COMBINING DIFFERENTIAL PRIVACY AND CRYPTOGRAPHY FOR MODERN APPLICATIONS

by

Amrita Roy Chowdhury

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2022

Date of final oral examination: 15 December, 2021

The dissertation is approved by the following members of the Final Oral Committee:
Somesh Jha, Professor, Computer Sciences
Kassem Fawaz, Assistant Professor, Electrical and Computer Engineering
Earlence Fernandes, Assistant Professor, Computer Sciences
Parameswaran Ramanathan, Professor, Electrical and Computer Engineering

Dedicated to my mom and grandfather

TABLE OF CONTENTS

Page

Page

Page

Page

ABSTRACT

Data is the new oil of the 21st century. The modern data economy is built on a two-pronged approach: (1) collect large amounts of data from diverse domains, and (2) efficiently analyze the data at scale to glean useful information. Its success and continued sustenance is rooted in the ubiquity of technology in our daily lives – every individual carrying a smart device today constitutes a source of data. This has resulted in a distributed data ecosystem where the data collected often comprises sensitive, personal information. Hence, there is an urgent need to develop technical solutions that can support analysis over distributed data without violating data privacy.

Over the past few years, differential privacy (DP) has emerged as the de-facto standard for achieving data privacy. On the other hand, modern cryptography has been the backbone of building secure systems in the presence of mutually distrusting parties for over three decades now. Traditionally, DP and cryptography have been studied in isolation from each other. In this dissertation, we show that moving forward, privacy concerns of modern applications can be addressed by *combining techniques from both DP and cryptography.* We present three directions for leveraging this synergy to develop privacy-first solutions that (1) provide formal privacy guarantees, (2) support high utility data analysis, and (3) are practical for real-world usage. First, we present Cryptϵ, a system that demonstrates how cryptography can enable high utility differentially private query analytics on distributed data. Second, we present OPϵ, a novel differentially private order-preserving encryption scheme that allows efficient data analysis involving the order of the data on an untrusted server. Finally, we present EIFFeL that allows privacy-preserving decentralized learning by co-designing both DP and

cryptography. Through the work presented in this dissertation, we show that it is possible for us to enjoy the benefits of modern computing while protecting the privacy of our data.

Chapter 1

Introduction

The landscape of data privacy has changed in the 21st century due to a multitude of reasons. First, machine learning (ML) has become ubiquitous in our lives. Ranging from spam filtering for emails to diagnosing cancer, ML is driving almost all the decision-making in modern society. The success of ML is rooted in the continued availability of large amounts of data from diverse domains. Consequently, there has been an explosion in the amount of personal data being collected and analyzed. In fact, every individual today constitutes a source of data, owing to our daily interactions with smart devices. This has led to the development of a decentralized data ecosystem which consists of multiple resource constrained data owners (clients) and a resource-heavy service provider (server), aiming at performing analysis on the joint dataset. Second, we spend more than quarter of our lives online today [Lov19], leaving behind a massive digital footprint. With access to such rich auxiliary information about individuals just a Google search away, adversaries are now more powerful than ever – the extent of privacy attacks range from de-identification of anonymized records to full data reconstruction [Cen20]. Hence, the threat of privacy violation is becoming increasingly real, as evident from the plethora of recent privacy breach incidents [dat16a, dat16b, dat17a, dat17b, dat18, dat19]. Finally, public awareness about the urgency of ensuring data privacy is growing. For instance, in 2017, a landmark decision by the Supreme Court of India granted her 1.3 billion citizens the Right to Privacy [SC17]. Additionally, several government agencies across the world have introduced privacy regulations, such as EU's GDPR [gdp] and California's CCPA [ccp], that make companies legally obligated to uphold the privacy of individuals. Hence, a key question facing us today is:

> How to disseminate results of data analysis on sensitive data, that is distributed across multiple parties, without violating privacy?

An emerging answer to the question of safe dissemination of the outputs of data analysis is differential privacy (DP). DP limits the amount of information leaked by an algorithm.

Semantically, this provides a rigorous guarantee of privacy for individuals in a dataset, regardless of an adversary's auxiliary knowledge [TSD20]. DP is currently the de-facto standard for data privacy and has been adopted by both government agencies [Cen20, MKA+08, VSA17] as well as major commercial organizations, such as Microsoft [DKY17], Apple [Gre16], Google [EPK14, FPE15] and Uber [JNS18]. It is characterized by a parameter $\epsilon > 0$ where lower the value of ϵ, greater the privacy achieved.

On the other hand, modern cryptography is the backbone of building secure systems and holds the key to the problem of secure data analysis over distributed data. Specifically, multi-party computation (MPC) enables multiple mutually distrusting parties to collaborate and compute over their joint dataset *without seeing the data*. Tremendous advancement have been made over the past few decades towards the goal of making secure computation practical for real-world usage.

Traditionally, DP and cryptography have been studied in isolation from each other. However, as motivated above, DP and cryptography are complementary techniques that are key to addressing the privacy concerns of modern applications. We argue that combining techniques from the two areas would lead to mutually beneficial solutions that pushes the frontiers of practical deployment of privacy-first technologies. In this dissertation, we **explore the synergy between DP and cryptography through novel algorithms that expose the rich interconnections between the two areas, both in theory and practice**. Specifically, we explore three paradigms: (1) Cryptography for DP, (2) DP for Cryptography, and (3) Co-design of DP and Cryptography.

1.1 Contributions

(1) Cryptography for DP

Two popular models of DP are *centralized differential privacy* (CDP) and *local differential privacy* (LDP). In CDP, data from individuals are collected and stored *in the clear* in a *trusted* centralized data curator which then executes DP programs on the sensitive data and releases outputs to an untrusted data analyst. However, this assumption of a trusted server is ill-suited in practice as it constitutes a single point of failure for data breaches, and saddles the trusted curator with legal and ethical obligations to uphold data privacy. Hence, most commercial deployments have opted for the alternative LDP model which does not require a trusted curator; each individual perturbs their data using a DP algorithm. The data analyst uses these noisy data to infer aggregate statistics of the datasets. For instance, Microsoft follows the LDP model to collect Window's usage statistics. LDP's attractive privacy properties, however, come at a cost. Under the CDP model, the expected additive error for an aggregate count over a dataset of size n is at most $\Theta(1/\epsilon)$ to achieve ϵ-DP. In

contrast, under the LDP model, at least $\Omega(\sqrt{n}/\epsilon)$ additive expected error must be incurred by any ϵ-DP program, owing to the randomness of each data owner. In other words, under the LDP model, for a database of a billion people, one can only learn properties that are common to at least $30,000$ people. Under CDP, on the other hand, one can learn properties that are shared by as few as a 100 people. The LDP model imposes additional penalties on the algorithmic expressibility; there exists an exponential separation between the accuracy and sample complexity of LDP and CDP algorithms. Thus, the LDP model operates under more practical trust assumptions than CDP, but as a result incurs higher loss in data utility. This raises the following question:

> Is it possible to design a practical system that can bridge the trust-utility gap between LDP and CDP?

Proposed Solution. Chapter 2 of the dissertation discusses Cryptϵ [RCWH+20], a system and a programming framework for executing DP programs that: (1) never stores or computes on sensitive data in the clear (2) achieves the accuracy guarantees and algorithmic expressibility of the CDP model. Cryptϵ employs a pair of untrusted but non-colluding servers – Analytics Server (AS) and Cryptographic Service Provider (CSP). The AS executes DP programs (like the data curator in CDP) but on *encrypted* records. The CSP initializes and manages the cryptographic primitives, and collaborates with the AS to generate the program outputs. Under the assumption that the AS and the CSP are semi-honest and do not collude (a common assumption in cryptographic systems), Cryptϵ ensures ϵ-DP guarantee for its programs via two cryptographic primitives – linear homomorphic encryption (LHE) and garbled circuits.

(2) DP for Cryptography

Resource-constrained data owners often rely on outsourcing their data for efficient data storage and management. However, as exposed by frequent mass data breaches, such outsourced data storage is vulnerable to privacy threats in practice. This has lead to a rapid development of systems that aim to protect the data (even in the event of a whole-system compromise) while enabling statistical analysis on the dataset. Encrypted database systems that allow query computation over the encrypted data is a popular approach in this regard. Typically such systems rely on *property-preserving encryption* schemes to enable efficient computation. *Order-preserving encryption* (OPE) is one such cryptographic primitive that preserves the numerical order of the plaintexts even after encryption. This allows actions like sorting, ranking, and answering range queries to be performed directly over the encrypted data. However, this efficiency of encrypted databases comes at a cost – such systems are

vulnerable to inference attacks that can reveal the plaintexts with good accuracy. Most of these attacks are inherent to any property-preserving encryption scheme – they do not leverage any weakness in the cryptographic security guarantee of the schemes but are rather carried out based on just the preserved property. For example, the strongest cryptographic guarantee for OPEs (IND-FA-OCPA) informally states that *only* the order of the plaintexts will be revealed from the ciphertexts. However, inference attacks can be carried out by leveraging only this ordering information. In contrast, an appealing property of DP is that any post-processing (inferential) computation performed on the noisy output of a DP algorithm does not incur additional privacy loss. This prompts the following question:

> Is it possible to leverage the properties of DP for providing a formal security guarantee for OPEs even in the face of inference attacks?

Proposed Solution. Chapter 3 discusses a novel *differentially private order preserving encryption* scheme, OPϵ [CDJ$^+$21], that combines the two approaches. Recall that standard OPE schemes are designed to reveal nothing but the order of the plaintexts. Our proposed scheme, OPϵ, ensures that this leakage of order is differentially private. In other words, the cryptographic guaratantee of OPEs is bolstered with a layer of DP guarantee. As a result, even if the cryptographic security guarantee of standard OPEs proves to be inadequate (in the face of inference attacks), the DP guarantee would continue to hold true. Intuitively, the reason behind is DP's resilience to post-processing computations. To the best of our knowledge, *this is the first work to combine DP with a property-preserving encryption scheme.*

(3) Co-design of DP and Cryptography

Federated learning (FL; [MMR$^+$17]) is a learning paradigm for decentralized data in which multiple clients collaborate with a server to train a machine-learning (ML) model. Each client computes an update on its *local* training data and shares it with the server; the server aggregates the local updates into a *global* model update. This allows clients to contribute to model training without sharing their private data. However, the local updates can still reveal information about a client's private data [MSDCS19, BDF$^+$18, ZLH19, YMV$^+$21, NSH19]. FL addresses this problem via a two-fold mechanism. First, a federated learner performs *secure aggregation* – clients mask the updates they share, and the server can *only* recover the aggregate in the clear. Next, suitable noise is added to the aggregate to guarantee DP which protects the clients from any privacy violation arising from the aggregate itself.

An additional challenge is that the distributed nature of federated learning makes it vulnerable to *Byzantine* failures wherein clients submit malformed updates to the server [BCMC19, KMA$^+$19, SKSM19a]. Specifically, one or more (colluding) malicious clients can ingest

specially crafted inputs to stage poisoning attacks that can either reduce model accuracy [BNL12, MZ15, FCJG20] or implant targeted back-doors in that model that can be exploited later [CLL+17, BVH+18]. Even a single malformed update can significantly alter the trained model [BMGS17b]. Thus, ensuring the well-formedness of the updates, *i.e.*, upholding their *integrity* is a primary task for ensuring robustness in federated learning. This problem is especially challenging in the context of secure aggregation where the individual updates are masked from the server which prevents any audits on them. These vulnerabilities in FL lead to the research question:

> How can a federated learner efficiently verify the integrity of clients' updates without violating their privacy?

Proposed Solution. In Chapter 4, we formalize this problem by proposing *secure aggregation of verified inputs* (SAVI, [CGJvdM21]) protocols that: (1) securely verify the integrity of each local update, (2) aggregate *only* well-formed updates, and (3) release only the final aggregate in the clear. A SAVI protocol allows for checking the well-formedness of updates *without observing them*, thereby ensuring *both* the privacy and integrity of updates.

In order to demonstrate the feasibility of SAVI, we propose EIFFeL: a system that instantiates a SAVI protocol that can perform *any integrity check that can be expressed as an arithmetic circuit with public parameters*. This provides EIFFeL the flexibility to implement a plethora of modern ML approaches that ensure robustness to Byzantine errors by checking the integrity of per-client updates before aggregating them [SKSM19b, SKL17, XKG20, XKG19, LCW+20, DMG+18, BVH+18, SH21]. EIFFeL is a general framework that empowers a federated learner to deploy (multiple) *arbitrary* integrity checks of their choosing on the "masked" updates. With EIFFeL, we take the first step towards designing aggregation protocols for federated learning that ensures both input privacy and integrity. EIFFeL can be used as a building block and easily extended to release differentially private aggregates.

In this dissertation, we show that it is possible to build practical systems that address the privacy needs of modern applications by *combining techniques from differential privacy and cryptography*. We present three directions for leveraging this synergy to develop privacy-first solutions that (1) provide formal privacy guarantees, (2) support high utility data analysis, and (3) are practical for real-world usage.

Chapter 2

Cryptϵ: Crypto-Assisted Differential Privacy on Untrusted Servers

Differential privacy (DP) is typically implemented in one of two models – *centralized differential privacy* (CDP) and *local differential privacy* (LDP). In CDP, data from individuals are collected and stored *in the clear* in a *trusted* centralized data curator which then executes DP programs on the sensitive data and releases outputs to an untrusted data analyst. In LDP, there is no trusted data curator. Rather, each individual perturbs his/her own data using a (local) DP algorithm. The data analyst uses these noisy data to infer aggregate statistics of the datasets. In practice, CDP's assumption of a trusted server is ill-suited for many applications as it constitutes a single point of failure for data breaches, and saddles the trusted curator with legal and ethical obligations to uphold data privacy. Hence, recent commercial deployments of DP [EPK14, Gre16] have preferred LDP over CDP. However, LDP's attractive privacy properties comes at a cost. Under the CDP model, the expected additive error for a aggregate count over a dataset of size n is at most $\Theta(1/\epsilon)$ to achieve ϵ-DP. In contrast, under the LDP model, at least $\Omega(\sqrt{n}/\epsilon)$ additive expected error must be incurred by any ϵ-DP program [BNO08, CSS12a, DJW13], owing to the randomness of each data owner. In other words, under the LDP model, for a database of a billion people, one can only learn properties that are common to at least $30,000$ people. Under CDP, on the other hand, one can learn properties that are shared by as few as a 100 people. The LDP model in fact imposes additional penalties on the algorithmic expressibility; the power of LDP is equivalent to that of the statistical query model [Kea98] and there exists an exponential separation between the accuracy and sample complexity of LDP and CDP algorithms [KLN$^+$08].

