The proof requires the computation of 4 GEX for $\mathcal{P}$ and 6 for $\mathcal{V}$. The proof size is of $5\mathbb{G}$ bits.

Composition Let Π_1 and Π_2 be two Σ-protocols for ZKP of relations $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively. A single Σ-protocol to prove the relation

$$\mathcal{R}_\wedge = \{(a, a'; w, w') : (a; w) \in \mathcal{R}_1 \wedge (a'; w') \in \mathcal{R}_2\}$$

can be easily constructed. Basically, we just make both proofs to share the same challenge. The protocol is as follows. The first message is (m_1, m_1'), where m_1 and m_1' are exactly as generated for the first messages of Π_1 and Π_2 respectively. Then $\mathcal{V}$ generates the challenge t and sends it to $\mathcal{P}$. Next, $\mathcal{P}$ generates the third message (m_2, m_2') where m_2 and m_2' are generated from (m_1, t) and from (m_1', t) as they will be generated for the third messages of Π_1 and Π_2 respectively. Finally $\mathcal{V}$ verifies (m_1, t, m_2) and (m_1', t, m_2') are accepting as in Π_1 and Π_2. Using this composition, if we wish to prove statements over many relations, we can do it with a single Σ-protocol.

Another technique allows to construct ZKPs of disjunctions of statements. The work by [48] proposed a Σ-protocol for the relation

$$\mathcal{R}_\vee = \{(a, a'; w, w') : (a; w) \in \mathcal{R}_1 \vee (a'; w') \in \mathcal{R}_2\},$$

where $\Pi_1, \Pi_2, \mathcal{R}_1$ and $\mathcal{R}_2$ are defined as for conjunctions. We briefly describe the protocol below. Without loss of generality, assume that $\mathcal{P}$ knows $(a; w) \in \mathcal{R}_1$, but he does not know $(a'; w') \in \mathcal{R}_2$ and that $\mathcal{V}$ does not know which of the two witnesses is known by $\mathcal{P}$. As Π_2 is zero knowledge, an accepting transcript (m_1', t_b, m_2') can be generated in reverse as shown for the proof of knowledge above and without $\mathcal{P}$ knowing a valid witness.

1. $\mathcal{P}$ generates (m_1', t_b, m_2') in reverse and generates m_1 as he would normally do for the first message of Π_1.

2. $\mathcal{P}$ sends (m_1, m_1') to $\mathcal{V}$.

3. $\mathcal{V}$ draws a challenge $t \leftarrow_R \mathbb{Z}_q$ and sends it to $\mathcal{P}$.

4. $\mathcal{P}$ computes $t_a \leftarrow t - t_b \bmod q$ and computes m_2 from m_1 and t_a as he would have done in the original protocol Π_1.

5. $\mathcal{P}$ sends (m_2, m_2') to $\mathcal{V}$.

6. $\mathcal{V}$ checks that (m_1, t_a, m_2) and (m_1', t_b, m_2') are accepting as in protocols Π_1 and Π_2, and that $t_a + t_b \bmod q =_? t$. If all checks pass, then **accept**, else **reject**.

Here, $\mathcal{P}$ can generate (m_1, t_a, m_2) as he knows a witness for $\mathcal{R}_1$ and he can simulate an accepting transcript for $\mathcal{R}_2$. As $t_a + t_b \bmod q = t$, $\mathcal{V}$ is ensured that at least either t_a or t_b is unpredictable for $\mathcal{P}$ when generating the first message of the proof, so he would not be able to pass the proof with significant probability unless he knows at least one witness. We refer to the paper for the security proofs and further details.

The composition techniques outlined above allow one to prove logical formulas of conjunctions and disjunctions of statements. The computational and communication cost for the resulting protocol is similar to the sum of the costs of the proof of each composed statement.

Proof of linear relation Let $\bar{x} = (x_i)_{i=1}^k$ for some $k > 0$ be a vector of private values, $\bar{P} = (P_i)_{i=1}^k$ a public vector of commitments such that P_i is a commitment of x_i, $\bar{a} = (a_i)_{i=1}^k$ a public vector of integer coefficients, and b a public scalar. The goal of $\mathcal{P}$ is to prove the equality $\langle \bar{x}, \bar{a} \rangle = b \pmod q$, where $\langle \cdot, \cdot \rangle$ is the inner product. The proof follows almost directly from the homomorphic property of commitments. We have that $P_L = \prod_{i=1}^k P_i^{a_i}$ is a valid commitment of $\langle \bar{x}, \bar{a} \rangle$. Then, $\mathcal{P}$ and $\mathcal{V}$ prove that values committed with P_L and $P_b = g^b$ are equal. Note that it is not necessary to use the base h to compute the commitment P_b as b is publicly known and hence we can use randomness equal to 0. The proven relation is

$$\Big\{(\bar{P} \in \mathbb{G}^k, \bar{a} \in \mathbb{Z}_q^k, b \in \mathbb{Z}_q; \bar{x}, \bar{r} = (r_i)_{i=1}^k \in \mathbb{G}^k) \\ : \bigwedge_{i=1}^k P_i = g^{x_i} h^{r_i} \wedge \langle \bar{a}, \bar{x} \rangle = b \pmod q \Big\}.$$

where $\bar{r}$ is the randomness for each commitment. For $\mathcal{P}$ and $\mathcal{V}$, the proof requires $k+1$ GEX to compute P_L and P_b plus one proof of equality. This leads to a final cost of $k+5$ GEX for $\mathcal{P}$ and $k+7$ for $\mathcal{V}$. Considering that initial commitments $\bar{P}$ are already shared with $\mathcal{V}$ and that $\mathcal{P}$ has already proven knowledge of them, the proof size is equal to that of the proof of equality, which is $5\mathbb{G}$ bits.

Proof of a committed bit A handy special case of the composition using disjunction is a proof that a secret b is a bit, i.e. that $\{b = 0 \vee b = 1\}$. This instantiation is described by ([99] Section 3.2 therein). It requires 3 GEX for $\mathcal{P}$ and 4 for $\mathcal{V}$. The size of the proof is of $4\mathbb{G}$ bits.

Range proof Here, the statement to prove is that, for a positive integer $M < q - 1$ of B_M bits and a commitment P, $\mathcal{P}$ knows an opening x of P that lies in the range $[0, M]$. This is a folklore proof that can be done by committing to the bits $b_1, \ldots, b_{B_M}$ of the bit representation of x, i.e. that $x = \sum_{i=1}^{B_M} 2^{i-1} b_i$. For $i \in \{1, \ldots, B_M\}$ it is proven that b_i is a bit with the protocol mentioned above. The validity of the decomposition of x can be proven by a linear proof. This implies that $x \in [0, 2^{B_M} - 1]$. For ranges whose upper bound is not of the form $2^{B_M} - 1$, $\mathcal{P}$ commits to $M - x$ and proves that it also lies in $[0, 2^{B_M} - 1]$. This finishes the proof for arbitrary ranges.

Committing to every bit b_i has a cost of B_M GEX for $\mathcal{P}$. The cost of proofs of bits is dominated by $3B_M$ GEX for $\mathcal{P}$ and $4B_M$ for $\mathcal{V}$. The proof that these bits are the decomposition of x is dominated by B_M GEX for both $\mathcal{P}$ and $\mathcal{V}$. For arbitrary ranges, the cost duplicates as described above. The total cost of the proof is dominated by $10 \log_2(M)$ GEX for $\mathcal{P}$ and the same cost for $\mathcal{V}$. The size of the proof is dominated by $10 \log_2(M)\mathbb{G}$ bits.

Proof of product In this proof, $\mathcal{P}$ wants to prove the knowledge of openings a, b and d of commitments $P_a = g^a h^{r_1}$, $P_b = g^b h^{r_2}$ and $P_d = g^d h^{r_3}$ and that they satisfy $ab = d \pmod q$. This proof can be done in a straightforward way with the knowledge and equality proofs. We use the fact that, by properties of our cyclic group $\mathbb{G}$, P_a is not only a commitment but also a base, and that $P_d = (P_a)^b h^{(r_3 - b r_1)}$ is a valid commitment for b using bases (P_a, h). For the proof, $\mathcal{P}$ first proves the knowledge of an opening of P_a, and then it proves the equality of two openings P_b and P_d with the proof of equality, where

commitments are computed with the pairs of bases (g, h) and (P_a, h) respectively. If $\mathcal{P}$ does not know the commitment of a product of ab hidden in P_d it would not pass the proof. The proof is the instantiation for Pedersen commitments of the protocol by [47], and requires 6 GEX for $\mathcal{P}$ and 9 for $\mathcal{V}$. The proof size is of $8\mathbb{G}$ bits.

Proof of modular sum Here, $\mathcal{P}$ wants to prove that, for a public modulus $M < q/2$, a public value $t \in [0, M-1]$ and two secrets $x, z \in [0, M-1]$, the relation $x = z + t \mod M$ is satisfied. Let P_x and P_z be the commitments of x and z respectively. First, $\mathcal{P}$ does two range proofs to prove that x and z lie in $[0, M-1]$. Then it proves that an auxiliary value b is a bit. Finally it proves that $x = z + t - bM \pmod q$ with a linear proof. Because of their range, we have that $z + t < q$, hence the modulus q does not interfere and the linear equality holds if and only if the modular sum is satisfied. This proof is inspired by the more general proof of modular sum proposed by [29]. Its complexity is given by the composition of the proofs mentioned above, where the dominant term comes from the two range proofs, and amounts to $20\log_2(M)$ GEX for $\mathcal{P}$ and the same for $\mathcal{V}$. The size of the proof is of $20\log_2(M)\mathbb{G}$ bits.

2.5.3 Linear relations for compression

We now show a Σ-protocol to prove linear relations over secret committed values which is the key to understand compressed Σ-protocols. It relies on vector commitments explained in Section 2.4 and its parameters: the commitment domain $\mathbb{Z}_q$, our underlying cryptographic group $\mathbb{G}$ and vector of elements $\bar{g} \in \mathbb{G}^k$. Let $L : \mathbb{Z}_q^k \to \mathbb{Z}_q$ be a linear function in $\mathbb{Z}_q$. That is, $L(x_1, \ldots, x_k) = a_1x_1 + \cdots + a_kx_k$ for coefficients $a_1, \ldots, a_k \in \mathbb{Z}_q$. For a vector commitment $P \in \mathbb{G}$ and a value $y \in \mathbb{Z}_q$, a prover $\mathcal{P}$ proves to know an opening $\bar{x} \in \mathbb{Z}_q^k$ of P such that $L(\bar{x}) = y$. We formally describe our linear relation by

$$\mathcal{R}_L = \{(P \in \mathbb{G}, y \in \mathbb{Z}_q; \bar{x} \in \mathbb{Z}_q^k) : P = \bar{g}^{\bar{x}} \wedge y = L(\bar{x})\}.$$

The relation is similar to the linear relation proof of Section 2.5.2, with the difference that $\mathcal{R}_L$ involves vector commitments. Note that in our case $\bar{x}$ has a coordinate reserved for randomness of the hiding property of commitments, so in valid relations the correspondent coefficient of L must be 0. However, in later auxiliary protocols we will not impose such restriction. Protocol Π_0 below describes a classic proof of $\mathcal{R}_L$.

Protocol $\Pi_0(P, y; \bar{x})$**:**

1. $\mathcal{P}$ computes: $\bar{r} \leftarrow_R \mathbb{Z}_q^k$, $A = \bar{g}^{\bar{r}}$, $t = L(\bar{r})$
2. $\mathcal{P}$ sends to $\mathcal{V}$: A, t
3. $\mathcal{V}$ sends to $\mathcal{P}$: $c \leftarrow_R \mathbb{Z}_q$
4. $\mathcal{P}$ sends to $\mathcal{V}$: $\bar{z} = c\bar{x} + \bar{r}$
5. $\mathcal{V}$: if $\bar{g}^{\bar{z}} = AP^c$ and $L(\bar{z}) = cy + t$ then **accept**, else **reject**.

The properties of Π_0 are obtained from similar arguments as in the proof of knowledge in Section 2.5.2. Completeness follows directly from the homomorphic property. For soundness, consider a prover $\mathcal{P}^*$ that by following Π_0 can produce an accepting transcripts $((A, t), c_1, \bar{z}_1)$ with non-negligible probability. Then it can produce with significant

probability as second transcript $((A,t),c_2,\bar{z}_2)$ such that the first message is equal in both transcripts and $c_1 \neq c_2$. Now, since both transcripts are accepting, we have $\bar{g}^{\bar{z}_1} = AP^{c_1}$ and $\bar{g}^{\bar{z}_2} = AP^{c_2}$ so $\mathcal{P}^*$ can efficiently compute the witness $\bar{x} = \frac{\bar{z}_1 - \bar{z}_2}{c_1 - c_2}$ such that $\bar{g}^{\bar{x}} = P$. Acceptance also implies that $L(\bar{z}_1) = c_1 y + t$ and $L(\bar{z}_2) = c_2 y + t$ and it follows that $L(\bar{x}) = y$. Therefore is $\bar{x}$ a valid witness. Either $\mathcal{P}^*$ already knew it or they can efficiently compute a non-trivial discrete logarithm relation between components of $\bar{g}$, which is a contradiction due to the DLA.

For zero knowledge, by only knowing the statement (P,y) one can compute $\bar{z}' \leftarrow_R \mathbb{Z}_q$, $c' \leftarrow_R \mathbb{Z}_q$, $A' = P^{-c'}\bar{g}^{\bar{z}'}$ and $t' = L(\bar{z}') - c'y$, and the transcript $((A',t'),c',\bar{z}')$ has the same distribution as a conversation between an honest prover and a honest verifier.

2.5.4 Compressing Proofs

Now we reduce the communication cost of Π_0 using ideas of compressed Σ-protocols [5], which are also present in the previous works of [23, 27]. The transfer in Π_0 is dominated by the third message of the protocols in Step 4, with size of k elements of $\mathbb{Z}_q$. It can be reduced if instead of sending $\bar{z}$, $\mathcal{P}$ proves that $(AP^c, cy + t; \bar{z}) \in \mathcal{R}_L$ which would imply the condition tested in Step 5. Note that this proof does not need to be zero knowledge as $\bar{z}$ is originally revealed in Π_0. We first present Π_1, a proof of $\mathcal{R}$ that halves communication cost by "folding" $\bar{z}$ before sending it. Next, we show how to use this protocol to reduce cost of Π_0. By assuming that k is even, we define $\bar{g}_L = (g_1, \ldots, g_{k/2}) \in \mathbb{G}^{k/2}$ and $\bar{g}_R = (g_{(k/2)+1}, \ldots, g_k) \in \mathbb{G}^{k/2}$ and analogously for $\bar{x}_L \in \mathbb{Z}_q^{k/2}$ and $\bar{x}_R \in \mathbb{Z}_q^{k/2}$. We also define $L_L : \mathbb{Z}_q^{k/2} \to \mathbb{Z}_q$ and $L_R : \mathbb{Z}_q^{k/2} \to \mathbb{Z}_q$ such that $L_L(\bar{a}) = L(\bar{a}, 0)$ and $L_R(\bar{a}) = L(0, \bar{a})$. We use an additional group element $\hat{g} \in \mathbb{G}$ generated in the same way as the components of $\bar{g}$.

Protocol $\Pi_1(P, y; \bar{x})$:

1. $\mathcal{P}$ computes: $A = \bar{g}_R^{\bar{x}_L}\hat{g}^{L_R(\bar{x}_L)}$, $B = \bar{g}_L^{\bar{x}_R}\hat{g}^{L_L(\bar{x}_R)}$
2. $\mathcal{P}$ sends to $\mathcal{V}$: A, B
3. $\mathcal{V}$ sends to $\mathcal{P}$: $c \leftarrow_R \mathbb{Z}_q$
4. $\mathcal{P}$ sends to $\mathcal{V}$: $\bar{z} = \bar{x}_L + c\bar{x}_R$
5. $\mathcal{V}$: if $(\bar{g}_L^c * \bar{g}_R)^{\bar{z}}\hat{g}^{cL_L(\bar{z}) + L_R(\bar{z})} = A(P\hat{g}^{L(\bar{x})})^c B^{c^2}$ then **accept**, else **reject**

Π_1 is a complete and sound proof of $\mathcal{R}_L$ with half the communication of Π_0. The communication of Step 4 can be further reduced by applying Π_1 recursively until the size of $\bar{z}$ is sufficiently small. Let $\Pi_B \diamond \Pi_A$ be the interactive proof obtained executing Π_A except for the last message and then executing Π_B. Now we can define to $\Pi_c = \Pi_1 \diamond \ldots \diamond \Pi_1 \diamond \Pi_0$ where the $\diamond$ is applied $\log_2(k) - 2$ times. Note that k requires to be a power of 2, but padding vectors with 0's is sufficient to fix this. Presented with more detail, it is proven in Theorem 3 of [5], that Π_c is a complete, sound and zero knowledge protocol for $\mathcal{R}_L$. Completeness is straightforward, and for zero knowledge it is sufficient to see that Π_0 is already zero knowledge. The rest of the protocol only reveals as much as Π_0. Soundness follows from similar ideas than those shown for Π_0.

Amortization techniques can be applied to prove many nullity checks, where the prover claims for linear relations $L_1, \ldots, L_r$ that $L_i(\bar{x}) = 0$ for all $i \in \{1, \ldots, r\}$. For that, $\mathcal{V}$ sends a random value $\rho \leftarrow \mathbb{Z}_q$ and then $\mathcal{P}$ and $\mathcal{V}$ execute Π_c on input $(P, \sum_{i=1}^{r} \rho^{i-1} L_i, 0; \bar{x})$.

If $L(\bar{x}) = 0$ then $L_i(\bar{x}) = 0$ for all i with overwhelming probability $1 - (r-1)/q$. Amortized nullity checks also hold when replacing linear forms by affine forms $\Phi_1, \ldots, \Phi_r$ where each one is the application of a linear form plus a constant. We denote this protocol by Π_N and its input by $(P, (\Phi_1, \ldots, \Phi_r); \bar{x})$. A prover can prove the opening of an affine map $\Phi : \mathbb{Z}_q^k \to \mathbb{Z}_q^r$ to $\bar{y} = (y_1, \ldots, y_r) \in \mathbb{Z}_q^r$ by running Π_N on input $(P, (\Phi_1 - y_1, \ldots, \Phi_r - y_r); \bar{x})$ where $\Phi_1, \ldots, \Phi_r : \mathbb{Z}_q^k \to \mathbb{Z}_q$ are the affine forms that compose Φ. The communication cost of these protocols is of $r-1$ elements of $\mathbb{Z}_q$ more than Π_c, which account for the size of $\bar{y}$.

2.5.5 Proving multiplications and circuits

Now, we show the idea of [5] to prove multiplicative relations only with black-box access to Π_N. For a set committed triplets $(\alpha_1, \beta_1, \gamma_1), \ldots, (\alpha_m, \beta_m, \gamma_m)$, $\mathcal{P}$ proves to $\mathcal{V}$ that $\alpha_i \beta_i = \gamma_i$ for all $i \in \{1, \ldots, m\}$. Let $\alpha = (\alpha_1, \ldots, \alpha_m)$, $\beta = (\beta_1, \ldots, \beta_m)$ and $\gamma = (\gamma_1, \ldots, \gamma_m)$.

Protocol Π_M:

1. $\mathcal{P}$: samples random polynomials $f(X), g(X)$ in $\mathbb{Z}_q$ of degree at most m that define a secret sharing over α and β by fixing $f(i) = \alpha_i$ and $g(i) = \beta_i$ for all $i \in \{1, \ldots, m\}$ and sampling $f(0), g(0) \leftarrow_R \mathbb{Z}_q$.

2. $\mathcal{P}$: computes the product polynomial $h(X) = f(X)g(X)$ which has degree at most $2m$.

3. $\mathcal{P}$ sends to $\mathcal{V}$: a vector commitment of $\bar{x} = (\alpha, \beta, f(0), g(0), h(0), \ldots, h(2m))$. Note that $\gamma = (h(1), \ldots, h(n))$.

4. $\mathcal{V}$ sends to $\mathcal{P}$: $c \leftarrow_R \mathbb{Z}_q \setminus \{0, \ldots, m\}$

5. $\mathcal{P}$ and $\mathcal{V}$ run Π_N to prove that $f(c)$, $g(c)$, $h(c)$ open to some points u, v and w respectively. This is possible by Lagrange interpolation, where by having sufficient points of f, g and h, which are in the commitment of $\bar{x}$, they can be evaluated its domain by applying an affine form to $\bar{x}$

6. $\mathcal{V}$: if $uv = w$ then **accept** else **reject**

As before, completeness is straightforward. Zero knowledge follows from fact that f, g and h are random polynomials and their evaluations do not reveal information, and from the fact that Π_N is zero knowledge. If the multiplicative relations do not hold in all triplets, the probability that $uv = w$ is negligible. From that and the soundness of Π_N, soundness is obtained.

Now, let $C : \mathbb{Z}_q^k \to \mathbb{Z}_q^s$ be an arithmetic circuit (see Section 2.4). For a vector commitment P, we show how ideas from Π_M are adapted to construct a proof that $\mathcal{P}$ knows a opening $\bar{x} \in \mathbb{Z}_q^k$ of P such that $C(\bar{x}) = 0$. Suppose that C has m multiplication gates. We enumerate the multiplication gates from 1 to m. For $i \in \{1, \ldots, m\}$, let α_i and β_i be inputs of the ith multiplication gate and γ_i its output. Let α, β and γ as defined in the multiplication protocol. It is not necessary to commit to α and β as a commitment of them can be obtained from affine forms on inputs $(\bar{x}, \gamma)$ which are only dependent on C. Similarly, the output of C can be computed from an affine map $\omega : \mathbb{Z}_q^{k+2m} \to \mathbb{Z}_q^s$ that takes as input $(\bar{x}, \gamma)$. When committing to $\bar{x}$ and γ, the prover only needs to prove that multiplication gates hold and that ω opens to 0. This can be done with amortized nullity

checks. Let $[\bar{a}]$ be a hiding vector commitment of $\bar{a}$ i.e., $[\bar{a}] = \bar{g}^{(\bar{a},r)}$ where $r \in \mathbb{Z}_q$ is chosen at random.

Protocol $\Pi_{cs}(C)$:

1. $\mathcal{P}$: Computes f, g and h from α, β as in Steps 1 and 2 of Π_M.
2. $\mathcal{P}$ sends to $\mathcal{V}$: $[\bar{y}]$, where $\bar{y} = (\bar{x}, f(0), g(0), h(0), h(1), \ldots, h(2m)) \in \mathbb{Z}_q^{k+2m+3}$.
3. $\mathcal{V}$ sends to $\mathcal{P}$: $c \leftarrow_R \mathbb{Z}_q \setminus \{1, \ldots, m\}$.
4. $\mathcal{P}$ sends to $\mathcal{V}$: $z_1 = f(c)$, $z_2 = g(c)$, $z_3 = h(c)$.
5. $\mathcal{P}$ and $\mathcal{V}$ run Π_N with input $([\bar{y}], (\omega, f(c) - z_1, g(c) - z_2, h(c) - z_3); \bar{y})$, where linear forms are obtained by Lagrange interpolation as in Step 5 of Π_M.
6. $\mathcal{V}$: if $z_1 z_2 = z_3$ then **accept** else **reject**

Note that $[\bar{y}]$, additionally to α and β, is an implicit commitment to $\gamma = (h(1), \ldots, h(m))$. We denote the protocol by Π_C. Properties of completeness, zero knowledge and soundness, which can be found in Theorem 4 of [5], follow from the same arguments as the multiplication protocol.

2.5.6 Compressed Range Proofs

A straightforward application of circuit proofs are range proofs. Namely, for a secret $x \in \mathbb{Z}_q$ and an integer $k < \log_2(q)$, that $x \in [0, 2^k)$. For that you commit to the first k bits $b_1, \ldots, b_k$ of x. Then, for $\bar{b} = (b_1, \ldots, b_k)$, a circuit proof for circuit

$$C_{Ra}(x, \bar{b}) = \begin{bmatrix} \bar{b} * (1 - \bar{b}) \\ x - \sum_{i=1}^{k} 2^{i-1} b_i \end{bmatrix}$$

implies the range constraint. Note that in C_{Ra} the output of the multiplication gates is 0. Therefore, in protocol Π_{cs}, it is not necessary to include in $\bar{y}$ the elements $h(1), \ldots, h(k)$, which reduces the proof cost.

2.5.7 Cost of Compressed Proofs

We now briefly summarize communication and computational cost of the proofs. As previously stated in this section, k is the number of inputs and m of multiplication gates in circuits.

Theorems 3 and 4 of [5] give the communication costs for Π_c and Π_{cs} respectively. As we use versions of the protocols transformed by the Fiat-Shamir heuristic, the verifier does not send any message. Therefore, the proof sizes are $2\lceil \log(k+1) \rceil$ elements of $\mathbb{G}$ and 3 elements of $\mathbb{Z}_q$ for Π_c, and $2\lceil \log(k+2m+4) \rceil - 1$ elements of $\mathbb{G}$ and 6 elements of $\mathbb{Z}_q$ for Π_{cs}.

For the computational cost, we count the amount of group exponentiations in $\mathbb{G}$ (GEX) as they dominate the work. In protocol Π_0, $\mathcal{P}$ performs k GEX to compute A, and $\mathcal{V}$ performs $k+1$ GEX, which corresponds to the verification of Step 5. In Π_1, $\mathcal{P}$ performs $k+2$ GEX to compute A and B, and $\mathcal{V}$ does $k/2+4$ GEX in the verification of Step 4. Π_c is a composition of one instance of Π_0 and $\mu = \lceil \log_2(k) \rceil - 2$ instances of Π_1. After the first Π_1 proof, k halves at each instance of Π_1. Additionally, $\mathcal{P}$ and $\mathcal{V}$ have to compute

$\bar{g'} = \bar{g}_L^c + \bar{g}_R$ after the first Π_1 to update parameters for each of the following sub-protocols. $\mathcal{V}$ avoids each of the verification checks except for the last one, which requires a constant amount of GEX. Therefore, $\mathcal{P}$ performs $k + 2 + \sum_{i=1}^{\mu} \frac{k}{2^{i-1}} + \frac{k}{2^i} = 4k + 2\mu - 10$ GEX and $\mathcal{V}$ does $3 + \sum_{i=1}^{\mu} \frac{k}{2^i} + 2 = k + 2\mu - 1$ GEX. Protocol Π_N requires the same amount of GEX than Π_c.

In Π_{cs}, $\mathcal{P}$ is required to compute a (hiding) commitment of $\bar{y} \in \mathbb{Z}_q^{k+2m+3}$, which costs $l = k + 2m + 4$ GEX. Then $\mathcal{P}$ and $\mathcal{V}$ engage in Π_N for an affine form of l inputs. The final costs for Π_{cs} are then $5k + 8m + 2\lceil \log_2(k + 2m + 4) \rceil + 6$ GEX for $\mathcal{P}$ and $k + 2m + 2\lceil \log_2(k + 2m + 4) \rceil - 1$ GEX for $\mathcal{V}$. For a proof of membership to the range $[0, 2^k)$, as discussed in Section 2.5.6, $h(1), \ldots, h(k)$ are not included in $\bar{y}$ which then has $2k + 4$ elements. The costs are of $9k + 2\lceil \log_2(2k + 5) \rceil + 11$ GEX for $\mathcal{P}$, and $2k + 2\lceil \log_2(2k + 5) \rceil$ GEX for $\mathcal{V}$.

We apply the same optimization done for range proofs to all of our circuits. That is, multiplication gates that are expected to be equal to 0 are not included in $\bar{y}$. Therefore, for circuits with k inputs, m multiplication gates, and m_0 multiplication gates that will be equal to 0,

- $\mathcal{P}$ performs $5k + 8m - 4m_0 + 2\lceil \log_2(k + 2m - m_0 + 4) \rceil + 6$ GEX
- $\mathcal{V}$ performs $k + 2m - m_0 + 2\lceil \log_2(k + 2m - m_0 + 4) \rceil - 1$ GEX.

Chapter 3

GOssip for Private Averaging (GOPA)

In this chapter, we present the first contribution of the dissertation. We introduce GOPA, our proposed protocol for privacy-preserving averaging. We analyze its privacy guarantees and present a methodology to provide robustness against malicious participants. In addition, we position it with respect to other existing work and illustrate its performance in privacy-preserving machine learning tasks.

3.1 Introduction

Individuals are producing ever growing amounts of personal data, which in turn fuel innovative services based on machine learning (ML). The classic *centralized* paradigm consists in collecting, storing and analyzing this data on a (supposedly trusted) central server or in the cloud, which poses well documented privacy risks for the users. With the increase of public awareness and regulations, we are witnessing a shift towards a more *decentralized* paradigm where personal data remains on each user's device, as can be seen from the growing popularity of federated learning [84]. In this setting, users typically do not trust the central server (if any), or each other, which introduces new issues regarding privacy and security. First, the information shared by users during the decentralized training protocol can reveal a lot about their private data (see [106, 116, 67] for inference attacks on federated learning). Formal guarantees such as differential privacy (DP) [58] are needed to provably mitigate this and convince users to participate. Second, *malicious* users may send incorrect results to bias the learned model in arbitrary ways [77, 17, 7]. Ensuring the correctness of the computation is crucial to persuade service providers to move to a more decentralized and privacy-friendly setting.

In this contribution, we provide a protocol for private distributed averaging. In this canonical problem, the objective is to privately compute an estimate of the average of values owned by many users who do not want to disclose them. Beyond simple data analytics, distributed averaging is of high relevance to modern ML. Indeed, it is the key primitive used to aggregate user updates in gradient-based distributed and federated learning algorithms [97, 125, 104, 80, 3, 84]. It also allows to train ML models whose sufficient statistics are averages (e.g., linear models and decision trees). Distributed averaging with differential privacy guarantees has thus attracted a lot of interest in recent years. In the strong model of local differential privacy (LDP) [87, 57, 85, 86, 82, 37], each user randomizes its input locally before sending it to an untrusted aggregator. Unfortunately, the best possible error for the estimated average with n users is of the order $O(\sqrt{n})$ larger than in the centralized model of DP where a trusted curator aggregates data in the clear and perturbs

the output [33]. To fill this gap, some work has explored relaxations of LDP that make it possible to match the utility of the trusted curator model. This is achieved through the use of cryptographic primitives such as secure aggregation [59, 34, 122, 22, 80] and secure shuffling [63, 38, 9, 68]. Many of these solutions however assume that all users truthfully follow the protocol (they are *honest-but-curious*) and/or give significant power (ability to reveal sensitive data) to a small number of servers. Furthermore, their practical implementation poses important challenges when the number of parties is large: for instance, the popular secure aggregation approach of [22] requires all $O(n^2)$ pairs of users to exchange messages.

In this context, our contribution is fourfold.

- First, we propose GOPA, a novel decentralized differentially private averaging protocol that relies on users exchanging (directly or through a server) some pairwise-correlated Gaussian noise terms along the edges of a graph so as to mask their private values without affecting the global average. This ultimately canceling noise is complemented by the addition of independent (non-canceling) Gaussian noise by each user.

- Second, we analyze the differential privacy guarantees of GOPA. Remarkably, we establish that our approach can achieve nearly the same privacy-utility trade-off as a trusted curator who would average the values of honest users, provided that the graph of honest-but-curious users is connected and the pairwise-correlated noise variance is large enough. In particular, for n_H honest-but-curious users and any fixed DP guarantee, the variance of the estimated average is only $n_H/(n_H - 1)$ times larger than with a trusted curator, a factor which goes to 1 as n_H grows. We further show that if the graph is well-connected, the pairwise-correlated noise variance can be significantly reduced.

- Third, to ensure both scalability and robustness to malicious users, we propose a randomized procedure in which each user communicates with only a *logarithmic* number of other users while still matching the privacy-utility trade-off of the trusted curator. Our analysis is novel and requires to leverage and adapt results from random graph theory on embedding spanning trees in random graphs. Additionally, we show our protocol is robust to a fraction of users dropping out.

- Finally, we propose a procedure to make GOPA verifiable by untrusted external parties, i.e., to enable users to prove the correctness of their computations without compromising the efficiency or the privacy guarantees of the protocol. To the best of our knowledge, we are the first to propose such a procedure. It offers a strong preventive countermeasure against various attacks such as data poisoning or protocol deviations aimed at reducing utility. Our construction relies on commitment schemes and zero knowledge proofs (ZKPs), which are very popular in auditable electronic payment systems and cryptocurrencies. These cryptographic primitives scale well both in communication and computational requirements and are perfectly suitable in our untrusted decentralized setting. We use classic ZKPs to design a procedure for the generation of noise with verifiable distribution, and ultimately to prove the correctness of the final computation (or detect malicious users who did not follow the protocol). Crucially, the privacy guarantees of the protocol are not compromised by this procedure, while the integrity of the computation relies on a standard discrete logarithm assumption. In the end, we argue that our protocol offers correctness

guarantees that are essentially equivalent to the case where a trusted curator would hold the private data of users.

The remaining of the chapter is organized as follows. Section 3.2 introduces the problem setting. We discuss the related work in more details in Section 3.3. The GOPA protocol is introduced in Section 3.4 and we analyze its differential privacy guarantees in Section 3.5. We present our procedure to ensure correctness against malicious behavior in Section 3.6, and summarize computational and communication costs in Section 3.7. Finally, we present some experimental results in Section 3.8 and conclude with future lines of research in Section 3.9

3.2 Notations and Setting

We consider a set $U = \{1, \dots, n\}$ of $n \geq 3$ users (parties). Each user $u \in U$ holds a private value X_u, which can be thought of as being computed from the private dataset of user u. We assume that X_u lies in a bounded interval of $\mathbb{R}$ (without loss of generality, we assume $X_u \in [0, 1]$). The extension to the vector case is straightforward. We denote by X the column vector $X = [X_1, \dots, X_n]^\top \in [0, 1]^n$ of private values. Unless otherwise noted, all vectors are column vectors. Users communicate over a network represented by a connected undirected graph $G = (U, E)$, where $\{u, v\} \in E$ indicates that users u and v are neighbors in G and can exchange secure messages. For a given user u, we denote by $N(u) = \{v : \{u, v\} \in E\}$ the set of its neighbors. We note that in settings where users can only communicate with a server, the latter can act as a relay that forwards (encrypted and authenticated) messages between users, as done in secure aggregation [22].

The users aim to collaboratively estimate the average value $X^{avg} = \frac{1}{n}\sum_{u=1}^{n} X_u$ without revealing their individual private values. Such a protocol can be readily used to privately execute distributed ML algorithms that interact with data through averages over values computed locally by the participants, but do not actually need to see the individual values. We give two concrete examples below.

Example 1 (Linear regression)**.** *Let $\iota \geq 0$ be a public parameter. Each user u holds a private feature vector $\phi_u = [\phi_u^1, \dots, \phi_u^d] \in \mathbb{R}^d$ and a private label $y_u \in \mathbb{R}$. The goal is to solve a ridge regression task, i.e. find $\theta^* \in \arg\min_\theta \frac{1}{n}\sum_{u\in U}(\phi_u^\top\theta - y_u)^2 + \iota\|\theta\|^2$. θ^* can be computed in closed form from the quantities $\frac{1}{n}\sum_{u\in U}\phi_u^i y_u$ and $\frac{1}{n}\sum_{u\in U}\phi_u^i\phi_u^j$ for all $i, j \in \{1, \dots, d\}$.*

Example 2 (Federated ML)**.** *In federated learning [84] and distributed empirical risk minimization, each user u holds a private dataset $\mathcal{D}_u$ and the goal is to find θ^* such that $\theta^* \in \arg\min_\theta \frac{1}{n}\sum_{u\in U} f(\theta; \mathcal{D}_u)$ where f is some loss function. Popular algorithms [97, 125, 104, 80, 3] all follow the same high-level procedure: at round t, each user u computes a local update θ_u^t based on $\mathcal{D}_u$ and the current global model θ^{t-1}, and the updated global model is computed as $\theta^t = \frac{1}{n}\sum_u \theta_u^t$.*

Threat model We adopt the threat model described in Section 2.1. Our privacy guarantees will hold under the assumption that honest users communicate through secure channels, while the correctness of our protocol will be guaranteed under some form of the Discrete Logarithm Assumption (DLA), described in Section 2.4.

For a given execution of the protocol, we denote by U^O the set of the users who remained online until the end (i.e., did not drop out). Users in U^O are either honest or

Approach	Com. per party	MSE	Verif	Risks
Central DP [60]	$O(1)$	$O(1/n^2)$	No	Trusted curator
Local DP [87]	$O(1)$	$O(1/n)$	No	
Verifiable secret sharing [59]	$O(n)$	$O(1/n^2)$	Yes	
Secure agg. [22] + DP [83, 2]	$O(n)$	$O(1/n^2)$	No	Honest users
CAPE [79]	$O(n)$	$O(1/n^2)$	No	
Shuffling [9]	$O(1+\log(1/\delta))$	$O(1/n^2)$	No	Trusted shuffler
GOPA (this work)	$O(\log n)$	$O(1/n^2)$	Yes	

Table 3.1: Comparison of GOPA with previous DP averaging approaches with their communication cost per party, mean squared error (MSE), verifiability (Verif) and additional risks.

malicious: we denote by $U^H \subseteq U^O$ those who are honest, by $n_H = |U^H|$ their number and by $\rho = n_H/n$ their proportion with respect to the total number of users.

We also denote by $G^H = (U^H, E^H)$ the subgraph of G induced by the set of honest users U^H, i.e., $E^H = \{\{u, v\} \in E : u, v \in U^H\}$. The properties of G and G^H will play a key role in the privacy and scalability guarantees of our protocol.

Privacy definition Our goal is to design a protocol that satisfies differential privacy (DP) defined in Section 2.2, which has become a gold standard in private information release.

3.3 Related Work

In this section we review the most important work related to ours. A set of key approaches together with their main features are summarized in Table 3.1.

Distributed averaging is a key subroutine in distributed and federated learning [97, 125, 104, 80, 3, 84]. Therefore, any improvement in the privacy-utility-communication trade-off for averaging implies gains for many ML approaches downstream.

Local differential privacy (LDP) [87, 57, 85, 86, 82] requires users to locally randomize their input before they send it to an untrusted aggregator. This very strong model of privacy comes at a significant cost in utility: the best possible mean squared is of order $1/n^2$ in the trusted curator model while it is of order $1/n$ in LDP [33, 37]. This limits the usefulness of the local model to industrial settings where the number of participants is huge [64, 54]. Our approach belongs to the recent line of work which attempts to relax the LDP model so as to improve utility without relying on a trusted curator (or similarly on a small fixed set of parties).

Previous work considered the use of cryptographic primitives like secure aggregation protocols, which can be used to compute the (exact) average of private values [59, 122, 22, 34]. While secure aggregation allows in principle to recover the utility of the trusted curator model, it suffers three main drawbacks. Firstly, existing protocols require $\Omega(n)$ communication per party, which is hardly feasible beyond a few hundred or thousand users. In contrast, we propose a protocol which requires only $O(\log n)$ communication[1].

[1]We note that, independently and in parallel to our work, [12] recently proposed a secure aggregation protocol with $O(\log n)$ communication at the cost of relaxing the functionality under colluding/malicious users.

Secondly, combining secure aggregation with DP is nontrivial as the noise must be added in a distributed fashion and in the discrete domain. Existing complete systems [83, 2] assume an ideal secure aggregation functionality which does not reflect the impact of colluding/malicious users. In these more challenging settings, it is not clear how to add the necessary noise for DP and what the resulting privacy/utility trade-offs would be. Alternatively, [80] adds the noise within the secure protocol but relies on two non-colluding servers. Thirdly, most of the above schemes are not verifiable. One exception is the verifiable secret sharing approach of [59], which again induces $\Omega(n)$ communication. Finally, we note that secure aggregation typically uses uniformly distributed pairwise masks, hence a single residual term completely destroys the utility. In contrast, we use Gaussian pairwise masks that have zero mean and bounded variance, which provides more robustness but requires the more involved privacy analysis we present in Section 3.5.

Recently, the shuffle model of privacy [38, 63, 76, 9, 68], where inputs are passed to a trusted/secure shuffler that obfuscates the source of the messages, has been studied theoretically as an intermediate point between the local and trusted curator models. For differentially private averaging, the shuffle model allows to match the utility of the trusted curator setting [9]. However, practical implementations of secure shuffling are not discussed in these works. Existing solutions typically rely on multiple layers of routing servers [55] with high communication overhead and non-collusion assumptions. Anonymous communication is also potentially at odds with the identification of malicious parties. To the best of our knowledge, all protocols for averaging in the shuffle model assume honest-but-curious parties.