In this chapter, we strive to bridge the gap between LDP and CDP. We propose, Cryptϵ, a system and a programming framework for executing DP programs that:

- never stores or computes on sensitive data in the clear, but still
- achieves the accuracy guarantees and algorithmic expressibility of the CDP model

Cryptϵ employs a pair of untrusted but non-colluding servers – Analytics Server (AS) and Cryptographic Service Provider (CSP). The AS executes DP programs (like the data curator in CDP) but on *encrypted* data records. The CSP initializes and manages the cryptographic primitives, and collaborates with the AS to generate the program outputs. Under the assumption that the AS and the CSP are semi-honest and do not collude (a common assumption in cryptographic systems [NWI+13, NIW+13, GSB+17a, KKK+16, MZ17, GJJ+18, GSB+16]), Cryptϵ ensures ϵ-DP guarantee for its programs via two cryptographic primitives – linear homomorphic encryption (LHE) and garbled circuits. One caveat here is that due to the usage of cryptographic primitives, the DP guarantee obtained in Cryptϵ is that of computational differential privacy or SIM-CDP [MPRV09] (details in Chapter 3.4.2).

Cryptϵ provides a data analyst with a programming framework to author logical DP programs just like in CDP. Like in prior work [McS09, ES15, ZMK+18], access to the sensitive data is restricted via a set of predefined transformations operators (inspired by relational algebra) and DP measurement operators (Laplace mechanism and Noisy-Max [DR14a]). Thus, any program that can be expressed as a composition of the above operators automatically satisfies ϵ-DP (in the CDP model) giving the analyst a proof of privacy for free. Cryptϵ programs support constructs like looping, conditionals, and can arbitrarily post-process outputs of measurement operators.

The main contributions of this work are:

- **New Approach.** We present the design and implementation of Cryptϵ, a novel system and programming framework for executing DP programs over encrypted data on two non-colluding and untrusted servers.

- **Algorithm Expressibility.** Cryptϵ supports a rich class of state-of-the-art DP programs expressed in terms of a small set of transformation and measurement operators. Thus, Cryptϵ achieves the accuracy guarantees of the CDP model without the need for a trusted data curator.

- **Ease Of Use.** Cryptϵ allows the data analyst to express the DP program logic using high-level operators. Cryptϵ automatically translates this to the underlying implementation specific secure protocols that work on encrypted data and provides a DP guarantee (in the CDP model) for free. Thus, the data analyst is relieved of all concerns regarding secure computation protocol implementation.

- **Performance Optimizations.** We propose optimizations that speed up computation on encrypted data by at least an order of magnitude. A novel contribution of this work

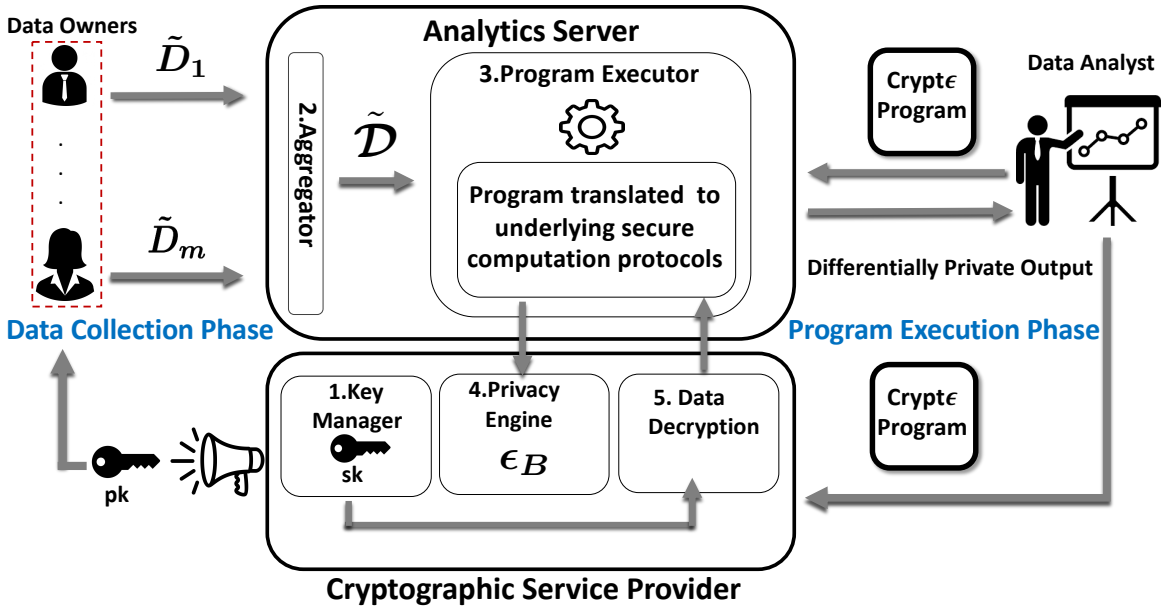


Figure 2.1: Cryptϵ System

is a DP indexing optimization that leverages the fact that noisy intermediate statistics about the data can be revealed.

- **Practical for Real World Usage.** For the same tasks, Cryptϵ programs achieve accuracy comparable to CDP and 50× more than LDP for a dataset of size $\approx 30,000$. Cryptϵ runs within 3.6 hours for a large class of programs on a dataset with 1 million rows and 4 attributes.

- **Generalized Multiplication Using LHE.** Our implementation uses an efficient way for performing n-way multiplications using LHE which maybe of independent interest.

2.1 Cryptϵ Overview

2.1.1 System Architecture

Figure 2.1 shows Cryptϵ's system architecture. Cryptϵ has two servers: Analytics server (AS) and Cryptographic Service Provider (CSP). At the very outset, the CSP records the total privacy budget, ϵ^B (provided by the data owners), and generates the key pair, $\langle sk, pk \rangle$ (details in Chapter 3.2), for the encryption scheme. The data owners, $\mathsf{DO}_i, i \in [m]$ (m = number of data owners), encrypt their data records, D_i, in the appropriate format with the public key, pk, and send the encrypted records, $\tilde{\boldsymbol{D}}_{\boldsymbol{i}}$, to the AS which aggregates them into a single encrypted database, $\tilde{\boldsymbol{\mathcal{D}}}$. Next, the AS inputs logical programs from the data analyst and translates them to Cryptϵ's implementation specific secure protocols that work

on $\tilde{\mathcal{D}}$. A Cryptϵ program typically consists of a sequence of transformation operators followed by a measurement operator. The AS can execute most of the transformations on its own. However, each measurement operator requires an interaction with the CSP for (1) decrypting the answer, and (2) checking that the total privacy budget, ϵ^B, is not exceeded. In this way, the AS and the CSP compute the output of a Cryptϵ program with the data owners being offline.

2.1.2 Cryptϵ Design Principles

Minimal Trust Assumptions. As mentioned above, the overarching goal of Cryptϵ is to mimic the CDP model but without a trusted server. A natural solution for dispensing with the trust assumption of the CDP model is using cryptographic primitives [BEM+17, CSU+18, EFM+18, SHCGR+11, CSS12b, AHKM18b, RN10, DKM+06, BNO11, BHT+18]. Hence, to accommodate the use of cryptographic primitives, we assume a computationally bounded adversary in Cryptϵ. However, a generic m-party SMC would be computationally expensive. This necessitates a third-party entity that can capture the requisite secure computation functionality in a 2-party protocol instead. This role is fulfilled by the CSP in Cryptϵ. For this two-server model, we assume semi-honest behaviour and non-collusion. This is a very common assumption in the two-server model [NWI+13, NIW+13, GSB+17a, KKK+16, MZ17, GJJ+18, GSB+16].

Programming Framework. Conceptually, the aforementioned goal of achieving the *best of both worlds* can be obtained by implementing the required DP program using off-the-self secure multi-party computation (SMC) tools like [EMP, MPC, Sca, ABY]. However, when it comes to real world usage, Cryptϵ outperforms such approaches due to the following reasons.

First, without the support of a programming framework like that of Cryptϵ, every DP program must be implemented from scratch. This requires the data analyst to be well versed in both DP and SMC techniques; he/she must know how to implement SMC protocols, estimate sensitivity of transformations and track privacy budget across programs. In contrast, Cryptϵ allows the data analyst to write the DP program using a high-level and expressive programming framework. Cryptϵ abstracts out all the low-level implementation details like the choice of input data format, translation of queries to that format, choice of SMC primitives and privacy budget monitoring from the analyst thereby reducing his/her burden of complex decision making. Thus, every Cryptϵ program is automatically translated to protocols corresponding to the underlying implementation.

Second, SMC protocols can be prohibitively costly in practice unless they are carefully tuned to the application. Cryptϵ supports optimized implementations for a small set of operators, which results in efficiency for all Cryptϵ programs.

Third, a DP program can be typically divided into segments that (1) transform the private data, (2) perform noisy measurements, and (3) post-process the noisy measurements without touching the private data. A naive implementation may implement all the steps using SMC protocols even though post-processing can be performed in the clear. Given a DP program written in a general purpose programming language (like Python), automatically figuring out what can be done in the clear can be subtle. In Cryptϵ programs, however, transformation and measurement are clearly delineated, as the data can be accessed only through a pre-specified set of operators. Thus, SMC protocols are only used for transformations and measurements, which improves performance.

For example, the AHP algorithm for histogram release [ZCX$^+$] works as follows: first, a noisy histogram, $\hat{H}$, is released using budget ϵ_1. This is followed by post-processing steps of thresholding, sorting and clustering resulting in $\bar{H}$. Then a final histogram, $\tilde{H}$, is computed with privacy budget $\epsilon - \epsilon_1$. An implementation of the entire algorithm in a single SMC protocol using the EMP toolkit [EMP] takes 810s for a dataset of size $\approx 30K$ and histogram size 100. In contrast, Cryptϵ uses SMC protocols only for the first and third steps. Cryptϵ automatically detects that the second post-processing step can be performed in the clear. A Cryptϵ program for this runs in 238s (3.4$\times$ less time than that of the EMP implementation) for the same dataset and histogram sizes.

Last, the security (privacy) proofs for just stand-alone cryptographic and DP mechanisms can be notoriously tricky [BR06, LSL17]. Combining the two thus exacerbates the technical complexity, making the design vulnerable to faulty proofs [HMFS17]. For example, given any arbitrary DP program written under the CDP model, the distinction between intermediate results that can be released and the ones which have to be kept private is often ambiguous. An instance of this is observed in the Noisy-Max algorithm, where the array of intermediate noisy counts is private. However, these intermediate noisy counts correspond to valid query responses. Thus, an incautious analyst, in a bid to improve performance, might reuse a previously released noisy count query output for a subsequent execution of the Noisy-Max algorithm leading to privacy leakage. In contrast, Cryptϵ is designed to reveal nothing other than the outputs of the DP programs to the untrusted servers; every Cryptϵ program comes with an automatic proof of security (privacy). Referring back to the aforementioned example, in Cryptϵ, the Noisy-Max algorithm is implemented as a secure measurement operator

thereby preventing any accidental privacy leakage. The advantages of a programming framework is further validated by the popularity of systems like PINQ [McS09], Featherweight PINQ [ES15], Ektelo [ZMK+18] - frameworks for the CDP setting.

Data Owners are Offline. Recall, Cryptϵ's goal is to mimic the CDP model with untrusted servers. Hence, it is designed so that the data owners are offline after submitting their encrypted records to the AS.

Low Burden on CSP. Cryptϵ views the AS as an extension of the analyst; the AS has a vested interest in obtaining the result of the programs. Thus, we require the AS to perform the majority of the work for any program; interactions with the CSP should be minimal and related to data decryption. Keeping this in mind, the AS performs most of the data transformations by itself (Table 2.3). Specifically, for every Cryptϵ program, the AS processes the whole database and transforms it into concise representations (an encrypted scalar or a short vector) which is then decrypted by the CSP. An example real world setting can be when Google and Symantec assumes the role of the AS and the CSP respectively.

Separation of Logical Programming Framework and Underlying Physical Implementation. The programming framework is independent of the underlying implementation. This allows certain flexibility in the choice for implementation. For example, we use one-hot-encoding as the input data format (Chapter 4.1). However, any other encoding scheme like range based encoding can be used instead. Another example is that for Cryptϵ, we use ϵ-DP (pure DP) for our privacy analysis. However, other DP notions like (ϵ, δ)-DP, Rènyi DP [Mir17a] can also be used instead. Similarly, it is straightforward to replace LHE with the optimized HE scheme in [BGP+19] or garbled circuits with the ABY framework [DSZ15].

Yet another alternative implementation for Cryptϵ could be where the private database is equally shared between the two servers and they engage in a secret share-based SMC protocol for executing the DP programs. This would require both the servers to do almost equal amount of work for each program. Such an implementation would be justified only if both the servers are equally invested in learning the DP statistics and is ill-suited for our context. A real world analogy for this can be if Google and Baidu decide to compute some statistics on their combined user bases.

2.2 Background

In this chapter, we give a brief introduction to the definitions and primitives relevant to Cryptϵ.

2.2.1 Differential Privacy

Differential privacy is a quantifiable measure of the stability of the output of a randomized mechanism to changes to its input.

Definition 1. *An algorithm $\mathcal{A}$ satisfies ϵ-differential privacy (ϵ-DP), where $\epsilon > 0$ is a privacy parameter, iff for any two neighboring datasets D and D' such that $D = D' - t$ or $D' = D - t$, we have*

$$\forall S \subset Range(\mathcal{A}), \Pr\big[\mathcal{A}(D) \in S\big] \leq e^{\epsilon} Pr\big[\mathcal{A}(D') \in S\big] \tag{2.1}$$

The above definition is sometimes called *unbounded DP*. A variant is *bounded-DP* where neighboring datasets D and D' have the same number of rows and differ in one row. Any ϵ-DP algorithm also satisfies 2ϵ-bounded DP [LLSY16].

When applied multiple times, the DP guarantee degrades gracefully as follows.

Theorem 1. *(Sequential Composition) If $\mathcal{A}_1$ and $\mathcal{A}_2$ are ϵ_1-DP and ϵ_2-DP algorithms with independent randomness, then releasing $\mathcal{A}_1(D)$ and $\mathcal{A}_2(D)$ on database D satisfies $\epsilon_1 + \epsilon_2$-DP.*

Another important result is that any post-processing computation performed on the noisy output of a differentially private algorithm does not degrade privacy.

Theorem 2. *(Post-Processing) Let $\mathcal{A} : D \mapsto R$ be a randomized algorithm that is ϵ-DP. Let $f : R \mapsto R'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{A} : D \mapsto R'$ is ϵ- DP.*

The *stability* of a transformation operation is defined as

Definition 2. *A transformation $\mathcal{T}$ is defined to be t-stable if for two datasets D and D', we have*

$$|\mathcal{T}(D) \ominus \mathcal{T}(D')| \leq t \cdot |D \ominus D'| \tag{2.2}$$

where (i.e., $D \ominus D' = (D - D') \cup (D' - D)$.

Transformations with bounded stability scale the DP guarantee of their outputs, by their stability constant [McS09].

Theorem 3. *If $\mathcal{T}$ is an arbitrary t-stable transformation on dataset D and $\mathcal{A}$ is an ϵ-DP algorithm which takes output of $\mathcal{T}$ as input, the composite computation $\mathcal{A} \circ \mathcal{T}$ provides $(\epsilon \cdot t)$-DP.*

2.2.2 Cryptographic Primitives

Linearly Homomorphic Encryption (LHE). If $(\mathcal{M}, +)$ is a finite group, an LHE scheme for messages in $\mathcal{M}$ is:

- **Key Generation** (Gen). This algorithm takes the security parameter κ as input and outputs a pair of secret and public keys, $\langle s_k, p_k \rangle \xleftarrow{\$} Gen(\kappa)$.
- **Encryption** (Enc). This is a randomized algorithm that encrypts a message, $m \in \mathcal{M}$, using the public key, p_k, to generate the ciphertext, $\mathbf{c} \xleftarrow{\$} Enc_{pk}(m)$.
- **Decryption** (Dec). This uses the secret key, s_k, to recover the plaintext, m, from the ciphertext, $\mathbf{c}$, deterministically.

In addition, LHE supports the operator $\oplus$ that allows the summation of ciphers as follows:
Operator $\oplus$. Let $c_1 \leftarrow Enc_{pk}(m1), \ldots, c_a \leftarrow Enc_{pk}(m_a)$ and $a \in \mathcal{Z}_{>0}$. Then we have $Dec_{sk}(c_1 \oplus c_2 ... \oplus c_a) = m_1 + \ldots + m_a$.
One can multiply a cipher $c \leftarrow Enc_{sk}(m)$ by a plaintext positive integer a by a repetitions of $\oplus$. We denote this operation by $cMult(a, c)$ such that $Dec_{sk}\big(cMult(a, c)\big) = a \cdot m$.

Labeled Homomorphic Encryption(labHE). Let (Gen, Enc, Dec) be an LHE scheme with security parameter κ and message space $\mathcal{M}$. Assume that a multiplication operation exists in $\mathcal{M}$, i.e., is a finite ring. Let $\mathcal{F} : \{0,1\}^s \times \mathcal{L} \rightarrow \mathcal{M}$ be a pseudo-random function with seed space $\{0,1\}^s$(s= poly(κ)) and the label space $\mathcal{L}$. A labHE scheme is defined as

- **labGen**(κ). Runs $Gen(\kappa)$ and outputs (sk, pk).
- **localGen**(pk). For each user i and with the public key as input, it samples a random seed $\sigma_i \in \{0,1\}^s$ and computes $pk_i = Enc_{pk}(\underline{\sigma_i})$ where $\underline{\sigma_i}$ is an encoding of σ_i as an element of $\mathcal{M}$. It outputs (σ_i, pk_i).
- **labEnc**$_{pk}(\sigma_i, m, \tau)$. On input a message $m \in \mathcal{M}$ with label $\tau \in \mathcal{L}$ from user i, it computes $b = \mathcal{F}(\sigma_i, \tau)$ (mask) and outputs the labeled ciphertext $\mathbf{c} = (a, d) \in \mathcal{M} \times \mathcal{C}$ with $a = m - b$ (hidden message) in $\mathcal{M}$ and $d = Enc_{pk}(b)$. For brevity we just use notation **labEnc**$_{pk}(m)$ to denote the above functionality, in the rest of paper.
- **labDec**$_{sk}(\mathbf{c})$. This functions inputs a cipher $\mathbf{c} = (a, d) \in \mathcal{M} \times \mathcal{C}$ and decrypts it as $m = a - Dec_{sk}(d)$.
- **labMult**$(\mathbf{c}_1, \mathbf{c}_2)$: On input two labHE ciphers $\mathbf{c}_1 = (a_1, d_1)$ and $\mathbf{c}_2 = (a_2, d_2)$, it computes a "multiplication" ciphertext $\mathbf{e} = labMult(\mathbf{c_1}, \mathbf{c_2}) = Enc_{pk}(a_1, a_2) \oplus cMult(d_1, a_2) \oplus cMult(d_2, a_1)$. Observe that $Dec_{sk}(\mathbf{e}) = m_1 \cdot m_2 - b_1 \cdot b_2$.

- **labMultDec**$_{sk}(d_1, d_2, \mathbf{e})$: On input two encrypted masks d_1, d_2 of two labHE ciphers $\mathbf{c_1}, \mathbf{c_2}$, this algorithm decryts the output $\mathbf{e}$ of $labMult(\mathbf{c_1}, \mathbf{c_2})$ as $m_3 = Dec_{sk}(\mathbf{e}) + Dec_{sk}(d_1) \cdot Dec_{sk}(d_2)$ which is equals to $m_1 \cdot m_2$.

Garbled Circuit. Garbled circuit [Yao86, LP09a] is a generic method for secure computation. Two data owners with respective private inputs x_1 and x_2 run the protocol such that, no data owner learns more than $f(x_1, x_2)$ for a function f. One of the data owners, called generator, builds a "garbled" version of a circuit for f and sends it to the other data owner, called evaluator, alongside the garbled input values for x_1. The evaluator, then, obtains the garbled input for x_2 from the generator via oblivious transfer and computes $f(x_1, x_2)$.

2.3 Cryptϵ System Description

In this chapter, we describe Cryptϵ in detail. First, we start with workflow (Chapter 2.3.1), followed by its modules (Chapter 2.3.2) and trust assumptions (Chapter 2.3.3).

2.3.1 Cryptϵ Workflow

Cryptϵ operates in three phases:

- **Setup Phase.** At the outset, data owners initialize the CSP with a privacy budget, ϵ^B, which is stored in its *Privacy Engine* module. Next, the CSP's *Key Manager* module generates key pair (sk, pk) for labHE, publishes pk and stores sk.

- **Data Collection Phase.** In the next phase, each data owner encodes and encrypts his/her record using the *Data Encoder* and *Data Encryption* modules and sends the encrypted data records to the AS. The data owners are relieved of all other duties and can go completely offline. The *Aggregator* module of the AS, then, aggregates these encrypted records into a single encrypted database, $\tilde{\mathcal{D}}$.

- **Program Execution Phase.** In this phase, the AS executes a Cryptϵ program provided by the data analyst. Cryptϵ programs (details in Chapters 2.4 and 2.5) access the sensitive data via a restricted set of transformation operators, that filter, count or group the data, and measurement operators, which are DP operations to release noisy answers. Measurement operators need interactions with the CSP as they require (1) decryption of the answer, and (2) a check that the privacy budget is not exceeded. These functionalities are achieved by CSP's *Data Decryption* and *Privacy Engine* modules.

The *Setup* and *Data Collection* phases occur just once at the very beginning, every subsequent program is handled via the corresponding *Program Execution* phase.

2.3.2 Cryptϵ Modules

Cryptographic Service Provider (CSP)

- **Key Manager.** The *Key Manager* module initializes the labHE scheme for Cryptϵ by generating its key pair, $\langle sk, pk \rangle$. It stores the secret key, sk, with itself and releases the public key, pk. The CSP has exclusive access to the secret key, sk, and is the only entity capable of decryption in Cryptϵ.

- **Privacy Engine.** Cryptϵ starts off with a total privacy budget of ϵ^B chosen by the data owners. The choice of value for ϵ^B should be guided by social prerogatives [AS19, HGH+14, LC11] and is currently outside the scope of Cryptϵ. For executing any program, the AS has to interact with the CSP at least once (for decrypting the noisy answer), thereby allowing the CSP to monitor the AS's actions in terms of privacy budget expenditure. The *Privacy Engine* module gets the program, P, and its allocated privacy budget, ϵ, from the data analyst, and maintains a public ledger that records the privacy budget spent in executing each such program. Once the privacy cost incurred reaches ϵ^B, the CSP refuses to decrypt any further answers. This ensures that the total privacy budget is never exceeded. The ledger is completely public allowing any data owner to verify it.

- **Data Decryption.** The CSP being the only entity capable of decryption, any measurement of the data (even noisy) has to involve the CSP. The *Data Decryption* module is tasked with handling all such interactions with the AS.

Data Owners (DO)

- **Data Encoder.** Each data owner, $\mathsf{DO}_i, i \in [m]$, has a private data record, D_i, of the form $\langle A_1, ...A_l \rangle$ where A_j is an attribute. At the very outset, every data owner, DO_i, represents his/her private record, D_i, in its respective per attribute one-hot-encoding format. The one-hot-encoding is a way of representation for categorical attributes and is illustrated by the following example. If the database schema is given by $\langle Age, Gender \rangle$, then the corresponding one-hot-encoding representation for a data owner, $\mathsf{DO}_i, i \in [m]$, with the record $\langle 30, Male \rangle$, is given by $\tilde{D}_i = \langle [\underbrace{0, \ldots, 0}_{29}, 1, \underbrace{0, \ldots, 0}_{70}], [1, 0] \rangle$.

- **Data Encryption.** The *Data Encryption* module stores the public key pk of labHE which is announced by the CSP. Each data owner, $\mathsf{DO}_i, i \in [m]$, performs an element-wise encryption of his/her per attribute one-hot-encodings using pk and sends the

encrypted record, $\tilde{\mathbf{D}}_\mathbf{i}$, to the AS via a secure channel. This is the only interaction that a data owner ever participates in and goes offline after this.

Analytics Server (AS)

- **Aggregator.** The *Aggregator* collects the encrypted records, $\tilde{\mathbf{D}}_i$, from each of the data owners, DO_i, and collates them into a single encrypted database, $\tilde{\boldsymbol{\mathcal{D}}}$.

- **Program Executor.** This module inputs a logical Cryptϵ program, P, and privacy parameter, ϵ, from the data analyst, translates P to the implementation specific secure protocol and computes the noisy output with the CSP's help.

2.3.3 Trust Model

There are three differences in Cryptϵ from the LDP setting:

- **Semi-honest Model.** We assume that the AS and the CSP are *semi-honest*, i.e., they follow the protocol honestly, but their contents and computations can be observed by an adversary. Additionally, each data owner has a private channel with the AS. For real world scenarios, the semi-honest behaviour can be imposed via legal bindings. Specifically, both the AS and the CSP can swear to their semi-honest behavior in legal affidavits; there would be loss of face in public and legal implications in case of breach of conduct.

- **Non-collusion.** We assume that the AS and the CSP are *non-colluding*, i.e., they avoid revealing information [KMR11] to each other beyond what is allowed by the protocol definition. This restriction can be imposed via strict legal bindings as well. Additionally, in our setting the CSP is a third-party entity with no vested interested in learning the program outputs. Hence, the CSP has little incentive to collude with the AS. Physical enforcement of the non-collusion condition can be done by implementing the CSP inside a trusted execution environment (TEE) or via techniques which involve using a trusted mediator who monitors the communications between the servers [AKL$^+$09].

- **Computational Boundedness.** The adversary is *computationally bounded*. Hence, the DP guarantee obtained is that of computational differential privacy or SIM-CDP [MPRV09]. There is a separation between the algorithmic power of computational DP and information-theoretic DP in the multi-party setting [MPRV09]. Hence, this assumption is inevitable in Cryptϵ.

Table 2.1: Cryptϵ Operators

Types	Name	Notation	Input	Output	Functionality
Transformation	CrossProduct	$\times_{A_i,A_j \to A'}(\cdot)$	$\tilde{\boldsymbol{T}}$	$\tilde{\boldsymbol{T}}'$	Generates a new attribute A' (in one-hot-coding) to represent the data for both the attributes A_i and A_j
	Project	$_{A^*}(\cdot)$	$\tilde{\boldsymbol{T}}$	$\tilde{\boldsymbol{T}}'$	Discards all attributes but A^*
	Filter	$\sigma_{\phi}(\cdot)$	$\tilde{\boldsymbol{T}}$	$\boldsymbol{B}'$	Zeros out records not satisfying ϕ in $\boldsymbol{B}$
	Count	$count(\cdot)$	$\tilde{\boldsymbol{T}}$	$\boldsymbol{c}$	Counts the number of 1s in $\boldsymbol{B}$
	GroupByCount	$\gamma_A^{count}(\cdot)$	$\tilde{\boldsymbol{T}}$	$\boldsymbol{V}$	Returns encrypted histogram of A
	GroupByCountEncoded	$\tilde{\gamma}_A^{count}(\cdot)$	$\tilde{\boldsymbol{T}}$	$\tilde{\boldsymbol{V}}$	Returns encrypted histogram of A in one-hot-encoding
	CountDistinct	$countD(\cdot)$	$\boldsymbol{V}$	$\boldsymbol{c}$	Counts the number of non-zero values in $\boldsymbol{V}$
Measurement	Laplace	$Lap_{\epsilon,\Delta}(\cdot)$	$\boldsymbol{V}n\boldsymbol{c}$	$\hat{V}$	Adds Laplace noise to $\boldsymbol{V}$
	NoisyMax	$NoisyMax^k_{\epsilon,\Delta}(\cdot)$	$\boldsymbol{V}$	$\hat{\mathcal{P}}$	Returns indices of the top k noisy values

2.4 Cryptϵ Operators

Let us consider an encrypted instance of a database, $\tilde{\boldsymbol{\mathcal{D}}}$, with schema $\langle A_1, \ldots, A_l \rangle$. In this chapter, we define the Cryptϵ operators (summarized in Table 2.1) and illustrate how to write logical Cryptϵ programs for DP algorithms on $\tilde{\boldsymbol{\mathcal{D}}}$. The design of Crypt$\epsilon$ operators are inspired by previous work [ZMK+18, McS09].

2.4.1 Transformation operators

Transformation operators input encrypted data and output a transformed encrypted data. These operators thus work completely on the encrypted data without expending any privacy budget. Three types of data are considered in this context: (1) an encrypted table, $\tilde{\boldsymbol{T}}$, of x rows and y columns/attributes where each attribute value is represented by its encrypted one-hot-encoding; (2) an encrypted vector, $\boldsymbol{V}$; and (3) an encrypted scalar, $\boldsymbol{c}$. In addition, every encrypted table, $\tilde{\boldsymbol{T}}$, of x rows has an encrypted bit vector, $\boldsymbol{B}$, of size x to indicate whether the rows are relevant to the program at hand. The i-th row in $\tilde{\boldsymbol{T}}$ will be used for answering the current program only if the i-th bit value of $\boldsymbol{B}$ is 1. The input to the first transformation operator in Cryptϵ program is $\tilde{\boldsymbol{\mathcal{D}}}$ with all bits of $\boldsymbol{B}$ set to 1. For brevity, we use just $\tilde{\boldsymbol{T}}$ to represent both the encrypted table, $\tilde{\boldsymbol{T}}$, and $\boldsymbol{B}$. The transformation operators are:

- **CrossProduct** $\times_{(A_i,A_j)\to A'}(\tilde{\boldsymbol{T}})$. This operator transforms the two encrypted one-hot-encodings for attributes A_i and A_j in $\tilde{\boldsymbol{T}}$ into a single encrypted one-hot-encoding of a new attribute, A'. The domain of the new attribute, A', is the cross product of the domains for A_i and A_j. The resulting table, $\tilde{\boldsymbol{T}}'$, has one column less than $\tilde{\boldsymbol{T}}$. Thus, the construction of the one-hot-encoding of the entire y-dimensional domain can be computed by repeated application of this operator.

- **Project** $\pi_{\bar{A}}(\tilde{\boldsymbol{T}})$. This operator projects $\tilde{\boldsymbol{T}}$ on a subset of attributes, $\bar{A}$, of the input table. All the attributes that are not in $\bar{A}$ are discarded from the output table $\tilde{\boldsymbol{T}}'$.