The protocol proposed in [79] uses correlated Gaussian noise to achieve trusted curator utility for averaging, but the dependence structure of the noise must be only at the global level (i.e., noise terms sum to zero over all users). Generating such noise actually requires a call to a secure aggregation primitive, which incurs $\Omega(n)$ communication per party as discussed above. In contrast, our pairwise-canceling noise terms can be generated with only $O(\log n)$ communication. Furthermore, [79] assume honest parties, while our protocol is robust to malicious participants.

In summary, an original aspect of our work is to match the privacy-utility trade-off of the trusted curator model at a relatively low cost without requiring to trust a fixed small set of parties. By spreading trust over sufficiently many parties, we ensure that even in the unlikely case where many parties collude they will not be able to infer much sensitive information, reducing the incentive to collude. We are not aware of other differential privacy work sharing this feature. Overall, our protocol provides a unique combination of three important properties: (a) utility of same order as trusted curator setting, (b) logarithmic communication per user, and (c) robustness to malicious users.

3.4 Proposed Protocol

In this section we describe our protocol called GOPA (GOssip noise for Private Averaging). The high-level idea of GOPA is to have each user u mask its private value by adding two different types of noise. The first type is a sum of pairwise-correlated noise terms $\Delta_{u,v}$ over the set of neighbors $v \in N(u)$ such that each $\Delta_{u,v}$ cancels out with the $\Delta_{v,u}$ of user v in the final result. The second type of noise is an independent term η_u which does not cancel out. At the end of the protocol, each user has generated a noisy version $\hat{X}_u$ of its

Algorithm 1 GOPA protocol

Input: $G = (U, E)$, $(X_u)_{u \in U}$, $\sigma_\Delta^2, \sigma_\eta^2 \in \mathbb{R}^+$

1: **for all** neighbor pairs $\{u, v\} \in E$ s.t. $u < v$ **do**
2: u and v draw a random $y \sim \mathcal{N}(0, \sigma_\Delta^2)$ and set $\Delta_{u,v} \leftarrow y$, $\Delta_{v,u} \leftarrow -y$
3: **end for**
4: **for all** users $u \in U$ **do**
5: u draws a random $\eta_u \sim \mathcal{N}(0, \sigma_\eta^2)$ and reveals $\hat{X}_u \leftarrow X_u + \sum_{v \in N(u)} \Delta_{u,v} + \eta_u$
6: **end for**

private value X_u, which takes the following form:

$$\hat{X}_u = X_u + \textstyle\sum_{v \in N(u)} \Delta_{u,v} + \eta_u. \tag{3.1}$$

Algorithm 1 presents the detailed steps. Neighboring nodes $\{u, v\} \in E$ contact each other to draw a real number from the Gaussian distribution $\mathcal{N}(0, \sigma_\Delta^2)$, that u adds to its private value and v subtracts. Intuitively, each user thereby distributes noise masking its private value across its neighbors so that even if some of them are malicious and collude, the remaining noise values will be enough to provide the desired privacy guarantees. The idea is reminiscent of uniformly random pairwise masks in secure aggregation [22] but we use Gaussian noise and restrict exchanges to the edges of the graph instead of requiring messages between all pairs of users. As in gossip algorithms [25], the pairwise exchanges can be performed asynchronously and in parallel. Additionally, every user $u \in U$ adds an independent noise term $\eta_u \sim \mathcal{N}(0, \sigma_\eta^2)$ to its private value. This noise will ensure that the final estimate of the average satisfies differential privacy (see Section 3.5). The pairwise and independent noise variances σ_Δ^2 and σ_η^2 are public parameters of the protocol.

Utility of GOPA The protocol generates a set of noisy values $\hat{X} = [\hat{X}_1, \ldots, \hat{X}_n]^\top$ which are then publicly released. They can be sent to an untrusted aggregator, or averaged in a decentralized way via gossiping [25]. In any case, the estimated average is given by $\hat{X}^{avg} = \frac{1}{n} \sum_{u \in U} \hat{X}_u = X^{avg} + \frac{1}{n} \sum_{u \in U} \eta_u$, which has expected value X^{avg} and variance σ_η^2 / n. Recall that the local model of DP, where each user releases a locally perturbed input without communicating with other users, would require $\sigma_\eta^2 = O(1)$. In contrast, we would like the total amount of independent noise to be of order $O(1/n_H)$ as needed to protect the average of honest users with the standard Gaussian mechanism in the trusted curator model of DP [60]. We will show in Section 3.5 that we can achieve this privacy-utility trade-off by choosing an appropriate variance σ_Δ^2 for our pairwise noise terms.

Dealing with dropout A user $u \notin U^O$ who drops out during the execution of the protocol does not actually publish any noisy value (i.e., $\hat{X}_u$ is empty). The estimated average is thus computed by averaging only over the noisy values of users in U^O. Additionally, any residual noise term that a user $u \notin U^O$ may have exchanged with a user $v \in U^O$ before dropping out can be "rolled back" by having v reveal $\Delta_{u,v}$ so it can be subtracted from the result (we will ensure this does not threaten privacy by having sufficiently many neighbors, see Section 3.5.4). We can thus obtain an estimate of $\frac{1}{|U^O|} \sum_{u \in U^O} X_u$ with variance $\sigma_\eta^2 / |U^O|$. Note that even if some residual noise terms are not rolled back, e.g. to avoid extra communication, the estimate remains unbiased (with a larger variance that

depends on σ_Δ^2). This is a rather unique feature of GOPA which comes from the use of Gaussian noise rather than the uniformly random noise used in secure aggregation [22]. We discuss strategies to handle users dropping out in more details in Section 3.6.4.

3.5 Privacy Guarantees

Our goal is to prove differential privacy guarantees for GOPA. First, we develop in Section 3.5.1 a general result providing privacy guarantees as a function of the structure of the communication graph G^H, i.e., the subgraph of G induced by U^H. Then, in Sections 3.5.2 and 3.5.3, we study the special cases of the path graph and the complete graph respectively, showing they are the worst and best cases in terms of privacy. Yet, we show that as long as G^H is connected and the variance σ_Δ^2 for the pairwise (canceling) noise is large enough GOPA can (nearly) match the privacy-utility trade-off of the trusted curator setting. In Section 3.5.4, we propose a randomized procedure to construct the graph G and show that it strikes a good balance between privacy and communication costs. In each section, we first discuss the result and its consequences, and then present the proof. In Section 3.5.5, we show how to scale noise in practice and in Section 3.5.6 we compare our theoretical results in random graphs with empirical results via numerical simulations.

3.5.1 Effect of the Communication Structure on Privacy

The strength of the privacy guarantee we can prove depends on the communication graph G^H over honest users. Intuitively, this is because the more terms $\Delta_{u,v}$ a given honest user u exchanges with other honest users v, the more he/she spreads his/her secret over others and the more difficult it becomes to estimate the private value X_u. We first introduce in Section 3.5.1.1 a number of preliminary concepts. Next, in Section 3.5.1.2, we prove an abstract result, Theorem 3, which gives DP guarantees for GOPA that depend on the choice of a labeling t of the graph G^H.

In Section 3.5.1.3 we discuss a number of implications of Theorem 3 which provide some insight into the dependency between the structure of G^H and the privacy of GOPA, and will turn out helpful in the proofs of Theorems 4, 5 and 6.

3.5.1.1 Preliminary Concepts

Recall that each user $u \in U^O$ who does not drop out generates $\hat{X}_u$ from its private value X_u by adding pairwise noise terms $\bar{\Delta}_u = \sum_{v \in N(u)} \Delta_{u,v}$ (with $\Delta_{u,v} + \Delta_{v,u} = 0$) as well as independent noise η_u. All random variables $\Delta_{u,v}$ (with $u < v$) and η_u are independent. We thus have the system of linear equations

$$\hat{X} = X + \bar{\Delta} + \eta,$$

where $\bar{\Delta} = (\bar{\Delta}_u)_{u \in U^O}$ and $\eta = (\eta_u)_{u \in U^O}$.

We now define the knowledge acquired by the adversary (colluding malicious users) during a given execution of the protocol. It consists of the following:

i. the noisy value $\hat{X}_u$ of all users $u \in U^O$ who did not drop out,

ii. the private value X_u and the noise η_u of the malicious users, and

iii. all $\Delta_{u,v}$'s for which u or v is malicious.

We also assume that the adversary knows the full network graph G and all the pairwise noise terms exchanged by dropped out users (since they can be rolled back, as explained in Section 3.4). The only unknowns are thus the private value X_u and independent noise η_u of each honest user $u \in U^H$, as well as the $\Delta_{u,v}$ values exchanged between pairs of honest users $\{u, v\} \in E^H$.

Letting $N^H(u) = \{v : \{u, v\} \in E^H\}$, from the above knowledge the adversary can subtract $\sum_{v \in N(u) \setminus N^H(u)} \Delta_{u,v}$ from $\hat{X}_u$ to obtain

$$\hat{X}_u^H = X_u + \sum_{u \in N^H(u)} \Delta_{u,v} + \eta_u$$

for every honest $u \in U^H$. The view of the adversary can thus be summarized by the vector $\hat{X}^H = (\hat{X}_u^H)_{u \in U^H}$ and the correlation between its elements. Let $\hat{X}_u^H = \hat{X}_u - \sum_{v \in N(u) \setminus N^H(u)} \Delta_{u,v}$. Let $X^H = (X_u)_{u \in U^H}$ be the vector of private values restricted to the honest users and similarly $\eta^H = (\eta_u)_{u \in U^H}$. Let the directed graph $(U^H, \vec{E}^H)$ be an arbitrary orientation of the undirected graph $G^H = (U^H, E^H)$, i.e., for every edge $\{u, v\} \in E^H$, the set $\vec{E}^H$ either contains the arc (u, v) or the arc (v, u). For every arc $(u, v) \in \vec{E}^H$, let $\Delta_{(u,v)} = \Delta_{u,v} = -\Delta_{v,u}$. Let $\Delta^H = (\Delta_e^H)_{e \in \vec{E}^H}$ be a vector of pairwise noise values indexed by arcs from $\vec{E}^H$. Let $K \in \mathbb{R}^{U^H \times \vec{E}^H}$ denote the oriented incidence matrix of the graph G^H, i.e., for $(u, v) \in \vec{E}^H$ and $w \in U^H \setminus \{u, v\}$ there holds $K_{u,(u,v)} = -1$, $K_{v,(u,v)} = 1$ and $K_{w,(u,v)} = 0$. In this way, we can rewrite the system of linear equations as

$$\hat{X}^H = X^H + K\Delta^H + \eta^H. \tag{3.2}$$

Now, adapting differential privacy (Definition 2.1) to our setting, for any input X and any possible outcome $\hat{X}$, we need to compare the probability of the outcome being equal to $\hat{X}$ when a (non-malicious) user $v_1 \in U$ participates in the computation with private value $X_{v_1}^A$ to the probability of obtaining the same outcome when the value of v_1 is exchanged with an arbitrary value $X_{v_1}^B \in [0, 1]$. Since honest users drop out independently of X and do not reveal anything about their private value when they drop out, in our analysis we will fix an execution of the protocol where some set U^H of n_H honest users have remained online until the end of the protocol. For notational simplicity, we denote by X^A the vector of private values $(X_u)_{u \in U^H}$ of these honest users in which a user v_1 has value $X_{v_1}^A$, and by X^B the vector where v_1 has value $X_{v_1}^B$. X^A and X^B differ in only in the v_1-th coordinate, and their maximum difference is 1.

All noise variables are zero mean, so the expectation and covariance matrix of $\hat{X}^H$ are respectively given by:

$$\mathbb{E}\left[\hat{X}^H\right] = X^H, \quad \text{var}\left(\hat{X}^H\right) = \sigma_\eta^2 I_{U^H} + \sigma_\Delta^2 L,$$

where $I_{U^H} \in \mathbb{R}^{n_H \times n_H}$ is the identity matrix and $L = KK^\top$ is the graph Laplacian matrix of G^H.

Now consider the real vector space $Z = \mathbb{R}^{n_H} \times \mathbb{R}^{|E^H|}$ of all possible values pairs (η^H, Δ^H) of noise vectors of honest users. For the sake of readability, in the remainder of this section we will often drop the superscript H and write (η, Δ) when it is clear from the context that we work in the space Z.

Let

$$\Sigma^{(g)} = \begin{bmatrix} \sigma_\eta^2 I_{U^H} & 0 \\ 0 & \sigma_\Delta^2 I_{E^H} \end{bmatrix},$$

and let $\Sigma^{(-g)} = \left(\Sigma^{(g)}\right)^{-1}$, we then have a joint probability distribution of independent Gaussians:

$$P((\eta, \Delta)) = C_1 \exp\left(-\frac{1}{2}(\eta, \Delta)^\top \Sigma^{(-g)}(\eta, \Delta)\right),$$

where $C_1 = (2\pi)^{-(n_H + |E^H|)/2} |\Sigma^{(g)}|^{-1/2}$.

Consider the following subspaces of Z:

$$\begin{aligned} Z^A &= \{(\eta, \Delta) \in Z \mid \eta + K\Delta = \hat{X}^H - X^A\}, \\ Z^B &= \{(\eta, \Delta) \in Z \mid \eta + K\Delta = \hat{X}^H - X^B\}. \end{aligned}$$

Assume that the (only) vertex for which X^A and X^B differ is v_1. Recall that without loss of generality, private values are in the interval $[0, 1]$. Hence, if we set $X^A_{v_1} - X^B_{v_1} = 1$ then X^A and X^B are maximally apart and also the difference between $P(\hat{X} \mid X^A)$ and $P(\hat{X}|X^B)$ will be maximal.

Now choose any $t = (t_\eta, t_\Delta) \in Z$ such that $t_\eta + Kt_\Delta = X^A - X^B$. It follows that $Z^A = Z^B + t$, i.e., $Y \in Z^A$ if and only if $Y + t \in Z^B$.

3.5.1.2 Abstract Differential Privacy Result

We start by proving differential privacy guarantees which depend on the particular choice of labeling t. Theorem 3 holds for all possible choices of t, but some choices will lead to more advantageous results than others. Later, we will apply this theorem for specific choices of t for proving theorems giving privacy guarantees for communication graphs G^H with specific properties.

We first define the function $\Theta_{max} : \mathbb{R}_+ \times (0, 1) \mapsto \mathbb{R}_+$ such that Θ_{max} maps pairs (ε, δ) on the largest positive value of θ satisfying

$$\varepsilon \geq \theta^{1/2} + \theta/2, \tag{3.3}$$

$$\frac{(\varepsilon - \theta/2)^2}{\theta} \geq 2\log\left(\frac{2}{\delta\sqrt{2\pi}}\right). \tag{3.4}$$

Note that for any ε and δ, any $\theta \in (0, \Theta_{max}]$ satisfies Eqs (3.3) and (3.4).

Theorem 3. *Let $\varepsilon, \delta \in (0, 1)$. Choose a $t \in Z$ and let $\theta = t^\top \Sigma^{(-g)} t$. Under the setting introduced above, if $\theta \leq \Theta_{max}(\varepsilon, \delta)$ then* GOPA *is (ε, δ)-DP, i.e.,*

$$P(\hat{X} \mid X^A) \leq e^\varepsilon P(\hat{X} \mid X^B) + \delta.$$

Proof. We adapt ideas from [60] to our setting. It is sufficient to prove that

$$\left|\log \frac{P((\eta, \Delta))}{P((\eta, \Delta) + t)}\right| \leq \varepsilon \tag{3.5}$$

with probability $1 - \delta$ over (η, Δ). Denoting $\gamma = (\eta, \Delta)$ for convenience, we need to prove that with probability $1 - \delta$ it holds that $|\log(P(\gamma)/P(\gamma + t))| \geq e^\varepsilon$. We have

$$\begin{aligned} \left|\log \frac{P(\gamma)}{P(\gamma + t)}\right| &= \left| -\frac{1}{2}\gamma^\top \Sigma^{(-g)}\gamma + \frac{1}{2}(\gamma + t)^\top \Sigma^{(-g)}(\gamma + t)\right| \\ &= \left|\frac{1}{2}(2\gamma + t)^\top \Sigma^{(-g)} t\right|. \end{aligned}$$

To ensure Equation (3.5) holds with probability at least $1-\delta$, since we are interested in the absolute value, we will show that

$$P\Big(\frac{1}{2}(2\gamma+t)^\top \Sigma^{(-g)} t \geq \varepsilon\Big) \leq \delta/2,$$

the proof of the other direction is analogous. This is equivalent to

$$P(\gamma\Sigma^{(-g)}t \geq \varepsilon - t^\top\Sigma^{(-g)}t/2) \leq \delta/2. \tag{3.6}$$

The variance of $\gamma\Sigma^{(-g)}t$ is

$$\begin{aligned}
\text{var}(\gamma\Sigma^{(-g)}t) &= \sum_v \text{var}\left(\eta_v\sigma_\eta^{-2}t_v\right) + \sum_e \text{var}\left(\Delta_e\sigma_\Delta^{-2}t_e\right) \\
&= \sum_v \text{var}\left(\eta_v\right)\sigma_\eta^{-4}t_v^2 + \sum_e \text{var}\left(\Delta_e\right)\sigma_\Delta^{-4}t_e^2 \\
&= \sum_v \sigma_\eta^2\sigma_\eta^{-4}t_v^2 + \sum_e \sigma_\Delta^2\sigma_\Delta^{-4}t_e^2 \\
&= \sum_v \sigma_\eta^{-2}t_v^2 + \sum_e \sigma_\Delta^{-2}t_e^2 \\
&= t^\top\Sigma^{(-g)}t.
\end{aligned}$$

For any centered Gaussian random variable Y with variance σ_Y^2, we have that

$$P(Y \geq \tau) \leq \frac{\sigma_Y}{\tau\sqrt{2\pi}}\exp\left(-\tau^2/2\sigma_Y^2\right). \tag{3.7}$$

Let $Y = \gamma\Sigma^{(-g)}t$, $\sigma_Y^2 = t^\top\Sigma^{(-g)}t$ and $\tau = \varepsilon - t^\top\Sigma^{(-g)}t/2$, then satisfying

$$\frac{\sigma_Y}{\tau\sqrt{2\pi}}\exp\left(-\tau^2/2\sigma_Y^2\right) \leq \delta/2 \tag{3.8}$$

implies (3.6). Equation (3.8) is equivalent to

$$\frac{\tau}{\sigma_Y}\exp\left(\tau^2/2\sigma_Y^2\right) \geq 2/\delta\sqrt{2\pi},$$

or, after taking logarithms on both sides, to

$$\log\left(\frac{\tau}{\sigma_Y}\right) + \frac{1}{2}\left(\frac{\tau}{\sigma_Y}\right)^2 \geq \log\left(\frac{2}{\delta\sqrt{2\pi}}\right).$$

To make this inequality hold, we require

$$\log\left(\frac{\tau}{\sigma_Y}\right) \geq 0 \tag{3.9}$$

and

$$\frac{1}{2}\left(\frac{\tau}{\sigma_Y}\right)^2 \geq \log\left(\frac{2}{\delta\sqrt{2\pi}}\right). \tag{3.10}$$

Equation (3.9) is equivalent to $\tau \geq \sigma_Y$. Substituting τ and σ_Y we get

$$\varepsilon - t^\top\Sigma^{(-g)}t/2 \geq (t^\top\Sigma^{(-g)}t)^{1/2},$$

which is equivalent to (3.3). Substituting τ and σ_Y in Equation (3.10) gives (3.4). Hence, if Equations (3.3) and (3.4) are satisfied the desired differential privacy follows. □

3.5.1.3 Discussion

Essentially, given some ε, Equation (3.3) provides a lower bound for the noise (the diagonal of $\Sigma^{(g)}$) to be added. Equation (3.3) also implies that the left hand side of Equation (3.4) is larger than 1. Equation (3.4) may then require the noise or ε to be even higher if $2\log(2/\delta\sqrt{2\pi}) \geq 1$, i.e., $\delta \leq 0.48394$.

If δ is fixed, both (3.3) and (3.4) allow for smaller ε if θ is smaller. Let us analyze the implications of this result. We know that $\theta = \sigma_\eta^{-2} t_\eta^\top t_\eta + \sigma_\Delta^{-2} t_\Delta^\top t_\Delta$. As we can make σ_Δ arbitrarily large without affecting the variance of the output of the algorithm (the pairwise noise terms canceling each other) and thus make the second term $\sigma_\Delta^{-2} t_\Delta^\top t_\Delta$ arbitrarily small, our first priority to achieve a strong privacy guarantee will be to chose a t making $\sigma_\eta^{-2} t_\eta^\top t_\eta$ small. We have the following lemma.

Lemma 1. *In the setting described above, for any t satisfying $t_\eta + K t_\Delta = 0$ we have:*

$$\sum_{u \in U^H} t_u = 1. \tag{3.11}$$

Proof. Due to the properties of the incidence matrix K, i.e., $\forall u, v : K_{u,\{u,v\}} = -K_{v,\{u,v\}}$, the sum of the components of the vector $K\Delta$ is zero, i.e.,

$$\sum_{u \in U^H} (K\Delta)_u = \sum_{u \in U^H} \left(\sum_{\{u,v\} \in E^H} K_{u,\{u,v\}} \Delta_{\{u,v\}} \right)_u = 0$$

Combining this with the fact that $t_\eta + K t_\Delta = X^A - X^B$ with $\sum_{u \in U^H} (X^A - X^B)_u = 1$ we obtain Equation (3.11). □

Therefore, the vector t_η satisfying Equation (3.11) and minimizing $t_\eta^\top t_\eta$ is the vector $\mathbb{1}_{n_H}/n_H$, i.e., the vector containing n_H components with value $1/n_H$. The proofs of the several specializations of Theorem 3 we will present will all be based on this choice for t_η. Below, we derive another constraint from these observations.

Lemma 2. *In the setting described above, if $t_\eta + K\Delta = X^A - X^B$ and $t_\eta = \mathbb{1}_{n_H}/n_H$, then G^H must be connected.*

Proof. Suppose G^H is not connected, then there is a connected component $C \subseteq U^H \setminus \{v_1\}$. Let $t_C = (t_u)_{u \in C}$ and $\Delta_C = (\Delta_e)_{e \in \vec{E}^H \cap (C \times C)}$. Let $K_C = (K_{u,e})_{u \in C, e \in \vec{E}^H \cap (C \times C)}$ be the incidence matrix of $G^H[C]$, the subgraph of G^H induced by C. Due to the properties of the incidence matrix of a graph there would hold $\sum_{u \in C} (K_C \Delta_C)_u = 0$. As there would be no edges between vertices in C and vertices outside C, we would have $\sum_{u \in C} (K\Delta)_u = 0$. There would follow $\sum_{u \in C} t_u = \sum_{u \in C} (X^A - X^B - K\Delta)_u = 0$ which would contradict with $t_\eta = \mathbb{1}_{n_H}/n_H$. In conclusion, G^H must be connected. □

Given a fixed privacy level and fixed variance of the output, a second priority is to minimize σ_Δ, as this may be useful when a user drops out and the noise he/she exchanged cannot be rolled back or would take too much time to roll back (see Section 3.6.4). For this, having more edges in G^H implies that the vector t_Δ has more components and therefore typically allows a solution to $t_\eta + K t_\Delta = X^A - X^B$ with a smaller $t_\Delta^\top t_\Delta$ and hence a smaller σ_Δ.

3.5.2 Worst Case Topology

We now specialize Theorem 3 to obtain a worst-case result.

Theorem 4 (Privacy guarantee for worst-case graph)**.** *Let X^A and X^B be two databases (i.e., graphs with private values at the vertices) which differ only in the value of one user. Let $\varepsilon, \delta \in (0,1)$ and $\theta_{\mathrm{P}} = \frac{1}{\sigma_\eta^2 n_H} + \frac{n_H}{3\sigma_\Delta^2}$. If G^H is connected and $\theta_{\mathrm{P}} \leq \Theta_{max}(\varepsilon, \delta)$, then* GOPA *is (ε, δ)-differentially private, i.e., $P(\hat{X} \mid X^A) \leq e^\varepsilon P(\hat{X} \mid X^B) + \delta$.*

Crucially, Theorem 4 holds as soon as the subgraph G^H of honest users who did not drop out is connected. Note that if G^H is not connected, we can still obtain a similar but weaker result for each connected component separately (n_H is replaced by the size of the connected component).

In order to get a constant ε, inspecting the term θ_{P} shows that the variance σ_η^2 of the independent noise must be of order $1/n_H$. This is in a sense optimal as it corresponds to the amount of noise required when averaging n_H values in the trusted curator model. It also matches the amount of noise needed when using secure aggregation with differential privacy in the presence of colluding users, where honest users need to add n/n_H more noise to compensate for collusion [122].

Further inspection of the conditions in Theorem 4 also shows that the variance σ_Δ^2 of the pairwise noise must be large enough. How large it must be actually depends on the structure of the graph G^H. Theorem 4 describes the worst case, which is attained when every node has as few neighbors as possible while still being connected, i.e., when G^H is a path. In this case, Theorem 4 shows that the variance σ_Δ^2 needs to be of order n_H. Recall that this noise cancels out, so it does not impact the utility of the final output. It only has a minor effect on the communication cost (the representation space of reals needs to be large enough to avoid overflows with high probability), and on the variance of the final result if some residual noise terms of dropout users are not rolled back (see Section 3.4).

Proof of Theorem 4. Let T be a spanning tree of the (connected) communication graph G^H. Let E^T be the set of edges in T. Let $t \in \mathbb{R}^{n_H + |E^H|}$ be a vector such that:

- For vertices $u \in U^H$, $t_u = 1/n_H$.
- For edges $e \in E^H \setminus E^T$, $t_e = 0$.
- Finally, for edges $e \in E^T$, we choose t_e in the unique way such that $t_\eta + K t_\Delta = (X^A - X^B)$.

In this way, $t_\eta + K t_\Delta$ is a vector with a 1 on the v_1 position and 0 everywhere else. We can find a unique vector t using this procedure for any communication graph G^H and spanning tree T. It holds that

$$t_\eta^\top t_\eta = \left(\frac{\mathbb{1}_{n_H}}{n_H}\right)^\top \left(\frac{\mathbb{1}_{n_H}}{n_H}\right) = \frac{1}{n_H}. \tag{3.12}$$

Both Equations (3.3) and (3.4) of Theorem 3, require $t^\top \Sigma^{(-g)} t$ to be sufficiently small. We can see $t_\Delta^\top \sigma_\Delta^{-2} t_\Delta$ is maximized (thus producing the worst case) if the spanning tree T is a path $(v_1\, v_2 \ldots v_{n_H})$, in which case $t_{\{v_i, v_{i+1}\}} = (n_H - i)/n_H$. Therefore,

$$\begin{aligned} t_\Delta^\top t_\Delta \leq \sum_{i=1}^{n_H - 1} \left(\frac{n_H - i}{n_H}\right)^2 &= \frac{n_H(n_H-1)(2n_H-1)/6}{n_H^2} \\ &= \frac{(n_H-1)(2n_H-1)}{6n_H}. \end{aligned} \tag{3.13}$$

Combining Equations (3.12) and (3.13) we get

$$\theta = t^\top \Sigma^{(-g)} t \leq \sigma_\eta^{-2} \frac{1}{n_H} + \sigma_\Delta^{-2} \frac{n_H(n_H-1)(2n_H-1)/6}{n_H^2}$$

We can see that $\theta \leq \theta_{\text{P}}$ and hence $\theta \leq \Theta_{max}(\varepsilon, \delta)$ satisfies the conditions of Theorem 3 and GOPA is (ε, δ)-differentially private. □

In conclusion, we see that in the worst case σ_Δ^2 should be large (linear in n_H) to keep ε small, which has no direct negative effect on the utility of the resulting $\hat{X}$. On the other hand, σ_η^2 can be small (of the order $1/n_H$), which means that independent of the number of participants or the way they communicate a small amount of independent noise is sufficient to achieve DP as long as G^H is connected.

3.5.3 The Complete Graph

The necessary value of σ_Δ^2 depends strongly on the network structure. This becomes clear in Theorem 5, which covers the case of the complete graph and shows that for a fully connected G^H, σ_Δ^2 can be of order $O(1/n_H)$, which is a *quadratic* reduction compared to the path case.

Theorem 5 (Privacy guarantee for complete graph)**.** *Let $\varepsilon, \delta \in (0,1)$ and let G^H be the complete graph. Let $\theta_C = \frac{1}{\sigma_\eta^2 n_H} + \frac{1}{\sigma_\Delta^2 n_H}$. If $\theta_C \leq \Theta_{max}(\varepsilon, \delta)$, then* GOPA *is (ε, δ)-DP.*

Proof. If the communication graph is fully connected, we can use the following values for the vector t:

- As earlier, for $v \in U^H$, let $t_v = 1/n_H$.
- For edges $\{u, v\}$ with $v_1 \notin \{u, v\}$, let $t_{\{u,v\}} = 0$.
- For $u \in U^H \setminus \{v_1\}$, let $t_{\{u,v_1\}} = 1/n_H$.

Again, one can verify that $t_\eta + K t_\Delta = X^A - X^B$ is a vector with a 1 on the v_1 position and 0 everywhere else. In this way, again $t_\eta^\top t_\eta = 1/n_H$ but now $t_\Delta^\top t_\Delta = (n_H - 1)/n_H^2$ is much smaller. We now get

$$\theta = t^\top \Sigma^{(-g)} t = \sigma_\eta^{-2}/n_H + \sigma_\Delta^{-2}(n_H - 1)/n_H^2 \leq \theta_C \leq \Theta_{max}(\varepsilon, \delta).$$

Hence, we can apply Theorem 3 and GOPA is (ε, δ)-differentially private. □

A practical communication graph will be between the two extremes of the path and the complete graph, as shown in the next section.

3.5.4 Random Graphs

Our results so far are not fully satisfactory from the practical perspective, when the number of users n is large. Theorem 4 assumes that we have a procedure to generate a graph G such that G^H is guaranteed to be connected (despite dropouts and malicious users), and requires a large σ_Δ^2 of $O(n_H)$. Theorem 5 applies if we pick G to be the complete graph, which ensures connectivity of G^H and allows smaller $O(1/n_H)$ variance but is intractable as all n^2 pairs of users need to exchange noise.

To overcome these limitations, we propose a simple randomized procedure to construct a sparse network graph G such that G^H will be well-connected with high probability, and prove a DP guarantee *for the whole process* (random graph generation followed by GOPA), under much less noise than the worst-case. The idea is to make each (honest) user select k other users uniformly at random among all users. Then, the edge $\{u, v\} \in E$ is created if u selected v or v selected u (or both). Such graphs are known as *random k-out* or *random k-orientable* graphs [21, 65]. They have very good connectivity properties [65, 131] and are used in creating secure communication channels in distributed sensor networks [32]. Note that GOPA can be conveniently executed while constructing the random k-out graph. Recall that $\rho = n_H/n$ is the proportion of honest users. We have the following privacy guarantees.

Theorem 6 (Privacy guarantee for random k-out graphs)**.** *Let $\varepsilon, \delta \in (0, 1)$ and let G be obtained by letting all (honest) users randomly choose $k \leq n$ neighbors. Let k and $\rho = n_H/n$ be such that $\rho n \geq 81$, $\rho k \geq 4\log(2\rho n/3\delta)$, $\rho k \geq 6\log(\rho n/3)$ and $\rho k \geq \frac{3}{2} + \frac{9}{4}\log(2e/\delta)$. Let*

$$\theta_R = \frac{1}{n_H \sigma_\eta^2} + \frac{1}{\sigma_\Delta^2}\Big(\frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + \frac{12 + 6\log(n_H)}{n_H}\Big)$$

If $\theta_R \leq \Theta_{max}(\varepsilon, \delta)$ then GOPA *is $(\varepsilon, 3\delta)$-differentially private.*

This result has a similar form as Theorems 4 and 5 but requires k to be large enough (of order $\log(\rho n)/\rho$) so that G^H is sufficiently connected despite dropouts and malicious users. Crucially, σ_Δ^2 only needs to be of order $1/k\rho$ to match the utility of the trusted curator, and each user needs to exchange with only $2k = O(\log n)$ peers in expectation, which is much more practical than a complete graph.

Notice that up to a constant factor this result is optimal. Indeed, in general, random graphs are not connected if their average degree is smaller than logarithmic in the number of vertices. The constant factors mainly serve for making the result practical and (unlike asymptotic random graph theory) applicable to moderately small communication graphs, as we illustrate in the next section.

In the remaining of Section 3.5.4, we will study the differential privacy properties of selecting k neighbors randomly, leading to a proof of Theorem 6. We will start by analyzing the properties of G^H (Section 3.5.4.1). Section 3.5.4.2 consists of preparations for embedding a suitable spanning tree in G^H. Next, in Section 3.5.4.3 we will prove a number of lemmas showing that such suitable spanning tree can be embedded almost surely in G^H. Finally, we will apply these results to proving differential privacy guarantees for GOPA when communicating over such a random k-out graph G in Section 3.5.4.4, proving Theorem 6.

In this section, all newly introduced notations and definitions are local and will not be used elsewhere. At the same time, to follow more closely existing conventions in random graph theory, we may reuse in this section some variable names used elsewhere and give them a different meaning.

3.5.4.1 The Random Graph G^H

Recall that the communication graph G^H is generated as follows:

- We start with $n = |U|$ vertices where U is the set of agents.

- All (honest) agents randomly select k neighbors to obtain a k-out graph G.
- We consider the subgraph G^H induced by the set U^H of honest users who did not drop out. Recall that $n_H = |U^H|$ and that a fraction ρ of the users is honest and did not drop out, hence $n_H = \rho n$.

Let $k_H = \rho k$. The graph G^H is a subsample of a k-out-graph, which for larger n_H and k_H follows a distribution very close to that of Erdős-Rényi random graphs $G_p(n_H, 2k_H/n_H)$. To simplify our argument, in the sequel we will assume G^H is such random graph as this does not affect the obtained result. In fact, the random k-out model concentrates the degree of vertices more narrowly around the expected value than Erdős-Rényi random graphs, so any tail bound our proofs will rely on that holds for Erdős-Rényi random graphs also holds for the graph G^H we are considering. In particular, for $v \in U^H$, the degree of v is a random variable which we will approximate for sufficiently large n_H and k_H by a binomial $B(n_H, 2k_H/n_H)$ with expected value $2k_H$ and variance $2k_H(1 - 2k_H/n_H) \approx 2k_H$.

3.5.4.2 The Shape of the Spanning Tree

Remember that our general strategy to prove differential privacy results is to find a spanning tree in G^H and then to compute the norm of the vector t_Δ that will "spread" the difference between X^A and X^B over all vertices (so as to get a σ_η of the same order as in the trusted curator setting). Here, we will first define the shape of a rooted tree and then prove that with high probability this tree is isomorphic to a spanning tree of G^H. Of course, we make a crude approximation here, as in the (unlikely) case that our predefined tree cannot be embedded in G^H it is still possible that other trees could be embedded in G^H and would yield similarly good differentially privacy guarantees. While our bound on the risk that our privacy guarantee does not hold will not be tight, we will focus on proving our result for reasonably-sized U and k, and on obtaining interesting bounds on the norm of t_Δ.

Let $G^H = ([n_H], E^H)$ be a random graph where between every pair of vertices there is an edge with probability $2k_H/n_H$. The average degree of G^H is $2k_H$.

Let $k_H \geq 4$. Let $q \geq 3$ be an integer. Let Δ_1, $\Delta_2 \ldots \Delta_q$ be a sequence of positive integers such that

$$\left(\sum_{i=1}^{q}\prod_{j=1}^{i}\Delta_j\right) - (\Delta_q + 1)\prod_{j=1}^{q-2}\Delta_j < n_H \leq \sum_{i=1}^{q}\prod_{j=1}^{i}\Delta_j. \tag{3.14}$$

Let $T = ([n_H], E_T)$ be a balanced rooted tree with n_H vertices, constructed as follows. First, we define for each level l a variable z_l representing the number of vertices at that level, and a variable Z_l representing the total number of vertices in that and previous levels. In particular: at the root $Z_{-1} = 0$, $Z_0 = z_0 = 1$ and for $l \in [q-2]$ by induction $z_l = z_{l-1}\Delta_l$ and $Z_l = Z_{l-1} + z_l$. Then, $z_{q-1} = \lceil (n_H - Z_{q-2})/(\Delta_q + 1) \rceil$, $Z_{q-1} = Z_{q-2} + z_{q-1}$, $z_q = n_H - Z_{q-1}$ and $Z_q = n_H$. Next, we define the set of edges of T:

$$E_T = \{\{Z_{l-2} + i, Z_{l-1} + z_{l-1}j + i\} \mid l \in [q] \wedge i \in [z_{l-1}] \wedge z_{l-1}j + i \in [z_l]\}.$$

So the tree consists of three parts: in the first $q-2$ levels, every vertex has a fixed, level-dependent number of children, the last level is organized such that a maximum of parents has Δ_q children, and in level $q-1$ parent nodes have in general $\Delta_{q-1} - 1$ or Δ_{q-1} children. Moreover, for $0 \leq l \leq q-2$, the difference between the number of vertices in the subtrees

rooted by two vertices in level l is at most $\Delta_q + 2$. We also define the set of children of a vertex, i.e., for $l \in [q]$ and $i \in [z_{l-1}]$,

$$ch(Z_{l-2} + i) = \{Z_{l-1} + z_{l-1}j + i \mid z_{l-1}j + i \in [z_l]\}.$$

In Section 3.5.4.3, we will show conditions on n_H, $\Delta_1 \dots \Delta_q$, k_H and δ such that for a random graph G^H on n_H vertices and a vertex v_1 of G^H, with high probability (at least $1 - \delta$) G^H contains a subgraph isomorphic to T whose root is at v_1.

3.5.4.3 Random Graphs Almost Surely Embed a Balanced Spanning Tree

The results below are inspired by [93]. We specialize this result to our specific problem, obtaining proofs which are also valid for graphs smaller than 10^{10} vertices, even if the bounds get slightly weaker when we drop terms of order $O(\log(\log(n_H)))$ for the simplicity of our derivation.

Let F be the subgraph of T induced by all its non-leafs, i.e., $F = ([Z_{q-1}], E_F)$ with $E_F = \{\{i, j\} \in E_T \mid i, j \leq Z_{q-1}\}$.