- **Filter** $\sigma_{\phi}(\tilde{\boldsymbol{T}})$. This operator specifies a filtering condition that is represented by a Boolean predicate, ϕ, and defined over a subset of attributes, $\bar{A}$, of the input table, $\tilde{\boldsymbol{T}}$. The predicate can be expressed as a conjunction of range conditions over $\bar{A}$, i.e., for a row $r \in \tilde{\boldsymbol{T}}$, $\phi(r) = \bigwedge_{A_i \in \bar{A}} \ (r.A_i \in V_{A_i})$, where $r.A_i$ is value of attribute A_i in row r and V_A is a subset of values (can be a singleton) that A_i can take. For example, $Age \in [30, 40] \wedge Gender = M$ can be a filtering condition. The Filter operator affects only the associated encrypted bit vector of $\tilde{\boldsymbol{T}}$ and keeps the actual table untouched. If any row, $r \in \tilde{\boldsymbol{T}}$, does not satisfy the filtering condition, ϕ, the corresponding bit in $\boldsymbol{B}$ will be set to $labEnc_{pk}(0)$; otherwise, the corresponding bit value in $\boldsymbol{B}$ is kept unchanged. Thus the Filter transformation suppresses all the records that are extraneous to answering the program at hand (i.e., does not satisfy ϕ) by explicitly zeroing the corresponding indicator bits and outputs the table, $\tilde{\boldsymbol{T}}'$, with the updated indicator vector.

- **Count** $count(\tilde{\boldsymbol{T}})$. This operator simply counts the number of rows in $\tilde{\boldsymbol{T}}$ that are pertinent to the program at hand, i.e. the number of 1s in its associated bit vector $\boldsymbol{B}$. This operator outputs an encrypted scalar, $\boldsymbol{c}$.

- **GroupByCount** $\gamma_A^{count}(\tilde{\mathbf{T}})$. The GroupByCount operator partitions the input table, $\tilde{\mathbf{T}}$, into groups of rows having the same value for an attribute, A. The output of this transformation is an encrypted vector, $\mathbf{V}$, that counts the number of unfiltered rows for each value of A. This operator serves as a preceding transformation for other Cryptϵ operators specifically, NoisyMax, CountDistinct and Laplace.

- **GroupByCountEncoded** $\tilde{\gamma}_A^{count}(\tilde{\mathbf{T}})$. This operator is similar to GroupByCount. The only difference between the two is that GroupByCountEncoded outputs a new table that has two columns – the first column corresponds to A and the second column corresponds to the number of rows for every value of A (in one-hot-encoding). This operator is useful for expressing computations of the form "count the number of age values having at least 200 records" (see P7 in Table 2.2).

- **CountDistinct** $countD(\mathbf{V})$. This operator is always preceded by GroupByCount. Hence the input vector, $\mathbf{V}$, is an encrypted histogram for attribute, A, and this operator returns the number of distinct values of A that appear in $\tilde{\boldsymbol{\mathcal{D}}}$ by counting the non-zero entries of V.

Cryptϵ Program	Description
P1: $\forall i \in [1,100], \hat{c}_i \leftarrow Lap_{\epsilon,1}(count(\sigma_{Age\in(0,i]}(_{Age}(\tilde{\mathcal{D}})))); post_{c.d.f}([\hat{c}_1,\ldots,\hat{c}_{100}])$	Outputs the c.d.f of *Age* with domain $[1,100]$.
P2: $\hat{P} \leftarrow NoisyMax^{5}_{\epsilon,1}(\gamma^{count}_{Age}(\tilde{\mathcal{D}}))$	Outputs the 5 most frequent age values.
P3: $\hat{V} \leftarrow Lap_{\epsilon,2}(\gamma^{count}_{Race\times Gender}(_{Race\times Gender\rightarrow Race\times Gender}(\tilde{\mathcal{D}})))$	Outputs the marginal over the attributes *Race* and *Gender*.
P4: $\hat{V} \leftarrow Lap_{\epsilon,2}(\gamma^{count}_{Age\times Gender}(\sigma_{NativeCountry=Mexico}(_{Age\times Gender,NativeCountry}(\times_{Age,Gender\rightarrow Age\times Gender}(\tilde{\mathcal{D}})))))$	Outputs the marginal over *Age* and *Gender* for Mexican employees.
P5: $\hat{c} \leftarrow Lap_{\epsilon,1}(count(\sigma_{Age=30\wedge Gender=Male\wedge NativeCountry=Mexico}(_{Age,Gender,NativeCountry}(\tilde{\mathcal{D}}))))$	Counts the number of male employees of Mexico having age 30.
P6: $\hat{c} \leftarrow Lap_{\epsilon,2}(countD(\gamma^{count}_{Age}(\sigma_{Gender=Male}(_{Age,Gender}(\tilde{\mathcal{D}})))))$	Counts the number of distinct age values for the male employees.
P7: $\hat{c} \leftarrow Lap_{\epsilon,2}(count(\sigma_{Count\in[200,m]}(\tilde{\gamma}^{count}_{Age}(_{Age}(\tilde{\mathcal{D}})))))$	Counts the number of age values having at least 200 records.

Table 2.2: Examples of Cryptϵ Program

2.4.2 Measurement operators

The measurement operators take encrypted vector of counts, $\boldsymbol{V}$ (or a single count, $\boldsymbol{c}$), as input and return noisy measurements on it in the clear. These two operators correspond to two classic DP mechanisms – Laplace mechanism and Noisy-Max [DR14a]. Both mechanisms add Laplace noise, η, scaled according to the transformations applied to $\tilde{\mathcal{D}}$.

Let the sequence of transformations applied on $\tilde{\mathcal{D}}$ to get V be $\bar{\mathcal{T}}(D) = \mathcal{T}_l(\cdots\mathcal{T}_2((\mathcal{T}_1(D))))$. The *sensitivity* of a sequence of transformations is defined as the maximum change to the output of this sequence of transformations [McS09] when changing a row in the input database, i.e., $\Delta_{\bar{\mathcal{T}}} = \max_{D,D'} \|\bar{\mathcal{T}}(D) - \bar{\mathcal{T}}(D')\|_1$ where D and D' differ in a single row. The sensitivity of $\bar{\mathcal{T}}$ can be upper bounded by the product of the stability [McS09] of these transformation operators, i.e., $\Delta_{\bar{\mathcal{T}}=(\mathcal{T}_l,\ldots,\mathcal{T}_1)} = \prod_{i=1}^{l} \Delta\mathcal{T}_i$. The transformations in Table 2.1 have a stability of 1, except for GroupByCount and GroupByCountEncoded which are 2-stable. Given ϵ and $\Delta_{\bar{\mathcal{T}}}$, we define the measurement operators:

- **Laplace** $Lap_{\epsilon,\Delta}(\mathbf{V}/\boldsymbol{c})$. This operator implements the classic Laplace mechanism [DR14a]. Given an encrypted vector, $\boldsymbol{V}$, or an encrypted scalar, $\boldsymbol{c}$, a privacy parameter ϵ and sensitivity Δ of the preceding transformations, the operator adds noise drawn from $Lap(\frac{2\Delta}{\epsilon})$ to $\boldsymbol{V}$ or $\boldsymbol{c}$ and outputs the noisy answer.

- **NoisyMax** $NoisyMax^{k}_{\epsilon,\Delta}(\mathbf{V})$. Noisy-Max is a differentially private selection mechanism [DR14a, GHIM19] to determine the top k highest valued queries. This operator takes in an encrypted vector $\boldsymbol{V}$ and adds independent Laplace noise from $Lap(\frac{2k\Delta}{\epsilon})$ to each count. The indices for the top k noisy values, $\hat{\mathcal{P}}$, are reported as the desired answer.

2.5 Implementation

In this chapter, we describe the implementation of Cryptϵ. First, we discuss our proposed technique for extending the multiplication operation of labHE to support $n > 2$ multiplicands which will be used for the CrossProduct operator. Then, we describe the implementations of Cryptϵ operators.

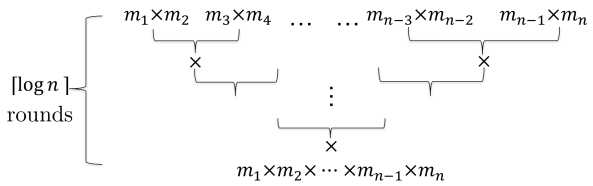


Figure 2.2: $genLabMult()$ - Batching of multiplicands for labHE

2.5.1 General n-way Multiplication for labHE

The $labMult()$ operator of a labHE scheme allows the multiplication of two ciphers. However, it cannot be used directly for a n-way muplication where $n > 2$. It is so because the "multiplication" cipher $\mathbf{e} = labMult(\mathbf{c_1}, \mathbf{c_2})$ does not have a corresponding label, i.e., it is not in the correct labHE cipher representation. Hence, we propose Algorithm 1 to generate a label τ' and a seed b' for every intermediary product of two multiplicands so that it we can do a generic n-way multiplication on the ciphers. Note that the mask r protects the value of $(m_1 \cdot m_2)$ from the CSP (Step 3) and b' hides $(m_1 \cdot m_2)$ from the AS (Step 6). For example, suppose we want to multiply the respective ciphers of 4 messages $\{m_1, m_2, m_3, m_4\} \in \mathcal{M}^4$ and obtain $\mathbf{e} = labEnc_{pk}(m_1 \cdot m_2 \cdot m_3 \cdot m_4)$. For this, the AS first generates $\mathbf{e_{1,2}} = labEnc_{pk}(m_1 \cdot m_2)$ and $\mathbf{e_{3,4}} = labEnc_{pk}(m_3 \cdot m_4)$ using Algorithm 1. Both operations can be done in parallel in just one interaction round between the AS and the CSP. In the next round, the AS can again use Algorithm 1 with inputs $\mathbf{e_{1,2}}$ and $\mathbf{e_{3,4}}$ to obtain the final answer $\mathbf{e}$. Thus for a generic $n - way$ multiplication the order of multiplication can be, in fact, parallelized as shown in Figure 2.2 to require a total of $\lceil \log n \rceil$ rounds of communication with the CSP.

2.5.2 Operator Implementation

We now summarize how Cryptϵ operators are translated to protocols that the AS and CSP can run on encrypted data.

Project $\pi_{\bar{A}}(\tilde{\boldsymbol{T}})$. The implementation of this operator simply drops off all but the attributes in $\bar{A}$ from the input table, $\tilde{\boldsymbol{T}}$, and returns the truncated table, $\tilde{\boldsymbol{T}}'$.

Filter $\sigma_\phi(\tilde{\boldsymbol{T}})$. The predicate ϕ in this operator is a conjunction of range conditions over $\bar{A}$, defined as: for a row r in input table $\tilde{\boldsymbol{T}}$, $\phi(r) = \bigwedge_{A_j \in \bar{A}} \quad (r.A_j \in V_{A_j})$, where $r.A_j$ is the value of attribute A_j in row r and $V_{A_j} \subseteq \{0, 1, \ldots, s_{A_j}\}$ (the indices for attribute values of A_j with domain size s_{A_j}).

Algorithm 1: *genLabMult* - generate label for *labMult*

Input: $\mathbf{c_1} = (a_1, d_1) = labEnc_{pk}(m_1)$ and $\mathbf{c_2} = labEnc_{pk}(m_2)$ where $a_1 = m_1 - b_1$, $d_1 = Enc_{pk}(b_1)$, $a_2 = m_2 - b_2, d_2 = Enc_{pk}(b_2)$

Output: $\mathbf{e} = labEnc_{pk}(m_1 \cdot m_2)$

AS:

1: Computes $\mathbf{e}' = labMult(\mathbf{c_1}, \mathbf{c_2}) \oplus Enc_{pk}(r)$ where r is a random mask
▷e' corresponds to $m_1 \cdot m_2 - b_1 \cdot b_2 + r$

2: Sends $\mathbf{e}', d_1, d_2$ to CSP

CSP:

3: Computes $e'' = Dec_{sk}(\mathbf{e}') + Dec_{sk}(d_1) \cdot Dec_{sk}(d_2)$
▷e'' corresponds to $m_1 \cdot m_2 + r$

4: Picks a seed σ' and label τ' and computes $b' = \mathcal{F}(\sigma', \tau')$

5: Sends $\bar{e} = (\bar{a}, d')$ to AS, where $\bar{a} = e'' - b'$ and $d' = Enc_{pk}(b')$
▷$\bar{a}$ corresponds to $m_1 \cdot m_2 + r - b'$

AS:

6: Computes true cipher $\mathbf{e} = (a', d')$ where $a' = \bar{a} - r$

First, we will show how to evaluate whether a row r satisfies $r.A_j \in V_{A_j}$. Let $\tilde{\mathbf{v}}_j$ be the encrypted one-hot-encoding of A_j, then the indicator function can be computed as

$$I_{r.A_j \in V_{A_j}} = \bigoplus_{l \in V_{A_j}} \tilde{\mathbf{v}}_j[l].$$

If the attribute of A_j in r has a value in V_{A_j}, then $I_{r.A_j \in V_{A_j}}$ equals 1; otherwise, 0.

Next, we can multiply all the indicators using $genLabMult()$ to check whether all attributes in $A_j \in \bar{A}$ of r satisfy the conditions in ϕ. Let $\bar{A} = \{A_1, \ldots, A_m\}$, then

$$\phi(r) = genLabMult(I_{A_1 \in V_{A_1}}, \ldots, I_{A_m \in V_{A_m}}).$$

Last, we update the bit of r in $\boldsymbol{B}$, i.e., $\boldsymbol{B}'[i] = labMult(\boldsymbol{B}[i], \phi(r))$, given r is the ith row in the input table. This step zeros out some additional records which were found to be extraneous by some preceding filter conditions.

Note that when the Filter transformation is applied for the very first time in a Cryptϵ program and the input predicate is conditioned on a single attribute $A \in V_A$, we can directly compute the new bit vector using $I_{r.A \in V_A}$, i.e., for the ith record r in input table $\tilde{\boldsymbol{T}}$, we have $\boldsymbol{B}'[i] = \bigoplus_{l \in V_A} \tilde{\mathbf{v}}_j[l]$. This avoids the unnecessary multiplication $labMult(\boldsymbol{B}[i], \phi(r))$.

CrossProduct $\times_{A_i, A_j \to A'}(\tilde{\boldsymbol{T}})$. This operator replaces the two attributes A_i and A_j by a single attribute A'. Given the encrypted input table $\tilde{\boldsymbol{T}}$, where all attributes are in one-hot-encoding

and encrypted, the attributes of $\tilde{\boldsymbol{T}}$ except A_i and A_j remain the same. For every row in $\tilde{\boldsymbol{T}}$, we denote the encrypted one-hot-encoding for A_i and A_j by $\tilde{\mathbf{v}}_1$ and $\tilde{\mathbf{v}}_2$. Let s_1 and s_2 be the domain sizes of A_i and A_j respectively. Then the new one-hot-encoding for A', denoted by $\tilde{\mathbf{v}}$, has a length of $s = s_1 \cdot s_2$. For $l \in \{0, 1, \ldots, s-1\}$, we have

$$\tilde{\mathbf{v}}[l] = labMult(\tilde{\mathbf{v}}_1[l/s_2], \tilde{\mathbf{v}}_2[l\%s_2]).$$

Only one bit in $\tilde{\mathbf{v}}$ for A' will be encrypted 1 and the others will be encrypted 0s. When merging more than two attributes, Cryptϵ uses the $genLabMult()$ described in Chapter 2.5.1 to speed up the computation.

Count $count(\tilde{\boldsymbol{T}})$.This operator simply adds up the bits in $\boldsymbol{B}$ corresponding to input table $\tilde{\boldsymbol{T}}$, i.e., $\bigoplus_i \boldsymbol{B}[i]$.

GroupByCount $\gamma_A^{count}(\tilde{\mathbf{T}})$. The implementations for Project, Filter and Count are reused here. First, Cryptϵ projects the input table $\tilde{\boldsymbol{T}}$ on attribute A, i.e. $\tilde{\boldsymbol{T}}_1 =_A (\tilde{\boldsymbol{T}})$. Then, Crypt$\epsilon$ loops each possible value of A. For each value v, Cryptϵ initializes a temporary $\boldsymbol{B}_v = \boldsymbol{B}$ and filters $\tilde{\boldsymbol{T}}'$ on $A = v$ to get an updated $\boldsymbol{B}'_v$. Finally, Cryptϵ outputs the number of 1s in $\boldsymbol{B}'_v$.

GroupByCountEncoded $\tilde{\gamma}_A^{count}(\tilde{\mathbf{T}})$. The implementation detail of this operator is given by Algorithm **??** and described below. First, the AS uses GroupByCount to generate the encrypted histogram, $\boldsymbol{V}$, for attribute A. Since each entry of $\mathbf{V}$ is a count of rows, its value ranges from $\{0, ..., |\tilde{\boldsymbol{T}}|\}$. The AS, then, masks $\boldsymbol{V}$ and sends it to the CSP. The purpose of this mask is to hide the true histogram from the CSP. Next, the CSP generates the encrypted one-hot-coding representation for this masked histogram $\tilde{\boldsymbol{\mathcal{V}}}$ and returns it to the AS. The AS can simply rotate $\tilde{\boldsymbol{\mathcal{V}}}[i], i \in [|V|]$ by its respective mask value $M[i]$ and get back the true encrypted histogram in one-hot-coding $\tilde{\boldsymbol{V}}$.

CountDistinct $countD(\boldsymbol{V})$. The implementation of this operator involves both AS and CSP. Given the input encrypted vector of counts $\boldsymbol{V}$ of length s, the AS first masks $\boldsymbol{V}$ to form a new encrypted vector $\mathcal{V}$ with a vector of random numbers M, i.e., for $i \in \{0, 1, \ldots, s-1\}$, $\mathcal{V}[i] = \boldsymbol{V}[i] \oplus labEnc_{pk}(M[i])$. This masked encrypted vector is then sent to CSP and decrypted by CSP to a plaintext vector $\mathcal{V}$ using the secret key.

Next, CSP generates a garbled circuit which takes (i) the mask M from the AS, and (ii) the plaintext masked vector $\mathcal{V}$ and a random number r from the CSP as the input. This circuit first removes the mask M from $\mathcal{V}$ to get V and then counts the number of non-zero entries in V, denoted by c. A masked count $c' = c + r$ is outputted by this circuit. CSP send both the circuit and the encrypted random number $labEnc_{pk}(r)$ to AS.

Algorithm 2: GroupByCountEncoded $\tilde{\gamma}_A^{count}(\tilde{\mathbf{T}})$

Input: $\tilde{\mathbf{T}}$

Output: $\tilde{V}$

AS:

1: Computes $\mathbf{V} = \gamma_A^{count}(\tilde{\boldsymbol{T}})$.

2: Masks the encrypted histogram $\mathbf{V}$ for attribute A as follows

$$\boldsymbol{\mathcal{V}}[i] = \mathbf{V}[i] \oplus labEnc_{pk}(M[i])$$
$$M[i] \in_R [m], i \in [|V|]$$

3: Sends $\boldsymbol{\mathcal{V}}$ to CSP.

CSP:

4: Decrypts $\mathcal{V}$ as $\mathcal{V}[i] = labDec_{sk}(\mathcal{V}), i \in [|V|]$.

5: Converts each entry of $\mathcal{V}$ to its corresponding one-hot-coding and encrypts it, $\tilde{\boldsymbol{\mathcal{V}}}[i] = labEnc_{pk}(\tilde{\mathcal{V}[i]}), i \in [|V|]$

6: Sends $\tilde{\boldsymbol{\mathcal{V}}}$ to AS.

AS:

7: Rotates every entry by its corresponding mask value to obtain the desired encrypted one-hot-coding $\tilde{\boldsymbol{V}}[i]$.

$$\tilde{\boldsymbol{V}}[i] = RightRotate(\tilde{\boldsymbol{\mathcal{V}}}, M[i]), i \in [|V|]$$

Last, the AS evaluates this circuit to the masked count c' and obtains the final output to this operator: $\mathbf{c} = labEnc_{pk}(c') - labEnc_{pk}(r)$.

Laplace $Lap_{\epsilon,\Delta}(\boldsymbol{V}/\boldsymbol{c})$: The Laplace operator has two phases (since both the AS and the CSP adds Laplace noise). In the first phase, the AS adds an instance of encrypted Laplace noise, $\eta_1 \sim Lap(\frac{2\Delta}{\epsilon})$, to the encrypted input to generate $\hat{\mathbf{c}}$. In the second phase, the CSP first checks whether $\sum_{i=1}^{t} \epsilon_i + \epsilon \leq \epsilon^B$ where ϵ_i represents the privacy budget used for a previously executed program, P_i (presuming a total of $t \in \mathbb{N}$ programs have been executed hitherto the details of which are logged into the CSP's public ledger). Only in the event the above check is satisfied, the CSP proceeds to decrypt $\hat{\mathbf{c}}$, and records ϵ and the current program details (description, sensitivity) in the public ledger. Next, the CSP adds a second instance of the Laplace noise, $\eta_2 \sim Lap(\frac{2\Delta}{\epsilon})$, to generate the final noisy output, $\hat{c}$, in the clear. The Laplace operator with an encrypted scalar, $\boldsymbol{V}$, as the input is implemented similarly.

NoisyMax $NoisyMax_{\epsilon,\Delta}^{k}$(Valid). The input to this operator is an encrypted vector of counts $\boldsymbol{V}$ of size s. Similar to Laplace operator, both AS and CSP are involved. First, the AS

adds to $\boldsymbol{V}$ an encrypted Laplace noise vector and a mask M, i.e., for $i \in \{0, 1, \ldots, s\}$, $\hat{\boldsymbol{V}}[i] = \boldsymbol{V}[i] \oplus labEnc_{pk}(\eta_i) \oplus M[i]$, where $\eta_i \sim Lap(\frac{2k\Delta}{\epsilon})$. This encrypted noisy, masked vector $\hat{\boldsymbol{V}}$ is then sent to the CSP.

The CSP first checks whether $\sum_{i=1}^{t} \epsilon_i + \epsilon \leq \epsilon^B$ where ϵ_i represents the privacy budget used for a previously executed program P_i (we presume that a total of $t \in \mathbb{N}$ programs have been executed hitherto the details of which are logged into the CSP's public ledger). Only in the event the above check is satisfied, the CSP proceeds to decrypt $\hat{\boldsymbol{V}}$ using the secret key, i.e., for $i \in \{0, 1, \ldots, s\}$, $\hat{V}[i] = labDec_{sk}(\hat{\boldsymbol{V}}[i])$. Next the CSP records ϵ and the current program details in the public ledger. This is followed by the CSP adding another round of Laplace noise to generate $\hat{\boldsymbol{V}}'[i] = \hat{\boldsymbol{V}}[i] \oplus labEnc_{pk}(\eta_i')$, where $\eta_i' \sim Lap(\frac{2k\Delta}{\epsilon}), i \in \{0, 1, \ldots, s\}$. (This is to ensure that as long as one of the parties is semi-honest, the output does not violate DP.) Finally, the CSP generates a garbled circuit which takes (i) the noisy, masked vector $\hat{V}$ from the CSP, and (ii) the mask M from the AS as the input. This circuit will remove the mask from $\hat{V}$ to get the noisy counts $\hat{V}'$ and find the indices of the top-k values in $\hat{V}'$.

Finally, the AS evaluates the circuit above and returns the indices as the output of this operator.

2.5.3 Classification of Cryptϵ Programs

Cryptϵ programs are grouped into three classes based on the number and type of interaction between the AS and the CSP.

Class I: Single Decrypt Interaction Programs
For releasing any result (noisy) in the clear, the AS needs to interact at least once with the CSP (via the two measurement operators) as the latter has exclusive access to the secret key. Cryptϵ programs like P1, P2 and P3 (Table 2.2) that require only a single interaction of this type fall in this class.

Class II: LabHE Multiplication Interaction Programs
Cryptϵ supports a n-way multiplication of ciphers for $n > 2$ as described in Chapter 2.5.1 which requires intermediate interactions with the CSP. Thus all Cryptϵ programs that require multiplication of more than two ciphers need interaction with the CSP. Examples include P4 and P5 (Table 2.2).

Class III: Other Interaction Programs
The GroupByCountEncoded operator requires an intermediate interaction with the CSP. The CountDistinct operator also uses a garbled circuit and hence requires interactions with the

CSP. Therefore, any program with the above two operators, like P6 and P7 (Table 2.2), requires at least two rounds of interaction.

2.6 Cryptϵ Security Analysis

In this chapter, we analysis the security guarantees of Cryptϵ in the semi-honest model using the well established simulation argument [Ode09]. Cryptϵ takes as input a DP program, P, and a privacy parameter, ϵ, and translates P into a protocol, Π, which in turn is executed by the AS and the CSP. In addition to revealing the output of the program P, Π also reveals the number of records in the dataset, $\mathcal{D}$. Let $P^{CDP}(\mathcal{D}, \epsilon/2)$ denote the random variable corresponding to the output of running P in the CDP model under $\epsilon/2$-DP (Definition 1). We make the following claims:

- The views and outputs of the AS and CSP are computationally indistinguishable from that of simulators with access to only $P^{CDP}(\mathcal{D}, \epsilon/2)$ and the total dataset size $|\mathcal{D}|$.
- For every P that satisfies $\epsilon/2$-DP (Definition 1), revealing its output (distributed identical to $P^{CDP}(\mathcal{D}, \epsilon/2)$) as well as $|\mathcal{D}|$ satisfies ϵ-bounded DP, where neighboring databases have the same size but differ in one row.
- Thus, the overall protocol satisfies computational differential privacy under the SIM-CDP model.

Now, let $P_B^{CDP}(\mathcal{D}, \epsilon)$ denote the random variable corresponding to the output of running P in the CDP model under ϵ-bounded DP such that $P_B^{CDP}(\mathcal{D}, \epsilon) \equiv (P^{CDP}(\mathcal{D}, \epsilon/2), |\mathcal{D}|)$.

Theorem 4. Let protocol Π correspond to the execution of program P in Cryptϵ. The views and outputs of the AS and the CSP are denoted as $View_1^{\Pi}(P, \mathcal{D}, \epsilon), Output_1^{\Pi}(P, \mathcal{D}, \epsilon)$ and $View_2^{\Pi}(P, \mathcal{D}, \epsilon), Output_2^{\Pi}(P, \mathcal{D}, \epsilon)$ respectively. Let $\equiv_c$ denote computational indistinguishability. There exists Probabilistic Polynomial Time (PPT) simulators, Sim_1 and Sim_2, such that:

- $Sim_1(P_B^{CDP}(\mathcal{D}, \epsilon))$ is $\equiv_c$ to $(View_1^{\Pi}(P, \mathcal{D}, \epsilon), Output^{\Pi}(P, \mathcal{D}, \epsilon))$, and
- $Sim_2(P_B^{CDP}(\mathcal{D}, \epsilon))$ is $\equiv_c$ to $(View_2^{\Pi}(P, \mathcal{D}, \epsilon), Output^{\Pi}(P, \mathcal{D}, \epsilon))$.

$Output^{\Pi}(P, \mathcal{D}, \epsilon))$ is the combined output of the two parties[1].

First, we present a proof sketch of the above theorem which is followed by the formal proof.

[1]Note that the simulators are passed a random variable $P_B^{CDP}(\mathcal{D}, \epsilon))$, i.e., the simulator is given the ability to sample from this distribution.

Proof Sketch. The main ingredient for the proof is the composition theorem [Ode09], which informally states: suppose a protocol, Π_f^g, implements functionality f and uses function g as an oracle (uses only input-output behavior of g). Assume that protocol Π_g implements g and calls to g in Π_f^g are replaced by instances of Π_g (referred to as the composite protocol). If Π_f and Π_g are correct (satisfy the above simulator definition), then the composite protocol is correct. Thus, the proof can be done in a modular fashion as long as the underlying operators are used in a blackbox manner (only the input-output behavior are used and none of the internal state are used).

Next, every Cryptϵ program expressed as a sequence of transformation operators followed by a measurement operator, satisfies $\epsilon/2$-DP (as in Definition 1). It is so because recall that the measurement operators add noise from $Lap(\frac{2\cdot\Delta}{\epsilon})$ (Chapter 2.4.2) where Δ denotes the sensitivity of P (computed w.r.t to Definition 1) [DR14a, GHIM19]. However, Cryptϵ reveals both the output of the program as well as the total size of the dataset $\mathcal{D}$. While revealing the size exactly would violate Definition 1, it does satisfy *bounded*-DP albeit with twice the privacy parameter, ϵ – changing a row in $\mathcal{D}$ is equivalent to adding a row and then removing a row.

Finally, since every program P executed on Cryptϵ satisfies ϵ-bounded DP, it follows from Theorem 4 that every execution of Cryptϵ satisfies computational DP.

Formal Proof. Cryptϵ has *nine operators* (see Table 2.1).

- NoisyMax and CountDistinct use "standard" garbled circuit construction and their security proof follows from the proof of these schemes.
- All other operators except Laplace essentially use homomorphic properties of our encryption scheme and thus there security follows from semantic-security of these scheme.
- The proof for the Laplace operator is given below.

The proof for an entire program P (which is a composition of these operators) follows from the composition theorem [Ode09, Section 7.3.1]

We will prove the theorem for the Laplace operator. In this case the views are as follows (the outputs of the two parties can simply computed from the views):

$$\begin{aligned} View_1^{\Pi}(P, \mathcal{D}, \epsilon) &= (pk, \tilde{\boldsymbol{\mathcal{D}}}, \eta_1, P(\mathcal{D}) + \eta_2 + \eta_1) \\ View_2^{\Pi}(pk, sk, P, \mathcal{D}, \epsilon) &= (\eta_2, labEnc_{pk}(P(\mathcal{D}) + \eta_1)) \end{aligned}$$

The random variables η_1 and η_2 are random variables generated according to the Laplace distribution $Lap(\frac{2\cdot\Delta}{\epsilon})$ where Δ is the program sensitivity (computed w.r.t Definition 1).