Lemma 3. *Let G^H and F be defined as above. Let v_1 be a vertex of G^H. Let $n_H \geq k_H \geq \Delta_i \geq 3$ for $i \in [l]$. Let $\gamma = \max_{l=1}^{q-1} \Delta_l / k_H$ and let $\gamma + 4(\Delta_q + 2)^{-1} + 2n_H^{-1} \leq 1$. Let $k_H \geq 4\log(2n_H/\delta_F(\Delta_q + 2))$. Then, with probability at least $1 - \delta_F$, there is an isomorphism ϕ from F to a subgraph of G^H, mapping the root 1 of F on v_1.*

Proof. We will construct ϕ by selecting images for the children of vertices of F in increasing order, i.e., we first select $\phi(1) = v_1$, then map children $\{2 \dots \Delta_1 + 1\}$ of 1 to vertices adjacent to v_1, then map all children of 2 to vertices adjacent to $\phi(2)$, etc. Suppose we are processing level $l \in [q-1]$ and have selected $\phi(j)$ for all $j \in ch(i')$ for all $i' < i$ for some $Z_{l-1} < i \leq Z_l$. We now need to select images for the Δ_l children $j \in ch(i)$ of vertex i (or in case $l = q - 1$ possibly only $\Delta_l - 1$ children). This means we need to find Δ_l neighbors of i not already assigned as image to another vertex (i.e., not belonging to $\cup_{0 \leq i' < i} \phi(ch(i'))$). We compute the probability that this fails. For any vertex $j \in [n_H]$ with $i \neq j$, the probability that there is an edge between i and j in G^H is $2k_H/n_H$. Therefore, the probability that we fail to find Δ_l free neighbors of i can be upper bounded as

$$\begin{aligned} Pr\left[\text{FAIL}_F(i)\right] &= Pr\left[\text{Bin}\left(n_H - Z_l, \frac{2k_H}{n_H}\right) < \Delta_l\right] \\ &\leq \exp\left(\frac{-\left((n_H - Z_l)\frac{2k_H}{n_H} - \Delta_l\right)^2}{2(n_H - Z_l)\frac{2k_H}{n_H}}\right). \end{aligned} \tag{3.15}$$

We know that $n_H - Z_l \geq z_q$. Moreover, $(z_q + \Delta_q - 1)/\Delta_q \geq z_{q-1}$ and $Z_{q-2} + 1 \leq z_{q-1}$, hence $2(z_q + \Delta_q - 1)/\Delta_q \geq Z_{q-2} + z_{q-1} + 1 = Z_{q-1} + 1$ and $2(z_q + \Delta_q - 1)/\Delta_q + z_q \geq n_H + 1$. There follows

$$z_q(2 + \Delta_q) \geq n_H + 1 - 2(\Delta_q - 1)/\Delta_q \geq n_H - 1.$$

Therefore,

$$n_H - Z_l \geq z_q \geq n_H(1 - 2(\Delta_q + 2)^{-1} - n_H^{-1}). \tag{3.16}$$

Substituting this and $\Delta_l \geq \gamma k_H$ in Equation (3.15), we get

$$\begin{aligned} Pr\left[\textsc{Fail}_F(i)\right] &\leq \exp\left(\frac{-\left(n_H(1-2(\Delta_q+2)^{-1}-n_H^{-1})\frac{2k_H}{n_H}-k_H\gamma\right)^2}{2n_H(1-2(\Delta_q+2)^{-1}-n_H^{-1})\frac{2k_H}{n_H}}\right) \\ &\leq \exp\left(\frac{-k_H^2\left(2(1-2(\Delta_q+2)^{-1}-n_H^{-1})-\gamma\right)^2}{4k_H}\right) \\ &\leq \exp\left(\frac{-k_H^2}{4k_H}\right) = \exp\left(\frac{-k_H}{4}\right), \end{aligned}$$

where the latter inequality holds as $\gamma+4(\Delta_q+2)^{-1}+2n_H^{-1} \leq 1$. As $k_H \geq 4\log(2n_H/\delta_F(\Delta_q+2))$ we can conclude that

$$Pr\left[\textsc{Fail}_F(i)\right] \leq \frac{\delta_F(\Delta_q+2)}{2n_H}.$$

The total probability of failure to embed F in G^H is therefore given by

$$\begin{aligned} \sum_{i=2}^{Z_{q-1}} \textsc{Fail}_F(i) &\leq (Z_{q-1}-1)\frac{\delta_F(\Delta_q+2)}{2n_H} \\ &\leq \left(n_H(2(\Delta_q+2)^{-1}+n_H^{-1})-1\right)\frac{\delta_F(\Delta_q+2)}{2n_H} \\ &= \frac{2n_H}{\Delta_q+2}\frac{\delta_F(\Delta_q+2)}{2n_H} = \delta_F, \end{aligned}$$

where we again applied (3.16) □

Now that we can embed F in G^H, we still need to embed the leaves of T. Before doing so, we review a result on matchings in random graphs. The next lemma mostly follows [21] (Theorem 7.11 therein), we mainly adapt to our notation, introduce a confidence parameter and make a few less crude approximations[2].

Lemma 4. *Let $m \geq 27$ (in our construction, $m = z_q$) and $\Upsilon \geq 4$. Consider a random bipartite graph with vertex partitions $A = \{a_1 \dots a_m\}$ and $B = \{b_1 \dots b_m\}$, where for every $i, j \in [m]$ the probability of an edge $\{a_i, b_j\}$ is $p = \Upsilon(\log(m))/m$. Then, the probability of a complete matching between A and B is higher than*

$$1 - \frac{em^{-2(\Upsilon-1)/3}}{1-m^{-(\Upsilon-1)/3}}.$$

Proof. For a set X of vertices, let $\Gamma(X)$ be the set of all vertices adjacent to at least one member of X. Then, if there is no complete matching between A and B, there is some set X, with either $X \subset A$ or $X \subset B$, which violates Hall's condition, i.e., $|\Gamma(X)| < |X|$. Let X be the smallest set satisfying this property (so the subgraph induced by $X \cup \Gamma(X)$ is connected). The probability that such sets X and $\Gamma(X)$ of respective sizes i and $j = |\Gamma(X)|$ exist is upper bounded by appropriately combining:

[2]In particular, even though Bollobas's proof is asymptotically tight, its last line uses the fact that $(e\log n)^{3a}n^{1-a+a^2/n} = o(1)$ for all $a \leq n/2$. This expression is only lower than 1 for $n \geq 5.6 \cdot 10^{10}$, and as the sum of this expression over all possible values of a needs to be smaller than δ_F, we do not expect this proof applies to graphs representing current real-life datasets.

- the number of choices for X, i.e., 2 (for selecting A or B) times $\binom{m}{i}$,
- the number of choices for $\Gamma(X)$, i.e., $\binom{m}{i-1}$ (considering that $j \leq i-1$),
- an upper bound for the probability that under these choices of X and $\Gamma(X)$ there are at least $2i-2$ edges (as the subgraph induced by $X \cup \Gamma(X)$ is connected), i.e., $\binom{ij}{i+j-1}$ possible choices of the vertex pairs and p^{i+j-1} the probability that these vertex pairs all form edges, and
- the probability that there is no edge between any of $X \cup \Gamma(X)$ and the other vertices, i.e., $(1-p)^{i(m-j)+j(m-i)} = (1-p)^{m(i+j)-2ij}$.

Thus, we upper bound the probability of observing such sets X and $\Gamma(X)$ of sizes i and j as follows:

$$\begin{aligned}\textsc{Fail}_B(i,j) &\leq \binom{m}{i}\binom{m}{j}\binom{ij}{i+j-1}p^{i+j-1}(1-p)^{m(i+j)-2ij} \\ &\leq \left(\frac{me}{i}\right)^i\left(\frac{me}{j}\right)^j\left(\frac{ije}{i+j-1}\right)^{i+j-1}p^{i+j-1}(1-p)^{m(i+j)-2ij}.\end{aligned}$$

Here, in the second line the classic upper bound for combinations is used: $\binom{m}{i} < \left(\frac{me}{i}\right)^i$. As $2j \leq i+j-1$, we get

$$\begin{aligned}\textsc{Fail}_B(i,j) &\leq \left(\frac{me}{i}\right)^i\left(\frac{me}{j}\right)^j\left(\frac{ie}{2}\right)^{i+j-1}p^{i+j-1}(1-p)^{m(i+j)-2ij} \\ &\leq \frac{m^{i+j}e^{2i+2j-1}i^{j-1}}{j^j 2^{i+j-1}}p^{i+j-1}(1-p)^{m(i+j)-2ij}. \qquad (3.17)\end{aligned}$$

As $0 < p < 1$, there also holds

$$(1-p)^{1/p} < 1/e,$$

and therefore

$$(1-p)^{m(i+j)-2ij} = (1-p)^{\frac{1}{p}p(m(i+j)-2ij)} < (1/e)^{p(m(i+j)-2ij)}.$$

We can substitute $p = \Upsilon(\log(m))/m$ to obtain

$$(1-p)^{m(i+j)-2ij} < (1/e)^{(\Upsilon(\log(m))/m)(m(i+j)-2ij)} = \left(\frac{1}{m}\right)^{\Upsilon(m(i+j)-2ij)/m}.$$

Substituting this into Equation (3.17), we get

$$\begin{aligned}\textsc{Fail}_B(i,j) &\leq \frac{m^{i+j}e^{2i+2j-1}i^{j-1}}{j^j 2^{i+j-1}}\left(\frac{\log(m)\Upsilon}{m}\right)^{i+j-1}\left(\frac{1}{m}\right)^{\Upsilon(m(i+j)-2ij)/m} \\ &= \frac{mei^{j-1}}{j^j}\left(\frac{\log(m)\Upsilon e^2}{2}\right)^{i+j-1}\left(\frac{1}{m}\right)^{\Upsilon((i+j)-2ij/m)}.\end{aligned}$$

Given that $m^{\Upsilon/3} \geq \Upsilon \log(m) e^2/2$ holds for $m \geq 27$ and $\Upsilon \geq 4$, we get

$$\begin{aligned}
\textsc{Fail}_B(i,j) &\leq \frac{mei^{j-1}}{j^j}\left(\frac{\log(m)\Upsilon e^2}{2m^{\Upsilon/3}}\right)^{i+j-1} m^{-\Upsilon((i+j)-2ij/m)+\Upsilon(i+j-1)/3} \\
&\leq \frac{ei^{j-1}}{j^j} m^{-\Upsilon(\frac{2}{3}(i+j)+1/3-2ij/m)+1} \\
&\leq \frac{ei^{j-1}}{j^j} m^{-\Upsilon(\frac{1}{3}i+\frac{1}{3})+1}.
\end{aligned}$$

As $\Upsilon \geq 4$, this implies

$$\textsc{Fail}_B(i,j) \leq \frac{e}{i}\left(\frac{i}{j}\right)^j m^{-\frac{\Upsilon i}{3}-\frac{1}{3}}. \tag{3.18}$$

There holds:

$$\begin{aligned}
\sum_{j=1}^{i-1}\left(\frac{i}{j}\right)^j &= \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + \sum_{j=\lfloor i/3\rfloor+1}^{i}\left(\frac{i}{j}\right)^j \\
&\leq \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + \sum_{j=\lfloor i/3\rfloor+1}^{i} 3^j = \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + 3^{\lfloor i/3\rfloor+1}\frac{3^{i-\lfloor i/3\rfloor}-1}{3-1} \\
&< \sum_{j=1}^{\lfloor i/3\rfloor}\left(\frac{i}{j}\right)^j + \frac{3^{i+1}}{2} < \sum_{j=1}^{\lfloor i/3\rfloor} i^{i/3} + \frac{3^{i+1}}{2} \\
&\leq \frac{i}{3}i^{i/3} + \frac{3^{i+1}}{2}.
\end{aligned}$$

Substituting in Equation (3.18) gives

$$\begin{aligned}
\textsc{Fail}_B &= \sum_{i=2}^{m/2}\sum_{j=1}^{i-1}\textsc{Fail}_B(i,j) \\
&< \sum_{i=2}^{m/2}\sum_{j=1}^{i-1}\frac{e}{i}\left(\frac{i}{j}\right)^j m^{-\frac{\Upsilon i}{3}-\frac{1}{3}} < \sum_{i=2}^{m/2}\left(\frac{i^{i/3+1}}{3}+\frac{3^{i+1}}{2}\right)\frac{e}{i}m^{-\frac{\Upsilon i}{3}-\frac{1}{3}} \\
&< \sum_{i=2}^{m/2}\left(i^{i/3}+3^{i+1}\right)\frac{e}{2}m^{-\frac{\Upsilon i}{3}-\frac{1}{3}} < \sum_{i=2}^{m/2} i^{i/3}\frac{e}{2}m^{-\frac{\Upsilon i}{3}-\frac{1}{3}} + 3^{i+1}\frac{e}{2}m^{-\frac{\Upsilon i}{3}-\frac{1}{3}}.
\end{aligned}$$

As $m \geq 27 = 3^3$ we can now write

$$\begin{aligned}
\textsc{Fail}_B &< \frac{e}{2}\sum_{i=2}^{m/2} m^{-\frac{(\Upsilon-1)i}{3}-\frac{1}{3}} + m^{(i+1)/3}m^{-\frac{\Upsilon i}{3}-\frac{1}{3}} < \frac{e}{2}\sum_{i=2}^{m/2} m^{-\frac{(\Upsilon-1)i}{3}-\frac{1}{3}} + m^{-\frac{(\Upsilon-1)i}{3}} \\
&< e\sum_{i=2}^{m/2} m^{-\frac{(\Upsilon-1)i}{3}} = em^{-\frac{2(\Upsilon-1)}{3}}\sum_{i=0}^{m/2-2}\left(m^{-\frac{\Upsilon-1}{3}}\right)^i \\
&= em^{-\frac{2(\Upsilon-1)}{3}}\frac{1-\left(m^{-\frac{\Upsilon-1}{3}}\right)^{m/2-1}}{1-\left(m^{-\frac{\Upsilon-1}{3}}\right)} < em^{-\frac{2(\Upsilon-1)}{3}}\frac{1}{1-\left(m^{-\frac{\Upsilon-1}{3}}\right)}.
\end{aligned}$$

This concludes the proof. □

Lemma 5. *Let $m \geq 27$ and $\delta_B > 0$. Let*

$$\Upsilon = \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right).$$

Consider a random bipartite graph as described in Lemma 4 above. Then, with probability at least $1 - \delta_B$ there is a complete matching between A and B.

Proof. From the given Υ, we can infer that

$$\begin{gathered}\Upsilon - 1 \geq \tfrac{3\log(2e/\delta_B)}{2\log(m)} \\ \Upsilon - 1 \geq \tfrac{-3\log(\delta_B/2e)}{2\log(m)} \\ -\tfrac{2}{3}(\Upsilon - 1)\log(m) \leq \log(\delta_B/2e) \\ m^{-2(\Upsilon-1)/3} \leq \delta_B/2e\end{gathered}$$

We also know that $\Upsilon \geq 4$ and $m \geq 27$, hence $(\Upsilon - 1)/3 \geq 1$ and $1 - m^{-(\Upsilon-1)/3} \geq 26/27 \geq 1/2$. We know from Lemma 4 that the probability of having a complete matching is at least

$$1 - \frac{em^{-2(\Upsilon-1)/3}}{1 - m^{-(\Upsilon-1)/3}} \geq 1 - \frac{e(\delta_B/2e)}{1/2} = 1 - \delta_B.$$

□

Lemma 6. *Let $\delta_B > 0$, $\Delta \geq 1$ (in our construction, $\Delta = \Delta_q$) and $m \geq 27$. Let $d_1 \dots d_l$ be positive numbers, with $d_i = \Delta$ for $i \in [l-1]$, $d_l \in [\Delta]$ and $\sum_{i=1}^{l} d_i = m$. Let $A = \{a_1 \dots a_l\}$ and $B = \{b_1 \dots b_m\}$ be disjoint sets of vertices in a random graph G^H where the probability to have an edge $\{a_i, b_j\}$ is $p = 2k_H/n_H$ for any i and j. Let*

$$p \geq 1 - \left(1 - \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right)\frac{\log(m)}{m}\right)^{\Delta}. \tag{3.19}$$

Then with probability at least $1 - \delta_B$, G^H contains a collection of disjoint d_i-stars with centers a_i and leaves in B.

Proof. Define an auxiliary random bipartite graph G' with sides $A' = \{a'_1 \dots a'_m\}$ and $B = \{b_1 \dots b_m\}$. For every $i, j \in [m]$, the probability of having an edge between a_i and b_j in G' is $p' = 1 - (1-p)^{1/\Delta}$. We relate the distributions on the edges of G^H and G' by requiring there is an edge between a_i and b_j if and only if there is an edge between $a'_{\Delta(i-1)+i'}$ and b_j for all $i' \in [\Delta]$.

From Equation (3.19) we can derive

$$p' \geq \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right)\frac{\log(m)}{m}. \tag{3.20}$$

Setting

$$\Upsilon = \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right),$$

this ensures p' satisfies the constraints of Lemma 5:

$$p' = \Upsilon\log(m)/m.$$

As a result, there is a complete matching in G' with probability at least $1 - \delta_B$, and hence the required stars can be found in G^H with probability at least $1 - \delta_B$. □

Lemma 7. *Let $\delta_F > 0$ and $\delta_B > 0$. Let G^H and T and their associated variables be as defined above. Assume that the following conditions are satisfied:*

$$
\begin{aligned}
&(a) \quad n_H \geq 27(\Delta_q + 2)/\Delta_q, \\
&(b) \quad \gamma + 2(\Delta_q + 2)^{-1} + n_H^{-1} \leq 1, \\
&(c) \quad k_H \geq 4\log(2n_H/\delta_F(\Delta_q + 2)), \\
&(d) \quad k_H \geq \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(n_H\Delta_q/(\Delta_q+2))}\right)\frac{\Delta_q+2}{2}\log\left(\frac{n_H\Delta_q}{\Delta_q+2}\right), \\
&(e) \quad \gamma = \max_{l=1}^{q-1}\Delta_l/k_H.
\end{aligned}
$$

Let G^H be a random graph where there is an edge between any two vertices with probability p. Let v_1 be a vertex of G^H. Then, with probability at least $1 - \delta_F - \delta_B$, there is a subgraph isomorphism between the tree T defined above and G^H such that the root of T is mapped on v_1.

Proof. The conditions of Lemma 3 are clearly satisfied, so with probability $1 - \delta_F$ there is a tree isomorphic to F in G^H. Then, from condition (d) above and knowing that the edge probability is $p = 2k_H/n_H$, we obtain

$$
p \geq \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(n_H\Delta_q/(\Delta_q+2))}\right)\frac{1}{n_H\Delta_q/(\Delta_q+2)}\log\left(\frac{n_H\Delta_q}{\Delta_q+2}\right)\Delta_q.
$$

Taking into account that $m = n_H\Delta_q/(\Delta_q + 2)$, we get

$$
p \geq \max\left(4, 1 + \frac{3\log(2e/\delta_B)}{2\log(m)}\right)\frac{1}{m}\log(m)\,\Delta_q,
$$

which implies the condition on p in Lemma 5. The other conditions of that lemma can be easily verified. As a result, with probability at least $1 - \delta_B$ there is a set of stars in G^H linking the leaves of F to the leaves of T, so we can embed T completely in G^H. □

3.5.4.4 Running GOPA on Random Graphs

Assume we run GOPA on a random graph satisfying the properties above, what can we say about the differential privacy guarantees? According to Theorem 3, it is sufficient that there exists a spanning tree and vectors t_η and t_Δ such that $t_\eta + Kt_\Delta = X^A - X^B$. We fix t_η in the same way as for the other discussed topologies (see sections 3.5.2 and 3.5.3) in order to achieve the desired σ_η and focus our attention on t_Δ. According to Lemma 7, with high probability there exists in G^H a spanning tree rooted at the vertex where X^A and X^B differ and a branching factor Δ_l specified per level. So given a random graph on n_H vertices with edge density $2k_H/n_H$, if the conditions of Lemma 7 are satisfied we can find such a tree isomorphic to T in the communication graph between honest users G^H. In many cases (reasonably large n_H and k_H), this means that the lemma guarantees a spanning tree with branching factor as high as $O(k_H)$, even though it may be desirable to select a small value for the branching factor of the last level in order to more easily satisfy condition (d) of Lemma 7, e.g., $\Delta_q = 2$ or even $\Delta_q = 1$.

Lemma 8. *Under the conditions described above,*

$$
\begin{aligned}
t_\Delta^\top t_\Delta &\leq \frac{1}{\Delta_1}\left(1 + \frac{1}{\Delta_2}\left(1 + \frac{1}{\Delta_3}\left(\ldots\frac{1}{\Delta_q}\right)\right)\right) + \frac{(\Delta_q+2)(\Delta_q+2+2q)}{n_H} \qquad (3.21) \\
&\leq \frac{1}{\Delta_1}\left(1 + \frac{2}{\Delta_2}\right) + O(n_H^{-1}).
\end{aligned}
$$

Proof. Let q be the depth of the tree T. The tree is balanced, so in every node the number of vertices in the subtrees of its children differs in at most $\Delta_q + 2$. For edges e incident with the root (at level 0 node), $|t_e - \Delta_1^{-1}| \leq n_H^{-1}(\Delta_q + 2)$. In general, for a node at level l (except leaves or parents of leaves), there are $\prod_{i=1}^{l} \Delta_i$ vertices, each of which have Δ_{l+1} children, and for every edge e connecting such a node with a child,

$$\left| t_e - \prod_{i=1}^{l+1} \Delta_i^{-1} \right| \leq (\Delta_q + 2)/n_H.$$

For a complete tree (of $1 + \Delta + \ldots + \Delta^q$ vertices), we would have

$$t_\Delta^\top t_\Delta \;=\; \sum_{l=1}^{q} \prod_{i=1}^{l} \Delta_i \left(\prod_{i=1}^{l} \Delta_i^{-1} \right)^2 = \sum_{l=1}^{q} \left(\prod_{i=1}^{l} \Delta_i \right)^{-1},$$

which corresponds to the first term in Equation (3.21). As the tree may not be complete, i.e., there may be less than $\prod_{i=1}^{q} \Delta_i$ leaves, we analyze how much off the above estimate is. For an edge e connecting a vertex of level l with one of its children,

$$\left| t_e - \prod_{i=1}^{l+1} \Delta_i \right| \leq (\Delta_q + 2)/n_H,$$

and hence

$$\begin{aligned}
\left| t_e^2 - \left(\prod_{i=1}^{l+1} \Delta_i \right)^2 \right| &\leq \left| t_e - \prod_{i=1}^{l+1} \Delta_i \right| \left(t_e + \prod_{i=1}^{l+1} \Delta_i \right) \\
&\leq \frac{\Delta_q + 2}{n_H} \left(t_e - \prod_{i=1}^{l+1} \Delta_i + 2 \prod_{i=1}^{l+1} \Delta_i \right) \\
&\leq \left(\frac{\Delta_q + 2}{n_H} \right)^2 + 2 \frac{\Delta_q + 2}{n_H} \prod_{i=1}^{l+1} \Delta_i.
\end{aligned}$$

Summing over all edges gives

$$\begin{aligned}
t_\Delta^\top t_\Delta - \sum_{l=1}^{q} \left(\prod_{i=1}^{l} \Delta_i \right)^{-1} &\leq \sum_{l=1}^{q} z_l \left((\Delta_q + 2)/n_H^2 + 2 \left(\prod_{i=1}^{l+1} \Delta_i \right)^{-1} (\Delta_q + 2)/n_H \right) \\
&= (\Delta_q + 2)^2/n_H + \sum_{l=1}^{q} 2 z_l \left(\prod_{i=1}^{l+1} \Delta_i \right)^{-1} (\Delta_q + 2)/n_H \\
&\leq (\Delta_q + 2)^2/n_H + \sum_{l=1}^{q} 2(\Delta_q + 2)/n_H \\
&= \frac{(\Delta_q + 2)(\Delta_q + 2 + 2q)}{n_H}.
\end{aligned}$$

□

So if we choose parameters Δ for the tree T, the above lemmas provide values δ_F and δ_B such that T can be embedded in G^H with probability at least $1 - \delta_F - \delta_B$ and an upper bound for $t_\Delta^\top t_\Delta$ that can be obtained with the resulting spanning tree in G^H.

Theorem 6 in the main text summarizes these results, simplifying the conditions by assuming that $\Delta_i = \lfloor (k-1)\rho/2 \rfloor$ for $i \leq q-1$ and $\Delta_q = 2$.

Proof of Theorem 6. Let us choose $\Delta_i = \lfloor (k-1)\rho/3 \rfloor$ for $i \in [q-1]$ and $\Delta_q = 1$ for some appropriate q such that Equation (3.14) is satisfied. We also set $\delta = \delta_F = \delta_B$.

Then, the conditions of Lemma 7 are satisfied. In particular, condition (a) holds as $n_H = \rho n \geq 81 = 27(\Delta_q + 2)/\Delta_q$. Condition (e) implies that

$$\gamma \;=\; \max_{i=1}^{q-1} \Delta_i / k_H = \frac{1}{k_H} \left\lfloor \frac{(k-1)\rho}{3} \right\rfloor .$$

Condition (b) holds as

$$\begin{aligned}
\gamma + 2(\Delta_q + 2)^{-1} + n_H^{-1} &= \frac{1}{k_H} \left\lfloor \frac{(k-1)\rho}{3} \right\rfloor + \frac{2}{3} + n_H^{-1} \\
&\leq \frac{1}{k_H} \frac{(k-1)\rho}{3} + \frac{2}{3} + n_H^{-1} \\
&\leq \frac{1}{3} - \frac{\rho}{3k_H} + \frac{2}{3} + n_H^{-1} = \frac{1}{3} - \frac{1}{3k} + \frac{2}{3} + n_H^{-1} \\
&\leq \frac{1}{3} - \frac{1}{n_H} + \frac{2}{3} + n_H^{-1} = 1.
\end{aligned}$$

Condition (d) holds because we know that $\rho k \geq 6 \log(\rho n/3)$, which is equivalent to

$$k_H \geq 4 \frac{\Delta + 2}{\Delta} \log \left(\frac{n_H \Delta_q}{\Delta_q + 2} \right),$$

and we know that $\rho k \geq \frac{3}{2} + \frac{9}{4} \log(2e/\delta)$, which is equivalent to

$$k_H \geq \left(1 + \frac{3 \log(2e/\delta_B)}{2 \log(n_H \Delta_q / (\Delta_q + 2))} \right) \frac{\Delta + 2}{\Delta} \log \left(\frac{n_H \Delta_q}{\Delta_q + 2} \right).$$

Finally, condition (c) is satisfied as we know that $\rho k \geq 4 \log(\rho n / 3\delta)$. Therefore, applying the lemma, we can with probability at least $1 - 2\delta$ find a spanning tree isomorphic to T. If we find one, Lemma 8 implies that

$$\begin{aligned}
t_\Delta^\top t_\Delta &\leq \sum_{l=1}^{q} \left(\prod_{i=1}^{l} \Delta_i \right)^{-1} + \frac{(\Delta_q + 2)(\Delta_q + 2 + 2q)}{n_H} \\
&= \sum_{l=1}^{q-1} \Delta_1^{-l} + \Delta_1^{1-q} \Delta_q^{-1} + \frac{3(3+2q)}{n_H} = \Delta_1^{-1} \frac{1 - \Delta_1^{1-q}}{1 - \Delta_1^{-1}} + \Delta_1^{1-q} \Delta_q^{-1} + \frac{9 + 6q}{n_H} \\
&\leq \frac{1}{\Delta_1 - 1} + \frac{3}{n_H} + \frac{9 + 6q}{n_H} = \frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + \frac{12 + 6q}{n_H} \\
&= \frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + \frac{12 + 6 \log(n_H)}{n_H}
\end{aligned}$$

This implies the conditions related to σ_Δ and t_Δ are satisfied. From Theorem 3, it follows that with probability $1 - 2\delta$ GOPA is (ε, δ)-differentially private, or in short GOPA is $(\varepsilon, 3\delta)$-differentially private. □

3.5.5 Matching the Utility of the Centralized Gaussian Mechanism

Using the results of previous theorems, we can precisely quantify the amount of independent and pairwise noise needed to achieve a desired privacy guarantee depending on the topology, as illustrated in the corollary below.

Corollary 1. *Let $\varepsilon, \delta' \in (0,1)$, and $\sigma_\eta^2 = c^2/n_H\varepsilon^2$, where $c^2 > 2\log(1.25/\delta')$. Given some $\kappa > 0$, let $\sigma_\Delta^2 = \kappa\sigma_\eta^2$ if G is complete, $\sigma_\Delta^2 = \kappa\sigma_\eta^2 n_H(\frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + (12 + 6\log(n_H))/n_H)$ if it is a random k-out graph with k and ρ as in Theorem 6, and $\sigma_\Delta^2 = \kappa\sigma_\eta^2 n_H^2/3$ for an arbitrary connected G^H. Then* GOPA *is (ε, δ)-DP with $\delta \geq a(\delta'/1.25)^{\kappa/\kappa+1}$, where $a = 3.75$ for the k-out graph and 1.25 otherwise.*

Proof. In the centralized (trusted curator) setting, the standard centralized Gaussian mechanism ([60] Theorem A.1 therein) states that in order for the noisy average $(\frac{1}{n}\sum_{u\in U} X_u) + \eta$ to be (ε', δ')-DP for some $\varepsilon', \delta' \in (0,1)$, the variance of η needs to be:

$$\sigma_{gm}^2 = \frac{c^2}{(\varepsilon' n)^2}. \tag{3.22}$$

where $c^2 > 2\log(1.25/\delta')$.

Based on this, we let the independent noise η_u added by each user in GOPA to have variance

$$\sigma_\eta^2 \;=\; \frac{n^2}{n_H}\sigma_{gm}^2 = \frac{c^2}{(\varepsilon')^2 n_H}, \tag{3.23}$$

which, for the approximate average $\hat{X}^{avg}$, gives a total variance of:

$$Var\Big(\frac{1}{n_H}\sum_{u\in U^H}\eta_u\Big) = \frac{1}{n_H^2} n_H \sigma_\eta^2 = \frac{c^2}{(\varepsilon' n_H)^2}. \tag{3.24}$$

We can see that when $n_H = n$ (no malicious user, no dropout), Equation (3.24) exactly corresponds to the variance required by the centralized Gaussian mechanism in Equation (3.22), hence GOPA will achieve the same utility. When there are malicious users and/or dropouts, each honest user needs to add a factor n/n_H more noise to compensate for the fact that drop out users do not participate and malicious users can subtract their own inputs and independent noise terms from $\hat{X}^{avg}$. This is consistent with previous work on distributed noise generation under malicious parties [122].

Now, given some $\kappa > 0$, let $\sigma_\Delta^2 = \kappa\sigma_\eta^2$ if G is the complete graph,

$$\sigma_\Delta^2 = \kappa\sigma_\eta^2 n_H\Big(\frac{1}{\lfloor (k-1)\rho/3 \rfloor - 1} + (12 + 6\log(n_H))/n_H\Big)$$

for the random k-out graph, and $\sigma_\Delta^2 = \kappa n_H^2\sigma_\eta^2/3$ for an arbitrary connected G^H. In all cases, the value of θ in Theorems 4, 5 and 6 after plugging σ_Δ^2 gives

$$\theta = \frac{\varepsilon^2}{c^2} + \frac{\varepsilon^2}{\kappa c^2} = \frac{(\kappa+1)\varepsilon^2}{\kappa c^2}.$$

We set $\varepsilon = \varepsilon'$ and require that $\theta \leq \Theta_{max}(\varepsilon, \delta)$ as in conditions of Theorems 4, 5 and 6. Then, by Equation (3.3) we have

$$\varepsilon \geq \frac{(\kappa+1)\varepsilon^2}{2\kappa c^2} + \sqrt{\frac{(\kappa+1)}{\kappa}}\frac{\varepsilon}{c}.$$

For $d^2 = \frac{\kappa}{\kappa+1}c^2$ we can rewrite the above as $\varepsilon \geq \frac{\varepsilon^2}{2d^2} + \frac{\varepsilon}{d}$. Since $\varepsilon \leq 1$, this is satisfied if $d - \frac{\varepsilon}{2d} \geq 1$ and in turn when $d \geq 3/2$, or equivalently when $c \geq \frac{3}{2}\sqrt{\frac{\kappa+1}{\kappa}}$. Now analyzing the inequality in Equation (3.4) we have:

$$\begin{aligned}
\Big(\varepsilon - \frac{(k+1)\varepsilon^2}{2\kappa c^2}\Big)^2 &\geq 2\log(2/\delta\sqrt{2\pi})\Big(\frac{\varepsilon^2}{c^2} + \frac{\varepsilon^2}{\kappa c^2}\Big) \\
\varepsilon^2 + \frac{(\kappa+1)^2\varepsilon^4}{4\kappa^2 c^4} - \frac{(\kappa+1)\varepsilon^3}{\kappa c^2} &\geq 2\log(2/\delta\sqrt{2\pi})\Big(\frac{(\kappa+1)\varepsilon^2}{\kappa c^2}\Big) \\
\frac{1}{2}\Big(\frac{\kappa c^2}{\kappa+1} + \frac{(\kappa+1)\varepsilon^2}{4\kappa c^2} - \varepsilon\Big) &\geq \log(2/\delta\sqrt{2\pi}).
\end{aligned}$$

Again denoting $d^2 = \frac{\kappa}{\kappa+1}c^2$ we can rewrite the above as

$$\frac{1}{2}\Big(d^2 + \frac{\varepsilon^2}{4d^2} - \varepsilon\Big) \geq \log(2/\delta\sqrt{2\pi}).$$

For $d \geq 3/2$ and $\varepsilon \leq 1$, the derivative of $d^2 + \frac{\varepsilon^2}{4d^2} - \varepsilon$ is positive, so $d^2 + \frac{\varepsilon^2}{4d^2} - \varepsilon > d^2 - 8/9$. Thus, we only require $d^2 \geq 2\log(1.25/\delta)$. Therefore Equation (3.4) is satisfied when:

$$\frac{\kappa}{\kappa+1}\log(1.25/\delta') \geq \log(1.25/\delta),$$

which is equivalent to $\delta \geq 1.25\Big(\frac{\delta'}{1.25}\Big)^{\frac{\kappa}{\kappa+1}}$. The constant 3.75 instead of 1.25 for the random k-out graph case is because Theorem 6 guarantees $(\varepsilon, 3\delta)$-DP instead of (ε, δ) in Theorems 4 and 5.

□

In Corollary 1, σ_η^2 is set such that after all noisy values are aggregated, the variance of the residual noise matches that required by the Gaussian mechanism [60] to achieve (ε, δ')-DP for an average of n_H values in the *centralized* setting. The privacy-utility trade-off achieved by GOPA is thus the same as in the trusted curator model up to a small constant in δ, as long as the pairwise variance σ_Δ^2 is large enough. As expected, we see that as $\sigma_\Delta^2 \to +\infty$ (that is, as $\kappa \to +\infty$), we have $\delta \to \delta'$ for worst case and complete graphs, or $\delta \to 3\delta'$ for k-out graphs. Given the desired $\delta \geq \delta'$, we can use Corollary 1 to determine a value for σ_Δ^2 that is sufficient for GOPA to achieve (ε, δ)-DP.

Remark 1. *For clarity of presentation, our privacy guarantees protect against an adversary that consists of colluding malicious users. To* simultaneously *protect against each single honest-but-curious user (who knows his own independent noise term), we can simply replace n_H by $n'_H = n_H - 1$ in our results. This introduces a factor $n_H/(n_H - 1)$ in the variance, which is negligible for large n_H. Note that the same applies to other approaches which distribute noise-generation over data-providing users, e.g., [59].*

3.5.6 Smaller k and σ_Δ^2 via Numerical Simulation

For random k-out graphs, the conditions on k and σ_Δ^2 given by Theorem 6 are quite conservative. While we are confident that they can be refined by resorting to tighter approximations in our analysis in Section 3.5.4, an alternative option to find smaller, yet admissible values for k and σ_Δ^2 is to resort to numerical simulation.

Table 3.2: Examples of admissible values for k and σ_Δ, obtained by numerical simulation, to ensure (ε, δ)-DP with trusted curator utility for $\varepsilon = 0.1$, $\delta' = 1/n_H^2$, $\delta = 10\delta'$.

$n = 100$	$\rho = 1$	$k = 3$	$\sigma_\Delta = 60.8$
		$k = 5$	$\sigma_\Delta = 41.3$
	$\rho = 0.5$	$k = 20$	$\sigma_\Delta = 26.8$
		$k = 30$	$\sigma_\Delta = 17.2$
$n = 1000$	$\rho = 1$	$k = 5$	$\sigma_\Delta = 63.4$
		$k = 10$	$\sigma_\Delta = 41.1$
	$\rho = 0.5$	$k = 20$	$\sigma_\Delta = 45.4$
		$k = 30$	$\sigma_\Delta = 27.3$
$n = 10000$	$\rho = 1$	$k = 10$	$\sigma_\Delta = 54.6$
		$k = 20$	$\sigma_\Delta = 34.7$
	$\rho = 0.5$	$k = 20$	$\sigma_\Delta = 55.5$
		$k = 40$	$\sigma_\Delta = 28.4$

Table 3.3: Value of σ_Δ needed to ensure (ε, δ)-DP with trusted curator utility for $n = 10000$, $\varepsilon = 0.1$, $\delta' = 1/n_H^2$, $\delta = 10\delta'$ depending on the topology, as obtained from Corollary 1 or numerical simulation.

	$\rho = 1$	$\rho = 0.5$
Complete	1.7	2.1
k**-out** (Theorem 6)	44.7 ($k = 105$)	34.4 ($k = 203$)
k**-out** (simulation)	34.7 ($k = 20$)	28.4 ($k = 40$)
Worst-case	9655.0	6114.8

Given the number of users n, the proportion ρ of nodes who are honest and do not drop out, and a value for k, we implemented a program that generates a random k-out graph, checks if the subgraph G^H is connected, and if so finds a suitable spanning tree for G^H and computes the corresponding value for $t_\Delta^\top t_\Delta$ needed by our differential privacy analysis (see for instance sections 3.5.2 and 3.5.3). From this, we can in turn deduce a sufficient value for σ_Δ^2 using Corollary 1.

Table 3.2 gives examples of values obtained by simulations for various values of n, ρ and several choices for k. In each case, the reported σ_Δ corresponds to the *worst-case* value required across 10^5 random runs, and the chosen value of k was large enough for G^H to be connected in *all* runs. This was the case even for slightly smaller values of k. Therefore, the values reported in Table 3.2 can be considered safe to use in practice.

Table 3.3 shows a numerical illustration with δ only a factor 10 larger than δ'. For random k-out graphs, we report the values of σ_Δ and k given by Theorem 6, as well as smaller values obtained by numerical simulation. Although the conditions of Theorem 6 are a bit conservative (constants can likely be improved), they still lead to practical values. Clearly, random k-out graphs provide a useful trade-off in terms of scalability and robustness. Note that in practice, one often does not know in advance the exact proportion ρ of users who are honest and will not drop out, so a lower bound can be used instead.

3.6 Correctness Against Malicious Users

While the privacy guarantees of Section 3.5 hold regardless of the behavior of the (bounded number of) malicious users, the utility guarantees discussed in Section 3.4 are not valid if malicious users tamper with the protocol. In this section, we add to our protocol the capability of being audited to ensure the correctness of the computations while preserving privacy guarantees.

Objective Our goal is to (i) verify that all calculations are performed correctly even though they are encrypted, and (ii) identify any malicious behavior. As a result, we guarantee that given the input vector X a truthfully computed $\hat{X}^{avg}$ is generated which excludes any faulty contributions.

Concretely, users will be able to prove the following properties:

$$\hat{X}_u = X_u + \sum_{v \in N(u)} \Delta_{u,v} + \eta_u, \qquad \forall u \in U, \quad (3.25)$$

$$\Delta_{u,v} = -\Delta_{v,u}, \qquad \forall \{u, v\} \in E, \quad (3.26)$$

$$\eta_u \sim \mathcal{N}(0, \sigma_\eta^2), \qquad \forall u \in U, \quad (3.27)$$

$$X_u \text{ is a valid input}, \qquad \forall u \in U. \quad (3.28)$$

It is easy to see that the correctness of the computation is guaranteed if Properties (3.25)-(3.28) are satisfied. Note that, as long as they are self-canceling and not excessively large (avoiding overflows and additional costs if a user drops out, see Section 3.6.4), we do not need to ensure that pairwise noise terms $\Delta_{u,v}$ have been drawn from the prescribed distribution, as these terms do not influence the final result and only those involving honest users affect the privacy guarantees of Section 3.5. In contrast, Properties (3.27) and (3.28) are necessary to prevent a malicious user from biasing the outcome of the computation. Indeed, (3.27) ensures that the independent noise is generated correctly, while (3.28) ensures that input values are in the allowed domain. Moreover, we can force users to commit to input data so that they consistently use the same values for data over multiple computations.

We first explain the involved tools to verify computations in Section 3.6.1. We present our verification protocol in Section 3.6.2. In the following sections, we explain in more detail different verification aspects. In Section 3.6.3 we discuss the setup of the protocol in which cryptographic and other parameters are generated. In Section 3.6.4 we discuss measures against dropouts. In section 3.6.5 we discuss prevention against attacks on efficiency. In Section 3.6.6 we discuss issues with finite precision. Finally, in Section 3.6.7 we discuss details on the verification of correct sampling from the Gaussian distribution, required for Property 3.27 and which is more involved than proving other properties.

3.6.1 Tools for verifying computations

Our approach consists in publishing an encrypted log of the computation using *cryptographic commitments* and proving that it is performed correctly without revealing any additional information using *zero knowledge proofs*. These techniques are popular in a number of applications such as privacy-friendly financial systems such as [115, 14]. We explain below different tools to robustly verify our computations. Namely, a structure to post the encrypted log of our computations, commitments and zero knowledge proofs.

Public bulletin board We implement the publication of commitments and proofs using a public *bulletin board* so that any party can verify the validity of the protocol, avoiding the need for a trusted verification entity. Users sign their messages so they cannot deny them. More general purpose distributed ledger technology [113] could be used here, but we aim at an application-specific, light-weight and hence more scalable solution.