The simulators $Sim_1(z_1^B)$ (where $z_1 = (y_1, |\mathcal{D}|)$ is the random variable distributed according to $P_B^{CDP}(\mathcal{D}, \epsilon)$), y_1 being the random variable distributed as $P^{CDP}(\mathcal{D}, \epsilon/2)$) performs the following steps:

- Generates a pair of keys (pk_1, sk_1) for the encryption scheme and generates random data set $\mathcal{D}_1$ of the same size as $\mathcal{D}$ and encrypts it using pk_1 to get $\tilde{\boldsymbol{\mathcal{D}}}_1$.
- Generates η_1' according to the Laplace distribution $Lap(\frac{2\cdot\Delta}{\epsilon})$.

The output of $Sim_1(z_1)$ is $(\tilde{\boldsymbol{\mathcal{D}}}_1, \eta_1', y_1 + \eta_1')$. Recall that the view of the AS is $(\tilde{\boldsymbol{\mathcal{D}}}, \eta_1, P(\mathcal{D}) + \eta_2 + \eta_1)$. The computational indistinguishability of $\tilde{\boldsymbol{\mathcal{D}}}_1$ and $\tilde{\boldsymbol{\mathcal{D}}}$ follows from the semantic security of the encryption scheme. The tuple $(\eta_1', y_1 + \eta_1')$ has the same distribution as $(\eta_1, P(\mathcal{D}) + \eta_2 + \eta_1)$ and hence the tuples are computationally indistinguishable. Therefore, $Sim_1(z_1)$ is computational indistinguishable from $View_1^{\Pi}(P, \mathcal{D}, \epsilon)$.

The simulators $Sim_2(z_2)$ (where $z_2 = (y_2, |\mathcal{D}|)$ is the random variable distributed according to $P_B^{CDP}(\mathcal{D}, \epsilon)$), y_2 being the random variable distributed as $P^{CDP}(\mathcal{D}, \epsilon/2)$) performs the following steps:

- Generates a pair of keys (pk_2, sk_2) for our encryption scheme.
- Generates η_2' according to the Laplace distribution $Lap(\frac{2\cdot\Delta}{\epsilon})$.

The output of $Sim_2(z_2)$ is $(\eta_2', labEnc_{pk}(y_2) + \eta_2')$. By similar argument as before $Sim_2(z_2)$ is computationally indistinguishable from $View_2^{\Pi}(P, \mathcal{D}, \epsilon)$.

Corollary 1. *Protocol* Π *satisfies computational differential privacy under the* SIM-CDP *notion [MPRV09].*

Note that Theorem 4 assumes that AS and the CSP do not collude with the users (data owners). However, if the AS colludes with a subset of the users, U, then Sim_1 (Sim_2) has to be given the data corresponding to users in U as additional parameters. This presents no complications in the proof (see the proof in [GJJ+18]). If a new user u joins, their data can be encrypted and simply added to the database.

2.7 Cryptϵ Optimizations

In this chapter, we present the optimizations used by Cryptϵ.

2.7.1 DP Index Optimization

This optimization is motivated by the fact that several programs, first, filter out a large number of rows in the dataset. For instance, P5 in Table 2.2 constructs a histogram over *Age* and *Gender* on the subset of rows for which *NativeCountry* is Mexico. Cryptϵ's filter implementation retains all the rows as the AS has no way of telling whether the filter condition is satisfied. As a result, the subsequent GroupbyCount is run on the full dataset. If there were an index on *NativeCountry*, Cryptϵ could run the GroupbyCount on only the subset of rows with *NativeCountry*=Mexico. But an exact index would violate DP. Hence, we propose a DP index to bound the information leakage while improving the performance.

At a high-level, the DP index on any ordinal attribute A is constructed as follows: (1) securely sort the input encrypted database, $\tilde{\boldsymbol{\mathcal{D}}}$, on A and (2) learn a mapping, $\mathcal{F}$, from the domain of A to $[1, |\tilde{\boldsymbol{\mathcal{D}}}|]$ such that most of the rows with index less than $\mathcal{F}(v), v \in domain(A)$, have a value less than v. The secure sorting is done via the following garbled circuit that (1) inputs $\tilde{\boldsymbol{\mathcal{D}}}$ (just the records without any identifying features) and indexing attribute A from the AS (2) inputs the secret key sk from the CSP (3) decrypts and sort $\mathcal{D}$ on A (4) re-encrypt the sorted database using pk and outputs $\tilde{\boldsymbol{\mathcal{D}}}_s = labEnc_{pk}(sort(\mathcal{D}))$. The mapping, $\mathcal{F}$, must be learned under DP, and we present a method for that below. Let $P = (P_1, \ldots, P_k)$ be an equi-width partition on the sorted domain of A such that each partition (bin) contains $\frac{s_A}{k}$ consecutive domain values where s_A is the domain size of A. The index is constructed using a Cryptϵ program that firstly computes the noisy prefix counts, $\hat{V}[i] = \sum_{v \in \cup_{l=1}^{i} P_l} ct_{A,v} + \eta_i$ for $i \in [k]$, where $\eta_i \sim Lap(2 \cdot k/\epsilon_A)$ and $ct_{A,v}$ denotes the number of rows with value v for A. Next, the program uses isotonic regression [HRMS10a] on $\hat{V}$ to generate a noisy cumulative histogram $\tilde{\mathcal{C}}$ with non-decreasing counts. Thus, each prefix count in $\tilde{\mathcal{C}}$ gives an *approximate index* for the sorted database where the values of attribute A change from being in P_i to a value in P_{i+1}. When a Cryptϵ program starts with a filter $\phi = A \in [v_s, v_e]$, we compute two indices for the sorted database, i_s and i_e, as follows. Let v_s and v_e fall in partitions P_i and P_j respectively. If P_i is the first partition, then we set $i_s = 0$; otherwise set i_s to be 1 more than the $i-1$-th noisy prefix count from $\tilde{\mathcal{C}}$. Similarly, if P_j is the last partition, then we set $i_e = |\tilde{\boldsymbol{\mathcal{D}}}|$; otherwise, we set i_e to be the $j+1$-th noisy prefix count from $\tilde{\mathcal{C}}$. This gives us the DP mapping $\mathcal{F}$. We then run the program on the subset of rows in $[i_s, i_e]$. For example, in Figure 2.3, the indexing attribute with domain $\{v_1, \cdots, v_{10}\}$ has been partitioned into $k = 5$ bins and if $\phi \in [v_3, v_6]$, $i_s = \tilde{\mathcal{C}}[1] + 1 = 6$ and $i_e = \tilde{\mathcal{C}}[3] = 13$.

Lemma 1. *Let P be the program that computes the mapping $\mathcal{F}$. Let Π be the Cryptϵ protocol corresponding to the construction of the DP index. The views and outputs of the AS and the CSP are denoted as $View_1^{\Pi}(P, \mathcal{D}, \epsilon_A)$, $Output_1^{\Pi}(P, \mathcal{D}, \epsilon_A)$ and $View_2^{\Pi}(P, \mathcal{D}, \epsilon_A)$, $Output_2^{\Pi}(P, \mathcal{D}, \epsilon_A)$ respectively. There exists PPT simulators Sim_1 and Sim_2 such that:*

- $Sim_1(P_B^{CDP}(\mathcal{D}, \epsilon_A)) \equiv_c (View_1^{\Pi}(P, \mathcal{D}, \epsilon_A), Output^{\Pi}(\mathcal{D}, \epsilon_A))$, *and*
- $Sim_2(P_B^{CDP}(\mathcal{D}, \epsilon)) \equiv_c (View_2^{\Pi}(P, \mathcal{D}, \epsilon_A), Output^{\Pi}(\mathcal{D}, \epsilon_A))$.

$Output^{\Pi}(P, \mathcal{D}, \epsilon_A))$ *is the combined output of the two parties*

Proof. Recall that protocol Π consists of two parts; in the first part Π_1, the AS obtains the sorted encrypted database $\tilde{\boldsymbol{\mathcal{D}}}_s$ via a garbled circuit. Next Π_2 computes $\mathcal{F}$ via a Cryptϵ program. The security of the garbled circuit in Π_1 follows from standard approaches [LP09b]. Hence, in this chapter we concentrate on Π_2. The proof of the entire protocol Π follows from the composition theorem [[Ode09], Section 7.3.1]. The views of the servers for Π_2 are as follows:

$$View_1^{\Pi_2}(P, \mathcal{D}, \epsilon_A) = (pk, \tilde{\boldsymbol{\mathcal{D}}}, \tilde{\boldsymbol{\mathcal{D}}}_s, \mathcal{F})$$
$$View_2^{\Pi_2}(pk, sk, P, \mathcal{D}, \epsilon_A) = (\mathcal{F})$$

The simulators $Sim_1(z_1)$ (where $z_1 = (y_1, |\mathcal{D}|)$ is the random variable distributed according to $P_B^{CDP}(\mathcal{D}, \epsilon_A)$, y_1 being the random variable distributed as $P^{CDP}(\mathcal{D}, \epsilon_A/2)$) performs the following steps:

- Generates a pair of keys (pk_1, sk_1) for the encryption scheme and generates random data set $\mathcal{D}_1$ of the same size as $\mathcal{D}$ and encrypts it using pk_1 to get $\tilde{\mathcal{D}}_{\mathbf{1}}$
- Generates another random dataset $\mathcal{D}_2$ of the same size and encrypts it with pk to get $\tilde{\mathcal{D}}_{\mathbf{2}}$.

The computational indistinguishability of $\tilde{\mathcal{D}}_{\mathbf{1}}$ and $\tilde{\boldsymbol{\mathcal{D}}}$ follows directly from the semantic security of the encryption scheme. From the construction of the secure sorting algorithm, it is evident that the records in $\tilde{\boldsymbol{\mathcal{D}}}_s$ cannot be associated back with the data owners by the AS. This along with the semantic security of the encryption scheme ensures that $\tilde{\mathcal{D}}_{\mathbf{2}}$ and $\tilde{\boldsymbol{\mathcal{D}}}_s$ are computationally indistinguishable as well. The tuples $(pk_1, \tilde{\mathcal{D}}_{\mathbf{1}}, \tilde{\mathcal{D}}_{\mathbf{2}}, y_1)$ has the same distribution as $(pk, \tilde{\boldsymbol{\mathcal{D}}}, \tilde{\boldsymbol{\mathcal{D}}}_s, \mathcal{F})$ and hence are computationally indistinguishable. Therefore, $Sim_1(z_1)$ is computational indistinguishable from $View_1^{\Pi_2}(P, \mathcal{D}, \epsilon_A)$.
For the simulator $Sim_2(z_2)$ (where $z_2 = (y_2, |\mathcal{D}|)$ is the random variable distributed according to $P_B^{CDP}(\mathcal{D}, \epsilon_A)$, y_2 being the random variable distributed as $P^{CDP}(\mathcal{D}, \epsilon_A/2)$), clearly tuples (y_2) and $(\mathcal{F})$ have identical distribution. Thus, $Sim_2(z_2)$ is also computationally indistinguishable from
$View_2^{\Pi_2}(P, \mathcal{D}, \epsilon_A)$ thereby concluding our proof. □

- **Optimized feature.** This optimization speeds up the program execution by reducing the total number of rows to be processed for the program.

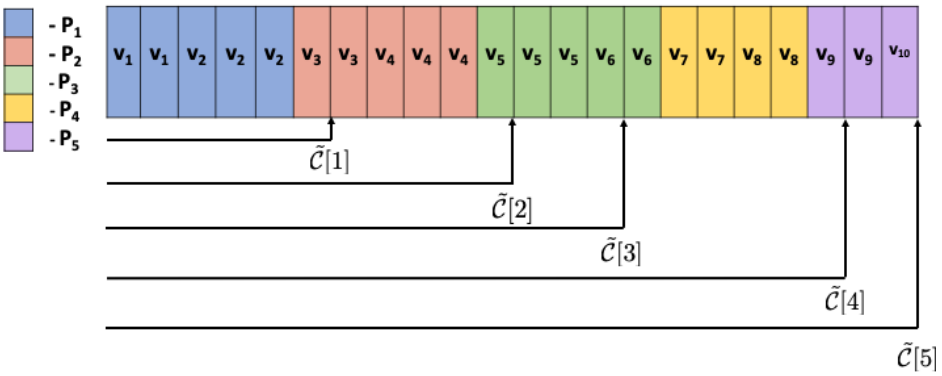


Figure 2.3: Illustrative example for DP Index

- **Trade-off.** The trade-off is a possible increase in error as some of the rows that satisfy the filter condition may not be selected due to the noisy index.

- **Privacy Cost.** Assuming the index is constructed with privacy parameter ϵ_A, the selection of a subset of rows using it will be ϵ_A-bounded DP (Lemma 1). If ϵ_L is the parameter used for the subsequent measurement primitives, then by Theorem 1, the total privacy parameter is $\epsilon_A + \epsilon_L$.

Discussion. Here, we discuss the various parameters in the construction of a DP index. The foremost parameter is the indexing attribute A which can be chosen with the help of the following two heuristics. First, A should be frequently queried so that a large number of queries can benefit from this optimization. Second, choose A such that the selectivity of the popularly queried values of A is high. This would ensure that the first selection performed alone on A will filter out the majority of the rows, reducing the intermediate dataset size to be considered for the subsequent operators. The next parameter is the fraction of the program privacy budget, ρ ($\epsilon_A = \rho \cdot \epsilon$ where ϵ is the total program privacy budget) that should be used towards building the index. The higher the value of ρ, the better is the accuracy of the index (hence better speed-up). However, the privacy budget allocated for the rest of the program decreases resulting in increased noise in the final answer. This trade-off is studied in Figures 2.5a and 2.5b in Chapter 2.8. Another parameter is the number of bins k. Finer binning gives more resolution but leads to more error due to DP noise addition. Coarser binning introduces error in indexing but has lower error due to noise. We explore this trade-off in Figures 2.5c and 2.5d. To increase accuracy we can also consider bins preceding i_s and bins succeeding i_e. This is so because, since the index is noisy, it might miss out on some rows that satisfy the filter condition. For example, in Figure 2.3, both the indices $i_s = \tilde{\mathcal{C}}[1]+1 = 6$ and $i_e = \tilde{\mathcal{C}}[3] = 14$ miss a row satisfying the filter condition $\phi = A \in [v_3, v_6]$; hence including an extra neighboring bin would reduce the error.

Thus, in order to gain in performance, the proposed DP index optimization allows some DP leakage of the data. This is in tune with the works in [MG18a,HMFS17,CCMS19a,GRR19a]. However, our work differs from earlier work in the fact that we can achieve pure DP (albeit SIM-CDP). In contrast, previous work achieved a weaker version of DP, approximate DP [BNS14], and added one-sided noise (i.e., only positive noise). One-sided noise requires addition of dummy rows in the data, and hence increases the data size. However, in our Cryptϵ programs, all the rows in the noisy set are part of the real dataset.

2.7.2 Crypto-Engineering Optimizations

DP Range Tree. If range queries are common, pre-compu-ted noisy range tree is a useful optimization. For example, building a range tree on *Age* attribute can improve the accuracy for P1 and P2 in Table 2.2. The sensitivity for such a noisy range tree is $\log s_A$ where s_A is the domain size of the attribute on which the tree is constructed. Any arbitrary range query requires access to at most $2 \log s_A$ nodes on the tree. Thus to answer all possible range queries on A, the total squared error accumulated is $O(\frac{s^2(\log s_A)^2}{\epsilon})$. In contrast for the naive case, we would have incurred error $O(\frac{s_A^3}{\epsilon})$ [HRMS10a]. Note that, if we already have a DP index on A, then the DP range tree can be considered to be a secondary index on A.

- **Optimized Feature.** The optimization reduces both execution time and expected error when executed over multiple range queries.
- **Trade-off.** The trade-off for this optimization is the storage cost of the range tree ($O(2 \cdot s_A)$).
- **Privacy Cost.** If the range tree is constructed with privacy parameter ϵ_R, then any measurement on it is post-processing. Hence, the privacy cost is ϵ_R-bounded DP.

Precomputation. The CrossProduct primitive generates the one-hot-coding of data across two attributes. However, this step is costly due to the intermediate interactions with the CSP. Hence, a useful optimization is to pre-compute the one-hot-codings for the data across a set of frequently used attributes $\bar{A}$ so that for subsequent program executions, the AS can get the desired representation via simple look-ups. For example, this benefits P3 (Table 2.2).

- **Optimized Feature.** This reduces the execution time of Cryptϵ programs. The multi-attribute one-hot-codings can be re-used for all subsequent programs.
- **Trade-off.** The trade-off is the storage cost ($\mathrm{O}(m \cdot s_{\bar{A}} = m \cdot \prod_{A \in \bar{A}} s_A)$, m = the number of data owners) incurred to store the multi-attribute one-hot-codings for $\bar{A}$.

- **Privacy Cost.** The computation is carried completely on the encrypted data, no privacy budget is expended.

(3) **Offline Processing.** For GroupByCountEncoded, the CSP needs to generate the encrypted one-hot-codings for the masked histogram. Note that the one-hot-encoding representation for any such count would simply be a vector of $(|\tilde{\boldsymbol{\mathcal{D}}}|-1)$ ciphertexts for '0', $labEnc_{pk}(0)$ and 1 ciphertext for '1', $labEnc_{pk}(1)$. Thus one useful optimization is to generate these ciphertexts offline (similar to offline generation of Beaver's multiplication triples [Bea95] used in SMC). Hence, the program execution will not be blocked by encryption.

- **Optimized Feature.** This optimization results in a reduction in the run time of Cryptϵ programs.

- **Trade-off.** A storage cost of $\mathrm{O}(m \cdot s_A)$ is incurred to store the ciphers for attribute A.

- **Privacy Cost.** The computation is carried completely on the encrypted data, no privacy budget is expended.

2.8 Experimental Evaluation

In this chapter, we describe our evaluation of Cryptϵ along two dimensions, accuracy and performance of Cryptϵ programs. Specifically, we address the following questions:

- **Q1.** Do Cryptϵ programs have significantly lower errors than that for the corresponding state-of-the-art LDP implementations? Additionally, is the accuracy of Cryptϵ programs comparable to that of the corresponding CDP implementations?

- **Q2.** Do the proposed optimizations provide substantial performance improvement over unoptimized Cryptϵ?

- **Q3.** Are Cryptϵ programs practical in terms of their execution time and do they scale well?

Evaluation Highlights:

- Cryptϵ can achieve up to $50\times$ smaller error than the corresponding LDP implementation on a data of size $\approx 30K$ (Figure 4.3). Additionally, Cryptϵ errors are at most $2\times$ more than that of the corresponding CDP implementation.

- The optimizations in Cryptϵ can improve the performance of unoptimized Cryptϵ by up to $5667\times$ (Table 2.3).

- A large class of Cryptϵ programs execute within 3.6 hours for a dataset of size 10^6, and they scale linearly with the dataset size (Figure 2.6). The AS performs majority of the work for most programs (Table 2.3).

2.8.1 Methodology

Programs. To answer the aforementioned questions, we ran the experiments on the Cryptϵ programs previously outlined in Table 2.2. Specifically, we choose P1, P3, P5 and P7 since these four cover all three classes of programs (Chapter 2.5.3) and showcase the advantages for all of the four proposed optimizations.

Dataset. We ran our experiments on the Adult dataset from the UCI repository [AN10]. The dataset is of size $32,651$. For the scaling experiments (Figure 2.6), we create toy datasets of sizes 100K and 1 million by copying over the Adult dataset.

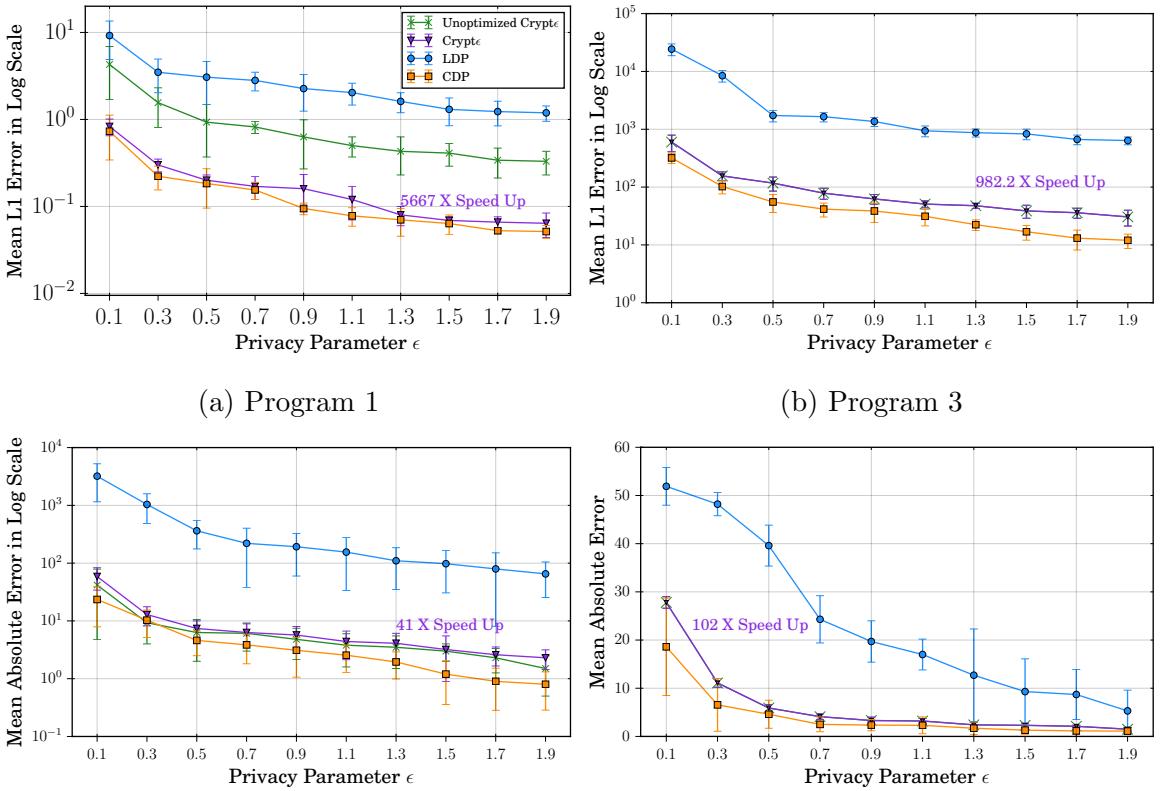
Accuracy Metrics. Programs with scalar outputs (P5, P7) use *absolute error* $|c-\hat{c}|$ where c is the true count and $\hat{c}$ is the noisy output. Programs with vector outputs (P1, P3) use the *L1 error metric* given by $Error = \sum_i |V[i] - \hat{V}[i]|, i \in [|V|]$ where V is the true vector and $\hat{V}$ is the noisy vector. We report the mean and s.t.d of error values over 10 repetitions.

Performance Metrics. We report the mean total execution time in seconds for each program, over 10 repetitions.

Configuration. We implemented Cryptϵ in Python with the garbled circuit implemented via EMP toolkit [EMP]. We use Paillier encryption scheme [Pai99]. All the experiments have been performed on the Google Cloud Platform [GCP] with the configuration c2-standard-8. For Adult dataset, Cryptϵ constructs a DP index optimization over the attribute $NativeCountry$ that benefits programs like P4 and P5. Our experiments assign 20% of the total program privacy parameter towards constructing the index and the rest is used for the remaining program execution. Cryptϵ also constructs a DP range tree over Age. This helps programs like P1, P2 and P3. This is our default Cryptϵ implementation.

2.8.2 End-to-end Accuracy Comparison

In this chapter, we evaluate **Q1** by performing a comparative analysis between the empirical accuracy of the aforementioned four Cryptϵ programs (both optimized and unoptimized) and that of the corresponding state-of-the-art LDP [WBLJ17a] and CDP (under bounded DP; specifically, using the CDP view Cryptϵ is computationally indistinguishable from as shown in Chapter 3.4.2) [DR14a] implementations.



(a) Program 1

(b) Program 3

(c) Program 5

(d) Program 7

Figure 2.4: Accuracy Analysis of Cryptϵ Programs

The first observation with respect to accuracy is that the mean error for a single frequency count for Cryptϵ is at least 50× less than that of the corresponding LDP implementation. For example, Figure 2.4b shows that for P3, $\epsilon = 0.1$ results in a mean error of 599.7 as compared to an error of 34301.02 for the corresponding LDP implementation. Similarly, P5 (Figure 2.4c) gives a mean error of only 58.7 for $\epsilon = 0.1$. In contrast, the corresponding LDP implementation has an error of 3199.96. For P1 (c.d.f on *Age*), the mean error for Cryptϵ for $\epsilon = 0.1$ is given by 0.82 while the corresponding LDP implementation has an error of 9.2. The accuracy improvement on P7 (Figure 2.4d) by Cryptϵ is less significant as compared to the other programs, because P7 outputs the number of age values ($[1 - 100]$) having 200 records. At $\epsilon = 0.1$, at least 52 age values out of 100 are reported incorrectly on whether their counts pass the threshold. Cryptϵ reduces the error almost by half. Note that the additive error for a single frequency count query in the LDP setting is at least $\Omega(\sqrt{n}/\epsilon)$, thus the error increases with dataset size. On the other hand, for Cryptϵ the error is of the order $\Theta(1/\epsilon)$, hence with increasing dataset size the relative the error improvement for Cryptϵ over that of an equivalent implementation in LDP would increase.

Table 2.3: Execution Time Analysis for Cryptϵ Programs

Time in (s)		**Program**			
		1	**3**	**5**	**7**
Unoptimized Cryptϵ	AS	1756.71	6888.23	650.78	290
	CSP	0.26	6764.64	550.34	30407.73
	Total	1756.97	13652.87	1201.12	30697.73
Cryptϵ	Total	0.31	13.9	29.21	299.5
	Speed Up ×	5667.64	982.2	41.1	102.49

For P1 (Figure 2.4a), we observe that the error of Cryptϵ is around 5× less than that of the unoptimized implementation. The reason is that P1 constructs the c.d.f over the attribute *Age* (with domain size 100) by first executing 100 range queries. Thus, if the total privacy budget for the program is ϵ, then for unoptimized Cryptϵ, each query gets a privacy parameter of just $\frac{\epsilon}{100}$. In contrast, the DP range tree is constructed with the full budget ϵ and sensitivity $\lceil \log 100 \rceil$ thereby resulting in lesser error. For P5 (Figure 2.4c) however, the unoptimized implementation has slightly better accuracy (around 1.4×) than Cryptϵ. It is because of two reasons; first, the noisy index on *NativeCountry* might miss some of the rows satisfying the filter condition (*NativeCountry*=Mexico). Second, since only 0.8% of the total privacy parameter is budgeted for the Laplace operator in the optimized program execution, this results in a higher error as compared to that of unoptimized Cryptϵ. However, this is a small cost to pay for achieving a performance gain of 41×. The optimizations for P3 (Figure 2.4b) and P7 (Figure 2.4d) work completely on the encrypted data and do not expend the privacy budget. Hence they do not hurt the program accuracy in any way.

Another observation is that for frequency counts the error of Cryptϵ is around 2× higher than that of the corresponding CDP implementation. This is intuitive because we add two instances of Laplace noise in Cryptϵ (Chapter 2.5.2). For P1, the CDP implementation also uses a range tree.

2.8.3 Performance Gain From Optimizations

In this chapter, we evaluate **Q2** (Table 2.3) by analyzing how much speed-up is brought about by the proposed optimizations in the program execution time.

DP Index. For P5, we observe from Table 2.3 that the unoptimized implementation takes around 20 minutes to run. However, a DP index over the attribute *NativeCountry* reduces the execution time to about 30s giving us a 41× speed-up. It is so because, only about 2%

of the data records satisfy $NativeCountry$=Mexico. Thus the index drastically reduces the number of records to be processed for the program.

Additionally, we study the dependency of the accuracy and execution time of P5 implemented with the DP index on three parameters – (1) fraction of privacy budget ρ used for the index (2) total number of domain partitions (bins) considered (3) number of neighboring bins considered. The default configuration for Cryptϵ presented in this chapter uses $\epsilon = 2.2$, $\rho = 0.2$, total 10 bins and considers no extra neighboring bin.

In Figure 2.5a and 2.5b we study how the mean error and execution time of the final result varies with ρ for P5. From Figure 2.5a, we observe that the mean error drops sharply from $\rho = 0.1$ to $\rho = 0.2$, stabilises till $\rho = 0.5$, and starts increasing again. This is because, at $\rho = 0.2$, the index correctly identifies almost all the records satisfying the Filter condition. However, as we keep increasing ρ, the privacy budget left for the program after Filter (Laplace operator) keeps decreasing resulting in higher error in the final answer. From Figure 2.5b, we observe that the execution time increases till $\rho = 0.5$ and then stabilizes; the reason is that the number of rows returned after $\rho = 0.5$ does not differ by much.

We plot the mean error and execution time for P5 by varying the total number of bins from 2 to 40 (domain size of *NativeCountry* is 40) in Figure 2.5c and 2.5d respectively. From Figure 2.5c, we observe that the error of P5 increases as the number of bins increase. It is so because from the computation of the prefix counts (Chapter 2.7.1), the amount of noise added increases with k (as noise is drawn from $Lap(\frac{k}{\epsilon})$). Figure 2.5d shows that the execution time decreases with k. This is intuitive because increase in k results in smaller bins, hence the number of rows included in $[i_s, i_e]$ decreases.

To avoid missing relevant rows, more bins that are adjacent to the chosen range $[i_s, i_e]$ can be considered for the subsequent operators. We increase the number of neighbouring bins from 0 to 8. As shown in Figure 2.5e, the error decreases and all the relevant rows are included when 4 neighbouring bins are considered. However, the execution time naturally increases with extra neighbouring bins as shown in Figure 2.5f.

DP Range Tree. For P1, we see from Table 2.3 that the total execution time of the unoptimized Cryptϵ implementation is about half an hour. However, using the range tree optimization reduces the execution time by 5667×. The reason behind this huge speed-up is that the time required by the AS in the optimized implementation becomes almost negligible because it simply needs to do a memory fetch to read off the answer from the pre-computed range tree.