Representations We will represent numbers by elements of cyclic groups isomorphic to $\mathbb{Z}_q$ for some large prime q. To be able to work with signed fixed-precision values, we encode them in $\mathbb{Z}_q$ by multiplying them by a constant $1/\zeta$ and using the upper half of $\mathbb{Z}_q$, i.e., $\{x \in \mathbb{Z}_q : x \geq \lceil q/2 \rceil\}$ to represent negative values. Unless we explicitly state otherwise, properties (such as linear relationships) we establish for the $\mathbb{Z}_q$ elements translate straightforwardly to the fixed-precision values they represent. We choose the precision so that the error of approximating real numbers up to a multiple of ζ does not impact our results.

Pedersen Commitments For our protocol we use classical (as opposed to the vector variant) Pedersen commitments defined in Section 2.4. We will refer to Pedersen parameters as in the definition of the scheme. Namely, to parameters $\Theta = (\mathbb{G}, q, g, h)$ where $\mathbb{G}$ is a cyclic multiplicative group of prime order q, and g and h are two generators of $\mathbb{G}$ chosen at random. It is sometimes needed to let users prove that they know the values underlying commitments. In our discussion we will implicitly assume proofs of knowledge (see also below) are inserted where needed.

Zero Knowledge Proofs We use Zero Knowledge Proofs defined in Section 2.5. Particularly, we will use (classical) Σ-protocols defined in Section 2.5.2. They can prove arithmetic relations, range satisfiability between committed values and logical formulas composed of conjunctions and disjunctions of these statements. Importantly, The zero knowledge property of these proofs does not rely on any computational hardness assumption. Sometimes we will refer as *proof* T to the transcript T of a proof generated by the prover.

3.6.2 Verification Protocol

Our verification protocol, based on the primitives described in Section 3.6.1, consists of four phases:

1. *Private data commit.* At the start of our protocol, we assume users have committed to their private data. In particular, for every user u a commitment is available, either directly published by u or available through a distributed ledger or other suitable mechanism. This attenuates data poisoning, as it forces users to use the same value for X_u in each computation where it is needed.

2. *Setup.* In a setup phase at the start of our protocol, users generate Pedersen commitment parameters Θ and private random seeds that will be used to prove Property (3.27). Details are discussed in Section 3.6.3.

3. *Verification.* During our protocol, users can prove that execution is performed correctly and verify logs containing such proofs by others. If during the protocol one detects a user has cheated he/she is added to a cheater list. After the protocol, one can verify that all steps were performed correctly and that the protocol has been completed. We give details on this key step below.

4. *Mitigation.* Cheaters and drop-out users (who got off-line for a too long period of time) detected during the protocol are excluded from the computation, and their contributions are rolled back. Details are provided in Section 3.6.4.

Verification phase First, we use the homomorphic property of Pedersen commitments to prove Properties (3.25) and (3.26). Note that Property (3.26) involves secrets of two different users u and v. This is not a problem as these pairwise noise terms are known by both involved users, so they can use negated randomnesses $r_{\Delta_{u,v}} = -r_{\Delta_{v,u}}$ in their commitments of $\Delta_{u,v}$ and $\Delta_{v,u}$ such that everybody can verify that $Com_\Theta(\Delta_{u,v}, r_{\Delta_{u,v}}) \cdot Com_\Theta(\Delta_{v,u}, r_{\Delta_{v,u}}) = Com_\Theta(0,0)$. Users can choose how they generate pairwise Gaussian noise (e.g., by convention, the user that initiates the exchange can generate the noise). We just require that each user holds a message on the agreed noise terms signed by the other user before publishing commitments, so that if one of them cheats, it can be easily discovered.

Verifying the correct drawing of Gaussian numbers is more involved and requires private seeds $r_1, \ldots, r_n$ generated in Phase 2. We explain the procedure step by step in Section 3.6.7. The proof generates a transcript T_{η_u} for each user u.

To verify Property (3.28), we verify its domain and its consistency. For the domain, we prove that $X_u \in [0,1]$ with the range proof described in Section 2.5.2. For the consistency, users u publish a Pedersen commitment $P_{X_u} = Com_\Theta(X_u)$ and prove its consistency with private data committed to in Phase 1 denoted as P_D. Such proof depends on the nature of the commitment in Phase 1: if the same Pedersen commitment scheme is used nothing needs to be done, but users could also prove consistency with a record in a blockchain (which is also used for other applications) or they may need to prove more complex consistency relationships. We denote the entire proof transcript as T_{X_u}. As an illustration, consider ridge regression in Example 1. Every user u can publish commitments $P_{y_u} = Com_\Theta(y_u)$, $P_{\phi_u^i} = Com_\Theta(\phi_u^i)$ for $i \in \{1, \ldots, d\}$ (computed with the appropriately drawn randomness), and additionally commit to $\phi_u^i y_u$ and $\phi_u^i \phi_u^j$, for $i, j \in \{1, \ldots, d\}$. Then it can be verified that all these commitments are computed coherently, i.e, that the commitment of $\phi_u^i y_u$ is the product of secrets committed in P_{y_u} and $P_{\phi_u^i}$ for $i \in \{1, \ldots, d\}$, and analogously for the commitment of $\phi_u^i \phi_u^j$ in relation with $P_{\phi_u^i}$ and $P_{\phi_u^j}$, for $i, j \in \{1, \ldots, d\}$.

Compared to the central setting with a trusted curator, encrypting the input does not make the verification of input more problematic. Both in the central setting and in our setting one can perform domain tests, ask certification of values from independent sources, and require consistency of the inputs over multiple computations, even though in some cases both the central curator and our setting may be unable to verify the correctness of some input.

Algorithm 2 gives a high level overview of the 4 verification steps described above. By composition of ZKPs, these steps allow each user to prove the correctness of their computations and preserve completeness, soundness and zero knowledge properties, thereby leading to our security guarantees:

Algorithm 2 Verification of GOPA

(1) Input. Import any previous commitments to private data P_D
(2) Setup. All users jointly run Phase 2 Setup to generate Pedersen parameters Θ and private seeds $r_1, \ldots, r_n$. Each user u publishes $P_{r_u} = Com_\Theta(r_u)$
(3a) Verification - commits.
    for all user $u \in U$, **publish** as soon as available:
        $-$ $P_{X_u} = Com_\Theta(X_u)$ and proof T_{X_u} that X_u is valid
        $-$ $P_{\eta_u} = Com_\Theta(\eta_u)$ and proof T_{η_u} that η_u is not biased
        $-$ $P_{\Delta_{u,v}} = Com_\Theta(\Delta_{u,v})$
        $-$ $\hat{X}_u$ and randomness to compute its commitment
(3b) Verification - checks.
    for all $u \in U$ **verify** when commitments/proofs are available:
        $-$ (T_{X_u}, P_{X_u}, P_D) is correct,
        $-$ $P_{X_u} \cdot \left(\prod_{v \in N(u)} P_{\Delta_{u,v}}\right) \cdot P_{\eta_u} =_? Com_\Theta(\hat{X}_u)$,
        $-$ $(T_{\eta_u}, P_{r_u}, P_{\eta_u})$ is correct.
    If a check is incorrect, add u to cheaters list.
        for all user $v \in N(u)$ **do**:
            $-$ **if** $P_{\Delta_{u,v}} \cdot P_{\Delta_{v,u}} \neq Com_\Theta(0,0)$**:** add u and/or v as cheater
(4) Mitigation.
    $-$Roll back contributions of drop-outs and exchange more noise if necessary
    $-$If a harmless amount of non-canceled pairwise noise remains,
        declare the computation successful, otherwise abort.

Theorem 7 (Security guarantees of GOPA)**.** *Under the DLA, a user $u \in U$ that passes the verification protocol proves that $\hat{X}_u$ was computed correctly. Additionally, u does not reveal any additional information about X_u by running the verification, even if the DLA does not hold.*

To reduce the verification load, we note that it is possible to perform the verification for only a subset of users picked at random (for example, sampled using public unbiased randomness generated in Phase 2) after users have published the involved commitments. In this case, we obtain *probabilistic* security guarantees, which may be sufficient for some applications.

We can conclude that GOPA is an auditable protocol that, through existing efficient cryptographic primitives, can offer guarantees similar to the automated auditing which is possible for data shared with a central party.

3.6.3 Setup Phase

Our verification protocol requires public unbiased randomness to generate Pedersen commitment parameters Θ and private random seeds to generate Gaussian samples for Property (3.27). We describe below how to perform these tasks in a Setup phase.

Public randomness To generate a public random seed, a simple procedure is the following. First, all users draw uniformly a random number and publish a commit to it. When all users have done so, they all reveal their random number. Then, they sum the random numbers (modulo the order q of the cyclic group) and use the result as public random seed.

If at least one user was honest and drew a random number, this sum is random too, so no user can both claim to be honest and claim that the obtained seed is not random. Finally, the amount of randomness of the seed is expanded by the use of a PRG (see Section 2.4).

Below, we provide in more detail a folklore method which evenly distributes the work over users. The procedure generates n public random numbers in $\mathbb{Z}_q$ with a computational effort of $O(1)$ per user, and costs logarithmic in n for a negligible portion of users. We enumerate users from 1 to $n = |U|$ and use a cryptographic hash function H to generate random numbers over $\{0,1\}^T$, for some $T \geq \lambda$ (see Section 2.4). We also make use of a commitment function Com, for which a common trusted initialization is not required (see for example [20]). The procedure is depicted in Algorithm 3.

Algorithm 3 Generation of Public Randomness

1: **Input:** Hash function H and a commitment function Com.
2: **for all** $u \in \{1, \ldots, n\}$ **do**
3: Draw $s_u \leftarrow_R \mathbb{Z}_q$, compute $P_u \leftarrow Com(s_u)$ and publish P_u
4: **end for**
5: **for all** $u \in \{1, \ldots, n\}$ **do**
6: Set $s[u,u] \leftarrow s_u$ and publish it
7: **end for**
8: **for** $j = 1$ **to** $\lfloor \log_2(n) \rfloor + 1$ sequentially **do**
9: **for all** $i \in \{0, \ldots, \lfloor n/2^j \rfloor\}$ **do**
10: Let $u_{min} = 2^j i + 1$ and $u_{max} = \min(2^j(i+1), n)$
11: **if** $s[u_{min}, u_{max}]$ is not already published **then**
12: Let $u_{mid} = u_{min} + 2^{j-1} - 1$
13: A user $u \in \{u_{min}, \ldots, u_{max}\}$ wakes up and:
14: • queries $(s[u_{min}, u_{mid}], s[u_{mid}+1, u_{max}])$, if a value is not published, set it to 0
15: • publishes $s[u_{min}, u_{max}] \leftarrow s[u_{min}, u_{mid}] + [u_{mid}+1, u_{max}] \mod q$
16: **end if**
17: **end for**
18: **end for**
19: **for all** $u \in \{1, \ldots, n\}$ **do**
20: Query $S \leftarrow s[1,n]$ and publish $t_u \leftarrow H(S+u)$
21: **end for**
22: **Output:** A set of unbiased random numbers $t_1, \ldots, t_n \in \mathbb{Z}_{2^T}$

The "commit-then-reveal" protocol from in lines 2-6 is to avoid users from choosing the value s_u depending on the choice of other users, which could bias the final seed $S = \sum_{u \in U} s_u \mod q$. The value S, computed in lines 8-15 in a distributed way, requires at most $O(\log(n))$ queries and sums for a user in the worst case, but $O(1)$ for almost all users. Inactive users do not affect the computation, but their s_u terms might be ignored. As it is computed from modular sums, S is uniformly distributed over $\mathbb{Z}_q$ if at least one active user is honest. Finally, we compute our public random numbers $t_1, \ldots, t_n$ in lines 19 and 20, which are unpredictable due to the properties of H and the unpredictability and uniqueness of its input $S + u$.

To verify that a user u participated correctly, one needs to check that (1) the commitment P_u was properly computed from s_u, (2) all sums $s[u_{min}, u_{max}]$ that u published were computed correctly, and (3) the challenge t_u was correctly computed from $S + u$ by the

application of H.

The cost of the protocol for a user is dominated in the worst case at by $\log_2(n)$ sums and $3\log_2(n)\mathbb{G}$ bits in total, corresponding to the several queries of $s[u_{min}, u_{mid}]$ and $s[u_{mid}+1, u_{max}]$ and the publication of its sum. However, this cost applies to only $O(1)$ users, while the rest of the users computes one commitment, one evaluation of H and $O(1)$ sums and transfers $O(1)$ bits during the execution.

Private Seeds In a second part of Setup, users collaboratively generate samples $r_1, \ldots, r_n$ such that, for all $u \in U$, r_u is private to u and has uniform distribution in the interval $[0, M-1]$ for some public integer $M < q/2$, the number of bins to generate Gaussian samples (in Section 3.6.7). In particular,

1. **For all** $u \in U$**:** u draws uniformly $z_u \in [0, M-1]$, and publishes $P_{z_u} = Com_\Theta(z)$.
2. The users draw a public, uniformly distributed random number z (as above).
3. **For all** $u \in U$**:** u computes $r_u \leftarrow z + t_u \mod M$ and publishes $P_{r_u} \leftarrow Com_\Theta(r_u)$ together with the proof of the modular sum (see the Σ-protocol for modular sum in Section 2.5.2).

It is important that the P_{z_u}'s are published before generating the public random z to avoid that users would try to generate several r_u and check which one is most convenient for them.

3.6.4 Dealing with Dropout

In this section, we give additional details on the strategies for dealing with dropout outlined in Section 3.4. We consider that a user drops out of the computation if it is off-line for a period which is too long for the community to wait until their return. This can happen accidentally to honest users, e.g. due to lost network connection. Malicious users may also intentionally drop out to affect the progress of the computation. Finally, a user detected as cheater by the verification procedure of Section 3.6 and banned from the system may also be seen as a dropout.

Unaddressed drop outs affect the outcome of the computation as we rely on the fact that pairwise noise terms $\Delta_{u,v}$ and $\Delta_{v,u}$ cancel out for the correctness and utility of the computation.

We propose a three-step approach for handling dropout:

1. First, as a preventive measure, users should exchange pairwise noise with enough users so that the desired privacy guarantees hold even if some neighbors drop out. This is what we proposed and theoretically analyzed in Section 3.5.4, where the number of neighbors k in Theorem 6 depends on (a lower bound on) the proportion ρ of honest users who do not drop out. It is important to use a safe lower bound on ρ to make sure users will have enough safety margin to handle actual dropouts.
2. Second, as long as there is time, users attempt to repair as much as possible the problems incurred by dropouts. A user u who did not publish $\hat{X}_u$ yet can just remove the corresponding pairwise noise (and possibly exchange noise with another active user instead). Second, a user u who did publish $\hat{X}_u$ already but has still some safety margin thanks to step 1 can simply reveal the noise exchanged with the user who dropped out, and subtract it from his published $\hat{X}_u$.

3. Third, it is possible that a user u did publish $\hat{X}$ and afterwards still so many neighbors drop out that revealing all exchanged pairwise noise would affect the privacy guarantees for u. If that happens it means a significant fraction of the neighbors of u dropped out, while the neighbors of u form a random sample of all users. In such case, it is likely that also globally many users dropped out. If caused by a large-scale network failure the best strategy could be to just wait longer than initially planned. Else, given that u is unable to reveal more pairwise noise without risking their privacy, the only options are either that u discards all his pairwise noise and restarts with a new set of neighbors, or that u does not address the problem and his pairwise noise is not compensated by the noise of another active user. To avoid such problems, in addition to step 1, it can be useful to check which users went off-line just before publishing $\hat{X}$ and to have penalties for users who (repeatedly) drop out at the most inconvenient times.

4. Finally, when circumstances require and allow it, we can ignore the remaining problems and proceed with the algorithm, which will then output an estimated average with slightly larger error. This can be the case for instance when only a few drop outs have not yet been resolved, there is not much time available, and the corresponding pairwise terms $\Delta_{u,v}$ are known to be not too large (e.g., by proving that they were drawn from $\mathcal{N}(0, \sigma_\Delta^2)$ where σ_Δ^2 is small enough, or by the use of range proofs).

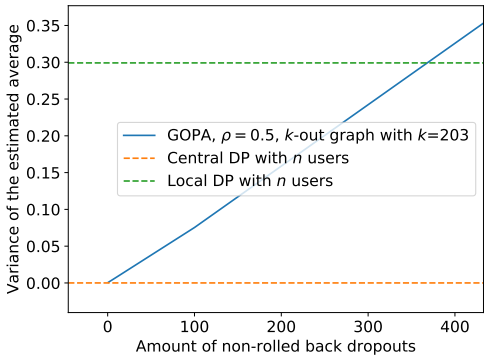


Figure 3.1: Impact of non-rolled back dropouts on the utility of GOPA. See text for details.

Figure 3.1 illustrates with a simple simulation the impact of a small number of residual pairwise noise terms of variance σ_Δ^2 in the final result, which may happen in the rare circumstances that the pairwise noise terms of some users who dropped out are not "rolled back" by their neighbors (step 2 above). We consider $n = 10000$, $\rho = 0.5$, $\epsilon = 0.1$, $\delta = 10/(\rho n)^2$ and set the values of k, σ_η^2 and σ_Δ^2 using Corollary 1 so that GOPA satisfies (ϵ, δ)-DP (in particular we set them in the same way as in Table 3.3). We simulate this by drawing a random k-out graph, selecting a certain number of dropout users at random, marking all their exchanged noise as not rolled back (in practice it is also possible that part of their noise gets rolled back) and computing the variance of the estimated average. The simulation is averaged over 100 runs (even though the standard deviation across random runs is negligible). We see that GOPA can tolerate a number of "catastrophic drop-outs" while remaining more accurate than local DP. This ability to retain a useful estimate despite residual pairwise noise terms is rather unique to GOPA, and not possible with secure aggregation methods which typically use uniformly random pairwise masks

[22]. We note that this robustness can be optimized by choosing smaller σ_Δ^2 (i.e., smaller κ) and compensating by adding a bit more independent noise according to Corollary 1.

3.6.5 Robustness Against Attacks on Efficiency

In this section, we study several attacks, their impact and GOPA's defense against them.

Dropout A malicious user could drop out of the computation intentionally, with the impact described in Section 3.6.4. However, dropping out is bad for the reputation of a user, and users dropping out more often than agreed could be banned from future participation. One can use techniques from identity management (a separate branch in the field of security) to ensure that creating new accounts is not free, and that possibilities to create new accounts are limited, e.g., by requiring to link accounts to unique persons, unique bank account or similar. This also ensures that the risk of being banned outweighs the incentive of the (bounded) delay of the system one could cause by intentionally dropping out.

Flooding with Neighbor Requests In Section 3.5, we discuss privacy guarantees for complete graphs, path graphs and random communication graphs. In the case of a complete communication graph, all users exchange noise with all other users. This is slow, but there is no opportunity for malicious users to interfere with the selection of neighbors as the set of neighbors is fixed. In other cases, e.g., when the number of users is too large and every user selects k neighbors randomly as in Section 3.5.4, one could wonder whether colluding malicious users could select neighbors in such a way that the good working of the algorithm is disturbed, e.g., a single honest user is flooded with neighbor requests and ends up exchanging noise with $O(n)$ others.

We first stress the fact that detecting such attacks is easy. If all agents randomly select neighbors uniformly at random as they should, then every agent expects to receive k neighbor invitations, perhaps plus a few standard deviations of order $O(\sqrt{k})$. As soon as a user receives a sufficiently unlikely number of neighbor invitations, we know that with overwhelming probability the user is targeted by malicious users.

To avoid this, we can let all users select neighbors in a deterministic way starting from a public random seed (e.g., take the public randomness generated in Section 3.6.3, add the ID of the user to it, apply a hash function to the sum, and use the result to start selecting neighbors). In this way, neighbor selection is public and cannot be tampered with. It is possible some neighbors of a user u were off-line and u skipped them, but unless so many users are off-line that the community should have noticed severe problems u should be able to find enough neighbors among the first ck in his random sequence for a small constant c.

Other Common Attacks We assume the algorithm is implemented on a system which is secure according to classic network-related attacks, such as denial-of-service (DoS) attacks. Such attacks are located at the network level rather than at the algorithm level. As such, they apply similarly to any distributed algorithm requiring communication over a network. To the extent such attacks can be mitigated, the solutions are on the network level, including (among others) a correct organization of the network and its routers.

Similarly, we assume that all (honest) communication is secure and properly encrypted, referring the reader to the state-of-the-art literature for details on possible implementations.

3.6.6 Further Discussion on the Impact of Finite Precision

In practice, we cannot work with real numbers but only with finite precision approximations (see Section 3.6.1). We provide a brief discussion of the impact of this on the guarantees offered by the protocol. There is already a large body of work addressing issues which could arise because of finite precision. Here are the main points:

1. Finite precision can be an issue for differential privacy in general, (see e.g. [8] for a study of the effect of floating point representations). Issues can be overcome with some care, and our additional encryption does not make the problem worse (in fact we can argue that encryption typically uses more bits and in our setting this may help).

2. The issue of finite precision has been studied in cryptography. While some operations such as multiplication can cause difficulties in the context of homomorphic encryption, in our work we use a partially homomorphic scheme with only addition. As a result, we can just represent our numbers with as many bits (after the decimal dot) as in plaintext memory.

3. If the number of users is high (and hence also the sum of the X_u and the $\Delta_{u,v}$ variables), working up to the needed precision does not cause a cost which is high compared to the cost of the digits before the decimal dot.

3.6.7 Private Gaussian Sampling

In our algorithm, every user u needs to generate a Gaussian distributed number η_u, which he does not publish, but for which we need to verify that it is generated correctly, as otherwise a malicious user could bias the result of the algorithm.

Proving the generation of a Gaussian distributed random number is more involved than ZKPs such as linear relationships and range proofs we need to verify in the other parts of the computation.

Recall that ζ is the desired fixed precision and B is a precision parameter such that B^2/ζ is an integer. We want to draw η_u from the Gaussian distribution approximated with $1/B$ equiprobable bins and we want to prove the correct drawing up to a precision B.

We will start from the private seed r_u, generated in the Phase 2 Setup (Section 3.6.3), which is only known to user u, for which u has published a commitment $Com_\Theta(r_u)$ and for which the other users know it has been generated uniformly randomly from an interval $[0, M-1]$ for $M \approx 1/B$. As it will be seen in Chapter 4, there are many ways now to exploit a uniformly distributed number to generate a Gaussian distributed one. In this section, we describe one of them.

User u will compute x' such that $((2r_u+1)/M) - 1 = \text{erf}(x'/\sqrt{2})$. We know that x' is normally distributed. The main task is then to provide a ZKP that $y = \text{erf}(x)$ for $y = ((2r_u+1)/M) - 1$ and $x = x'/\sqrt{2}$. As erf is symmetric, we do our analysis for positive values of x and y, while the extension for the negative case is straightforward. We want to achieve an approximation where the error on y as a function of x is at most B.

Approximating the error function The error function relates its input and output in a way that cannot be expressed with additive, multiplicative or exponential equations. We

therefore approximate erf using a converging series. In particular, we will rely on the series

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{l=0}^{\infty} \frac{(-1)^l x^{2l+1}}{l!(2l+1)}. \tag{3.29}$$

As argued by [40], this series has two major advantages. First, it only involves additions and multiplications, while other known series converging to $\text{erf}(x)$ often include multiple $\exp(-x^2/2)$ factors which would require additional evaluations and proofs. Second, it is an alternating series, which means we can determine more easily in advance how many terms we need to evaluate to achieve a given precision.

Nevertheless, Equation (3.29) converges slowly for large x. It is more efficient to prove either that

$$y = \text{erf}(x) \quad or \quad 1 - y = \text{erfc}(x),$$

as for $\text{erfc}(x) = 1 - \text{erf}(x)$ there exist good approximations requiring only a few terms for large x. An example is the asymptotic expansion

$$\text{erfc}(x) = \frac{e^{-x^2}}{x\sqrt{\pi}} S_{\text{erfc}}(x) + R_L(x), \tag{3.30}$$

where

$$S_{\text{erfc}}(x) = \sum_{l=0}^{L-1} \frac{(-1)^l (2l-1)!!}{(2x^2)^l} \tag{3.31}$$

with $l!! = 1$ for $l < 1$ and $(2l-1)!! = \prod_{i=1}^{l}(2i-1)$. This series diverges, but if x is sufficiently large then the remainder

$$R_L(x) \leq \frac{e^{-x^2}}{x\sqrt{\pi}} \frac{(2L-1)!!}{(2x^2)^L} \tag{3.32}$$

after the first L terms is sufficiently small to be neglected. So u could prove either part of the disjunction depending on whether the erf or erfc approximations achieve sufficient precision.

Zero Knowledge Proof of erf(x) We use fixed-precision rounding operations. The implied rounding does not cause major problems for several reasons. First, the Gaussian distribution is symmetric, and hence the probability of rounding up and rounding down is exactly the same, making the rounding error a zero-mean random variable. Second, discrete approximations of the Gaussian mechanism such as binomial mechanisms have been studied and found to give similar guarantees as the Gaussian mechanism [3]. Third, we can require the cumulated rounding error to be an order of magnitude smaller than the standard deviation of the noise we are generating, so that any deviation due to rounding has negligible impact. We will use a fixed precision for all numbers (except for small integer constants), and we will represent numbers as multiples of ζ.

Let $t_l = \frac{x^{2l+1}}{l!}$, and $L_{\text{erf}} + 1$ be the amount of terms of the series in Equation (3.29) we evaluate. We provide a ZKP of the evaluation of erf by proving that $t_0 = x$,

$$t_l = \frac{t_{l-1} x^2}{l} \quad \text{for all } l \in \{1, \ldots, L_{\text{erf}}\}, \tag{3.33}$$

and $y = \frac{2}{\sqrt{\pi}} \sum_{l=0}^{L_{\text{erf}}} (-1)^l \frac{t_l}{2l+1}$. The bulk of the ZKP is in Equation (3.33) and in the divisions by $2l+1$ of the latter relation. For any fixed-precision value d, let $\langle d \rangle = \frac{d}{\zeta}$ be its integer

encoding. We can achieve a ZKP of the fixed precision product $c = ab$ for private a, b and c by proving that $\frac{1}{\zeta}\langle c\rangle - \langle a\rangle\langle b\rangle \in [-1/2\zeta, 1/2\zeta]$. For round-off division, a ZKP that $a/b = c$ for private a, c and a public positive integer b is possible by proving that $\langle a\rangle - b\langle c\rangle \in [-b/2, b/2]$. The above proofs can be achieved using additions, products and range proofs in $\mathbb{Z}_q$ (see Section 2.5.2).

Similarly for erfc, let $m_l = (2l-1)!!/(2x)^l$ and L_{erfc} be the amount of terms we compute of the series defined in Equation (3.31), we can construct a ZKP of its evaluation by proving $m_0 = 0$,

$$m_l = m_{l-1}\frac{2l-1}{2x^2} \qquad \text{, for all } l \in \{1, \ldots, L_{\text{erfc}} - 1\} \tag{3.34}$$

and $y = \frac{e^{-x^2}}{x\sqrt{\pi}}\sum_{l=0}^{L_{\text{erfc}}-1}(-1)^l m_l$. Proving that $\langle m_l\rangle\langle x^2\rangle - \frac{2l-1}{\zeta}\langle m_{l-1}\rangle \in [-\langle x^2\rangle, \langle x^2\rangle]$ is equivalent to prove the relation in Equation (3.34). This can be done with $10B_{x^2}$ cost, where B_{x^2} is maximum number of bits of $\langle x^2\rangle$. We can assume that $B_{x^2} \leq \log_2(q)$ (where q is the order of the Pedersen group $\mathbb{G}$) which makes the cost for each term not bigger than $10\log_2(q)$. All other non-dominant computations can similarly be verified with addition and product proofs.

The rest of Section 3.6.7 mainly studies the less interesting details of approximations the erf and erfc series. Some readers may want to skip the derivation and continue at "computational cost" at the end of the section.

Amount of approximation terms Now we determine the magnitudes of L_{erf} and L_{erfc} needed to achieve an error smaller than B in our computation. We then require the approximation and rounding error to be smaller than $B/2$. The amount of terms of the series must not depend on x as the latter is a private value, but we must be able to achieve the expected precision for all possible x.

The erf series requires more approximation terms as x gets larger. On the other hand, the erfc series is optimal when $L_{\text{erfc}} = \lfloor x^2 + 1/2\rfloor$ terms are evaluated, and the error gets smaller as x increases. Then, we use erfc only when x is large enough to satisfy the required precision, and use erf for smaller values. We first compute the lower bound x_{erfc}^{min} for the application of erfc. Then the domain of erf is restricted to $[0, x_{\text{erfc}}^{min})$ so can obtain the an upper bound of L_{erf}. Similarly, the restricted domain of erfc allows us to upper bound L_{erfc}.

Developing Equation (3.32), we can achieve an approximation error of erfc of

$$\begin{aligned} E_{\text{erfc}}(x) &\leq \frac{1}{e^{x^2}x\sqrt{\pi}}\frac{(2L-1)!!}{(2x^2)^L} = \frac{1}{e^{x^2}x\sqrt{\pi}}\frac{(2L)!}{L!2^L(2x^2)^L} \\ &\approx \frac{1}{e^{x^2}x\sqrt{\pi}}\frac{\sqrt{4\pi L}(2L/e)^{2L}}{\sqrt{2\pi L}(L/e)^L 2^L(2x^2)^L} = \frac{\sqrt{2}}{e^{x^2}x\sqrt{\pi}}\frac{(2^{2L})(L^{2L})(e^L)}{(e^{2L})(L^L)(2^L)(2x^2)^L} \\ &= \frac{\sqrt{2}}{e^{x^2}x\sqrt{\pi}}\frac{2^L L^L}{e^L(2x^2)^L} = \frac{\sqrt{2}}{e^{x^2}x\sqrt{\pi}}\frac{L^L}{e^L(x^2)^L}. \end{aligned}$$

This is minimal in $\lfloor x^2 + 1/2\rfloor$ so given x we can achieve an error of

$$\frac{\sqrt{2}}{e^{x^2}x\sqrt{\pi}}\frac{L^L}{e^L(x^2)^L} \leq \frac{\sqrt{2}}{e^{x^2}x\sqrt{\pi}}\frac{\lfloor x^2+1/2\rfloor^{\lfloor x^2+1/2\rfloor}}{e^{\lfloor x^2+1/2\rfloor}(x^2)^{\lfloor x^2+1/2\rfloor}} \leq \frac{\sqrt{2}}{\sqrt{\pi}}\frac{(1+1/2x^2)^{\lfloor x^2+1/2\rfloor}}{xe^{(2x^2-1/2)}}.$$

As we use erfc for large values of x, we assume $x \geq 1$ which will not affect our reasoning, and then we can simplify the above by

$$E_{\text{erfc}}(x) \leq \frac{\sqrt{2}}{\sqrt{\pi}} \frac{(1+1/2x^2)^{\lfloor x^2+1/2 \rfloor}}{xe^{(2x^2-1/2)}} \leq \frac{\sqrt{2}}{\sqrt{\pi}} \frac{3\sqrt{6}/4}{xe^{(2x^2+3/2)}} \leq \frac{1}{2}\sqrt{\frac{27}{\pi}} \frac{1}{e^{(2x^2-1/2)}} \quad (3.35)$$

Then, by Equation (3.35) and if

$$\frac{B}{2} \geq \frac{1}{2}\sqrt{\frac{27}{\pi}} \frac{1}{e^{(2x^2-1/2)}} \geq E_{\text{erfc}}(x),$$

the prover will use the erfc series, else the erf series. In the latter case, we require that

$$\sqrt{\frac{27}{\pi}} \frac{1}{2e^{(2x^2-1/2)}} \geq \frac{B}{2},$$

which implies

$$\sqrt{\frac{27}{\pi}} \frac{1}{B} \geq e^{(2x^2-1/2)}.$$

The above is equivalent to

$$\ln(27/\pi)/2 + \ln(1/B) \geq 2x^2 - 1/2,$$

which implies

$$1.076 + \ln(1/B)/0.434 \geq (2x^2 - 1/2)$$

and

$$x_{\text{erfc}}^{min} = \sqrt{\frac{\ln(1/B)}{2} + 0.788} \geq x. \quad (3.36)$$

Now that we bounded the domains of erf and erfc, we can obtain the number of terms to evaluate for the series. As shown in Equation (3.29), this is an alternating series too, so to reach an error smaller than $B/2$, it is sufficient to truncate the series when terms get smaller than $B/2$ in absolute value. In particular, we need L terms with

$$\frac{2x^{2L+1}}{\sqrt{\pi}L!(2L+1)} \leq \frac{B}{2}.$$

Using again Stirling's approximation, this means

$$\frac{2x^{2L+1}}{\sqrt{\pi}\sqrt{2\pi L}(L/e)^L(2L+1)} \leq \frac{B}{2}.$$

Taking logarithms, we get

$$\ln(2) + (2L+1)\ln(x) - \ln(\pi\sqrt{2}) - (L+1/2)\ln(L) + L - \ln(2L+1) \leq \ln(B) - \ln(2).$$

We have that $2\ln(2) - \ln(\pi\sqrt{2}) - \ln(2L+1) \leq 0$, so the above inequality is satisfied if

$$(2L+1)\ln(x) - (L+1/2)\ln(L) + L \leq \ln(B)$$

which is equivalent to

$$(L+1/2)\ln(ex^2/L) \leq \ln(B),$$

or written differently:

$$(L+1/2)\ln\left(1-\frac{L-ex^2}{L}\right)\leq \ln(B)$$

As $\ln(1+\alpha)\leq\alpha$, this is satisfied if

$$-(L+1/2)\frac{L-ex^2}{L}\leq \ln(B),$$

which is equivalent to

$$(L+1/2)\frac{L-ex^2}{L}\geq \ln(1/B).$$

The above is satisfied if

$$L-ex^2\geq \ln(1/B).$$

It follows that we need

$$L\geq \ln(1/B)+ex^2.$$

Substituting the worst case value x_{erfc}^{min} of x from Equation (3.36), we get

$$L\geq\left\lceil\left(1+\frac{e}{2}\right)\ln(1/B)+0.79e\right\rceil$$

which, approximating, is implied if

$$L\geq\lceil 2.36\ln(1/B)+2.15\rceil = L_{\text{erf}}. \tag{3.37}$$

Now, to compute L_{erfc}, we just observe in Equation (3.35) that the error gets smaller as x increases, so the biggest error for a fixed number of terms of the series is in x_{erfc}^{min}. Therefore, plugging Equation (3.36) we have

$$\begin{aligned} L_{\text{erfc}} &= \lfloor (x_{\text{erfc}}^{min})^2+1/2\rfloor \\ &= \left\lfloor\frac{\ln(1/B)}{2}+0.788+\frac{1}{2}\right\rfloor \\ &= \left\lfloor\frac{\ln(1/B)}{2}+1.288\right\rfloor. \end{aligned} \tag{3.38}$$

Required precision ζ Now we determine the storage needed in our fixed precision representation such that the rounding error is smaller than $B/2$. We have to consider error propagation and the errors E_{div} and E_{mul} due to division and product ZKPs respectively, which are equal to $\zeta/2$. Let E_l be the error to compute the t_l terms of the erf series. The total erf rounding error is then

$$E_{\text{erf}}^{round}=\frac{2}{\sqrt{\pi}}\left(\sum_{l=0}^{L_{\text{erf}}}\frac{E_l}{2l+1}+E_{div}\right)+E_{mul}.$$

We require $1/\zeta M^2$ to be an integer so as the variable x is a multiple of $1/M$, we can represent x and x^2 without rounding. As $t_0=x$ we have that $E_0=0$, and for other terms

we have

$$\begin{aligned} t_{l+1} \pm E_{l+1} &= \frac{(t_l \pm E_l)x^2 \pm E_{mul}}{l+1} \pm E_{div} \\ &= \frac{t_l x^2 \pm E_l x^2 \pm \zeta/2}{l+1} \pm \frac{\zeta}{2} \\ &= \frac{t_l x^2}{l+1} \pm \left(\frac{E_l x^2 + \zeta/2}{l+1} + \frac{\zeta}{2} \right). \end{aligned}$$

Then the absolute error at term $l+1$ is

$$E_{l+1} = \frac{E_l x^2 + \zeta/2}{l+1} + \frac{\zeta}{2} \leq \frac{E_l x^2}{l+1} + \zeta.$$

For $1 \leq l \leq x^2 - 1$ we have $E_l \geq \zeta$ and

$$E_{l+1} = E_l \frac{x^2}{l+1} + \zeta \leq 2E_l \frac{x^2}{l+1}.$$

If $l+1 \leq x^2$, we can easily see that

$$E_l \leq \zeta 2^{l-1} \frac{x^{2(l-1)}}{l!}. \tag{3.39}$$

If $l+1 > x^2$, we have that

$$E_{l+1} = E_l \frac{x^2}{l+1} + \zeta \leq E_l + \zeta.$$

From the above and plugging Equation (3.39) we have

$$\begin{aligned} E_l &\leq E_{\lfloor x^2 \rfloor} + (l - \lfloor x^2 \rfloor)\zeta \\ &\leq \zeta 2^{\lfloor x^2 \rfloor} \frac{x^{2\lfloor x^2 \rfloor}}{\lfloor x^2 \rfloor!} + (l - \lfloor x^2 \rfloor)\zeta. \end{aligned}$$

From the above, independently of x^2 and l we have that

$$\begin{aligned} E_l &\leq \zeta 2^{\lfloor x^2 \rfloor} \frac{x^{2\lfloor x^2 \rfloor}}{\lfloor x^2 \rfloor!} + \sum_{k=\lfloor x^2 \rfloor + 1}^{l} \zeta \\ &= \zeta \frac{(2x^2)^{\lfloor x^2 \rfloor}}{\lfloor x^2 \rfloor!} + \sum_{k=\lfloor x^2 \rfloor + 1}^{l} \zeta. \end{aligned}$$

We assume $x \geq 1$ as E_{erf}^{round} is smaller when $x \in [0, 1)$. Using the Stirling approximation we can bound E_l to

$$\begin{aligned} E_l &\leq \zeta \frac{(2x^2)^{\lfloor x^2 \rfloor} e^{\lfloor x^2 \rfloor}}{\sqrt{2\pi \lfloor x^2 \rfloor} \lfloor x^2 \rfloor^{\lfloor x^2 \rfloor}} + \sum_{k=\lfloor x^2 \rfloor + 1}^{l} \zeta \\ &\leq \zeta \frac{(2e)^{x^2}}{\sqrt{2\pi} x} + \sum_{k=\lfloor x^2 \rfloor + 1}^{l} \zeta \\ &\leq \zeta \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{k=\lfloor x^2 \rfloor + 1}^{l} \zeta. \end{aligned}$$

Then

$$
\begin{aligned}
E_{\mathrm{erf}}^{round} &= E_{mul} + \frac{2}{\sqrt{\pi}} \sum_{l=0}^{L_{\mathrm{erf}}} E_{div} + \frac{E_l}{2l+1} \\
&\leq \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \sum_{l=0}^{L_{\mathrm{erf}}} \frac{\zeta}{2} + \frac{1}{2l+1} \left(\zeta \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{k=\lfloor x^2 \rfloor + 1}^{l} \zeta \right) \\
&= \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \zeta \left(\frac{L_{\mathrm{erf}}+1}{2} + \sum_{l=0}^{L_{\mathrm{erf}}} \frac{1}{2l+1} \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{l=0}^{L_{\mathrm{erf}}} \frac{1}{2l+1} \sum_{k=\lfloor x^2 \rfloor + 1}^{l} 1 \right) \\
&= \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \zeta \left(\frac{L_{\mathrm{erf}}+1}{2} + \sum_{l=0}^{L_{\mathrm{erf}}} \frac{1}{2l+1} \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{l=\lfloor x^2 \rfloor + 1}^{L_{\mathrm{erf}}} \frac{l - \lfloor x^2 \rfloor}{2l+1} \right) \\
&\leq \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \zeta \left(\frac{L_{\mathrm{erf}}+1}{2} + \sum_{l=0}^{L_{\mathrm{erf}}} \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{l=\lfloor x^2 \rfloor + 1}^{L_{\mathrm{erf}}} \frac{l - \lfloor x^2 \rfloor}{2l} \right) \\
&= \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \zeta \left(\frac{L_{\mathrm{erf}}+1}{2} + (L_{\mathrm{erf}}+1) \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{l=\lfloor x^2 \rfloor + 1}^{L_{\mathrm{erf}}} 1 - \frac{\lfloor x^2 \rfloor}{2l} \right) \\
&\leq \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \zeta \left(\frac{L_{\mathrm{erf}}+1}{2} + (L_{\mathrm{erf}}+1) \frac{(2e)^{x^2}}{\sqrt{2\pi}} + \sum_{l=\lfloor x^2 \rfloor + 1}^{L_{\mathrm{erf}}} 1 \right) \\
&\leq \frac{\zeta}{2} + \frac{2}{\sqrt{\pi}} \zeta \left(\frac{L_{\mathrm{erf}}+1}{2} + (L_{\mathrm{erf}}+1) \frac{(2e)^{x^2}}{\sqrt{2\pi}} + L_{\mathrm{erf}} + 1 \right) \\
&\leq \frac{\zeta}{2} + \frac{2(L_{\mathrm{erf}}+1)}{\sqrt{\pi}} \zeta \left(\frac{1}{2} + \frac{(2e)^{x^2}}{\sqrt{2\pi}} + 1 \right) \\
&\leq \frac{\zeta}{2} + \frac{2(L_{\mathrm{erf}}+1)}{\sqrt{\pi}} \zeta \frac{\sqrt{2}(2e)^{x^2}}{\sqrt{\pi}} \\
&= \left(\frac{1}{2} + \frac{\sqrt{8}(L_{\mathrm{erf}}+1)(2e)^{x^2}}{\pi} \right) \zeta \\
&\leq \frac{2\sqrt{8}(L_{\mathrm{erf}}+1)(2e)^{x^2}}{\pi} \zeta.
\end{aligned}
$$

By plugging Equation (3.37) we have

$$\begin{aligned}
E_{\text{erf}}^{round} &\leq \frac{2\sqrt{8}(2.36\ln(1/B)+2.15)(2e)^{x^2}}{\pi}\zeta \\
&\leq \frac{(13.36\ln(1/B)+12.17)(2e)^{(x_{\text{erfc}}^{min})^2}}{\pi}\zeta \\
&\leq \frac{(13.36\ln(1/B)+12.17)(2e)^{\ln(1/B)/2+0.788}}{\pi}\zeta \\
&\leq \frac{(13.36\ln(1/B)+12.17)(2e)^{\log_{2e}(1/B)\ln(2e)/2}(2e)^{0.788}}{\pi}\zeta \\
&\leq \frac{3.8(13.36\ln(1/B)+12.17)(1/B)^{\ln(2e)/2}}{\pi}\zeta \\
&\leq \frac{50.8\ln(1/B)+48.3}{\pi B^{0.85}}\zeta. \qquad (3.40)
\end{aligned}$$

The erfc case requires a similar analysis. To compute error of terms $m_0,\ldots,m_{L_{\text{erfc}}}$ defined in the ZKP we take into account the error propagation and the error of our finite precision. Let now F_i be the absolute error of the term m_i. We have that $F_0=0$ and

$$\begin{aligned}
m_{i+1}\pm F_{i+1} &= \frac{(m_i\pm F_i)(2i+1)}{2x^2}\pm E_{div} \\
&= \frac{m_i(2i+1)}{2x^2}\pm\left(\frac{F_i(2i+1)}{2x^2}+\frac{\zeta}{2}\right) \\
&= m_{i+1}\pm\left(\frac{F_i(2i+1)}{2x^2}+\frac{\zeta}{2}\right).
\end{aligned}$$

Hence,

$$F_{i+1}=\frac{F_i(2i+1)}{2x^2}+\frac{\zeta}{2}.$$

In erfc, $i<L_{\text{erfc}}=\lfloor(x_{\text{erfc}}^{min})^2+1/2\rfloor$ therefore

$$\frac{F_i(2i+1)}{2x^2}+\frac{\zeta}{2}\leq F_i+\frac{\zeta}{2}.$$

Then we have that $F_i\leq i\frac{\zeta}{2}$. The total error is

$$\begin{aligned}
E_{\text{erfc}}^{round} &= \frac{e^{-x^2}}{x\sqrt{\pi}}\sum_{i=0}^{L_{\text{erfc}}-1}F_i \\
&= \frac{e^{-x^2}}{x\sqrt{\pi}}\sum_{i=0}^{L_{\text{erfc}}-1}i\frac{\zeta}{2} \\
&= \frac{e^{-x^2}}{x\sqrt{\pi}}\frac{(L_{\text{erfc}}-1)L_{\text{erfc}}}{2}\frac{\zeta}{2}.
\end{aligned}$$

Plugging Equation (3.38) to the above we have

$$\begin{aligned} E_{\text{erfc}}^{round} &\leq \frac{(0.5\ln(1/B)+1.288)^2}{4e^{x^2}x\sqrt{\pi}}\zeta \\ &\leq \frac{0.25\ln^2(1/B)+1.288\ln(1/B)+1.659}{4e^{(x_{\text{erfc}}^{min})^2}\sqrt{\pi}}\zeta \\ &\leq \frac{0.25\ln^2(1/B)+1.288\ln(1/B)+1.659}{4e^{\ln(1/B)/2+0.788}\sqrt{\pi}}\zeta \\ &\leq \frac{0.25\ln^2(1/B)+1.288\ln(1/B)+1.659}{8\sqrt{1/B}\sqrt{\pi}}\zeta. \end{aligned} \tag{3.41}$$

Now we determine what value of ζ is needed. Equations (3.40) and (3.41) fix our requirements to

$$E_{\text{erf}}^{round} \leq \frac{50.8\ln(1/B)+48.3}{\pi B^{0.85}}\zeta \leq \frac{B}{2}$$

and to

$$E_{\text{erfc}}^{round} \leq \frac{0.25\ln^2(1/B)+1.288\ln(1/B)+1.659}{8\sqrt{1/B}\sqrt{\pi}}\zeta \leq \frac{B}{2}.$$

E_{erf}^{round} imposes the biggest constraint to ζ, as it requires that

$$\zeta \leq \frac{\pi B^{1.85}}{101.6\ln(1/B)+96.6} = O\left(\frac{B^{1.85}}{\ln(1/B)}\right).$$

Typically, one would like the total error (due to approximation and rounding) to be negligible with respect to the standard deviation σ_η, so one could choose $B = \sigma_\eta/10^6|U^H|$.

Computation Cost We now evaluate the computational cost of the proof. When we say a task has "cost c", it means that requires at most c cryptographic computations for generating the proof, c for another party to verify it, and the exchange of $c\mathbb{G}$ bits, recalling that we overload $\mathbb{G}$ to be the size in bits of an element of the set $\mathbb{G}$.

The main statement of the ZKP is

$$\begin{aligned} \Bigg\{ &\left(x \in \left[0, \left\lfloor \frac{y_{min}^{\text{erfc}}}{\zeta} \right\rfloor\right] \wedge y = \text{erf}(x)\right) \\ &\vee \left(y \in \left[\left\lfloor \frac{y_{min}^{\text{erfc}}}{\zeta} \right\rfloor + 1, \frac{1}{\zeta}\right] \wedge 1 - y = \text{erfc}(x)\right)\Bigg\} \end{aligned} \tag{3.42}$$

where $y_{min}^{\text{erfc}} = \text{erf}\left(x_{\text{erfc}}^{min}\right)$ is a public constant. The main costs are in the proofs of erf and erfc.

Proving computations of erf in Equation (3.33) requires 3 range proofs of cost of at most $10\log_2(1/\zeta)$, $10\log_2(l)$ and $10\log_2(2l+1)$. As $l \leq L_{\text{erf}} = 2.36\ln(1/B)$, the cost of evaluating a term is $10\log_2(1/\zeta) + 20\log_2(\ln(1/B))$. We evaluate L_{erf} terms. The total cost is

$$L_{\text{erf}}(10\log_2(1/\zeta) + 20\log_2(\ln(1/B))) = 10L_{\text{erf}}\log_2(1/\zeta)L_{\text{erf}} + 20\log_2(\ln(1/B)).$$

The dominating term above is

$$10\log_2(1/\zeta)L_{\text{erf}} = 23.6\log_2(1/\zeta)\ln(1/B) < 34.1\ln(1/\zeta)\ln(1/B).$$

For erfc, we require L_{erfc} proofs of the computation in Equation (3.34), one for each term, and its cost is dominated by

$$10\log_2(q)L_{\text{erfc}} = 10\log_2(q)\left\lfloor 0.5\ln(1/B) + 1.288\right\rfloor.$$

Neglecting lower order constants, the above is dominated by

$$10\log_2(q)0.5\ln(1/B) = 5\log_2(e)\ln(q)\ln(1/B) < 7.22\ln(q)\ln(1/B).$$

The total cost is the sum of the costs of the erf and erfc ZKPs, which is dominated by

$$\begin{aligned}&\ln(1/B)\left(34.1\ln(1/\zeta) + 7.22\ln(q)\right)\\ &\leq \log_2(1/B)\left(17.05\log_2(1/\zeta) + 3.61\log_2(q)\right).\end{aligned}$$

3.7 Computation and Communication Costs

This section summarizes each component of the cost of GOPA, relying in particular on the cost of proving computations of Section 3.6.

Dominant computations are exponentiations in the cryptographic group $\mathbb{G}$ defined for Pedersen commitments in Section 2.4. The cost of signing messages is negligible. We describe costs centered on any user $u \in U$, which is natural as most operations can be performed asynchronously and in parallel. For simplicity, when we say a task costs c, it means that costs at most c computations for proving a computation, c computations for another party verifying this computation and it requires the exchange of $c\mathbb{G}$ bits, where we overload $\mathbb{G}$ to be size in bits of an element of the set $\mathbb{G}$. Some computations depend on fixed-precision parameter ζ to represent numbers in $\mathbb{Z}_q$ (see Section 3.6.1) and the amount $1/B$ of equiprobable bins used to sample independent noise η_u (see Section 3.6.7).

Each phase of the verification protocol is summarized in Section 3.6.2. The cost breaks down as follows:

- The Phase 2 Setup requires generating public randomness which has constant cost (except for $O(1)$ parties that perform some extra computations, see Algorithm 3) and private seeds that require 2 range proofs in the interval $[0, 1/B\zeta]$, with a final cost of $20\log_2(1/B) + 20\log_2(1/\zeta)$.

- Validity of input at Phase 3 Verification requires a range proof in the interval $[0, 1/\zeta]$, with a cost of $10\log_2(1/\zeta)$. Extra computations of consistency cannot be accounted here as they depend on the nature of external computations.

- Correctness of computations of properties (3.25) and (3.26) at Phase 3 Verification cost at most $5|N(u)|+4$, accounting the computations over commitments and proofs of knowledge of terms of each property.

- The verification of Property (3.27) at Phase 3 Verification costs

  $$\log_2(1/B)(17.05\log_2(1/\zeta) + 3.61\log_2(q))$$

  (see Section 3.6.7).

The overall cost of a protocol for user u is at most

$$5|N_u| + 20\log_2(1/B) + 30\log_2(1/\zeta) + \log_2(1/B)(17.05\log_2(1/\zeta) + 3.61\log_2(q)) + 4.$$

The following statement summarizes our results.

Theorem 8 (Complexity of GOPA)**.** *Let $\zeta > 0$ be the desired fixed precision such that the number 1 is represented as $1/\zeta$. Let $B > 0$ be such that the η_u's are drawn from a Gaussian distribution approximated with $1/B$ equiprobable bins. Then, each user u, to perform and prove its contribution, requires $O(|N(u)| + \log(1/\zeta)\log(1/B) + \log(1/B) + \log(1/\zeta))$ computations and transferred bits. The verification of its contribution requires the same cost.*

3.8 Experiments

Private averaging We present some numerical simulations to study the empirical utility of GOPA and in particular the influence of malicious and dropped out users. We consider $n = 10000$, $\varepsilon = 0.1$, $\delta = 1/n^2$ and set the values of k, σ_η^2 and σ_Δ^2 using Corollary 1 so that GOPA satisfies (ε, δ)-DP. Figure 3.2 (left) shows the utility of GOPA as a function of ρ, which is the (lower bound on) the proportion of users who are honest and do not drop out. We see that even for reasonably small ρ, GOPA achieves a utility of the same order as a trusted curator that would average the values of *all* n users. In Section 3.6.4, we already described the ability of GOPA to tolerate a small number of residual pairwise noise terms of variance σ_Δ^2 in the final result. We note that this feature is rather unique to GOPA and is not possible with secure aggregation [22, 12].

Application to federated SGD We present some experiments on training a logistic regression model in a federated learning setting. We use a binarized version of UCI Housing dataset with standardized features and points normalized to unit L2 norm to ensure a gradient sensitivity bounded by 2. We set aside 20% of the points as test set and split the rest uniformly at random across $n = 10000$ users so that each user u has a local dataset D_u composed of 1 or 2 points.

We use the Federated SGD algorithm, which corresponds to FedAvg with a single local update [104]. At each epoch, each user computes a stochastic gradient using one sample of their local dataset; these gradients are then averaged and used to update the model parameters. To privately average the gradients, we compare GOPA (with $\rho = 0.5$) to (i) a *trusted* aggregator that averages all n gradients in the clear and adds Gaussian noise to the result as per central DP, and (ii) local DP. We fix the total privacy budget to $\varepsilon = 1$ and $\delta = 1/(\rho n)^2$ and use advanced composition to compute the budget allocated to each epoch. The step size is tuned for each approach, selecting the value with the highest accuracy after a predefined number T of epochs.

Figure 3.2 (middle) shows a typical run of the algorithm for $T = 50$ epochs. Local DP is not shown as it diverges unless the learning rate is overly small. On the other hand, GOPA is able to decrease the objective function steadily, although we see some difference with the trusted aggregator (this is expected since $\rho = 0.5$). Figure 3.2 (right) shows the final test accuracy (averaged over 10 runs) for different numbers of epochs T. Despite the

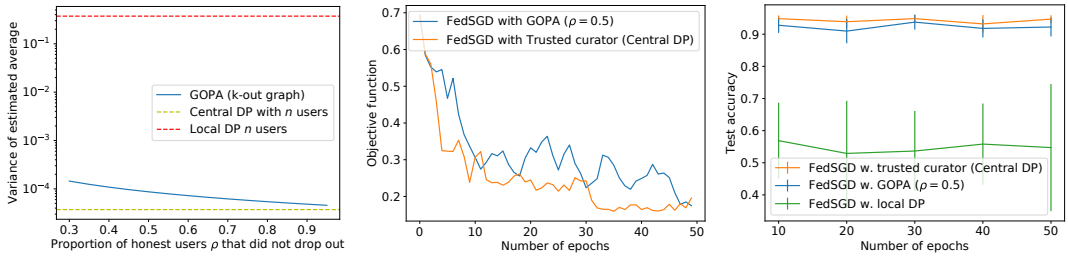


Figure 3.2: Comparing GOPA to central and local DP. *Left*: Utility of GOPA (measured by the variance of the estimated average) w.r.t. ρ. *Middle:* Evolution of the objective for a typical run of FedSGD. *Right:* Test accuracy of models learned with FedSGD. See text for details.

small gap in objective function, GOPA nearly matches the accuracy achieved by the trusted aggregator, while local DP is unable to learn useful models.

3.9 Conclusion

We proposed GOPA, a protocol to privately compute averages over the values of many users. GOPA satisfies DP, can nearly match the utility of a trusted curator, and is robust to malicious parties. It can be used in distributed and federated ML [80, 84] in place of more costly secure aggregation schemes. In future work, we plan to provide efficient implementations, to integrate our approach in complete ML systems, and to exploit scaling to reduce the cost per average. We think that our work is also relevant beyond averaging, e.g. in the context of robust aggregation for distributed SGD [19] and for computing pairwise statistics [11].

Chapter 4

Private Sampling with Malicious Samplers

In this chapter, we present our second contribution. We first discuss the importance of privacy-preserving verifiable sampling and define different types of sampling tasks. We propose a high-level methodology and several protocols to perform these tasks for different statistical distributions. Then, we analyze the security of these protocols in the presence of malicious participants. After that, we provide a comparison of accuracy and cost between the proposed and previously existent techniques. Finally, we show their application to distributed mechanisms for differential privacy.

4.1 Introduction

Nowadays, randomization is an important algorithmic technique. Its numerous applications include randomized algorithms, e.g., for many problems the simplest or most efficient known solution strategy is a randomized algorithm, and hiding information, e.g., in cryptography or in differential privacy. While true randomness is hard to achieve in most cases it is sufficient to be able to generate pseudo-random numbers. A wide range of approaches exist to generate pseudo-random numbers of good quality.

The situation becomes more complicated when we consider generating random numbers in the context of multi-party computation between parties which do not trust each other. We are particularly interested in algorithms which allow multiple parties to draw a random number from a specified probability distribution in such a way that all parties can be convinced that the number drawn is truly random and that either all parties, only one party, or none of the parties learn the drawn random number. This implies that no party should be able to influence the probability distribution or be able to predict or guess the random number.

Such algorithms are particularly useful for differentially private federated machine learning using sensitive data from multiple data owners. In this setting, one would like to learn a statistical model $\mathcal{M}$ with parameters θ on the sensitive data of multiple data owners. Such model could reveal sensitive information and therefore one possible technique is to perturb the model before publication sufficiently such that it becomes differentially private [60], i.e., such that from the perturbed model $\hat{\mathcal{M}}$ with parameters $\hat{\theta}$ one cannot distinguish a change in a single individual. This can be achieved by drawing some noise η from an appropriate probability distribution, e.g., η often is a vector of Laplace or Gaussian random variables, and setting $\hat{\theta} = \theta + \eta$. In such scenario it is important nobody knows η as else

that party could subtract η from the published $\hat{\theta}$ to obtain the sensitive model parameters θ. At the same time, all data owners want to be sure that η is drawn correctly: if anyone can bias the distribution of this noise, privacy may not be guaranteed anymore or the model parameters may be biased in a way similar as what one can achieve with data poisoning [123, 130].

In this paper we develop algorithms to verifiably draw random numbers. We consider uniform distributions, Laplace distributions, Gaussian distributions and arbitrary distributions. We develop strategies with three different privacy levels for the random number: strategies which verifiably draw a publicly known random number, strategies which verifiably draw a random number which is revealed to only one party and strategies which verifiably draw a random number and output it as a shared secret so that none of the parties knows the random number.

An important tool to prove correct behavior can be found in zero knowledge proofs (ZKP). These are cryptographic techniques that allow a party to prove statements without revealing anything else. Typically, one considers statements involving logical and arithmetic relations over private values which can be expressed using additions, multiplications and other elementary operations such as comparisons. For drawing from Laplace or Gaussian distributions, transcendental functions are needed. We work towards bridging this gap based on Cordic [129], a classic technique for computing such functions.

The main contributions of this paper can be summarized as follows: (1) we propose strategies to prove relationships involving logarithms or trigonometric functions in zero knowledge, (2) we propose and compare several strategies to let a party verifiably draw Gaussian random numbers, (3) we propose algorithms to let one party verifiably sample from the Laplace distribution and from an arbitrary distribution, (4) we propose algorithms to draw from the Gaussian or Laplace distribution a random number represented as a shared secret.

The remainder of this chapter is structured as follows: After reviewing some preliminary concepts in Section 4.2, we formalize our problem statement in Section 4.3. Next, in Section 4.4 we discuss related work and in Section 4.5 we provide a high-level overview of our method. After that, in Section 4.6 we review the Cordic algorithm and adapt it for zero-knowledge proofs. In Sections 4.7 and 4.8 we apply these techniques for sampling from the Laplace and Gaussian distributions. In Section 4.9, we analyze the security of our protocols. To show how our methods work in practice, in Section 4.10 we provide an experimental comparison of the several possible strategies to sample from the Gaussian distribution. In Section 4.11 we discuss the application of our techniques to the problem of differentially private machine learning. Finally, in Section 4.12 we conclude and outline directions of future work.

4.2 Preliminaries

We will follow the general notation in Section 2.1. We use the *vector* variant of Pedersen commitments, and pseudo-random generators (PRG) defined in Section 2.4. Parties communicate through secure channels and have access to a public bulletin board, as defined in Section 3.6.1 for GOPA, that they can use to post messages. When a party sends a message to the bulletin board, it is forwarded to all other agents as when using a broadcast channel. In addition, all broadcasted messages remain publicly visible in the bulletin board, which allows to have publicly verifiable protocols.

In parts of our protocols, we make use of specific Zero Knowledge Proofs that are non-interactive versions of compressed Σ-protocols defined in Section 2.5 and whose security relies on the Random Oracle Model [13].

We briefly summarize our threat model, which compressed Σ-protocols we use and describe secret sharing below.

Threat Model We consider a set of n parties $\mathbf{P} = \{P_1, \ldots, P_n\}$. We assume that a subset of parties $\mathbf{P}_{cor} \subset \mathbf{P}$ is corrupted and controlled by an adversary $\mathcal{A}$. $\mathcal{A}$ can make corrupted parties to deviate arbitrarily from the protocol and perform coordinated attacks. Our protocols are secure if at least one party is honest. The set $\mathbf{P}_{cor}$ of corrupted parties is assumed to be static, meaning that it does not change after the beginning of the execution.

We prove security in the simulation paradigm, using the *covert security model* in its *strong explicit cheat* formulation [6, Def. 3.3]. We assume parties are able to detect malicious actions and can in such cases abort the protocol. Deterrence measures may be in place to discourage parties from being detected as malicious. In fact, unless parties stop participating our protocols either complete successfully or abort with a proof that a specific party is a cheater, i.e., in case our protocols abort the message trace (which is kept on the bulletin board) allows for proving that a specific party did not follow the protocol.

Protocols with covert security provide in most cases weaker guarantees than in the framework of *malicious security with abort* [71, Ch 7], because the former allows a certain probability of success when cheating while in the latter it is guaranteed that protocols either finish correctly or abort. In our protocols, cheaters are detected with overwhelming probability, which provides the same guarantees as the model of malicious security with abort [6, Prop. 3.10]. We have nonetheless decided to use the definition of the covert security model as it explicitly formalizes the ability of honest parties to determine who has cheated. Assuming that adversaries will be deterred if they risk getting caught is a standard assumption that applies in many scenarios [6].

We provide a detailed description of our security framework and prove the security of our protocols in Section 4.9.

Compressed Σ-protocols We use compressed Σ-protocols described in sections 2.5.3 to 2.5.7. Particularly, we use Protocol Π_{cs} of in Section 2.5.5, that proves the nullity of the output of arithmetic circuits in $\mathbb{Z}_q$ applied to private inputs. Let $\mathbb{G}$, $\mathbb{Z}_q$, and $\bar{g}$ be as defined for vector commitments, then for any circuit $C : \mathbb{Z}_q^k \to \mathbb{Z}_q^s$, by applying Π_{cs} to C we obtain a complete, sound and zero knowledge proof for the relation

$$\{(P \in \mathbb{G}; \bar{x} \in \mathbb{Z}_q^k) : P \text{ is a commitment of } \bar{x} \wedge C(\bar{x}) = 0)\}.$$

While Π_{cs} is an interactive protocol between $\mathcal{P}$ and $\mathcal{V}$, it can be turned into a non-interactive proof using the Strong Fiat-Shamir heuristic [16]. By this transformation, ZKPs can be generated offline by $\mathcal{P}$ and later be verified by any party.

Let m be the number of multiplication gates of C, then the proof generated by the execution of Π_{cs} has a size of $2\lceil\log(k + 2m + 4)\rceil - 1$ elements of $\mathbb{G}$ and 6 elements of $\mathbb{Z}_q$. To generate such proof, the dominant computations are modular exponentiations in $\mathbb{G}$ (GEX). $\mathcal{P}$ performs $5k + 8m + 2\lceil\log_2(k + 2m + 4)\rceil + 6$ GEX, and the verification cost is of $k + 2m + 2\lceil\log_2(k + 2m + 4)\rceil - 1$ GEX. Details on costs are explained in more detail in Section 2.5.7.

Secret Sharing Consider a set of n parties $\{i\}_{i=1}^n$. For a positive prime q, group $\mathbb{Z}_q$, and a number $a \in \mathbb{Z}_q$, one can generate an additive secret share for a by drawing a random vector $(a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ subject to the constraint that $\sum_{i=1}^n a_i = a \bmod q$. We then denote this sharing of a as $[\![a]\!] = (a_1 \ldots a_n)$. The process of computing and revealing a from the sharing $[\![a]\!]$ is called opening the sharing $[\![a]\!]$. If every party i only receives a_i (for $i \in [n]$), then if not all parties collude each party can only see at most $n-1$ uniformly randomly distributed numbers, and hence has no information about the value of a.

If for one or more values a sharing is available, it is possible to perform various operations on them without revealing any new information, see [51] for an overview. If $[\![a]\!] = (a_1 \ldots a_n)$ is a sharing of a and $[\![b]\!]$ is a sharing of b, then $[\![a+b]\!] = (a_1 + b_1 \ldots a_n + b_n)$ is a sharing of $a+b$. Given a sharing $[\![a]\!]$ of a and a public constant c, then $[\![ca]\!] = (ca_1 \ldots ca_n)$ is a sharing of ca. For multiplying two sharings, one can use pre-computed triples of sharings $([\![x]\!], [\![y]\!], [\![z]\!])$ with x and y random and $xy = z$. Given such triple, and two sharings $[\![a]\!]$ and $[\![b]\!]$ which one wants to multiply, one can compute $[\![d]\!] = [\![a]\!] - [\![x]\!]$ and $[\![e]\!] = [\![b]\!] - [\![y]\!]$ and open both $[\![d]\!]$ and $[\![e]\!]$. Then, a sharing of $c = ab$ is obtained by $[\![c]\!] = [\![z]\!] + d[\![y]\!] + e[\![x]\!] + de$. Several approaches have been proposed to generate such triples of sharings $([\![a]\!], [\![b]\!], [\![c]\!])$ efficiently, typically involving a somewhat homomorphic encryption (SHE) scheme with distributed decryption, where the parties can generate random sharings $[\![a]\!]$ and $[\![b]\!]$ uniformly at random, encrypt them, multiply them and decrypt the product in a distributed way to obtain $[\![c]\!]$ [51].

We will adopt a number of ideas from [50]. In particular, we will represent sharings in binary form, denoting by $BITS(x, (x^{(i)})_{i=0}^{l-1})$ the relation $[\![x]\!] = \sum_{i=0}^{l-1} [\![x^{(i)}]\!] 2^i$ with $x^{(i)} \in \{0,1\}$.

The protocol BIT-ADD$((\![x^{(i)}]\!])_{i=0}^{l-1}; ([\![y^{(i)}]\!])_{i=0}^{l-1})$ returns $([\![z^{(i)}]\!])_{i=0}^{l-1}$ such that there holds $z = x + y$ for $BITS(x, ([\![x^{(i)}]\!])_{i=0}^{l-1})$, $BITS(y, ([\![y^{(i)}]\!])_{i=0}^{l-1})$ and $BITS(z, ([\![z^{(i)}]\!])_{i=0}^{l-1})$. To implement it, let $c^{(-1)} = 0$. For $i = 0 \ldots l-1$: $[\![z^{(i)}]\!] =$ BIT-XOR$([\![x^{(i)}]\!], [\![y^{(i)}]\!], [\![c^{(i-1)}]\!])$ where BIT-XOR$(a,b) = (a-b)^2$; and $[\![c^{(i)}]\!] = [\![x^{(i)}]\!][\![y^{(i)}]\!] + [\![c^{(i-1)}]\!]((1 - [\![x^{(i)}]\!])[\![y^{(i)}]\!] + [\![x^{(i)}]\!](1 - [\![y^{(i)}]\!]))$. After this loop, return $([\![z^{(i)}]\!])_{i=1}^{l}$.

4.3 Problem Statement

We call Π a *sampling protocol* over a domain $\mathcal{X}$ if Π is a randomized multi-party protocol which outputs sequences of elements of $\mathcal{X}$. We consider sampling protocols which take only one input per party at the beginning of the protocol. In particular, let $U = \{i\}_{i=1}^n$ be the set of n parties which participate to a sampling protocol Π, and let s_i be the input (also called *seed*) of party i (for $i \in [n]$). We denote the output of Π by $\Pi(\bar{s})$ where $\bar{s} = (s_i)_{i=1}^n$ is the vector of seeds. We assume that there is some increasing polynomial $p : \mathbb{N} \to \mathbb{N}$ such that if $\bar{s} \in \{0,1\}^{k \times n}$ then $\Pi(\bar{s}) \in \mathcal{X}^{p(k)}$. Let $\bar{s}_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_{|s|})$ denote the vector $\bar{s}$ without the i-th component.

We use the threat model defined in Section 2.1, but we do not consider the problem of parties dropping out. Therefore, malicious parties do not drop out as otherwise they will be immediately detected and considered cheaters. For a multi-party protocol Π, we say a party is *honest* if it follows the steps of protocol Π correctly and does not collude with other parties. We say that a sampling protocol Π *correctly samples* from a probability distribution $\mathcal{D}$ if there is a function μ with $\mu(k)$ negligible in k such that for every run of Π by parties $U = \{i\}_{i=1}^n$, there exists $i \in [n]$ such that party i is honest, for every $\bar{s}_{-i} \in \{0,1\}^{k \times (n-1)}$ and for any probabilistic polynomial time algorithm $A : \{0,1\}^{k \times (n-1)} \times \mathcal{X} \to \{0,1\}$,

there holds $|P_{s_i \leftarrow_R \{0,1\}^k}(A(\bar{s}_{-i}, \Pi(\bar{s})) = 1) - P_{x \leftarrow_R \mathcal{D}^{p(k)}}(A(x) = 1)| \leq \mu(k)$, where $\mathcal{D}^{p(k)}$ draws vectors from $\mathcal{X}^{p(k)}$ whose components are independently distributed according to $\mathcal{D}$. In other words, if there is at least one honest party, then Π acts as a PRG even if all parties except that honest party would disclose their seeds. As a result, as soon as a single party is honest it can trust that any output of Π used by any party is pseudo-random and no party could predict it in advance. We denote the fact that x is correctly drawn from $\mathcal{D}$ by $x \leftarrow^*_R \mathcal{D}$.

We say a protocol Π *verifiably samples* from $\mathcal{D}$ if Π correctly samples from $\mathcal{D}$ and after every execution of Π the value of x is uniquely defined given the union of the information obtained by all parties and the information published by Π is sufficient to convince any party that x has been correctly drawn. We denote the fact that x is verifiably drawn from $\mathcal{D}$ by $x \leftarrow^V_R \mathcal{D}$.

In this paper, we will often informally consider both discrete and continuous probability distributions, and $P_{\mathcal{D}}$ then either represents a probability mass or probability density according to the context. As computers work with finite precision, we will eventually discretize up to some parameter-defined precision. While in the end all distributions will be discrete, we will use the continuous representations whenever this simplifies the explanation.

In the sequel, unless made explicit otherwise, we will assume there are n parties among whom at least one is honest, and that $\mathcal{D}$ is a publicly agreed probability distribution. Also, to simplify the explanation we will often describe protocols generating just one random number, the extension to streams of random numbers is then straightforward.

We can distinguish several types of verifiable sampling protocols, depending on how they output the sampled number x. For a verifiable sampling protocol Π, we say it is a

- **public draw** if after running Π the value of x is published.
- **private draw** if after running Π exactly one party knows x, but the other parties have no information on x next to the prior distribution $\mathcal{D}$.
- **hidden draw** if after running Π the parties have received an additive secret share $(x_1, \ldots, x_n)$ for x, but still no party has any information about x next to the prior distribution $\mathcal{D}$.

In this paper, we study the problem of finding efficient verifiable sampling protocols of each of the three above types given the probability distribution $\mathcal{D}$.

This problem is reasonably straightforward if $\mathcal{D}$ is the uniform distribution over the integers in the interval $[0, L)$ for some $L > 0$:

Protocol 1 (Public uniform sampling)**.** *For each $i \in [n]$ let party i generate its own random number r_i uniformly distributed over $[0, L)$ from its own secret seed s_i and publish a commitment C_i to it. Then, all parties open their commitment, i.e., they publish r_i and the randomness associated to the commitment to prove that C_i was a commitment to r_i. Finally, all parties compute publicly $\sum_{i=1}^{n} r_i \bmod L$.*

It is easy to see Protocol 1 draws r verifiably: if at least one party i is honest, it has generated a uniformly distributed number r_i and r is also uniformly distributed because no dishonest party j can change their r_j as a function of other parties because they start with a commitment on their r_j.

Protocol 2 (Private uniform sampling)**.** *One can sample a vector of k numbers private to party 1 as follows: party 1 draws uniformly at random a vector $\bar{a} = (a_1, \ldots, a_k) \in [0, L)^k$ and publishes a vector commitment C to it. Then, all parties generate jointly a public random number $r \in [0, L)$ with Protocol 1. Party 1 expands r to random numbers $(r_1, \ldots, r_k) \in [0, L)^k$ using a PRG. Finally, for $i \in [k]$, party 1 computes $u_i = a_i + r_i \bmod L$ and performs a zero knowledge proof of the modular sum for each u_i. $(u_1, \ldots, u_k)$ is a vector of private uniform random numbers.*

Again, it is easy to see that $(u_1, \ldots, u_k)$ is drawn verifiably. Note that Protocols 1 and 2 are simplified versions of previously described methods for generation of uniform random numbers, explained in Section 3.6.3 of Chapter 3.

Protocol 3 (Hidden uniform sampling)**.** *For each $i \in [n]$ let party i generate its own random number r_i uniformly distributed over $[0, L)$ and publish a commitment C_i to it. Then, they consider $(r_1, \ldots, r_n)$ as a secret share of the random number $r = \sum_{i=1}^{n} r_i \bmod L$.*

After running Protocol 3, if there is a honest party, r is fixed and follows the right probability distribution, and as not all parties collude no party knows more about r than that it follows the uniform distribution over $[0, L)$.

The problem of finding efficient verifiable sampling protocols becomes more challenging when $\mathcal{D}$ is not the uniform distribution, but a more general distribution such as a normal distribution or a Laplace distribution. Even for single party computation there sometimes exist multiple approaches with varying cost and precision.

4.4 Related Work

Below we describe lines of work that are related to ours.

Multiparty Computation between unreliable participants The seminal work of [20] proposed the first protocol to sample a public random bit (i.e. tossing a coin) between two parties that do not trust each other. Subsequent works such as [26] proposed protocols to perform coin tossing between an arbitrary number of parties.

The work of [43] proved that in the malicious model without aborts it is impossible that a multiparty protocol is guaranteed to finish correctly and perform an unbiased coin toss if the number of malicious users is half or more of the total of participants. However, it is possible to produce an unbiased outcome if we do not guarantee that the protocol will always finish no matter what malicious participants do. Our protocols can generate unbiased random numbers even if only a few parties are honest-but-curious, by allowing for aborts when parties detect that other parties cheat.

The work [6] proposes covert security, a framework for security against adversaries where cheating adversaries can be caught. Our work fits in that framework. However, in our work, cheaters get caught with overwhelming probability, which makes our protocols very similar to malicious security with abort [71, Ch 7]. The malicious security with abort framework, however, is slightly less expressive as it does not allow for detecting which parties are cheating.

The work [72] proposes a method to securely perform a wide family of randomized computations (related to interactive games) over private data and private random numbers, using zero knowledge proofs to verify correctness. They prove that this is secure in the ideal paradigm without abort if the majority of parties is honest.

Sampling from Gaussians and other popular non-uniform distributions. Distributions such as the Gaussian distribution, the Laplace distribution, the Poisson distribution or the exponential distribution are important in the field of statistics. Algorithms to securely draw from such distributions have applications in federated machine learning. Several contributions concern the problem of verifiable noise for differential privacy [127, 88] and hence can benefit from secure drawing.

Even in the semi-honest model where parties follow the specified protocol, drawing hidden random numbers is sometimes non-trivial. For example, in [41] one needs to make a sum of statistics and a Laplacian-distributed noise term, hence the authors propose a protocol where parties generate random numbers summing to a Laplacian distributed value which can then be included in a secure aggregation without being revealed.

In [59], protocols are proposed to generate secret-shared samples for Gaussian, Exponential and Poisson distributions. For the Gaussian distribution, their approach generates samples by averaging uniform seeds, a method which we call the Central Limit Theorem (CLT) approach. We compare the CLT approach with our approaches in Section 4.10. Even if more than a decade has passed since [59], recent contributions still resort to these techniques to generate Gaussian samples among unreliable participants. For example, recent protocols use the technique of [59] by adapting it to generate private draws from the Exponential distribution [111] and to sample hidden draws from the Binomial distribution [18]. The work of [102] proposes techniques to securely sample from the geometrical and Gaussian distributions, both building on [59], and studies them in the light of differentially private memory access patterns. In addition, [102] defines an extension of the malicious security model which includes information leakage, as measured in differential privacy, and proves the security of their protocols within this model.

In our work, we propose new techniques for privately drawing from the Gaussian distribution and show that all our techniques for all but the lowest precision requirements outperform the technique of [59], which is the most efficient method known so far. The same dynamics are at play for exponential, Poisson and Laplace distributions. Compared to the techniques in [59], our methods have a better complexity as a function of the precision parameter. We extend our methodology to hidden draws of Gaussian, Laplacian and Arbitrary distributions. Achieving sufficient precision when sampling is important for both the statistical quality and the security of the algorithms [108].

Implementation of math functions using cryptographic primitives Using secret sharing techniques, there is a large body of work on how to compute math functions such as square roots, logarithms and trigonometric [53, 4, 96, 10, 75, 75]. However they usually rely on splines or other approximation techniques that approximate functions by splitting the domain and using low-degree polynomials for each part. Alternatively, they rely on rational approximations. Piecewise approximations require the use of conditionals which are expensive when computing with secret shares, and rational approximations only allow for a fixed precision. Our work uses iterative approximations which allow to customize the precision of the approximation and are easy to compute given that we avoid the use of comparison gates in our circuits. Furthermore, for the Gaussian distribution, piecewise approximations require an external method to sample from the tails of the distribution. We also show protocols for private sampling from Gaussian and Laplace distributions where we avoid the high cost of secret shared computation by letting one party perform the calculation and then prove correct behavior using compressed Σ-protocols.

Zero Knowledge Proofs for such functions, as we apply in our work, is a less explored

technique. [133] proposes techniques to prove a limited set of relations involving common activation functions in machine learning.

4.5 Method

We start with discussing two generic approaches: a strategy based on the inverse cumulative probability distribution and a strategy based on table lookup.

4.5.1 Inverse Cumulative Probability Distribution

Assume $\mathcal{D}$ is a probability distribution on $\mathcal{X} \subseteq \mathbb{R}$. The cumulative probability distribution is defined as

$$F_{\mathcal{D}}(x) = \int_{t \leq x} P_{\mathcal{D}}(t)\,\mathrm{d}t$$

To the extent $\mathcal{D}$ is discrete, we can see $P_{\mathcal{D}}$ as a sum of scaled Dirac delta functions over which integration is possible and results in a sum. Then, the inverse $F_{\mathcal{D}}^{-1}$ is a function on the interval $(0, 1)$.

An approach to sampling from arbitrary distributions $\mathcal{D}$ on domains $\mathcal{X} \subseteq \mathbb{R}$, known as the inversion method, consists of sampling uniformly from the $(0, 1)$ interval and applying the inverse of the cumulative distribution function $F_{\mathcal{D}}^{-1}$. Indeed, if $t \leftarrow_R (0, 1)$, then $P(F_{\mathcal{D}}^{-1}(t) = x) = P_{\mathcal{D}}(x)$.

Public Sampling from an Arbitrary Distribution This approach can easily be applied to draw random numbers publicly:

Protocol 4 (Public draw from arbitrary distribution)**.** *Run Protocol 1 to generate a public uniformly distributed random number r', and then publicly compute $r = F_{\mathcal{D}}^{-1}(r')$.*

Using the inversion method for private or hidden draws is more involved since one needs a multi-party algorithm to compute $F_{\mathcal{D}}^{-1}$ or a ZKP algorithm to prove to other parties that $F_{\mathcal{D}}^{-1}$ was applied correctly. In many practical cases, $F_{\mathcal{D}}^{-1}$ does not have a simple closed form. This especially holds for the Gaussian distribution which we will discuss in more detail in Section 4.8.

We can extend this method to multi-variate distributions. For example, consider a distribution $\mathcal{D}$ over $\mathbb{R}^2$. To sample a pair (x, y) according to $\mathcal{D}$, we first define $P_x(x) = \int P_{\mathcal{D}}(x, y)\,\mathrm{d}y$, apply the inversion method to draw a random number x according to P_x, and then define $P_{y|x}(y) = P_{\mathcal{D}}(y|x) = P_{\mathcal{D}}(x, y)/P_x(x)$ and apply again the inversion method to draw a random y.

4.5.2 Table Lookup

As pointed out above, practical inverse cumulative probability functions are often expensive to compute, especially in a secure multi-party setting. In such scenarios approaches such as the ones discussed in Sections 4.5.1 and 4.8 incur a high cost for each drawn random number. In this section we consider an approach based on *table lookup*. While the involved techniques are well-known, this approach is interesting as a baseline, especially as it has a number of properties which are different from the other methods considered in this paper. In particular, the method studied here has a high pre-processing cost but then allows for drawing random numbers at a low constant cost per drawn random number.

Protocol 5 (Table-lookup private sampling)**.**

1. *Preprocessing. Let $M \in \mathbb{N}$. The parties publicly pre-compute the pairs*

$$\left(i, F_{\mathcal{D}}^{-1}\left(\frac{2i-1}{2M}\right)\right)$$

for all $i \in [M]$ and store them into a database DB.

2. *Sampling. Party 1 privately draws using Protocol 2 a random number r' distributed uniformly in $[M]$. Then, party 1 sets $r = F_{\mathcal{D}}^{-1}\left(\frac{2r'-1}{2M}\right)$, publishes commitments to r' and r, and publishes a ZKP that $(r', r) \in$ DB.*

In Protocol 5 a zero knowledge set membership proof is needed. There is a large body of work on this topic since in [28] a seminal method was shown that has a large preprocessing cost (linear in M) but only a unit communication cost for proving membership. Several improvements have been proposed which vary in their assumptions and efficiency, [15] discusses some lines of recent work.

Only already storing the database DB may take a prohibitive amount of space if a high precision is needed, as M is exponential in the number of desired correct digits. As a result, this technique can only be used when the needed precision is not too high. If it is feasible, it is expensive for drawing only a few random numbers but it can become more efficient than other methods if a huge number of random numbers need to be drawn, as asymptotically the cost per sample will dominate.

4.5.3 Laplace distribution

Recall the Laplace distribution, denoted $Lap(b)$ and defined by

$$P_{Lap(b)}(x) = \frac{1}{2b}\exp\left(-\frac{|x|}{b}\right).$$

The cumulative distribution is

$$F_{Lap}(x) = \frac{1}{2} + \frac{\text{sign}(x)}{2} - \text{sign}(x)\exp\left(\frac{-|x|}{2}\right).$$

To sample a number r from $Lap(b)$ it is convenient to separately draw the sign s and absolute value a of r. Then, $P(s=-1) = P(s=1) = 1/2$ and $P(a) = \frac{1}{b}\exp\left(-\frac{a}{b}\right)$ and $P(a \leq t) = 1 - \exp\left(-\frac{t}{b}\right)$. In Section 4.7 we will describe protocols for both private and hidden Laplace-distributed draws.

4.5.4 Gaussian distribution

Recall the Gaussian distribution, denoted by $\mathcal{N}(\mu, \sigma^2)$ and defined by

$$P_{\mathcal{N}(\mu,\sigma^2)}(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right).$$

We will sometimes use the shorthand $P_{\mathcal{N}} = P_{\mathcal{N}(0,1)}$. The cumulative distribution is

$$F_{\mathcal{N}}(x) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{4.1}$$

where erf is the error function. There is no closed form for $P_{\mathcal{N}}$, $F_{\mathcal{N}}$ nor its inverse. In the single party setting multiple strategies have been investigated to sample from this important distribution:

- the Central Limit Theorem (CLT) approach, which consists of sampling repeatedly from a uniform distribution and computing the average, which is simple but requires $O(1/\Delta^2)$ time for a root mean squared error Δ,
- the Box-Müller method [24], that can obtain two Gaussian numbers from two uniform samples by the application of a closed form formula, but involves the computation of a square root, trigonometric functions and a logarithm,
- rejection sampling methods, such as the polar version of Box-Müller [91] or the Ziggurat Method [101] are efficient and highly accurate. While the former avoids the computation of trigonometric functions and leads to an efficient verifiable implementation, the latter uses several conditional branches which are expensive to prove in zero knowledge and requires an external method for sampling in the tails of the distribution,
- the inversion method for Gaussians that involves the approximation of the inverse error function erf^{-1}, which can be done with rational functions or Taylor polynomials (see in Section 3.6.7), and
- the recursive method of Wallace [128], which is very popular for its efficiency, but requires as input a vector of already generated Gaussian samples to generate an output vector of the same size; furthermore, samples from input and output vectors are correlated, which deteriorates the statistical quality.

Before studying some of these in the multi-party setting, we will first provide Σ-protocols of relations involving approximations of certain elementary functions.

4.6 Proofs of Elementary Functions

In this section, we construct zero knowledge proofs of statements that involve the approximation of elementary functions, i.e. sine, cosine, natural logarithm and square root. These functions can be numerically approximated using basic operations such as addition and multiplication. While classic cryptographic tools are used to prove statements over integers, we operate with real numbers which we approximate with fixed precision. Therefore, we use representations of integer multiples of $2^{-\psi}$ by multiplying our values with 2^{ψ} and rounding them deterministically to obtain elements of $\mathbb{Z}_q$. Negative numbers are represented in the upper half of $\mathbb{Z}_q$. For example, the number $a < 0$ is represented with $q + 2^{\psi}a$. The set of representable numbers is denoted by

$$\mathbb{Q}_{\langle q,\psi\rangle} = \left\{v \in \mathbb{Q} : 2^{\psi}v \in \mathbb{Z} \wedge -q/2 \leq 2^{\psi}v < q/2\right\}$$

which is closed under addition and multiplication modulo p (rounded up to $2^{-\psi}$). The encoding of $v \in \mathbb{Q}_{\langle q,\psi\rangle}$ is denoted by $\langle v\rangle = 2^{\psi}v \bmod q$.

We show circuits such that the nullity of their output is equivalent to the statements we want to prove. We will first construct circuits to describe low level statements and then use these as building blocks for higher level statements. In the end, we apply compressed

Σ-protocols (see Section 4.2) to produce zero knowledge proofs of these circuits. For parameters $(a; b)$ of all circuits defined below, a always contains public constants and b private values.

We present in Section 4.6.1 circuits for proving various types of simple statements. In Section 4.6.2, we introduce Cordic, the core approximation algorithm. We implement circuits to prove its correct execution in Section 4.6.3, and details on how to expand its domain of application, particularly for our sampling techniques, in Section 4.6.4.

4.6.1 Building Blocks

We introduce below proofs of basic statements that we will use to prove approximations, including the handling of some statements of numbers in $\mathbb{Q}_{\langle q,\psi\rangle}$. Note that additions, multiplication by an integer and range proofs port directly to $\mathbb{Z}_q$ by our encoding $\langle\cdot\rangle$. In Section 2.5.6, we show that to prove that an integer $x \in \mathbb{Z}_q$ belongs to $[0, 2^k)$ we can use the circuit

$$C_{Ra}(k; x, \bar{x}) := \begin{bmatrix} \bar{x} * (1 - \bar{x}) \\ x - \sum_{i=1}^{k} x_i 2^{i-1} \end{bmatrix}$$

where $\bar{x} = (x_1, \ldots, x_k)$ is the bit map of x. Here, $\bar{x} * (1 - \bar{x})$ is a vector with at position i the value $x_i(1 - x_i)$, which is 0 if $x_i \in \{0, 1\}$. The second expression evaluates to 0 if $\bar{x}$ is indeed the correct bit map of x. Hence, the nullity of the circuit, i.e., its righthandside evaluating to the zero vector, proves $x \in [0, 2^k)$.

Generalized range proof C_{Ra} can be used twice to prove membership in any range $[a, b] \subset \mathbb{Z}_q$. To prove $x \in [a, b]$ we use the circuit

$$C_{GRa}(a, b; x, \bar{s}_1, \bar{s}_2) := \begin{bmatrix} C_{Ra}(\lfloor \log(b - a) \rfloor + 1; x - a, \bar{s}_1) \\ C_{Ra}(\lfloor \log(b - a) \rfloor + 1; b - x + a, \bar{s}_2) \end{bmatrix}$$

where $\bar{s}_1, \bar{s}_2 \in \{0, 1\}^{\lfloor \log(b-a) \rfloor + 1}$ the auxiliary bit vectors required for C_{Ra}.

(Right) Bit-shift For $a \in \mathbb{Q}_{\langle q,\psi\rangle}$ and an integer $k > 0$, a bit shift $a >> k$ is equal to the biggest value in $\mathbb{Q}_{\langle q,\psi\rangle}$ smaller than $a/2^k$. We have that $b = a >> k$ if $\langle a\rangle - 2^k\langle b\rangle \in [0, 2^k)$. For the vector $\bar{s}$ of the bit decomposition of $\langle a\rangle - 2^k\langle b\rangle$, the circuit is

$$C_{>>}(k; \langle a\rangle, \langle b\rangle, \bar{s}) := C_{Ra}(k; \langle a\rangle - 2^k\langle b\rangle, \bar{s}).$$

Note that in the definition of $C_{>>}$, as in all subsequent circuits, the evaluation of inputs to sub-circuits such as C_{Ra} are computations performed within the circuit.

Approximate product For private $a, b, c \in \mathbb{Q}_{\langle q,\psi\rangle}$, it can be proven that c is the rounding of ab, that is by proving that $\langle c\rangle - \langle a\rangle\langle b\rangle + \langle 1/2\rangle \in [0, 2^\psi)$. For $\bar{s} \in \{0, 1\}^\psi$ the bitmap of $\langle c\rangle - \langle a\rangle\langle b\rangle + \langle 1/2\rangle$ our circuit is

$$C_{Prod}(\psi; \langle a\rangle, \langle b\rangle, \langle c\rangle, \bar{s}) := C_{Ra}(\psi; \langle c\rangle - \langle a\rangle\langle b\rangle + \langle 1/2\rangle, \bar{s}).$$

Approximate division For private $a, b, c \in \mathbb{Q}_{\langle q,\psi\rangle}$, we prove that c is approximately a/b with error $2^{-\psi}$. We also require that $b \in [A, B]$ for public $A, B \in \mathbb{Q}_{\langle q,\psi\rangle}$. We prove that $\langle b\rangle\langle c\rangle - 2^{\psi}\langle a\rangle + \langle b\rangle \in [0, 2\langle b\rangle)$. Our range proofs require that the bounds are public, so we prove that $\langle b\rangle\langle c\rangle - 2^{\psi}\langle a\rangle + \langle b\rangle \in [0, 2^{s_b+1})$ and $2^{\psi}\langle a\rangle - \langle b\rangle\langle c\rangle \in [0, 2^{s_b+1})$ where $s_b = \lfloor \log_2(\langle B\rangle - \langle A\rangle)\rfloor + 1$. For $\bar{s_1}, \bar{s_2} \in \mathbb{Z}_q^{s_b+1}$ auxiliary bit vectors, the circuit is

$$C_{Div}(\psi, s_b; \langle a\rangle, \langle b\rangle, \langle c\rangle, \bar{s}_1, \bar{s}_2) := \begin{bmatrix} C_{Ra}(s_b; \langle b\rangle\langle c\rangle - 2^{\psi}\langle a\rangle + \langle b\rangle, \bar{s_1}) \\ C_{Ra}(s_b; 2^{\psi}\langle a\rangle - \langle b\rangle\langle c\rangle, \bar{s_2}) \end{bmatrix}.$$

Exponentiation in $\mathbb{Z}_q$ Let $y, x \in \mathbb{Z}_q$ be private values with $x \in [0, 2^k)$ and $E \in \mathbb{Z}_q$ a public integer such that $E^x < p/2$. We prove that $y = E^x$. Let $\bar{x} \in \{0,1\}^k$ the vector of bits of x, we prove that $y = \prod_{y=1}^{k} y_i$ where for $i \in \{1, \ldots, k\}$, y_i is equal to $E^{2^{i-1}}$ if $\bar{x}_i = 1$, or to 1 if $\bar{x}_i = 0$. The circuit is

$$C_{IEx}(k, E; x, y, \bar{x}) := \begin{bmatrix} C_{Ra}(k; x, \bar{x}) \\ y - \prod_{i=1}^{k} 1 + \bar{x}_i(E^{2^{i-1}} - 1) \end{bmatrix}.$$

Modular sum We prove, for private $x, z \in \mathbb{Z}_q$ and public $y \in \mathbb{Z}_q$ such that all belong to $[0, M)$, that $z = x + y \bmod M$. Let $\bar{x}_1$, $\bar{x}_2$, $\bar{z}_1$ and $\bar{z}_2$ vectors of intermediate values for C_{GRa}, and let $b \in \{0,1\}$, our circuit is

$$C_{Mod}(M, y; x, z, \bar{x}_1, \bar{x}_2, \bar{z}_1, \bar{z}_2, b) := \begin{bmatrix} C_{GRa}(0, M-1; x, \bar{x}_1, \bar{x}_2) \\ C_{GRa}(0, M-1; z, \bar{z}_1, \bar{z}_2) \\ b(1-b) \\ z - (x + y - bM) \end{bmatrix}.$$

Ideas for C_{IEx} and C_{Mod} are taken from pages 112-115 of [29].

Private magnitude shift Here, we prove that $y = x >> k$ for public K and private $k \leq K$. Let $\bar{k}, \bar{k'}, \bar{k''} \in \{0,1\}^K$ and $h \in \mathbb{Z}_q$ be intermediate values for range proofs and integer exponentiations and $\bar{I}_{>>} = (h, \bar{k}, \bar{k'}, \bar{k''})$, our circuit is

$$C_{P>>}(K; x, k, y, \bar{I}_{>>}) := \begin{bmatrix} C_{IEx}(K, 2; k, h, \bar{k}) \\ C_{Ra}(K; x - hy, \bar{k'}) \\ C_{Ra}(K; h - x + hy - 1, \bar{k''}) \end{bmatrix}.$$

4.6.2 Cordic Algorithm

For approximations, we use the Cordic algorithm [129], which has long been state of the art for computations of elementary functions from simple operations [105, 110]. Essentially, it uses the same core iteration algorithm, which only uses additions and bit-shifts, for all elementary function approximations. We will use Cordic parameterized for two settings described below, the first is used for sinus and cosinus and the second for square root and logarithm. In what follows, we only provide an algorithmic description of the Cordic algorithm as is needed in order to understand our extension to the zero knowledge setting in Section 4.6.3.

Setting 1 (Sine and Cosine) Let $\theta_0 = 0$. From input values $X_0, Y_0, \theta \in \mathbb{Q}_{\langle q,\psi\rangle}$, the following iterations are performed:

$$\begin{aligned} X_i &= X_{i-1} - \xi_i(Y_{i-1} >> i) \\ Y_i &= Y_{i-1} + \xi_i(X_{i-1} >> i) \\ \theta_i &= \xi_i \tan^{-1}(1 >> i) + \theta_{i-1} \end{aligned}$$

where $\xi_i \in \{-1, 1\}$ is equal to $\text{sign}(\theta - \theta_{i-1})$ and $\tan^{-1}(1 >> i)$ is taken from a precomputed table. Let ν be the total number of iterations, and let constants $K_{1,\nu} = \prod_{j=0}^{\nu} \sqrt{1 + (1 >> 2j)}$ and $K_1 = \lim_{\nu\to\infty} K_{1,\nu} \approx 1.6$. We have that

$$\lim_{\nu\to\infty} \begin{bmatrix} X_\nu \\ Y_\nu \\ \theta_\nu - \theta \end{bmatrix} = K_1 \begin{bmatrix} X_0 \cos\theta - Y_0 \sin\theta \\ X_0 \sin\theta + Y_0 \cos\theta \\ 0 \end{bmatrix}.$$

Recall the representation parameter ψ of $\mathbb{Q}_{\langle q,\psi\rangle}$ defined at the beginning of the section. By the convergence rate of Cordic, if $\psi \geq \nu + \lceil \log_2(\nu) \rceil + 1$, with input $X_0 = 1/K_{1,\nu}$, $Y_0 = 0$, and $\theta \in [-\pi/2, \pi/2]$, then X_ν and Y_ν are approximations of $\sin(\theta)$ and $\cos(\theta)$ respectively with error at most $2^{1-\nu}$.

Setting 2 ($\ln(x)$ and $\sqrt{x}$) In this setting Cordic only takes two inputs $X_0, Y_0 \in \mathbb{Q}_{\langle q,\psi\rangle}$ and, with $\theta_0 = 0$, it performs the iterations

$$\begin{aligned} X_i &= X_{i-1} + \xi_i(Y_{i-1} >> F_i) \\ Y_i &= Y_{i-1} + \xi_i(X_{i-1} >> F_i) \\ \theta_i &= \xi_i \tanh^{-1}(1 >> F_i) + \theta_{i-1} \end{aligned}$$

with $\xi_i = \text{sign}(-Y_{i-1})$ and shift magnitude $F_i = i + 1 - k$ where the small value k is equal to the biggest integer such that $3^{k+1} + 2k - 1 \leq 2(i + 1)$. Now let $K_{2,\nu} = \prod_{j=1}^{\nu} \sqrt{1 + (1 >> 2F_{j-1})}$ and $K_2 = \lim_{\nu\to\infty} K_{2,\nu} \approx 0.8$. In Setting 2, we have that

$$\lim_{\nu\to\infty} \begin{bmatrix} X_\nu \\ Y_\nu \\ \theta_\nu \end{bmatrix} = \begin{bmatrix} K_2\sqrt{X_0^2 - Y_0^2} \\ 0 \\ \tanh^{-1}\left(\frac{Y_0}{X_0}\right) \end{bmatrix}.$$

With ψ as in Setting 1, $x \in [\frac{1}{4}, 1)$ and fixing $X_0 = x + 1$, $Y_0 = x - 1$ we get by the identity $\ln(x) = 2\tanh^{-1}(\frac{x-1}{x+1})$ that θ_ν is an approximation of $\frac{1}{2}\ln(x)$ with error at most 2^{1-F_ν}. Similarly, for the same ψ and domain of x, $\sqrt{x}$ can be obtained by setting $(X_0, Y_0) = \left(x + \frac{1}{K_{2,\nu+1}^2}, x - \frac{1}{K_{2,\nu+1}^2}\right)$ with error at most 2^{1-F_ν}.

4.6.3 Cordic in Zero Knowledge

We first specify a set of statements that together are equivalent to a correctly performed Cordic computation. Note that the iterations in Settings 1 and 2 are very similar. Except

for the correctness of the ξ_i values, they can be described by equations

$$\xi_i = -1 \vee \xi_i = 1 \quad \forall i \in \{1, \ldots, \nu\}, \tag{4.2}$$

$$Y_i = Y_{i-1} + \xi_i (X_{i-1} >> F_i) \quad \forall i \in \{1, \ldots, \nu\}, \tag{4.3}$$

$$X_i = X_{i-1} - m\xi_i (Y_{i-1} >> F_i) \quad \forall i \in \{1, \ldots, \nu\}, \tag{4.4}$$

$$\theta_\nu = \sum_{i=1}^{\nu} \xi_i \alpha_i, \tag{4.5}$$

where the constants in Setting 1 are $m = 1$, $F_i = i$ and $\alpha_i = \tan^{-1}(1 >> i)$ and in Setting 2, F_i is already defined, $m = -1$ and $\alpha_i = \tanh^{-1}(1 >> F_i)$. To prove the correct value of the ξ_i's, we avoid wide range checks at each iteration (on $\theta - \theta_i$ or Y_{i-1}), but instead we use properties of the convergence of Cordic: all of $\xi_1, \ldots, \xi_\nu \in \{-1, 1\}$ have been chosen correctly if

$$\theta_\nu - \theta \in [-\alpha_\nu, \alpha_\nu] \tag{4.6}$$

in Setting 1, and

$$Y_\nu \in [-2^{-F_{\nu-1}}, 2^{-F_{\nu-1}}] \tag{4.7}$$

in Setting 2.

We outline below the circuits that imply the above statements. Let $S \in \{1, 2\}$ be the Cordic setting that defines the involved constants. Let $\bar{\xi} = (\xi_1, \ldots, \xi_\nu)$ and let $\bar{I} = (\langle X_i \rangle, \langle Y_i \rangle)_{i=1}^{\nu-1}, (\bar{s}_i, \bar{s}'_i, \langle X'_i \rangle, \langle Y'_i \rangle)_{i=0}^{\nu-1}, \bar{\xi})$ be the vector of all intermediate values. The nullity of circuit

$$C_{Crd}(\nu, S; \langle X_0 \rangle, \langle Y_0 \rangle, \langle X_\nu \rangle, \langle Y_\nu \rangle, \langle \theta_\nu \rangle, \bar{I})$$
$$:= \begin{bmatrix} C_{>>}(F_i; \langle X_{i-1} \rangle, \langle X'_{i-1} \rangle, \bar{s}_{i-1}) & \forall i \in [\nu] \\ C_{>>}(F_i; \langle Y_{i-1} \rangle, \langle Y'_{i-1} \rangle, \bar{s}'_{i-1}) & \forall i \in [\nu] \\ (1 + \bar{\xi}) * (1 - \bar{\xi}) & \\ \langle Y_i \rangle - \langle Y_{i-1} \rangle - \xi_i \langle X'_{i-1} \rangle & \forall i \in [\nu] \\ \langle X_i \rangle - \langle X_{i-1} \rangle + m\xi_i \langle Y'_{i-1} \rangle & \forall i \in [\nu] \\ \langle \theta_\nu \rangle - \sum_{i=1}^{\nu} \xi_i \langle \alpha_i \rangle & \end{bmatrix}$$

is a proof of the core of the execution in eqs. (4.2) to (4.5). Here, for $i \in \{1, \ldots, \nu\}$, $\bar{s}_i, \bar{s}'_i \in \{0, 1\}^{F_i}$ are auxiliary bit vectors to prove bit shifts of X_i and Y_i with result X'_{i-1} and Y'_{i-1} respectively.

We complete the above core circuit for Setting 1. Let $\gamma = \lfloor \log_2(2\langle \alpha_\nu \rangle) \rfloor + 1$, and let $\bar{s}_\alpha \in \{0, 1\}^\gamma$ be the bit decomposition of $\langle \theta \rangle - \langle \theta_\nu \rangle$. The circuit

$$C_{Crd1}(\nu; \langle \theta \rangle, \langle X_0 \rangle, \langle Y_0 \rangle, \langle X_\nu \rangle, \langle Y_\nu \rangle, \langle \theta_\nu \rangle, \bar{I}, \bar{s}_\alpha)$$
$$:= \begin{bmatrix} C_{Crd}(\nu, 1; \langle X_0 \rangle, \langle Y_0 \rangle, \langle X_\nu \rangle, \langle Y_\nu \rangle, \langle \theta_\nu \rangle, \bar{I}) \\ C_{Ra}(\gamma; \langle \theta \rangle - \langle \theta_\nu \rangle + \langle \alpha_\nu \rangle, \bar{s}_\alpha) \end{bmatrix}.$$

proves eqs. (4.2) to (4.6). Similarly, we extend the core circuit to a complete one for setting 2: for $\bar{s}_Y$ equal to the bit decomposition of Y_ν,

$$C_{Crd2}(\psi, \nu; \langle X_0 \rangle, \langle Y_0 \rangle, \langle X_\nu \rangle, \langle Y_\nu \rangle, \langle \theta_\nu \rangle, \bar{I}, \bar{s}_Y)$$
$$:= \begin{bmatrix} C_{Crd}(\nu, 2; \langle X_0 \rangle, \langle Y_0 \rangle, \langle X_\nu \rangle, \langle Y_\nu \rangle, \langle \theta_\nu \rangle, \bar{I}) \\ C_{Ra}(\psi - F_\nu; \langle Y_\nu \rangle, \bar{s}_Y) \end{bmatrix}.$$

proves eqs. (4.2) to (4.5) and (4.7).

The instantiation of C_{Crd1} and C_{Crd2} for elementary functions is straightforward. Inputs that are intermediate values are defined as above. For trigonometric functions, we set $\bar{I}_T = (\langle\theta_\nu\rangle, \bar{I}, \bar{s}_\alpha)$ and use

$$\begin{aligned} &C_{Tr}(\nu; \langle\theta\rangle, \langle c\rangle, \langle s\rangle, \bar{I}_T) \\ &\quad := C_{Crd1}(n; \langle\theta\rangle, \langle 1/K_{1,\nu}\rangle, 0, \langle s\rangle, \langle c\rangle, \bar{I}_T) \end{aligned}$$

to compute $s = \sin(\theta)$ and $c = \cos(\theta)$. For the logarithm, let $\bar{I}_L = (\langle X_\nu\rangle, \langle Y_\nu\rangle, \langle\theta_\nu\rangle, \bar{I}, \bar{s}_Y)$. Then,

$$\begin{aligned} &C_{Log}(\psi, \nu; \langle x\rangle, \langle l\rangle, \bar{I}_L) \\ &\quad := \begin{bmatrix} C_{Crd2}(\psi, \nu; \langle x\rangle + \langle 1\rangle, \langle x\rangle - \langle 1\rangle, \bar{I}_L) \\ \langle l\rangle - 2\langle\theta_\nu\rangle \end{bmatrix} \end{aligned}$$

proves $l = \ln(x)$. For the square root let $\bar{I}_S = (\langle Y_\nu\rangle, \langle\theta_\nu\rangle, \bar{I}, \bar{s}_Y)$, then

$$\begin{aligned} &C_{Sqrt}(\psi, \nu; \langle x\rangle, \langle s\rangle, \bar{I}_S) = \\ &C_{Crd2}(\psi, \nu; \langle x\rangle + \langle 1/4K_{2,\nu+1}^2\rangle, \langle x\rangle - \langle 1/4K_{2,\nu+1}^2\rangle, \langle s\rangle, \bar{I}_S) \end{aligned}$$

proves $s = \sqrt{x}$.

Finally, we point out that, with minor adjustments to the above approximations, proofs of hyperbolic trigonometric functions and e^x can be obtained.

4.6.4 Extending the Domain

Here we extend the domain of approximations, which is necessary for our sampling applications. We sometimes do not define inputs such as bit vectors for range proofs and other intermediate values that are clear from the context or that are already defined in previous circuits.

Sine and cosine As shown, sine and cosine can be approximated in $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. For $Q \in \{1, 2, 3, 4\}$ we use the identity

$$\sin\left(Q\frac{\pi}{2} + \theta'\right) = \begin{cases} \cos(\theta') & \text{if } Q = 1 \\ -\sin(\theta') & \text{if } Q = 2 \\ -\cos(\theta') & \text{if } Q = 3 \\ \sin(\theta') & \text{if } Q = 4 \end{cases}$$

extend the domain to $[0, 2\pi]$. Let $\bar{s}_\pi, \bar{s}'_\pi$ be bit vectors as needed for C_{GRa}, and

$$\bar{I}_{Tg} = (\langle\theta'\rangle, \langle s'\rangle, \langle c'\rangle, Q, \bar{s}_\pi, \bar{s}'_\pi, \bar{I}_T).$$

Let

$$([j_1], [j_2], [j_3], [j_4]) = (\langle c'\rangle, -\langle s'\rangle, -\langle c'\rangle, \langle s'\rangle)$$

and

$$([k_1], [k_2], [k_3], [k_4]) = ([j_2], [j_3], [j_4], [j_1])$$

for $i \in \{1, 2, 3, 4\}$ be literal variable replacements. Circuit

$$C_{TrG}(\nu; \langle\theta\rangle, \langle s\rangle, \langle c\rangle, \bar{I}_{Tg}) := \begin{bmatrix} C_{Tr}(\nu; \langle\theta'\rangle, \langle s'\rangle, \langle c'\rangle, \bar{I}_T) \\ C_{GRa}(\langle -\frac{\pi}{2}\rangle, \langle\frac{\pi}{2}\rangle; \langle\theta'\rangle, \bar{s}_\pi, \bar{s}'_\pi) \\ \langle\theta\rangle - Q\langle\frac{\pi}{2}\rangle - \langle\theta'\rangle \\ \prod_{i=1}^{4}(Q-i)^2 + ([j_i] - \langle s\rangle)^2 + ([k_i] - \langle c\rangle)^2 \end{bmatrix}$$

proves $s = sin(\theta)$ and $c = \cos(\theta)$ in the extended domain.

Natural logarithm We extend the domain of $\ln(x)$ to $(0, 1)$. For $x' \in [\frac{1}{2}, 1)$ and non-negative integer e such that $x = 2^{-e}x' \in (0, 1)$. We prove that $l = \ln(x') - e\ln(2) = \ln(x)$. Let $\bar{I}_{Lg} = (e, h, \bar{e}, \bar{s}_{x'}, \langle x'\rangle, \langle l'\rangle, \bar{I}_L)$, then

$$C_{LogG}(\psi, \nu; \langle x\rangle, \langle l\rangle, \bar{I}_{Lg}) := \begin{bmatrix} C_{Log}(\psi, \nu; \langle x'\rangle, \langle l'\rangle, \bar{I}_L) \\ C_{Ra}(\psi - 1; \langle x'\rangle - \langle 0.5\rangle, \bar{s}_{x'}) \\ C_{IEx}(\psi - 1, 2; e, h, \bar{e}) \\ h\langle x\rangle - \langle x'\rangle \\ \langle l\rangle - \langle l'\rangle + e\langle\ln(2)\rangle \end{bmatrix}$$

proves our approximation.

Square root Now, for a public bound $B > 0$ and a private $x \in [0, B]$, we prove that $s = \sqrt{x}$. Let $\gamma = \lfloor\log_2(B)\rfloor + 1$. We choose $x' \in [\frac{1}{2}, 1)$ and an integer $e \in [-\psi, \gamma]$ such that $x = 2^e x'$, and we have that

$$\sqrt{x} = \begin{cases} 2^{e/2}\sqrt{x'} & \text{if } e \text{ is even} \\ 2^{(e+1)/2}\sqrt{x'/2} & \text{if } e \text{ is odd.} \end{cases}$$

We break the proof in several circuits to handle different cases. For that we use bit variables as flags to decide which computation will be proven. Let $n_e \in \{0, 1\}$ be the "negativity flag" of e and $e' \geq 0$ such that $e = (1 - n_e)e'$. Let $i_e \in \{0, 1\}$ be the "parity flag" of e, such that $e = 2f - i_e$ for an integer f. We also define $f' \geq 0$ such that $f = (1 - n_e)f'$. We first handle the relations between x, x', $s = \sqrt{x}$ and $s' = \sqrt{x'/(1 + i_e)}$ when e is non-negative, or equivalently, when $n_e = 0$. Let $\bar{I}_{D1} = (l, \bar{f}, \bar{I}_{>>})$, then our circuit is

$$C_{SDom1}(B; \langle x\rangle, \langle s\rangle, \langle x'\rangle, \langle s'\rangle, i_e, e', f', \bar{I}_{D1}) := \begin{bmatrix} C_{P>>}(\gamma; \langle x\rangle, e' + i_e, \langle x'\rangle, \bar{I}_{>>}) \\ e' - 2f' + i_e \\ C_{IEx}(\gamma, 2; f', l, \bar{f}) \\ \langle s\rangle - \langle s'\rangle l \end{bmatrix}.$$

Similarly, for $\bar{I}_{D2} = (h, \bar{e}, \bar{I}'_{>>})$ the case when e is negative is described by

$$C_{SDom2}(\psi; \langle x\rangle, \langle s\rangle, \langle x'\rangle, \langle s'\rangle, i_e, e', f', \bar{I}_{D2}) := \begin{bmatrix} C_{IEx}(\psi - 1, 2; e' - i_e, h, \bar{e}) \\ \langle x'\rangle - h\langle x\rangle \\ e' - 2f' - i_e \\ C_{P>>}(\psi - 1; \langle s\rangle, f', \langle s'\rangle, \bar{I}'_{>>}) \end{bmatrix}.$$

Now we describe the main circuit. For

$$\bar{I}_D = (\langle x' \rangle, \langle s' \rangle, i_e, e', f')$$

and

$$\bar{I}_{Sg} = (n_e, \bar{s}_{x'}, \bar{I}_S, \bar{I}_D, \bar{I}_{D1}, \bar{I}_{D2})$$

vectors of intermediate values, we prove $s = \sqrt{x}$ with

$$C_{SqrtG}(\psi, \nu, B; \langle x \rangle, \langle s \rangle, \bar{I}_{Sg}) \\ := \begin{bmatrix} C_{Sqrt}(\psi, \nu; \langle x' \rangle, \langle s' \rangle, \bar{I}_S) \\ i_e(1 - i_e) \\ n_e(1 - n_e) \\ i_e * C_{Ra}(\psi - 2; \langle x' \rangle - \langle 0.25 \rangle, \bar{s}_{x'}) \\ (1 - i_e) * C_{Ra}(\psi - 1; \langle x' \rangle - \langle 0.5 \rangle, \bar{s}_{x'}) \\ (1 - n_e) * C_{SDom1}(B; \langle x \rangle, \langle s \rangle, \bar{I}_D, \bar{I}_{D1}) \\ n_e * C_{SDom2}(\psi, \langle x \rangle, \langle s \rangle, \bar{I}_D, \bar{I}_{D2}) \end{bmatrix}.$$

While the circuit above is easier to read, the practical implementation contains a number of further optimizations to reduce the number of multiplications. In particular, additional variables are introduced to avoid multiplying flags such as i_e with larger vectors such as the output of a C_{Ra} circuit. This introduces additional variables, e.g., $i_e * C_{Ra}(\psi - 2; \langle x' \rangle - \langle 0.25 \rangle, \bar{s}_{x'})$ would become $i_e(x' - x'_{aux})$ and $C_{Ra}(\psi - 2; \langle x'_{aux} \rangle - \langle 0.25 \rangle, \bar{s}_{x'_{aux}})$.

4.7 The Laplace distribution

4.7.1 Private Laplace sampling

Protocol 6 (private drawing from Laplace)**.** *First, party* 1 *privately draws* s_0 *and* a' *uniformly at random in* $[0, L)$ *with* L *sufficiently large (Protocol 2). Then, party* 1 *computes* $a = -b \log(1 - a'/L)$, $s = 2(s_0 \bmod 2) - 1$ *and* $r = sa$*, and provides a ZKP for these relations (in Section 4.6.3 we showed a ZKP for the logarithm function, in Section 4.6.1 for approximate division).*

As Protocol 2 verifiably draws random numbers uniformly and for the other computations in Protocol 6 a ZKP is provided, Protocol 6 verifiably draws random numbers from the $Lap(b)$ distribution.

4.7.2 Hidden Laplace sampling

We can also make hidden draws from the Laplace distribution, i.e., drawing a Laplace-distributed random number r as a secret share $[\![r]\!]$. For this, we build on the basic operations discussed in Section 4.2.

First, we observe that one can sample a sign s uniformly from $\{-1, 1\}$ as follows: the parties apply Protocol 3 to draw a secret shared random number uniformly distributed in $\mathbb{Z}_p$, obtaining the sharing $[\![t]\!]$, next they multiply the sharing with itself to obtain $[\![t^2]\!]$ and open $[\![t^2]\!]$ to reveal t^2, and finally they multiply $[\![t]\!]$ with the public constant $1/\sqrt{t^2}$ to obtain $[\![s]\!] = \left[\!\left[t/\sqrt{t^2}\right]\!\right] \in \{-1, 1\}$. Drawing a secret shared random bit $b \in \{0, 1\}$ is then just drawing a sign $[\![s]\!]$ and computing $[\![b]\!] = ([\![s]\!] + 1)/2$ (this is protocol RAN_2 in [50]).

The Cordic algorithm for logarithm computation described in Section 4.6.2 requires only additions, bit shifts and comparisons (when setting $\xi_i = \text{sign}\,(-Y_{i-1})$). While it does not directly use multiplications, implementations of bit operations and comparisons, e.g., as in [50], often are using multiplications so the use of multiplications cannot be fully avoided. Alternative strategies to compute the logarithm suffer from similar challenges.

As in Section 4.7.1, we want to draw a number a' uniformly from $[0, L)$, compute $a = -b\log(1 - a'/L)$ and multiply it with a random sign s to get the random number as distributed according to $Lap(b)$. To compute $\log(x)$, Cordic expects $x \in [1/4, 1)$, so before applying Cordic we may need to scale its input to fit this interval.

We set $L = 2^l$ for some sufficiently large integer l and generate a' as an l-bit number, i.e., $[\![a']\!] = \sum_{i=0}^{l-1} [\![a^{(i)}]\!] 2^i$ where $a^{(i)}$ are random bits.

We can find the highest zero bit of a' as follows: set $h_l = 0$, $h'_l = 1$ and $a^{(-1)} = 0$, and for $i = l - 1 \ldots - 1$, set $[\![h'_i]\!] = [\![h'_{i+1}]\!][\![1 - h_{i+1}]\!]$ and $[\![h_i]\!] = [\![h'_i]\!][\![1 - a^{(i)}]\!]$. The meaning of h_i then is 'bit i is the highest 0-bit', and the meaning of h'_i is 'the bits higher than bit i are all ones'. Exactly one h_i equals 1 and all others are 0 among $i = -1 \ldots l - 1$. We then can write $\log(1 - a'/L) = a^{\text{h}} + \log(x)$ with $[\![a^{\text{h}}]\!] = \sum_{i=0}^{l-1} [\![h_i]\!] \log(2^{i+1-l})$ and $BITS(x, (x^{(i)})_{i=0}^{l})$ where

$$([\![x^{(i)}]\!])_{i=0}^{l} = \text{BIT-ADD}(2^l, ([\![x_-^{(i)}]\!])_{i=0}^{l-1})$$

and

$$[\![x_-^{(i)}]\!] = \sum_{j=0}^{l-1} [\![h_j]\!][\![a^{(i+j+1-l)}]\!]$$

with $a^{(i)} = 0$ for $i < 0$.

Now we can apply Cordic on x. Cordic needs additions (using the BIT-ADD protocol), bit shifts (moving bits to the right and duplicating the highest bit), the $\text{sign}(\cdot)$ function (check the highest bit) and negation (invert all bits and add 1).

Protocol 7 (hidden drawing from Laplace)**.** *One can verifiably draw a hidden Laplace-distributed random number by following the steps explained above, and by providing ZKP for all computations. The ZKP are similar to those for private sampling, where parts of secret shares are transfered between parties, the parties can agree on the commitment which will represent the shared number.*

4.8 The Gaussian Distribution

In this section we elaborate several strategies to sample from the Gaussian distribution. In particular, we are interested in protocols such that upon termination one party has a private number $y \in \mathbb{Q}_{\langle q,\psi\rangle}$ and has provided a zero knowledge argument that $y \sim \mathcal{N}(\mu, \sigma^2)$ for some public μ and σ.

All methods require as a subprotocol sampling uniformly distributed numbers. Therefore all our protocols for private drawing numbers from the Gaussian distribution follow the same high-level structure:

1. use Protocol 2 to verifiably draw uniformly distributed number(s),

2. transform the uniformly distributed number(s) into Gaussian distributed number(s), and

3. use an arithmetic circuit matching this transformation together with compressed Σ-protocols (see Section 4.2) to prove the transformation.

We implement ZKPs of the correct execution of Gaussian draws. The sampling methods we implement are (1) the central limit theorem approach (averaging over uniform samples), (2) the Box Müller method [24], (3) the Polar Method [100], (4) the inversion method using a series expansion for erf^{-1} [39], and (5) the inversion method using a fractional polynomial to approximate erf^{-1} [69].

4.8.1 The central limit theorem method

The uniform distribution over the interval $[0, L)$ has variance $L^2/12$. Let a party privately draw N random numbers $\{x_i\}_{i=1}^N$ uniformly distributed over $[0, L)$, and compute $x = \mu + \frac{\sigma\sqrt{12}}{L\sqrt{N}} \sum_{i=1}^N \left(x_i - \frac{L}{2}\right)$. Then, x is $\mathcal{N}(\mu, \sigma^2)$ distributed. For a ZKP for this relation between x and the x_i we only need the homomorphic property of Pedersen commitments (for the additions and the multiplication with a constant) and a range proof (for the rounding).

4.8.2 The Box Müller method

The Box Müller method [24] consists of drawing two uniform samples U_1 and U_2 in the interval $(0, 1)$ and to compute

$$\begin{aligned} \rho &= \sqrt{-2\ln(U_1)}, \\ X_1 &= \rho\cos(2\pi U_2), \\ X_2 &= \rho\sin(2\pi U_2). \end{aligned}$$

Then, X_1 and X_2 are distributed according to $\mathcal{N}(0, 1)$. Now we use circuits of elementary functions defined in Section 4.6 to construct our proof. Recall parameters ψ and the number of Cordic iterations ν defined therein. Let

$$\begin{aligned} \bar{I}_{BM} = (&\bar{s}_1, \bar{s}_2, \bar{s}_3, \langle s\rangle, \langle c\rangle, \bar{I}_{Tg}, \langle a_\pi\rangle, \langle X_1'\rangle, \langle X_2'\rangle \\ &\bar{I}_{Sg}, \langle\rho\rangle, \bar{I}_{Lg}, \langle l\rangle, \bar{a}_1', \bar{a}_2', \bar{U}_1', \bar{U}_2', \bar{a}_1'', \bar{a}_2'', \bar{U}_1'', \bar{U}_2'') \end{aligned}$$

be a vector containing all intermediate values of the computation. Then the approximation circuit is

$$\begin{aligned} &C_{BM}(\psi, \nu, z_1, z_2; \langle U_1\rangle, \langle U_2\rangle, \langle X_1\rangle, \langle X_2\rangle, \bar{I}_{BM}) \\ &:= \begin{bmatrix} C_{Mod}(2^\psi - 1, z_1; a_1, \langle U_1\rangle - 1, \bar{a}_1', \bar{a}_2', \bar{U}_1', \bar{U}_2') \\ C_{Mod}(2^\psi - 1, z_2; a_2, \langle U_2\rangle - 1, \bar{a}_1'', \bar{a}_2'', \bar{U}_1'', \bar{U}_2'') \\ C_{LogG}(\psi, \nu; \langle U_1\rangle, \langle l\rangle, \bar{I}_{Lg}) \\ C_{SqrtG}(\psi, \nu, 2\psi\ln(2); -2\langle l\rangle, \langle\rho\rangle, \bar{I}_{Sg}) \\ C_{Prod}(\psi; 2\langle\pi\rangle, \langle U_2\rangle, \langle a_\pi\rangle, \bar{s}_1) \\ C_{TrG}(\nu; \langle a_\pi\rangle, \langle s\rangle, \langle c\rangle, \bar{I}_{Tg}) \\ C_{Prod}(\psi; \langle\rho\rangle, \langle c\rangle, \langle X_1'\rangle, \bar{s}_2) \\ C_{Prod}(\psi; \langle\rho\rangle, \langle s\rangle, \langle X_2'\rangle, \bar{s}_3) \end{bmatrix}. \end{aligned}$$

Private values a_1 and a_2 and public challenges z_1 and z_2 are used in the modular proofs of circuit C_{Mod} to generate U_1 and U_2 with Protocol 2. To obtain a sample with standard deviation different than 1, the resulting samples can be scaled with an extra C_{Prod} circuit. Different mean requires an extra addition gate.

4.8.3 The Polar Box-Müller method

The polar method [91] is an optimization of Box-Müller that avoids the computation of sine and cosine by the use of rejection sampling. It samples two uniform values V_1 and V_2 in the $(-1, 1)$ interval, and keeps the result only if $0 < V_1^2 + V_2^2 \leq 1$. Otherwise V_1 and V_2 are re-sampled. For non rejected V_1 and V_2 it computes

$$\begin{aligned} \alpha &= V_1^2 + V_2^2, \\ Y_1 &= V_1\sqrt{-2\ln(\alpha)/\alpha}, \\ Y_2 &= V_2\sqrt{-2\ln(\alpha)/\alpha}. \end{aligned}$$

Y_1 and Y_2 have distribution $\mathcal{N}(0, 1)$.

In the private sampling, if V_1 and V_2 are rejected, the prover can just reveal them and start new uniform draws until acceptance, when it proves the correctness of accepting pairs. As in Box Müller, parameters ν and ψ define our elementary function approximations, and a_1, a_2, z_1, z_2 are used to generate V_1 and V_2 in Protocol 2. Let

$$\begin{aligned} \bar{I}_{Pol} = (&\langle s\rangle, \bar{I}_{Sg}, \langle d\rangle, \langle\alpha\rangle, \langle l\rangle, \bar{I}_{Lg}, \bar{s}_6, \bar{s}_5, \bar{s}_4, \bar{s}_3, \bar{s}_2, \bar{s}_1, \langle\alpha\rangle, \\ &\langle V_2'\rangle, \langle V_1'\rangle, \bar{v}', \bar{v}, \bar{a}_1', \bar{a}_2', \bar{V}_1', \bar{V}_2', \bar{a}_1'', \bar{a}_2'', \bar{V}_1'', \bar{V}_2'') \end{aligned}$$

be a vector of intermediate computation values, then the implemented circuit is

$$\begin{aligned} &C_{Pol}(\psi, \nu, z_1, z_2; \langle V_1\rangle, \langle V_2\rangle, \langle Y_1\rangle, \langle Y_2\rangle, \bar{I}_{Pol}) \\ &:= \begin{bmatrix} C_{Mod}(2^{2\psi} - 2, z_1; a_1, \langle V_1\rangle + \langle 1\rangle - 1, \\ \bar{a}_1', \bar{a}_2', \bar{V}_1', \bar{V}_2') \\ C_{Mod}(2^{2\psi} - 2, z_2; a_2, \langle V_2\rangle + \langle 1\rangle - 1, \\ \bar{a}_1'', \bar{a}_2'', \bar{V}_1'', \bar{V}_2'') \\ C_{Prod}(\psi; \langle V_1\rangle, \langle V_1\rangle, \langle V_1'\rangle, \bar{v}) \\ C_{Prod}(\psi; \langle V_2\rangle, \langle V_2\rangle, \langle V_2'\rangle, \bar{v}') \\ \langle\alpha\rangle - \langle V_1'\rangle - \langle V_2'\rangle \\ C_{GRa}(1, \langle 1\rangle - 1; \langle\alpha\rangle, \bar{s}_1, \bar{s}_2) \\ C_{LogG}(\psi, \nu; \langle\alpha\rangle, \langle l\rangle, \bar{I}_{Lg}) \\ C_{Div}(\psi; -2\langle l\rangle, \langle\alpha\rangle, \langle d\rangle, \bar{s}_3, \bar{s}_4) \\ C_{SqrtG}(\psi, \nu, 2^{\psi+1}\psi\ln(2); \langle d\rangle, \langle s\rangle, \bar{I}_{Sg}) \\ C_{Prod}(\psi; \langle s\rangle, \langle V_1\rangle, \langle Y_1\rangle, \bar{s}_5) \\ C_{Prod}(\psi; \langle s\rangle, \langle V_2\rangle, \langle Y_2\rangle, \bar{s}_6) \end{bmatrix}. \end{aligned}$$

To avoid multiple interactions due to rejection, Protocol 2 is set to draw a sufficiently large number of uniform samples such that at least one pair is not rejected with high probability. The cost of the extra samples is negligible. To obtain a distribution with different mean and variance, we scale V_1 and V_2 with addition and product.

4.8.4 Inversion method

Inverting eq. (4.1) we get

$$F_{\mathcal{N}}^{-1}(x) = \sqrt{2}\text{erf}^{-1}(2x - 1).$$

There are many numerical strategies to approximate either erf or erf^{-1}, of which we implemented two.

A first strategy due to [39] is described as part of the GOPA protocol in Section 3.6.7. While it is described for classical Σ-protocols, its implementation with compressed protocols is straightforward using circuit building blocks of Section 4.6.1. A second strategy is proposed in [69] and uses a rational approximation. Therein, erf^{-1} is computed by

$$\mathsf{erfinv}_{\mathrm{SP}}(x) = \begin{cases} xp_1(w) & \text{if } w \leq 5 \text{ (central region)} \\ xp_2(s) & \text{if } w > 5 \text{ (tail region)} \end{cases}$$

where $w = -\log(1-x^2)$, $s = \sqrt{w}$ and p_1 and p_2 are two polynomials of degree 8. We use Cordic, product and range ZKPs to prove its computation.

4.8.5 Hidden drawing

For the several strategies for sampling the Gaussian distribution described above, one can construct a protocol based on secret sharing for hidden sampling. Similar considerations apply as for the discussion in Section 4.7.2. As an example, we show a protocol using the Central Limit Theorem approach.

Protocol 8. *Let $N/12$ be a power of 4. For $i = 0 \dots l-1$ and $j = 1 \dots N$, draw a random bit sharing $[\![x_j^{(i)}]\!]$. This yields N random numbers $\{x_j\}_{j=1}^N$ in the interval $[0, 2^l)$. For $j \in [N]$ and $i = l \dots l + \log_2(N)$, set $x_j^{(i)} = 0$. Let $([\![y_1^{(i)}]\!])_{i=0}^{l+\log_2(N)} = ([\![x_1]\!])_{i=0}^{l+\log_2(N)}$ and for $j = 2 \dots N$ let $([\![y_j^{(i)}]\!])_{i=0}^{l+\log_2(N)} = \text{BIT-ADD}(([\![y_{j-1}^{(i)}]\!])_{i=0}^{l+\log_2(N)}, ([\![x_j^{(i)}]\!])_{i=0}^{l+\log_2(N)})$. Finally let $[\![r]\!] = \sum_{i=\log_2(N/12)/2}^{l+\log_2(N)} y_N^{(i)} 2^{i-\log_2(N/12)/2}$. Then, r approximates $\mathcal{N}(2^l\sqrt{3N}, 2^l)$. The computations can be made verifiable using a ZKP where parties send a number to each other can agree on using the same commitment.*

4.9 Security of our protocols

In this section, we prove the security of the protocols presented in the main paper. In the sequel, we will often restate in more detail and more formally the protocols. We consider a set of n parties $\mathbf{P} = \{P_1 \dots P_n\}$. We will denote by P_{-i} the set of all parties except i, i.e., $P_{-i} = \mathbf{P} \setminus \{P_i\}$. We assume that a subset of parties $\mathbf{P}_{cor} \subset \mathbf{P}$ is corrupted and controlled by an adversary $\mathcal{A}$. The set $\mathbf{P}_{cor}$ of corrupted parties is static, i.e. does not change after the beginning of the execution.

For the description of our protocols, we will denote the fact that a party A sends a message M to a party B by "$A \to B$: M". Recall that we use a bulletin board for communication. Among others, this means that when a protocol contains a broadcast instruction a single message is sent from one party to all others, in practice by sending it from that party to the bulletin board, which forwards it to all other agents. We will also use hiding vector Pedersen commitments defined in Section 2.4. Recall the finite groups $\mathbb{Z}_q$ and $\mathbb{G}$ defined therein. For any integer $k > 0$, we will denote by $Com(\bar{x}, r) \in \mathbb{G}$ the commitment of the k-dimension vector $\bar{x} \in \mathbb{Z}_q^k$ with randomness $r \in \mathbb{Z}_q$.

We describe our security framework in Section 4.9.1. In Section 4.9.2, we describe our model of compressed Σ-protocols in our security analysis. In sections 4.9.3 and 4.9.4 we prove the security of Protocols 1 and 2 respectively. We conclude by discussing the security of our protocols for public and private draws from some other distributions in Section 4.9.5.

4.9.1 Security Definitions

We prove security in the simulation paradigm, using the *covert security model* in its *strong explicit cheat* formulation [6, Def. 3.3]. We assume parties are able to detect malicious actions and can in such cases abort the protocol. Deterrence measures may be in place to discourage parties from being detected as malicious. In fact, unless parties stop participating our protocols either complete successfully or abort while detecting that a specific party is a cheater. We note that covert security is in most cases weaker than the notion of *malicious security with abort* [71, Ch 7]. In our protocols, cheaters are detected with overwhelming probability, which provides the same guarantees as the model of malicious security with abort [6, Prop. 3.10]. We have nonetheless decided to use the definition of the covert security model as it explicitly formalizes the ability of honest parties to determine who has cheated.

We start by introducing the key concepts of multiparty computation under the covert security model. A multiparty computation between our n parties in $\mathbf{P}$ is a protocol that computes a stochastic process $\mathcal{F} : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, which is also called an *ideal functionality*, where for all $i \in \{1, \ldots, n\}$, the ith component of the input and output of $\mathcal{F}$ maps respectively to the private input and output of party P_i.

In the simulation paradigm, a multiparty protocol securely computes an ideal functionality in the presence of malicious adversaries if any possible malicious behavior in the protocol caused by colluding malicious parties is not more harmful than what such party can cause in the *ideal model* as defined below.

While in the covert ideal model, parties that try to cheat are caught with probability equal to ζ, called the *deterrence factor*. We use the Strong Explicit-Cheat formulation described in [6] (Section 3.4 therein). However, as in our protocols cheaters are caught with overwhelming probability, we omit the deterrence factor, as the negligible probability of cheating success will not produce a distinguishable impact in the distribution of the output. We describe such ideal model in the following before defining security.

Ideal model In this model, it is assumed that there exists a trusted party that computes $\mathcal{F}$. A malicious adversary $\mathcal{S}$ controls a set of corrupted parties $\mathbf{P}_{cor}$. The ideal execution comprises 7 phases and goes as follows:

1. **Inputs:** Each party P_i receives a private input u_i. Additionally, $\mathcal{S}$ has an auxiliary input u^* which represents the extra knowledge other than its regular input.

2. **Send inputs to the trusted party:** All honest parties send their input to the trusted party, while parties controlled by $\mathcal{S}$ might deviate and send what $\mathcal{S}$ wishes.

3. **Early abort or corrupted input:** $\mathcal{S}$ can send $\texttt{abort}_i$ instead of a valid input, which means that some corrupted party P_i performed an early abortion or sent a corrupted input. The trusted party sends either $\texttt{abort}_i$ (choosing the index i deterministically if many parties aborted or sent a corrupted input) to all other parties and halts.

4. **Detect cheating parties:** $\mathcal{S}$ can send $\texttt{cheat}_i$ instead of a valid input, which means that a corrupted party P_i attempted to cheat. In that case, the trusted party sends $\texttt{abort}_i$ to all parties (i.e. the cheating is detected). Note this is different than in the standard definition in [6], where the cheater succeeds with probability $1 - \zeta$. In our definition, note that $\texttt{abort}_i$ and $\texttt{cheat}_i$ are considered equivalent.

5. **Trusted party answers the adversary:** If no $\texttt{abort}_i$ (or $\texttt{cheat}_i$) is sent, then the trusted party sends to $\mathcal{S}$ the outputs of $\mathcal{F}$ of the corrupted parties. After receiving them, $\mathcal{S}$ can send $\texttt{abort}_i$ to the trusted party or instruct it to continue.

6. **Trusted party answers the honest parties:** if $\mathcal{S}$ instructed the trusted party to continue, then the latter sends their outputs of $\mathcal{F}$ to the honest parties.

7. **Output:** the hones parties always output what the trusted party sent to them. $\mathcal{S}$ outputs any arbitrary computable function of the inputs $\{u_i\}_{i\in\mathbf{P}_{cor}}$, the auxiliary input u^* and the messages obtained from the trusted party.

Let $\bar{u} = (u_1 \dots u_n)$ be the vector of inputs of all parties and u^* the auxiliary input of $\mathcal{S}$. Recall that λ is the security parameter. We denote by $\text{IDEAL}_{\mathcal{F},\mathcal{S}(u^*),\mathbf{P}_{cor}}(\bar{u},\lambda)$ to the vector of outputs of parties in the above execution.

The real model The real model of an n-party protocol Π describes its execution in the presence of non-uniform probabilistic polynomial time advesary $\mathcal{A}$ that corrupts a set of parties $\mathbf{P}_{cor}$. Parties in $\mathbf{P}\setminus\mathbf{P}_{cor}$ behave as described by Π. We denote by $\text{REAL}_{\Pi,\mathcal{A}(u^*)}(\bar{u},\lambda)$ the vector of outputs of parties in the real execution of Π with input $\bar{u}$ and where $\mathcal{A}$ has auxiliary input u^*.

We formalize the notion of covert security below.

Definition 1 (Covert security - strong explicit-cheat formulation with deterrence factor equal to 1)**.** *Let λ be the security parameter and $\mathcal{F} : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party ideal functionality. A protocol* Π securely computes $\mathcal{F}$ in the presence of covert adversaries with cheat detection *if for every non-uniform probabilistic polynomial time (PPT) adversary $\mathcal{A}$ in the real model, there exist a non-uniform PPT adversary $\mathcal{S}$ in the ideal model such that the distributions of*

$$\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}(u^*),\mathbf{P}_{cor}}(\bar{u},\lambda)\right\}_{\bar{u},u^*\in(\{0,1\}^*)^{n+1},\lambda\in\mathbb{N}}$$
$$\textit{and}$$
$$\left\{\text{REAL}_{\Pi,\mathcal{A}(u^*),\mathbf{P}_{cor}}(\bar{u},\lambda)\right\}_{\bar{u},u^*\in(\{0,1\}^*)^{n+1},\lambda\in\mathbb{N}}$$

are computationally indistinguishable.

Hybrid model In addition to the security definition, the simulation paradigm facilitates tools to prove security of protocols which use sub-protocols already known to be secure. Given functionalities $\mathcal{F}_1,\dots,\mathcal{F}_k$, where k is polynomial in λ, it allows to define a protocol in which both parties can send messages to each other and place "ideal calls" to a trusted party that computes $\mathcal{F}_i$ for $i \in \{1,\dots,k\}$. In these ideal calls, parties can send their input to the trusted party and wait for the output of $\mathcal{F}_i$. However, (1) they cannot send any messages between each other after invoking an ideal functionality and before its response is returned by the trusted party and (2) functionalities cannot be called concurrently. In other words, functionalities can only be *sequentially composed* with all other interactions. We denote such model as the $(\mathcal{F}_1,\dots,\mathcal{F}_k)$*-hybrid model*. When we describe a protocol in the $(\mathcal{F}_1,\dots,\mathcal{F}_k)$-hybrid model, we say that $\mathcal{F}_1$, …, $\mathcal{F}_k$ are *hybrid functionalities*.

Sequential Composition Consider the protocol Π and functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_k$ as defined above and let $\rho_1, \ldots, \rho_k$ be protocols. We define the protocol $\Pi^{\rho_1,\ldots,\rho_k}$ to be the protocol in the ideal model that behaves exactly as Π in real messages, but for all $i \in \{1, \ldots, k\}$ each ideal call to $\mathcal{F}_i$ is replaced by the execution of protocol ρ_i. We now state conditions such in which sequential composition is secure.

Theorem 9 (Sequential composition)**.** *Let $\mathcal{F}_1, \ldots, \mathcal{F}_k$ be secure multiparty functionalities and let $\rho_1, \ldots, \rho_k$ be protocols that securely compute $\mathcal{F}_1, \ldots, \mathcal{F}_k$ in the presence of covert adversaries with cheating detection. Let $\mathcal{G}$ be a multiparty functionality and let Π be a multiparty protocol that securely computes $\mathcal{G}$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_k)$-hybrid model in the presence of covert adversaries with cheat detection. Then $\Pi^{\rho_1,\ldots,\rho_k}$ securely computes $\mathcal{G}$ in the real model in the presence of covert adversaries with cheating detection.*

The proof of the above theorem is a special case of Theorem 4.2 in [6], with all deterrence factors of the theorem are equal to 1.

Assumptions on adversaries We can see $\mathcal{A}$ as a deterministic algorithm with a special input which is a random tape of uniformly distributed bits. In the ideal model, we can also rewind $\mathcal{A}$ to a previous state in execution, where the random tape also rewinds.

4.9.2 Compressed Σ-protocols as ideal functionalities

We will use compressed Σ-protocols implemented with the Fiat-Shamir heuristic which are proven secure in the random oracle model [119]. They have been proven secure against (by definition) malicious provers and there is no interaction with malicious verifiers. Therefore we can consider them as secure and abstract them as hybrid functionalities. In particular, when we write $P \to \mathcal{F}_\Sigma^R : (x; w)$; $\mathcal{F}_\Sigma^R \to \mathcal{O} : [b, x']$, we mean that $\mathcal{F}_\Sigma^R$ gets as input from party P the public data x and secret witness w, gets from all other relevant parties as input the empty string, and returns to all parties in $\mathcal{O}$ the same pair $[b, x']$ where x' is the data provided as input by P and b is 1 if $(x; w) \in R$ (the proof succeeds) and 0 otherwise.

4.9.3 Proof of Protocol 1

Let U be the random variable uniformly distributed over the interval $[0, L)$ for $L \leq q$. We consider the *ideal functionality*

$$\mathcal{F}_{p1}(\{\} \ldots \{\}) = (U \ldots U),$$

i.e., $\mathcal{F}_{p1}$ takes as input from each of the n parties an empty string and outputs to every party the same uniformly distributed U.

Protocol The protocol Π_{p1} is explained below.
Protocol Π_{p1}:
Security Parameter: λ
Hybrid Functionality sub-protocols: Functionalities $\mathcal{F}_\Sigma^{R_1}$ and $\mathcal{F}_\Sigma^{R_2}$ are zero knowledge proofs of respectively

- $R_1 = \{(C; x, r) : C = Com(x, r)\}$
- $R_2 = \{(C, x; r) : C = Com(x, r)\}$ (only r is secret),

Protocol:

1. For $i = 1 \dots n$:
   - P_i : choose $x_i \in [0, L), r_i \in \mathbb{Z}_q$ at random
   - P_i : compute the commitment $C_i = Com(x_i, r_i)$
   - P_i : broadcast C_i
   - $P_i \rightarrow \mathcal{F}_{\sum}^{R_1} : (C_i; x_i, r_i)$
   - $\mathcal{F}_{\sum}^{R_1}$: broadcast $[b_i, C_i']$
   - P_{-i}: if $b_i \neq 1$ or $C_i' \neq C_i$, detect P_i as a cheater and abort
2. For $i = 1 \dots n$:
   - $P_i \rightarrow \mathcal{F}_{\sum}^{R_2} : (C_i, x_i; r_i)$
   - $\mathcal{F}_{\sum}^{R_2}$: broadcast $[b_i', C_i'', x_i']$
   - P_{-i}: if $x_i \notin [0, L)$, $b_i' \neq 1$ or $C_i'' \neq C_i$, detect P_i as a cheater and abort
3. For $i = 1 \dots n$:
   - **output** $\sum_{i=1}^{n} x_i' \bmod L$.

We state the security of protocol Π_{p1} in the theorem below.

Theorem 10 (Security of Π_{p1})**.** *Let Com be a computationally binding and perfectly hiding commitment scheme and let $\mathcal{F}_{\sum}^{R_1}$ and $\mathcal{F}_{\sum}^{R_2}$ be secure multiparty functionalities of computationally sound zero-knowledge proofs of relations R_1 and R_2 respectively. Then, Protocol Π_{p1} securely computes $\mathcal{F}_{p1}$ in the $(\mathcal{F}_{\sum}^{R_1}, \mathcal{F}_{\sum}^{R_2})$-hybrid model in the presence of covert adversaries with cheating detection if at least one party is honest.*

Proof. It is clear that Π_{p1} securely computes $\mathcal{F}_{p1}$ in the honest-but-curious setting.

Below, we will use $\mathcal{A}$ to denote a non-uniform probabilistic polynomial-time (PPT) adversary that controls covert parties.

We define four very similar algorithms $\mathcal{S}^v$, $v \in \{0, 1, 2, 3\}$ where $\mathcal{S}^0$ is a simulator and for which we will prove that their ideal execution output is indistinguishable from the hybrid execution output.

Phase 1: choosing z (see also phase 4)
  if $v = 0$ **then**
    $\mathcal{F}_{p1} \rightarrow \mathcal{S}^0 : z$ ($\mathcal{S}^0$ invokes trusted party delivering ideal functionality)
  end if
Phase 2: $\mathcal{S}^v$ simulates honest parties:
  $\mathcal{S}^v : i' = \max\{i \mid P_i \in \mathbf{P} \setminus \mathbf{P}_{cor}\}$
  for $P_i \in (\mathbf{P} \setminus \mathbf{P}_{cor}) \setminus \{P_{i'}\}$ **do**
    $\mathcal{S}^v$: choose x_i randomly in $[0, L)$ and r_i randomly in $\mathbb{Z}_p$
    $\mathcal{S}^v$: $C_i = Com(x_i, r_i)$; $[b_i, C_i'] = \mathcal{F}_{\sum}^{R_1}(C_i; x_i, r_i))$
    $\mathcal{S}^v \rightarrow \mathcal{A} : [b_i, C_i']$ (as if it is sent by P_i)
  end for
  if $v \in \{0, 1, 2\}$ **then**
    draw $r_{i'}$ randomly; $C_{i'} = Com(0, r_{i'})$; $[b_{i'}, C_{i'}'] = [1, C_{i'}]$

14: **else**
15: draw $x_{i'}$ and $r_{i'}$ randomly; $C_{i'} = Com(x_{i'}; r_{i'})$; $[b_{i'}, C'_{i'}] = \mathcal{F}_{\sum}^{R_1}(C_{i'}; x_{i'}, r_{i'}))$
16: **end if**
17: $\mathcal{S}^v \to \mathcal{A} : [1, C'_{i'}]$ (as if it is the answer of $\mathcal{F}_{\sum}^{R_1}(C_{i'}, x_{i'}, r_{i'})$)
18: Phase 3: $\mathcal{S}^v$ simulates $\mathcal{A}$:
19: **for** $i \in \mathbf{P}_{cor}$ **do**
20: $\mathcal{A} \to \mathcal{S}^v : C_i$
21: $\mathcal{A} \to \mathcal{F}_{\sum}^{R_1} : (C_i; x_i, r_i)$; $\mathcal{S}^v$ records C_i, x_i and r_i.
22: $\mathcal{F}_{\sum}^{R_1} \to \mathcal{S}^v : [b_i, C'_i]$ (as if sent to all members of $\mathbf{P} \setminus \mathbf{P}_{cor}$)
23: if $C'_i \neq C_i$ or $C_i \neq Com(x_i, r_i)$ detect P_i as cheater
24: **end for**
25: Phase 4: choosing z and $x_{i'}$
26: **if** $v \in \{0, 1\}$ **then**
27: **if** $v = 1$ **then**
28: $\mathcal{S}^1$ chooses z uniformly at random
29: **end if**
30: $x_{i'} = z - \sum_{i \neq i'} x_i \bmod L$
31: **else**
32: (i.e., if $v \in \{2, 3\}$)
33: draw $x_{i'}$ uniformly at random
34: $z = \sum_{i \in \mathbf{P}} x_i \bmod L$
35: **end if**
36: Phase 5:
37: **for** $P_i \in \mathbf{P} \setminus \mathbf{P}_{cor}$ **do**
38: $\mathcal{S}^v$: set $x'_i = x_i$; $C''_i = C_i$; $b'_i = 1$ and send $[b'_i, C''_i, x'_i]$ to $\mathcal{A}$ (as if it is sent by $\mathcal{F}_{\sum}^{R_2}$ in answer to $(x_i, C_i; r_i)$)
39: **end for**
40: Phase 6: $\mathcal{S}^v$ continues simulation of $\mathcal{A}$
41: **for** $P_i \in \mathbf{P}_{cor}$ **do**
42: $\mathcal{A} \to \mathcal{F}_{\sum}^{R2} : (C''_i, x'_i; r'_i)$; $\mathcal{S}^v$ records C''_i, x'_i and r'_i
43: $\mathcal{F}_{\sum}^{R2} \to \mathcal{S}^v : [b'_i, C''_i, x'_i]$
44: If $x'_i \neq x_i$ or $C''_i \neq C_i$ or $b'_i = 0$ detect P_i as cheater
45: **end for**
46: Phase 7: Output
47: $\mathcal{A} \to \mathcal{S}^v : z'$ (the output of $\mathcal{A}$)
48: output z'

To see that the simulation of $\mathcal{S}^0$ is indistinguishable from the $(\mathcal{F}_{\sum}^{R_1}, \mathcal{F}_{\sum}^{R_2})$-hybrid model with adversary $\mathcal{A}$, first consider that without loss of generality we can assume that at the points where all parties send messages, the messages of the honest parties arrive first, as everything an adversary can infer from the messages of a subset of honest parties it can also infer from the messages of all honest parties. Then, observe that

- The outputs of $\mathcal{S}^0$ and $\mathcal{S}^1$ are identically distributed, because their only difference is the moment on which z is chosen uniformly at random (and hence independently from other variables).
- The outputs of $\mathcal{S}^1$ and $\mathcal{S}^2$ are identically distributed, the only difference is that $\mathcal{S}^1$ first draws z in line 28 and then computes $x_{i'}$ from it, while $\mathcal{S}^2$ first draws $x_{i'}$ and then computes z from it.

- The outputs of $\mathcal{S}^2$ and $\mathcal{S}^3$ are indistinguishable. All inputs to $\mathcal{A}$ are the same, except for a commitment $Com(0, r_{i'})$ versus a commitment $Com(x_{i'}, r_{i'})$. As the commitment scheme is hiding and $r_{i'}$ is chosen independetly, the distributions of these commitments are indistinguishable and for any $\mathcal{A}^3$ getting input $Com(x_{i'}, r_{i'})$ there is an $\mathcal{A}^2$ producing a computationally indistinguishable output.
- the output of $\mathcal{S}^3$ is identically distributed as the output of Π_{p1} in the hybrid model.

□

Vectorized version We can now consider the ideal functionality

$$\mathcal{F}_{p1}^{(k)}(\{\}\dots\{\}) = ((U_1 \dots U_k)\dots(U_1 \dots U_k)),$$

i.e., $\mathcal{F}_{p1}^{(k)}$ takes as input from each of the n parties an empty string and outputs to every party the same vector $(U_1 \dots U_k)$ where every U_i is uniformly distributed in $[0, L)$ independently for $i = 1 \dots k$. We can then similarly construct a protocol $\Pi_{p1}^{(k)}$ that draws a vector z uniformly distributed in $[0, L)^k$.

In this protocol we will use a pseudo-random number generator as we defined in Section 2.4. We use a similar but equivalent definition which is more convenient in our proof, in particular, for some polynomial p, this is a function $G : \{0,1\}^q \to [0, L)^{p(q)}$ such that for any randomized polynomial time algorithm $A : [0, L)^{p(q)} \to \{0, 1\}$ there holds that $|P_{x \leftarrow_R \{0,1\}^q}(A(G(x)) = 1) - P_{x \leftarrow_R [0,L)^{p(q)}}(A(x) = 1)| \leq \mu(q)$ with μ a negligible function. We choose q sufficiently large such that $p(q) \geq k\lceil \log(L) \rceil$ and $\mu(q)$ is sufficiently small.

We describe this protocol in the hybrid model using the ideal functionality $\mathcal{F}_{p1[0,2^q)}$, which is a variant on the $\mathcal{F}_{p1}$ functionality we introduced above where L is set to 2^q. The protocol $\Pi_{p1}^{(k)}$ is described below:

1: **Protocol** $\Pi_{p1}^{(k)}$
2: **Security parameter:** λ
3: **Hybrid functionality sub-protocols:**

- $\mathcal{F}_{p1[0,2^q)}$: draws a single public random number in $[0, 2^q)$.

4: **Protocol:**
5: All parties collaboratively perform:
6: $z' = \mathcal{F}_{p1[0,2^q)}(\{\}\dots\{\})$
7: For $P_i \in \mathbf{P}$:
8: P_i: Output $z = (z_1 \dots z_k)$ with z_j containing bits $(j-1)\lceil \log(L) \rceil + 1 \dots j\lceil \log(L) \rceil$ of $G(z')$ for $j = 1 \dots k$

Theorem 11 (Security of $\Pi_{p1}^{(k)}$)**.** *Let $\mathcal{F}_{p1[0,2^q)}$ be a secure multiparty functionality as defined above. Then, Protocol $\Pi_{p1}^{(k)}$ securely computes $\mathcal{F}_{p1}^{(k)}$ in the $\mathcal{F}_{p1[0,2^q)}$-hybrid model in the presence of covert adversaries with cheating detection if at least one party is honest.*

Proof. It follows from the definition of random number generator that the output of the protocol is indistinguishable from the output of the ideal functionality when all parties are honest-but-curious.

The protocol consists of first a call to a secure sub-protocol and then a public computation which each agent can do for himself without any interaction. It is hence easy to see that the protocol is secure. □

4.9.4 Proof of Protocol 2

Now we prove the security of Protocol 2, which performs a private uniform draw. Without loss of generality, we assume that the drawn sample should be private to P_1. The ideal functionality $\mathcal{F}_{p2} : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ is defined as

$$\mathcal{F}_{p2}(\{\} \dots \{\}) = ((z, r_z), C_z, \dots, C_z)$$

where $C_z = Com(z, r_z)$, z is uniformly randomly distributed in $[0, L)^k$ and r_z is uniformly randomly distributed in $\mathbb{Z}_q$ (as a consequence of the distributions of z and r_z, C_z is uniformly distributed over $\mathbb{G}$). The pair (z, r_z) is private to P_1 while C_z is known by all parties. We will describe a detailed version of Protocol 2 in the hybrid model. Our hybrid functionalities will be $\mathcal{F}_{p1}$ and $\mathcal{F}_{p1}^{(k)}$, which correspond to Protocol 1, and $\mathcal{F}_{\Sigma}^{R_m}$, which, for some modulus $L \leq q$ is the ideal functionality of a zero knowledge proof for relation

$$\begin{aligned} R_m = \{&(y \in [0, L)^k, r_y \in \mathbb{Z}_q, C_x \in \mathbb{G}, C_z \in \mathbb{G} \\ &; x \in [0, L)^k, r_x \in \mathbb{Z}_q, z \in [0, L)^k, r_z \in \mathbb{Z}_q) : \\ &C_x = Com(x, r_x) \wedge C_z = Com(z, r_z) \wedge x \in [0, L) \wedge z = x + y \bmod L \\ &\wedge r_z = r_x + r_y \bmod \mathbb{Z}_q\}. \end{aligned}$$

Such proof can be performed by applying a compressed Σ-protocol using the techiques decribed in Section 4.6.1. We define below the protocol Π_{p2} which describes Protocol 2 in more detail.

Protocol Π_{p2} (to generate single private sample):
Security Parameter: λ
Hybrid Functionalitiy sub-protocols:

- $\mathcal{F}_{p1}^{(k)}$ draws publicly a random number from $[0, L)^k$.
- $\mathcal{F}_{p1\mathbb{Z}_p}$ is a variant of $\mathcal{F}_{p1}$ that draws a random number from $\mathbb{Z}_q$ rather than from $[0, L)$.
- $\mathcal{F}_{\Sigma}^{R_m}$ performs a zero-knowledge proof for the relation R_m defined above

Protocol:
P_1:
    draw $x \in [0, L)^k$ and $r_x \in \mathbb{Z}_q$ uniformly at random
    compute $C_x = Com(x, r_x)$
    broadcast C_x
All parties in $\mathbf{P}$ collaboratively:
    call $\mathcal{F}_{p1}^{(k)}$ to obtain a public y uniformly distributed over $[0, L)^k$
    call $\mathcal{F}_{p1\mathbb{Z}_p}$ to obtain a public r_y uniformly distributed over $\mathbb{Z}_q$
P_1:

13: Compute $z = x + y \bmod L$ and $r_z = r_x + r_y \bmod q$
14: Compute $C_z = Com(z, r_z)$
15: broadcast C_z
16: Parties perform an ideal call to $\mathcal{F}_{\sum}^{R_m}$:
17: $P_1 \rightarrow \mathcal{F}_{\sum}^{R_m} : (y, r_y, C_x, C_z; x, r_x, z, r_z)$
18: $\mathcal{F}_{\sum}^{R_m}$: broadcast $[b, y', r'_y, C'_x, C'_z]$
19: P_{-1}: if $b = 0 \vee r_u \neq r'_y \vee y \neq y' \vee C_x \neq C'_x \vee C_Z \neq C'_z$, detect P_1 as a cheater, otherwise continue the execution
20: P_1: **output** (z, r_z)
21: P_{-1}: **output** C_z

Theorem 12. *Let Com be a computationally binding and perfectly hiding commitment scheme and let $\mathcal{F}_{\sum}^{R_m}$ be a secure multiparty functionality of a computationally sound zero-knowledge proof of relation R_m. Let $\mathcal{F}_{p1}^{(k)}$ and $\mathcal{F}_{p1[0,2^q)}$ be secure multiparty functionalities as defined above. Then, protocol Π_{p2} securely computes $\mathcal{F}_{p2}$ in the $(\mathcal{F}_{p1}^{(k)}, \mathcal{F}_{p1\mathbb{Z}_p}, \mathcal{F}_{\sum}^{R_m})$-hybrid model in the presence covert adversaries with cheating detection if at least one party is honest.*

Proof. First note that, except for P_1, the other parties are only supposed to perform ideal calls and do not interact with each other. We will consider first the most difficult case in which P_1 is among the corrupted parties controlled by $\mathcal{A}$. We define our adversary $\mathcal{S}$ that simulates the output of $\mathcal{A}$ in the ideal model as follows:

1: $\mathcal{S}$: internally run $\mathcal{A}$ until broadcast C_x (line 8 in Π_{p2})
2: $\mathcal{A} \rightarrow \mathcal{S} : C_x$
3: $\mathcal{S}$: continue the run of $\mathcal{A}$ until after the ideal call to $\mathcal{F}_{p1}^{(k)}$ (line 10 in Π_{p2})
4: $\mathcal{S}$: draw a random value $y \in [0, L)^k$
5: $\mathcal{S} \rightarrow \mathcal{A} : y$ (as if it came from $\mathcal{F}_{p1}^{(k)}$)
6: $\mathcal{S}$: continue the run of $\mathcal{A}$ until after the ideal call to $\mathcal{F}_{p1\mathbb{Z}_p}$ (line 11 in Π_{p2})
7: $\mathcal{S}$: draw a random value $r_y \in \mathbb{Z}_q$
8: $\mathcal{S} \rightarrow \mathcal{A} : r_y$ (as if it came from $\mathcal{F}_{p1\mathbb{Z}_p}$)
9: $\mathcal{A} \rightarrow \mathcal{S} : C_z$
10: $\mathcal{S}$: continue run of $\mathcal{A}$: $P_1 \rightarrow \mathcal{F}_{\sum}^{R_m} : (y', r'_y, C'_x, C'_z, x', r'_x, z', r'_z)$
11: $\mathcal{F}_{\sum}^{R_m}$: broadcast $[b, y', r'_y, C'_x, C'_z]$
12: if $b = 0 \vee y' \neq y \vee r'_y \neq r_y \vee C'_x \neq C_x \vee C'_z \neq C_z$, detect P_1 as a cheater
13: $\mathcal{S}$: invoke trusted party computing $\mathcal{F}_{p2}$
14: $\mathcal{F}_{p2}$ returns $((z, r_z) \in [0, L)^k \times \mathbb{Z}_q, (C_z, \ldots, C_z) \in \mathbb{G}^{|\mathbf{P}_{cor}|-1})$ to $\mathcal{S}$ and $(C_z, \ldots, C_z) \in \mathbb{G}^{|\mathbf{P} \setminus \mathbf{P}_{cor}|}$ to the honest parties
15: $\mathcal{S}$ sets $y = z - x \bmod L$ and $r_y = r_z - r_x \bmod q$
16: $\mathcal{S}$ rewinds $\mathcal{A}$ to before the invocation of $\mathcal{F}_{p1}^{(k)}$
17: $\mathcal{S}$ continues the internal run of $\mathcal{A}$, which performs ideal calls to $\mathcal{F}_{p1}^{(k)}$ and $\mathcal{F}_{p1\mathbb{Z}_p}$.
18: $\mathcal{S} \rightarrow \mathcal{A} : y, r_y$ (as if they were sent by $\mathcal{F}_{p1}^{(k)}$ and $\mathcal{F}_{p1\mathbb{Z}_p}$)
19: $\mathcal{A} \rightarrow \mathcal{S} : C''_z$; if $C''_z \neq C_z$, detect P_1 as a cheater
20: $\mathcal{S}$ continues run $\mathcal{A}$: $P_1 \rightarrow \mathcal{F}_{\sum}^{R_m} : (y'', r''_y, C''_x, C''_z, x'', r''_x, z'', r''_z)$
21: $\mathcal{F}_{\sum}^{R_m}$: broadcast $[b', y'', r''_y, C''_x, C''_z]$
22: if $b' = 0 \vee y'' \neq y \vee r''_y \neq r_y \vee C''_x \neq C_x \vee C''_z \neq C_z$, detect P_1 as a cheater

23: output whatever $\mathcal{A}$ outputs

Now we show that the ideal and hybrid execution outputs are indistinguishable. We first start by analyzing the view of $\mathcal{A}$ in the hybrid protocol and when interacting with $\mathcal{S}$.

- Since $\mathcal{A}$ has not seen any other message, the first commitment C_x of $\mathcal{A}$ is the same in both hybrid and ideal executions.

- y and r_y have the same distribution in the hybrid and ideal model as they are part of an ideal call. Therefore, also the C_z broadcasted by $\mathcal{A}$ follows the same distribution in the hybrid and ideal model.

- if $\mathcal{A}$ cheats in C_z (i.e., C_z is not a commitment according to relation R_m), it gets caught both in the hybrid and ideal executions

- before rewinding $\mathcal{A}$, $\mathcal{S}$ recovers x and r_x such that $C_x = Com(x, r_x)$ with overwhelming probability (this is because $\mathcal{A}$ cannot fake the zero knowledge proof in $\mathcal{F}_{\Sigma}^{R_m}$ except with negligible probability).

- in the ideal world, $\mathcal{S}$ sets y and r_y such that $z = x+y \bmod L$ and $r_z = r_x+r_y \bmod q$, where x and r_x are chosen by $\mathcal{A}$, z and y are uniformly randomly distributed in $[0, L)^k$ and r and r_y are uniformly randomly distributed in $\mathbb{Z}_q$. This is the same distribution as in the hybrid world.

- now $\mathcal{A}$ broadcast C_z'' and either is detected as cheater or $C_z'' = Com(z, r_z)$ with overwhelming probability both in the hybrid or ideal executions.

Up to this point, from the view of $\mathcal{A}$ the transcripts simulated by $\mathcal{S}$ in the ideal execution and the transcript in the hybrid execution are indistinguishable. Since $\mathcal{A}$ runs in polynomial time, his output must be indistinguishable in both executions.

Now we analyze the case where P_1 is among then honest parties. Particularly, consider the worst case where all other parties are malicious and P_1 is the only honest party. This case is still easy since the only interaction of malicious users is to perform ideal calls to $\mathcal{F}_{p1}^{(k)}$, $\mathcal{F}_{p1\mathbb{Z}_p}$ and $\mathcal{F}_{\Sigma}^{R_m}$. The only role of $\mathcal{S}$ is to send inputs that might be consistent according to the behavior of $\mathcal{A}$. Even if all the verifiers are malicious, the honest party can still detect them. □

4.9.5 Non-uniform Public and Private Draws

Public and Private draws from many distributions such as Gaussians or Laplace distributions can be performed by first drawing uniformly distributed samples with either Protocol 1 for public draws or Protocol 2 for private draws, then applying a non-interactive transformation. For private draws, this transformation is a non-interactive compressed Σ-protocol as described in Section 4.9.2. For each technique, we explain the appropriate transformation in Sections 4.7.1, 4.8.1, 4.8.2, 4.8.3 and 4.8.4. This procedure is secure as it is essentially a secure drawing of a uniformly distributed random number as discussed in the previous sections followed by a private but verifiable single-party post-processing.

4.10 Evaluation

In this section, we present an empirical comparison of several methods to privately sample from the Gaussian distribution. We will publish code to reproduce all experiments together with the final version of this paper.

4.10.1 Setup

We evaluate the costs of the methods presented in Section 4.8. Namely, the Central Limit Theorem approach (CLT), the Box Müller (BM), the Polar Method (PolM) and the inversion method. In the latter, we evaluate the two described strategies: using series (InvM-S) and rational approximations (InvM-R). Samples are generated for the $\mathcal{N}(0,1)$ distribution.

We evaluate the cost of each method instantiated for several parameters against the statistical quality of the generated samples. For the computational cost we measure the exponentiations in $\mathbb{G}$ (GEX), which dominate the computation. The total communication cost is the number of elements of $\mathbb{G}$ and $\mathbb{Z}_q$ sent (see Section 4.2). To measure the statistical quality, we generate 10^7 samples and measure the Mean Squared Error (MSE) from the ideal Gaussian CDF.

The varying parameter for BM and PolM is the number of iterations ν of their Cordic approximations, which is chosen between 2 and 14. For CLT we vary the number of averaged uniform terms between 2 and 400. For InvM-R, the number of terms of the approximation series is changed in order to obtain different approximations with errors between 0.5 and 2^{-20}. The rational approach InvM-R has no varying parameter. The representation parameter ψ which defines $\mathbb{Q}_{\langle q,\psi\rangle}$ is chosen to be the smallest as allowed by BM, PolM, InvM-S due to approximation constraints, and for CLT is set to optimize the quality/cost tradeoff .

4.10.2 Results

Figure 4.1 shows the communication costs, i.e., the number of elements of $\mathbb{G}$ required to prove one Gaussian draw. Note that, as described in Section 4.2, 6 elements of $\mathbb{Z}_q$ must be added to obtain the final cost. If high precision is not important, CLT performs well, but in general PolM, BM and Inv-R give the best precision for a given computational investment. The Inv-R method is much simpler but cannot be tuned to other precisions.

Figure 4.2 shows the number of GEX to prove (by the party who draws the number) or to verify (by another party) one Gaussian draw. Here too, BM and PolM are the most efficient methods as soon as a good statistical quality is required. We note that, as the communication cost is logarithmic in the number of inputs and multiplication gates, several parameter settings give different points in Figure 4.2 but may have the same communication cost, so in Figure 4.1 we just show the proof with best statistical quality.

For illustration, if we implement Pedersen commitments using the secp256k1 [1] elliptic curve, we obtain 128 bit security and an element of $\mathbb{G}$ can be represented with 257 bits. One GEX using this curve takes no more than 30 microseconds on an Intel Core i7-6600U at 2.60 GHz CPU. With BM, PolM and InvM-R, a sample with MSE $< 10^{-6}$ requires less than 900 Bytes of communication. With PolM, such sample takes less than 360

[1] See `https://www.secg.org/SEC2-Ver-1.0.pdf` and `https://github.com/bitcoin-core/secp256k1`

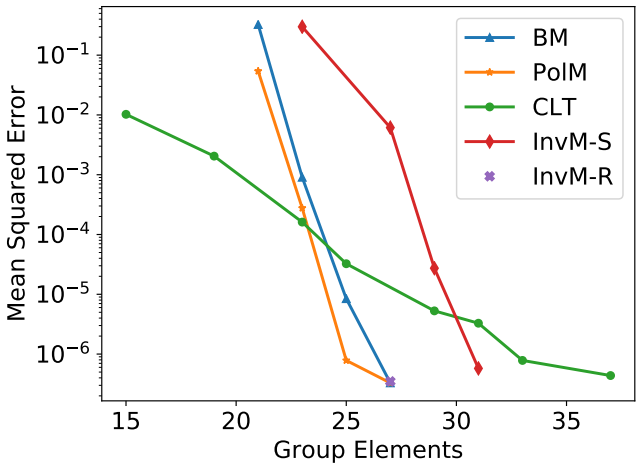


Figure 4.1: *(Comm. costs)* Required Group Elements for one sample against the MSE for BM, PolM, InvM and CLT approaches.

milliseconds (ms) to prove and 75 ms for its verification. While CLT quickly gets very expensive, if quality is less important and an MSE $> 10^{-4}$ is satisfactory, it is the most efficient approach. A proof of a sample using CLT with MSE 10^{-2} can be generated in less than 10 ms, verified in 3 ms and has a size of 482 Bytes.

We also note that it is possible to further optimize our implementation using special-purpose algorithms [112] to compute multiple exponentiations in the form $\bar{g}^{\bar{b}}$.

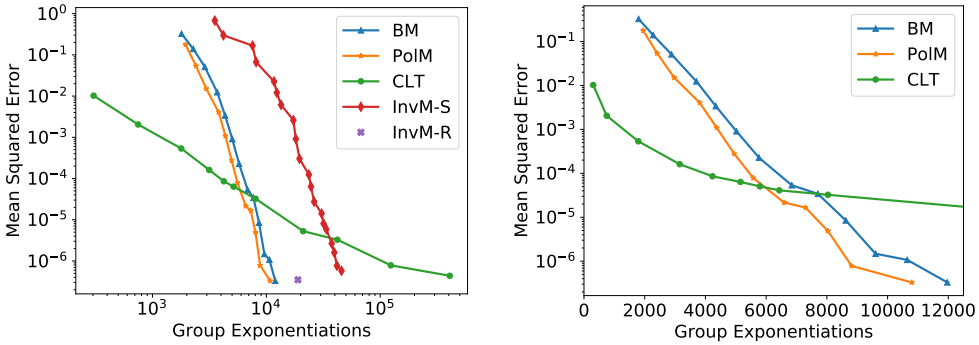
Finally, in Figure 4.3 we show the gap in the communication cost between our PolM and CLT sampling techniques implemented with classic (non-compressed) Σ-protocols [46, 47] to the presented compressed techniques.

4.11 Application: Differentially Private Machine Learning

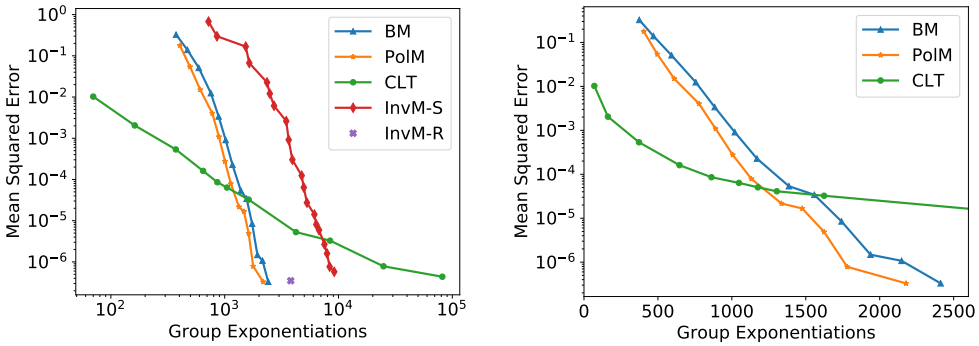
An important application of verifiable sampling can be found in the field of federated machine learning under differential privacy. Consider the parties $U = \{i\}_{i=1}^{n}$ where each party i has some sensitive private data x_i. The parties U want to keep their data x_i private but want to collaborate to obtain statistical information θ of common interest. For example, similar to GOPA in Chapter 3, assume that $\forall i \in [n] : 0 \leq x_i \leq 1$ and that the parties in U would like to compute $\theta = \frac{1}{n}\sum_{i=1}^{n} x_i$. As shown in Chapter 2.2, common strategies to make information DP before publication are to add noise from appropriately scaled Laplace or Gaussian distributions. Then, for any $\varepsilon > 0$, if we set $\hat{\theta} = \theta + \eta$ with $\eta \sim \text{Lap}(1/\varepsilon)$ there holds that $\hat{\theta}$ is ε-DP. Alternatively, for any $\varepsilon > 0$ and $\delta > 0$, if we set $\hat{\theta} = \theta + \eta$ with $\eta \sim \mathcal{N}\left(0, \frac{2\ln(1.25/\delta)}{\varepsilon^2}\right)$ there holds that $\hat{\theta}$ is (ε, δ)-DP.

It is important that no party knows the added noise η, because knowing both $\hat{\theta}$ and η would allow to reconstruct $\theta = \hat{\theta} - \eta$. We present two protocols, one privately drawing random numbers, which produces a less accurate result, and one based on the more expensive hidden drawing which has optimal precision.

Protocol 9 (DP learning using private sampling)**.** *Assume it is possible to partition U into groups of parties of size at most ρn (with $0 \leq \rho < 1$) such that parties in different groups*



(a) Proving cost.



(b) Verification cost.

Figure 4.2: *(Comp. costs)* Required group exponentiations for one sample against the MSE for BM, PolM, InvM and CLT approaches. Costs in the left side plots are in logarithmic scale. The right side plots are zooms of their left plots, in linear scale and with a detailed view on the most efficient methods.

do not collude with each other. As described in Section 4.8, let all parties i *(*$i \in [n]$*) privately verifiably draw a Gaussian random number* $\eta_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{n(1-\rho)}\right)$*, where noise with variance* σ^2 *on* θ *would be sufficient to achieve the desired privacy level. Then, securely sum* $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n}(x_i + \eta_i)$ *and publish* $\hat{\theta}$*.*

As discussed in Chapter 3, even if a group of colluding parties $U_{coll} = \{i\}_{i \in C}$ with $C \subseteq [n]$ and $|C| \leq \rho n$ collects all noise they have contributed $\eta_{coll} = \sum_{i \in C} \eta_i$ and subtracts it from $\hat{\theta}$ to obtain $\theta_{coll} = \hat{\theta} - \eta_{coll}/n$, then there is still Gaussian noise with variance $\frac{\sigma^2}{1-\rho} - \rho\frac{\sigma^2}{1-\rho} = \sigma^2$ left on their best estimation of θ.

Protocol 10 (DP learning using hidden sampling)**.** *Let all parties* i *(with* $i \in [n]$*) represent their private number* x_i *as a shared secret. Let the parties next together verifiably drawn a hidden random number* η*, i.e., a random number they obtain only as a shared secret. Finally, let them sum the secret shares and reveal* $\hat{\theta} = \sum_{i=1}^{n} x_i + \eta$

The advantage of this protocol is that no parties see η or parts of it, so it is impossible to get back towards the sensitive statistic θ. On the other hand, the full computation needs to be performed through multi-party computations, e.g., using shared secrets, which is clearly more expensive than the ZKPs which are needed in Protocol 9, especially as compressed Σ-protocols allow for ZKPs of size only logarithmic in the circuit size while calculations on secret shares have a linear communication cost.

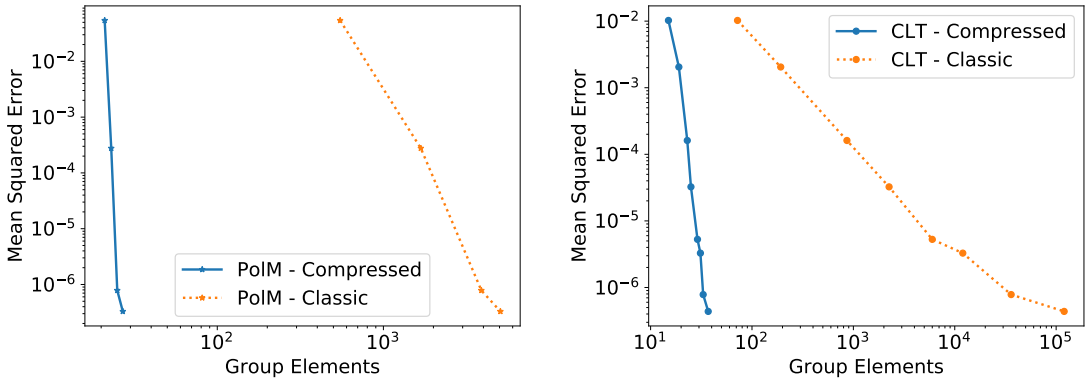


Figure 4.3: *(Comparison w. classic Σ-protocols)* Required Group Elements for one sample against the MSE using compressed and classic Σ-protocols for PolM (left) and CLT (right).

As such drawing of random noise is a basic building block and needs to be performed repeatedly by secure federated differentially private machine learning algorithms, being able to draw from these probability distributions with low communication cost is essential to make algorithms more efficient.

4.12 Conclusion

We have presented novel methods for drawing random numbers in a verifiable way in a public, private and hidden setting. We applied the ideas to the Laplace and Gaussian distribution, and evaluated several alternatives to sample from the Gaussian distribution.

We see several interesting directions for future work. First, we hope to develop novel strategies to let our methods scale better when in the course of an algorithm many random numbers are needed. Second, we would like to develop new methods which allow for more efficient sampling in the hidden setting where the random numbers are output as shared secrets. In particular, our current methods based on generic secret sharing techniques require multiple rounds of computation and communication, it may be possible to develop more efficient special-purpose strategies.

Chapter 5

Conclusion and Perspectives

We give in this chapter a conclusion to our dissertation. We summarize the presented contributions in Section 5.1 and describe lines of future work and limitations in Section 5.2.

5.1 Summary

We presented contributions to improve differentially private decentralized machine learning without a trusted curator. In our first contribution, we presented GOPA, a novel privacy-preserving averaging protocol that matches the accuracy of trusted curator DP. We show that if the communication graph between honest parties is connected, these honest parties enjoy differential privacy. Moreover, the more connected the communication graph is, the lower is the error if a party drops out and there is no time to roll back its contribution. We show that a random communication graph hits a nice balance between efficiency, needing only a logarithmic communication cost per party, and robustness against unknown colluding adversaries. Next to our worst-case bounds, we perform random graph simulations to obtain DP parameters that are less conservative than in our theoretical results. Our protocol weakens the tension between privacy, accuracy and communication cost. We achieve this using bounded pairwise noise, instead of modular unbounded uniform noise masks common in popular techniques. This makes uncancelled pairwise noise due to dropouts to have a bounded impact on utility and makes the protocol more robust.

Furthermore, GOPA can actively prevent malicious behavior to bias the result of the computation via Zero Knowledge Proofs. While ZKPs have been widely studied, using them to handle malicious behaviors in decentralized machine learning is novel.

In the second contribution, we further develop on techniques for robustness in the context of randomized computations. We propose methods for sampling Gaussian and Laplace distributions in malicious settings. These techniques can be applied in machine learning, for example, when sampling noise for differential privacy. In addition, we discuss strategies for sampling from arbitrary distributions in the malicious setting. This work brings a broad treatment of methods for private sampling. We also propose efficient zero knowledge proofs of transcendental relations, which can be used in robust machine learning as building blocks to other proofs. Finally, we provide a comparative study of the efficiency of drawing random Gaussian numbers using different approximation strategies and applications for differential privacy.

5.2 Directions for Future Work

Now we provide directions for future work. We cover aspects that range from concrete lines of work to long term perspectives.

Runtime evaluation It would be interesting to know how the practical execution time depends on the parameters of our protocols. To address this, prototype implementations that cover global aspects of our protocols are needed. A further question in this direction is how to simulate realistic dropouts and malicious parties. For verifiable sampling, a systematic experimental study is needed, including the complete spectrum of outlined techniques.

Batch Computation Another question is to evaluate the impact of vectorization. That is, averaging vectors of sensitive inputs in GOPA and generating many random numbers at once in our sampling techniques of Chapter 4. In verifiable sampling, such evaluation could provide interesting insights on the performance of sampling methods that are not efficient when only generating a few samples (for example, table lookups) but their cost is amortized when many samples are generated. For GOPA, compression methods might reduce the communication cost when averaging vectors which, with the current protocol, is linear in the number of vector dimensions.

Further Combination of our Contributions Another line of future work is to incorporate techniques of Chapter 4 into GOPA. To reduce communication, note that we use classical Σ-protocols for GOPA while a direct improvement is to use compressed Σ-protocols as described in Chapter 4. Furthermore, the more detailed study of generating random numbers in Chapter 4 can significantly improve aspects such as the cryptographic cost and utility of GOPA. For example, hidden random draws that no party knows can get GOPA from *nearly* as little noise as in the trusted curator setting to *exactly* the same amount of noise as in the trusted curator setting.

Efficiency and Finite Precision As explained in Chapter 3, issues of finite precision can be overcome with care, often requiring sufficiently large finite groups as parameters of our ZKPs. However, bigger finite groups yield to longer communication constants. In GOPA, finite groups are only required to prove the correctness of the computation but not for the computation itself. It could be explored how other ZKP primitives that perform proofs over all the integers might impact on the communication cost.

From Building Blocks to Machine Learning The purpose of our contributions is to be used as building blocks of bigger machine learning systems. While GOPA has a direct application in federated learning aggregation, linear regression and decision trees, a remaining question is how to compute other statistics from averages. Our ZKPs of transcendental computations could also be incorporated into bigger systems. For example, when implementing a ZKP of the correct evaluation of an activation function or a loss function. Proving in zero knowledge the correct evaluation of an entire machine learning model or a model update in federated learning is challenging. While techniques for this purpose exist, in most of the cases they are still prohibitive in practice due to their cost.

Related Challenges Other challenges are related to identity control and shuffling. The identity control problem is to check without compromising privacy that all enrolled participants are indeed true and different individuals. This protects the identity of each party while prevents from attacks such as the creation of fake identities to bias the computation. In this direction, existent tools such as anonymous credentials might provide promising solutions. The shuffling problem is to prevent that all messages of the same participant are linked together. This can help to reduce the amount of noise required for DP, as shown in the shuffle model of differential privacy. Algorithms that perform shuffling, i.e. a random permutation of the updates of each party, without requiring a trusted party is an interesting direction of research.

Limitations of this work Below, we describe scenarios in which our contributions might not be applicable.

Some AI techniques require intensive computation that can only be performed by certain clusters and not by multiple personal devices. This can happen if the computation is not easily parallelizable or due to the limited computational power of participants, even if a massive amount of them collaborate. Other approaches could be suitable in such cases: for example, a technique in which a single party with sufficient computational power collects encrypted data and obliviously exploits it by using fully homomorphic encryption (if such primitive becomes an efficient alternative).

In our protocols, the interaction between parties requires a certain level of synchronization. For instance, when generating noise samples, parties have sometimes to wait for the response of a message to continue their computation. This may be problematic for unstable networks or in the presence of participants with different computational power, where some of them might get blocked waiting for message responses, increasing latency and the risk of further network problems. Even if our protocols have measures to deal with some of these issues (e.g. for dropouts in the case of GOPA) other approaches might show better resilience to them. For example, protocols where parties send messages and can continue their own computations without further interaction with the receivers may benefit from a more predictable execution time and require simpler prevention measures.

Our verification protocol requires to commit to private values and to prove the correctness of computations. While this approach successfully hides information about the input and output, it still reveals the "structure" of the computation through the statements being proven. In some settings where participants use different learning techniques to compute their contributions, it might be desirable to keep these techniques hidden. This can prevent from attackers that exploit vulnerabilities of a particular algorithm and hide the algorithms that participants do not want to disclose, for example, for competitive advantage. Model poisoning in federated learning could be reduced in such scenarios by requiring parties to prove that their model updates perform "sufficiently good" instead of proving that the exact computation was done correctly.

Our threat model does not contemplate some relevant scenarios that can be present in practice. An example is a setting where each party is interested in biasing the outcome in a different way to obtain individual benefit. In that situation, all parties are malicious but they might not collude with each other. Even if malicious, parties may decide to have a "honest-looking" behavior to avoid being detected and giving advantage to other "competitors". A game-theoretical study on these more general scenarios could provide other solutions and a better understanding of the applicability of our protocols.

Long Term Perspectives Achieving privacy in machine learning and other artificial intelligence systems are problems that face many challenges. They require the integration of heterogeneous primitives such as the contributions presented in this dissertation together with other techniques to compute statistics, privacy of identity, proofs of consistency, etc. In the near future, composing primitives in large ecosystems of data exploitation might not be enough to ensure good levels of privacy or to design computationally tractable models. This is especially challenging for techniques that require complex data transformations. Further research will be required to achieve more efficient integrations. Two important lines of work are the derivation of tighter bounds on the composition of DP algorithms that do not exhaust the privacy budget of parties, and the search of computationally tractable techniques to verify higher level computations.

For a broad study of privacy, it is also crucial to understand what theoretical parameters are safe to be used in practice. Methods that can derive realistic values for parameters, possibly taking into account already existent perturbations when data is shared, are important to provide complete guarantees.

Privacy-preserving decentralized machine learning increases significantly the amount of interactions in the network and, in some cases, the amount of transferred information. Even if data is sent through secure channels, real implementations might expose vulnerabilities. Hence, it is important to evaluate if decentralized machine learning can increase potential risks such as eavesdroppers and other kinds of attacks.

Finally, we would like to point out that machine learning techniques require an intense consumption of energy. Privacy preserving techniques, especially when they use cryptographic primitives, might make this problem worse. Not all problems can be addressed at once, but it is important to do research in more environmental friendly tools and uses of our protocols.

Bibliography

[1] S. Advokat and K.-R. Newspapers. PUBLICATION OF BORK'S VIDEO RENTALS RAISES PRIVACY ISSUE. https://web.archive.org/web/20220322015153/https://www.chicagotribune.com/news/ct-xpm-1987-11-20-8703270590-story.html. Last Access: 2022-03-22.

[2] N. Agarwal, P. Kairouz, and Z. Liu. The skellam mechanism for differentially private federated learning. In *NeurIPS*, 2021.

[3] N. Agarwal, A. T. Suresh, F. X. Yu, S. Kumar, and B. McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *NeurIPS*, 2018.

[4] A. Aly and N. P. Smart. Benchmarking Privacy Preserving Scientific Operations. In R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, editors, *Applied Cryptography and Network Security*, Lecture Notes in Computer Science, pages 509–529, Cham, 2019. Springer International Publishing.

[5] T. Attema and R. Cramer. Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, Lecture Notes in Computer Science, pages 513–543, Cham, 2020. Springer International Publishing.

[6] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, 2010.

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *AISTATS*, 2020.

[8] V. Balcer and S. Vadhan. Differential Privacy on Finite Computers. In *ITCS*, 2018.

[9] B. Balle, J. Bell, A. Gascón, and K. Nissim. Private Summation in the Multi-Message Shuffle Model. In *CCS*, 2020.

[10] F. Bayatbabolghani, M. Blanton, M. Aliasgari, and M. Goodrich. Secure fingerprint alignment and matching protocols. *arXiv preprint arXiv:1702.03379*, 2017.

[11] J. Bell, A. Bellet, A. Gascón, and T. Kulkarni. Private Protocols for U-Statistics in the Local Model and Beyond. In *AISTATS*, 2020.

[12] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS*, 2020.

[13] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, Dec. 1993. Association for Computing Machinery.

[14] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *S&P*, 2014.

[15] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. Cryptology ePrint Archive, Report 2019/1255, 2019. `https://ia.cr/2019/1255`.

[16] D. Bernhard, O. Pereira, and B. Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In X. Wang and K. Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, Lecture Notes in Computer Science, pages 626–643, Berlin, Heidelberg, 2012. Springer.

[17] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[18] A. Biswas and G. Cormode. Verifiable differential privacy for when the curious become dishonest, 2022.

[19] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*, 2017.

[20] M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[21] B. Bollobás. *Random Graphs (2nd edition)*. Cambridge University Press, 2001.

[22] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*, 2017.

[23] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, Lecture Notes in Computer Science, pages 327–357, Berlin, Heidelberg, 2016. Springer.

[24] G. E. P. Box and M. E. Muller. A Note on the Generation of Random Normal Deviates. *Annals of Mathematical Statistics*, 29(2):610–611, June 1958. Publisher: Institute of Mathematical Statistics.

[25] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530, 2006.

[26] A. Broder and D. Dolev. Flipping coins in many pockets (byzantine agreement on uniformly random values). In *25th Annual Symposium onFoundations of Computer Science, 1984.*, pages 157–170, 1984.

[27] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, May 2018. ISSN: 2375-1207.

[28] J. Camenisch, R. Chaabouni, and A. Shelat. Efficient Protocols for Set Membership and Range Proofs. In J. Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, Lecture Notes in Computer Science, pages 234–252, Berlin, Heidelberg, 2008. Springer.

[29] J. Camenisch and M. Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, Lecture Notes in Computer Science, pages 107–122, Berlin, Heidelberg, 1999. Springer.

[30] 2020-21 survey of canadians on privacy-related issues. `https://web.archive.org/web/20220917162325/https://www.priv.gc.ca/en/opc-actions-and-decisions/research/explore-privacy-research/2021/por_2020-21_ca/`, 2021.

[31] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[32] H. Chan, A. Perrig, and D. X. Song. Random Key Predistribution Schemes for Sensor Networks. In *S&P*, 2003.

[33] T.-H. H. Chan, E. Shi, and D. Song. Optimal Lower Bound for Differentially Private Multi-party Aggregation. In *ESA*, 2012.

[34] T.-H. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography*, 2012.

[35] D. Chaum, J.-H. Evertse, and J. van de Graaf. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In D. Chaum and W. L. Price, editors, *Advances in Cryptology — EUROCRYPT' 87*, Lecture Notes in Computer Science, pages 127–141, Berlin, Heidelberg, 1988. Springer.

[36] D. Chaum and T. P. Pedersen. Wallet Databases with Observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, Lecture Notes in Computer Science, pages 89–105, Berlin, Heidelberg, 1993. Springer.

[37] W.-N. Chen, P. Kairouz, and A. Ozgur. Breaking the communication-privacy-accuracy trilemma. In *NeurIPS*, 2020.

[38] A. Cheu, A. D. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. Distributed Differential Privacy via Shuffling. In *EUROCRYPT*, 2019.

[39] S. Chevillard. The functions erf and erfc computed with arbitrary precision and explicit error bounds. *Information and Computation*, 216:72–95, July 2012.

[40] S. Chevillard and N. Revol. Computation of the error functions erf & erfc in arbitrary precision with correct rounding. Research Report RR-6465, INRIA, 2008.

[41] S. G. Choi, D. Dachman-Soled, M. Kulkarni, and A. Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. Cryptology ePrint Archive, Paper 2020/029, 2020. `https://eprint.iacr.org/2020/029`.

[42] Cisco consumer privacy survey. https://web.archive.org/web/20220901043636/https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-cybersecurity-series-2021-cps.pdf, 2021. Last accessed: 2022-09-01.

[43] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 364–369, New York, NY, USA, 1986. Association for Computing Machinery.

[44] N. Confessore. Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. https://web.archive.org/web/20220411013138/https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html, 2018. Last access: 2022-04-11.

[45] H. Corrigan-Gibbs and D. Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 259–282, 2017.

[46] R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, University of Amsterdam, Jan. 1997.

[47] R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, Lecture Notes in Computer Science, pages 424–441, Berlin, Heidelberg, 1998. Springer.

[48] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Y. G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, Lecture Notes in Computer Science, pages 174–187, Berlin, Heidelberg, 1994. Springer.

[49] E. Cyffers and A. Bellet. Privacy Amplification by Decentralization. *arXiv:2012.05326 [cs, stat]*, Mar. 2022. arXiv: 2012.05326.

[50] I. Damgard, M. Fitzi, E. Kiltz, J. Nielsen, and T. Toft. Unconditionally secure constant rounds multi-party computation for equality, comparison, bits and exponentiation. In *TCC 2006*, volume 3876 of *LNCS*, page 285–304. Springer, 2006.

[51] I. Damgard, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In *ESORICS*, volume 8134 of *LNCS*, pages 1–18. Springer, 2013.

[52] Q. Dang. Secure Hash Standard (SHS). Technical Report Federal Information Processing Standard (FIPS) 180-4, National Institute of Standards and Technology, Gaithersburg, MD, 2015.

[53] V. Dimitrov, L. Kerik, T. Krips, J. Randmets, and J. Willemson. Alternative Implementations of Secure Real Numbers. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 553–564, New York, NY, USA, Oct. 2016. Association for Computing Machinery.

[54] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *NIPS*, 2017.

[55] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[56] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local Privacy and Statistical Minimax Rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438, Oct. 2013. ISSN: 0272-5428.

[57] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, 2013.

[58] C. Dwork. Differential Privacy. In *ICALP*, 2006.

[59] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 486–503, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[60] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[61] C. Dwork and G. N. Rothblum. Concentrated Differential Privacy. *arXiv:1603.01887 [cs]*, Mar. 2016. arXiv: 1603.01887.

[62] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985. Conference Name: IEEE Transactions on Information Theory.

[63] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, and K. Talwar. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*, 2019.

[64] U. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[65] T. I. Fenner and A. M. Frieze. On the connectivity of random m-orientable graphs and digraphs. *Combinatorica*, 2(4):347–359, 1982.

[66] A. Fiat and A. Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, Lecture Notes in Computer Science, pages 186–194, Berlin, Heidelberg, 1987. Springer.

[67] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *NeurIPS*, 2020.

[68] B. Ghazi, R. Kumar, P. Manurangsi, and R. Pagh. Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *ICML*, 2020.

[69] M. Giles. Approximating the erfinv function. In *GPU Computing Gems Jade Edition*, pages 109–116. Elsevier, 2012.

[70] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[71] O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[72] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.

[73] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, 18(1):186–208, Feb. 1989. Publisher: Society for Industrial and Applied Mathematics.

[74] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[75] K. Gupta, D. Kumaraswamy, N. Chandran, and D. Gupta. Llama: A low latency math library for secure inference. Cryptology ePrint Archive, Paper 2022/793, 2022. `https://eprint.iacr.org/2022/793`.

[76] V. Hartmann and R. West. Privacy-Preserving Distributed Learning with Secret Gradient Descent. Technical report, arXiv:1906.11993, 2019.

[77] J. Hayes and O. Ohrimenko. Contamination attacks and mitigation in multi-party machine learning. In *NeurIPS*, 2018.

[78] M. Heikkilä, E. Lagerspetz, S. Kaski, K. Shimizu, S. Tarkoma, and A. Honkela. Differentially private Bayesian learning on distributed data. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[79] H. Imtiaz, J. Mohammadi, and A. D. Sarwate. Distributed differentially private computation of functions with correlated noise. *arXiv preprint arXiv:1904.10059*, 2021.

[80] B. Jayaraman, L. Wang, D. Evans, and Q. Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In *NeurIPS*, 2018.

[81] D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, Aug. 2001.

[82] P. Kairouz, K. Bonawitz, and D. Ramage. Discrete distribution estimation under local privacy. In *ICML*, 2016.

[83] P. Kairouz, Z. Liu, and T. Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *ICML*, 2021.

[84] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[85] P. Kairouz, S. Oh, and P. Viswanath. Secure multi-party differential privacy. In *NIPS*, 2015.

[86] P. Kairouz, S. Oh, and P. Viswanath. Extremal Mechanisms for Local Differential Privacy. *Journal of Machine Learning Research*, 17:1–51, 2016.

[87] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith. What Can We Learn Privately? In *FOCS*, 2008.

[88] F. Kato, Y. Cao, and M. Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. In K. Barker and K. Ghazinour, editors, *Data and Applications Security and Privacy XXXV*, pages 43–60, Cham, 2021. Springer International Publishing.

[89] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[90] D. Kifer and A. Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems*, 39(1):3:1–3:36, Jan. 2014.

[91] R. Knop. Remark on algorithm 334 [G5]: normal random deviates. *Communications of the ACM*, 12(5):281, 1969. Publisher: ACM New York, NY, USA.

[92] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[93] M. Krivelevich. Embedding spanning trees in random graphs. *SIAM J. Discret. Math.*, 24(4), 2010.

[94] K. K. R. Lai, N. Perlroth, T. Hsu, and J. Keller. How Many Times Has Your Personal Information Been Exposed to Hackers? `https://web.archive.org/web/20220411013830/https://www.nytimes.com/interactive/2015/07/29/technology/personaltech/what-parts-of-your-information-have-been-exposed-to-hackers-quiz.html`, July 2015. Last acces: 2022-04-11.

[95] B. M. Lal Srivastava, N. Vauquier, M. Sahidullah, A. Bellet, M. Tommasi, and E. Vincent. Evaluating Voice Conversion-Based Privacy Protection against Informed Attackers. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2802–2806, May 2020. ISSN: 2379-190X.

[96] M. Liedel. Secure Distributed Computation of the Square Root and Applications. In M. D. Ryan, B. Smyth, and G. Wang, editors, *Information Security Practice and Experience*, Lecture Notes in Computer Science, pages 277–288, Berlin, Heidelberg, 2012. Springer.

[97] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi. Don't Use Large Mini-batches, Use Local SGD. In *ICLR*, 2020.

[98] Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Journal of Cryptology*, 16(3):143–184, June 2003.

[99] W. Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, Lecture Notes in Computer Science, pages 60–71, Berlin, Heidelberg, 1998. Springer.

[100] G. Marsaglia and T. A. Bray. A Convenient Method for Generating Normal Variables. *SIAM Review*, 6(3):260–264, July 1964. Publisher: Society for Industrial and Applied Mathematics.

[101] G. Marsaglia and W. W. Tsang. The ziggurat method for generating random variables. *Journal of statistical software*, 5(8):1–7, 2000.

[102] S. Mazloom and S. D. Gordon. Secure computation with differentially private access patterns. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 490–507, New York, NY, USA, 2018. Association for Computing Machinery.

[103] K. McCarthy. 2018 ain't done yet... Amazon sent Alexa recordings of man and girlfriend to stranger. `https://web.archive.org/web/20220224164933/https://www.theregister.com/2018/12/20/amazon_alexa_recordings_stranger/`. Last access: 2022-02-24.

[104] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[105] P. K. Meher, J. Valls, T. Juang, K. Sridharan, and K. Maharatna. 50 Years of CORDIC: Algorithms, Architectures, and Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(9):1893–1907, Sept. 2009. Conference Name: IEEE Transactions on Circuits and Systems I: Regular Papers.

[106] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *S&P*, 2019.

[107] T. Mendel, A. Puddephatt, B. Wagner, D. Hawtin, and N. Torres. *Global survey on internet privacy and freedom of expression*. UNESCO, 2012.

[108] I. Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 650–661, New York, NY, USA, 2012. Association for Computing Machinery.

[109] I. Mironov. Rényi Differential Privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, Aug. 2017. ISSN: 2374-8303.

[110] J.-M. Muller. *Elementary Functions: Algorithms and Implementation*. Birkhäuser Basel, 3 edition, 2016.

[111] G. Munilla Garrido, J. Sedlmeir, and M. Babel. Towards verifiable differentially-private polling. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES '22, New York, NY, USA, 2022. Association for Computing Machinery.

[112] B. Möller. Algorithms for Multi-exponentiation. In S. Vaudenay and A. M. Youssef, editors, *Selected Areas in Cryptography*, Lecture Notes in Computer Science, pages 165–180, Berlin, Heidelberg, 2001. Springer.

[113] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online at http://bitcoin.org/bitcoin.pdf, 2008.

[114] A. Narayanan and V. Shmatikov. How To Break Anonymity of the Netflix Prize Dataset. *arXiv:cs/0610105*, Nov. 2007. arXiv: cs/0610105.

[115] N. Narula, W. Vasquez, and M. Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 65–80, Renton, WA, Apr. 2018. USENIX Association.

[116] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *S&P*, 2019.

[117] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, Lecture Notes in Computer Science, pages 129–140, Berlin, Heidelberg, 1992. Springer.

[118] Americans and privacy: Concerned, confused and feeling lack of control over their personal information. https://web.archive.org/web/20220921215648/https://www.pewresearch.org/internet/2019/11/15/americans-and-privacy-concerned-confused-and-feeling-lack-of-control-over-their-personal-information/, 2019. Last accessed: 2022-09-21.

[119] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 387–398, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[120] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux. POSEIDON: Privacy-Preserving Federated Neural Network Learning. *arXiv:2009.00349 [cs]*, Jan. 2021. arXiv: 2009.00349.

[121] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, Jan. 1991.

[122] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-Preserving Aggregation of Time-Series Data. In *NDSS*, 2011.

[123] M. B. Sinai, N. Partush, S. Yadid, and E. Yahav. Exploiting social navigation. *Arxiv*, 1410.0151, 2014.

[124] J. So, B. Güler, and A. S. Avestimehr. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, Mar. 2021. Conference Name: IEEE Journal on Selected Areas in Information Theory.

[125] S. U. Stich. Local SGD Converges Fast and Communicates Little. In *ICLR*, 2019.

[126] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[127] G. Tsaloli and A. Mitrokotsa. Differential privacy meets verifiable computation: Achieving strong privacy and integrity guarantees. In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2: SECRYPT, Prague, Czech Republic, July 26-28, 2019*, pages 425–430, 2019.

[128] C. S. Wallace. Fast pseudorandom generators for normal and exponential variates. *ACM Transactions on Mathematical Software*, 22(1):119–127, Mar. 1996.

[129] J. S. Walther. A unified algorithm for elementary functions. In *Proceedings of the May 18-20, 1971, spring joint computer conference*, AFIPS '71 (Spring), pages 379–385, New York, NY, USA, May 1971. Association for Computing Machinery.

[130] S. Weckert. Google maps hacks. `https://web.archive.org/web/20220325225255/http://www.simonweckert.com/googlemapshacks.html`.

[131] O. Yağan and A. M. Makowski. On the Connectivity of Sensor Networks Under Random Pairwise Key Predistribution. *IEEE Transactions on Information Theory*, 59(9):5754–5762, 2013.

[132] A. C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 80–91, Nov. 1982. ISSN: 0272-5428.

[133] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng. Veriml: Enabling integrity assurances and fair payments for machine learning as a service. *IEEE Transactions on Parallel and Distributed Systems*, 32(10):2524–2540, 2021.