Pre-computation. For P3, the unoptimized execution time on the dataset of 32561 records is around 4 hours (Table 2.3). This is so because the CrossProduct operator has to perform $10 \cdot 32561$ $labMult$ operations which is very time consuming. Hence, pre-computing the one-hot-codings for 2-D attribute over *Race* and *Gender* is very useful; the execution time reduces to less than a minute giving us a 982.2× speed up.

Offline Processing. The most costly operator for P7 is the GroupByCountEncoded operator since the CSP has to generate $\approx 3300K$ ciphertexts of 0 and 1 for the encrypted one-hot-codings. This results in a total execution time of about 8.5 hours in unoptimized Cryptϵ. However, by generating the ciphertexts off-line, the execution time can be reduced to just 5 minutes giving us a speed up of 102.49×.

Another important observation from Table 2.3 is that the AS performs the major chunk of the work for most program executions. This conforms with our discussion in Chapter 2.1.2.

2.8.4 Scalability

In this chapter, we evaluate **Q3** by observing the execution times of the aforementioned four Cryptϵ programs for dataset sizes up to 1 million. As seen from Figure 2.6, the longest execution time (P7) for a dataset of 1 million records is $\approx$ 3.6 hours; this shows the practical scalability of Cryptϵ. All the reported execution times are for default setting. For P1 we see that the the execution time does not change with the dataset size. This is so because once the range tree is constructed, the program execution just involves reading the answer directly from the tree followed by a decryption by the CSP. The execution time for the P3 and P7 is dominated by the $\oplus$ operation for the GroupByCount operator. The cost of $\oplus$ is linear to the data size. Hence, the execution time for P3 and P7 increases linearly with the data size. For P5, the execution time depends on the % of the records in the dataset that satisfy the condition $NativeCountry = Mexico$ (roughly this many rows are retrieved from the noisy index).

2.8.5 Communication Costs

We use Paillier encryption scheme [Pai99] in our prototype Cryptϵ. This means that each ciphertext is a random number in the group $(\mathbb{Z}/N^2\mathbb{Z})^*$ where N is a RSA moduli. Thus sending an encrypted data record entails in each data owner sending $\sum_j |domain(A_j)|$, where A_j is an attribute of the database schema, such numbers to the AS. Communication is also needed for the measurement operators and GroupByCountEncoded where the AS needs to send a ciphertext (or a vector of ciphertexts) to the CSP. Additionally operators like NoisyMax and CountDistinct need a round of communication for the garbled circuit however these circuits

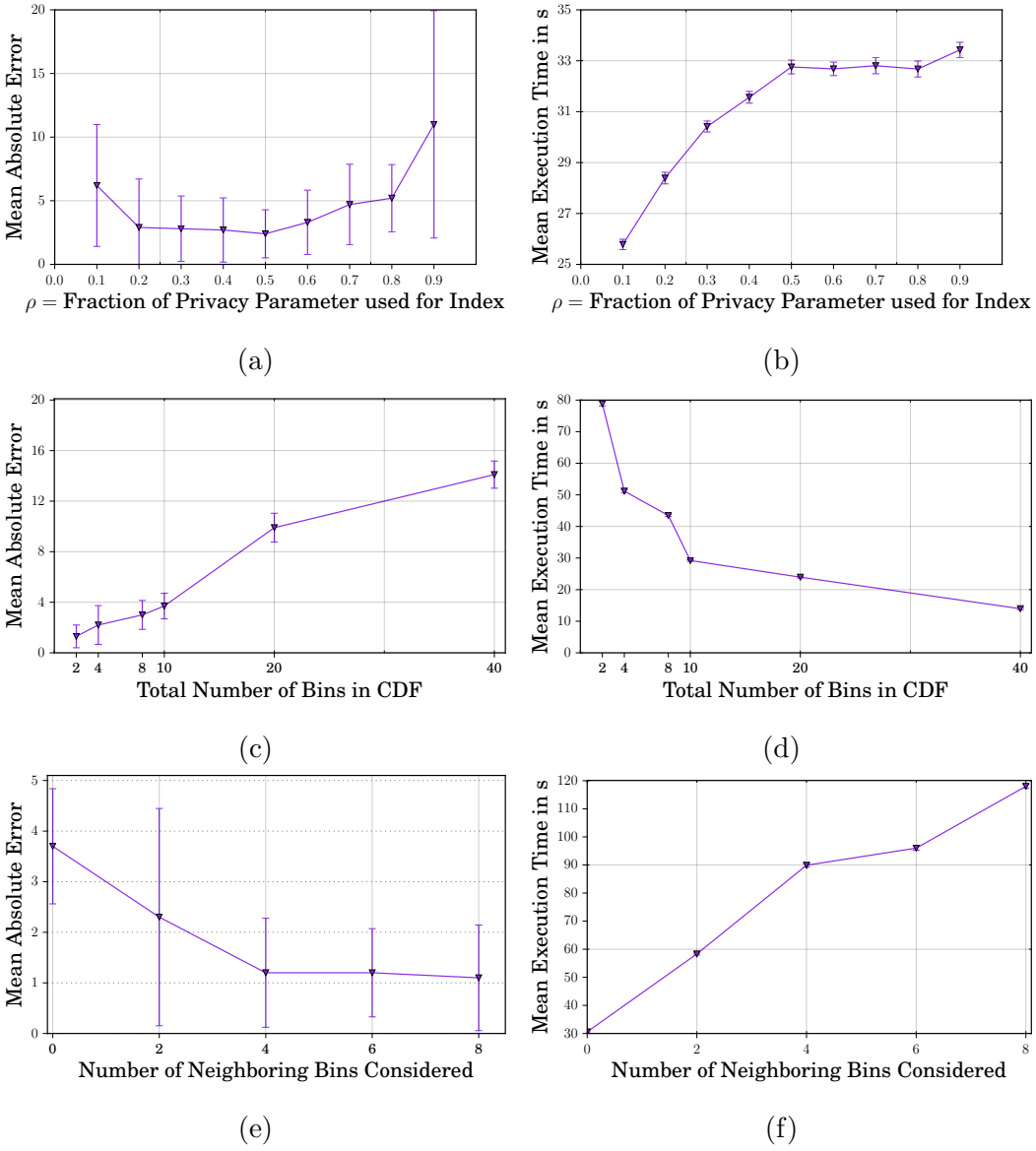


Figure 2.5: Accuracy and performance of P5 at different settings of the DP index optimization

are simple and dataset size independent. The most communication intensive operator is the CrossProduct which requires $log_2 m$ where m is the dataset size rounds of interactions. However, this can be done as a part of pre-processing and hence does not affect the actual program execution time. Hence overall, Cryptϵ programs are not communication intensive.

2.9 Extension of Cryptϵ to the Malicious Model

Here, we discuss how to extend the current Cryptϵ system to account for malicious adversaries. There can be two approaches for achieving this as follows.

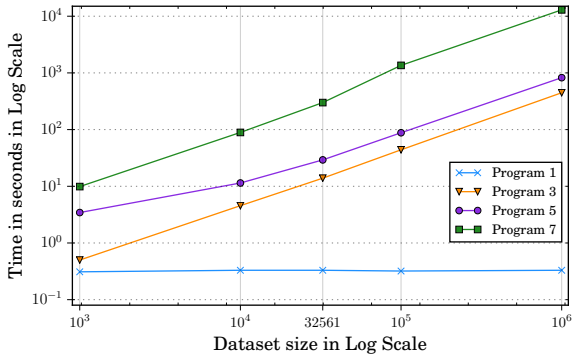


Figure 2.6: Scalability of Cryptϵ Programs

2.9.1 Approach 1

The first approach to extend Cryptϵ to the malicious threat model is to implement the CSP inside a trusted execution environment (TEE) [NIW+13,BEM+17,ABFMO08]. This ensures non-collusion (as the CSP cannot collude with the AS since its operations are vetted). The measurement operators are implemented as follows (the privacy budget over-expenditure checking remains unchanged from that in Chapter 2.5.2 and we skip re-describing it here).

Laplace $Lap_{\epsilon,\Delta}(\boldsymbol{V}/\boldsymbol{c})$. The new implementation requires only a single instance of noise addition by the CSP. The AS sends the ciphertext $\boldsymbol{c}$ to the CSP. The CSP decrypts the ciphertext, adds a copy of noise, $\eta \sim Lap(\frac{2\cdot\Delta}{\epsilon})$, and sends it to the AS.

NoisyMax $NoisyMax^{k}_{\epsilon,\Delta}(\mathbf{V})$. The new implementation works without the garbled circuit as follows. The AS sends the vector of ciphertexts, $\boldsymbol{V}$, to the CSP. The CSP computes $\tilde{V}[i] = labDecrypt_{sk}(\boldsymbol{V}[i]) + \eta[i], i \in [|\boldsymbol{V}|]$, where $\eta[i] \sim Lap(2k\Delta/\epsilon)$ and outputs the indices of the top k values of $\tilde{V}$.

Malicious AS. Recall that a Cryptϵ program, P, consists of a series of transformation operators that transform the encrypted database, $\tilde{\boldsymbol{\mathcal{D}}}$, to a ciphertext, $\boldsymbol{c}$ (or an encrypted vector, $\boldsymbol{V}$). This is followed by applying a measurement operator on $\boldsymbol{c}$ (or $\boldsymbol{V}$). Let P_1 represent the first part of the program P up to the computation of $\boldsymbol{c}$ and let P_2 represent the subsequent measurement operator (performed by the CSP inside a TEE). In the malicious model, the AS is motivated to misbehave. For example, instead of submitting the correct cipher $\boldsymbol{c} = P_1(\tilde{\boldsymbol{\mathcal{D}}})$ the AS could run a different program P' on the record of a single data owner only. Such malicious behaviour can be prevented by having the CSP validate the AS's work via zero knowledge proofs (ZKP) [Ode09] as follows (similar proof structure as prior work [NWI+13]). Specifically, the ZKP statement should prove that the AS 1) runs the correct program P_1 2) on the correct dataset $\tilde{\boldsymbol{\mathcal{D}}}$. For this, the CSP shares a random one-time MAC key, $mk_i, i \in [m]$ with each of the data owners, DO_i. Along with the encrypted

record $\tilde{\mathbf{D}}_i$, DO_i sends a Pedersen commitment [Ped92] Com_i to the one-time MAC [KL14a] on $\tilde{\mathbf{D}}_i$ and a short ZKP that the opening of this commitment is a valid one-time MAC on $\tilde{\mathbf{D}}_i$. The AS collects all the ciphertexts and proofs from the data owners and computes $\boldsymbol{c} = P_1(\tilde{\mathbf{D}}_1, \cdots, \tilde{\mathbf{D}}_m)$. Additionally, it constructs a ZKP that $\boldsymbol{c}$ is indeed the output of executing P_1 on $\tilde{\boldsymbol{\mathcal{D}}} = \{\tilde{\mathbf{D}}_1, \cdots, \tilde{\mathbf{D}}_m\}$ [CM99]. Formally, the proof statement is

$$\boldsymbol{c} = P_1(\tilde{\mathbf{D}}_1, \cdots, \tilde{\mathbf{D}}_m) \wedge \forall i \ \text{Open}(Com_i) = MAC_{mk_i}(\tilde{\mathbf{D}}_i) \tag{2.3}$$

The AS submits the ciphertext $\boldsymbol{c}$ along with all the commitments and proofs to the CSP. By validating the proofs, the CSP can guarantee $\boldsymbol{c}$ is indeed the desired ciphertext. The one-time MACs ensure that the AS did not modify or drop any of the records received from the data owners.

Efficient Proof Construction. Here we will outline an efficient construction for the aforementioned proof. First note that our setting suits that of designated verifier non-interactive zero knowledge (DV NIZK) proofs. In a DV NIZK setting, the proofs can be verified by a single designated entity (as opposed to publicly verifiable proofs) who possesses some secret key for the NIZK system. Thus in Cryptϵ, clearly the CSP can assume the role of the designated verifier. This relaxation of public verifiability leads to boast in efficiency for the proof system.

The authors in [CC17] present a framework for efficient DV NIZKs for a group-dependent language $\mathcal{L}$ where the abelian group $\mathcal{L}$ is initiated on is of order N and $\mathbb{Z}_N$ is the plaintext-space of an homomorphic cryptosystem. In other words, this framework enables proving arbitrary relations between cryptographic primitives such as Pedersen commitments or Paillier encryptions via DV NIZK proofs, in an efficient way. In what follows, we show that the proof statement given by Eq. (2.3) falls in the language $\mathcal{L}$ and consists of simple arithmetic computations.

The construction of the proof works as follows. First the AS creates linearly homomorphic commitments (we use Pederson commitments) Com_i^e on the encrypted data records $\tilde{\mathbf{D}}_i$ and proves that $Com_c = P_1(Com_1^e, \ldots, Com_m^e)$ where $Open(Com_c) = \mathbf{c}$. This is possible because of the homomorphic property of the Pederson commitment scheme; all the operations in P_1 can be applied to $\{Com_i^e\}$ instead. We use Paillier encryption scheme [Pai99] for our prototype Cryptϵ construction and hence base the rest of the discussion on it. Paillier ciphertexts are elements in the group $(\mathbb{Z}/N^2\mathbb{Z})^*$ where N is an RSA modulus. Pedersen commitments to such values can be computed as $Com = g^x h^r \in \mathbb{F}_p^*, 0 \leq r < N^2, g, h \in \mathbb{F}_p^*$, $\text{Order}(g) = \text{Order}(h) = N^2, p$ is a prime such that $p = 1 \text{mod } N^2$. This allows us to prove arithmetic relations on committed values modulo N^2. Finally, the AS just needs to show that Com_i opens to a MAC of the opening of Com_i^e. For this, the MACs we use are

built from linear hash functions $H(x) = ax + b$ [NWI+13] where the MAC signing key is the pair of random values $(a, b) \in (\mathbb{Z}/N^2\mathbb{Z})$. Proving to the CSP that the opening of Com_i is a valid MAC on the opening of Com_i^e is a simple proof of arithmetic relations. Thus, quite evidently an efficient DV NIZK proof for eq (2.3) can be supported by the framework in [CC17]. To get an idea of the execution overhead for the ZKPs, consider constructing a DV NIZK for proving that a Paillier ciphertext encrypts the products of the plaintexts of two other ciphertexts requires (this could be useful for proving the validity of our Filter operator). In the framework proposed in in [CC17], this involves $4logN$ bits of communication and the operations involve addition and multiplication of group elements. Each such operation takes order of 10^{-5}s execution time, hence for proving the above statement for 1 mil ciphers will take only a few tens of seconds.

Malicious CSP. Recall that our extension implements the CSP inside a TEE. Hence, this ensures that the validity of each of CSP's actions in the TEE can be attested to by the data owners. Since the measurement operators (P_2) are changed to be implemented completely inside the CSP, this guarantees the bounded ϵ-DP guarantee of Cryptϵ programs even under the malicious model. Additionally sending the CSP the true ciphers $\boldsymbol{c} = P_1(\tilde{\boldsymbol{\mathcal{D}}})$ also does not cause any privacy violation as it is decrypted inside the TEE.

Validity of the Data Owner's Records. The validity of the one-hot-coding of the data records, $\tilde{\mathbf{D}}_i$, submitted by the data owners DO_i can be checked as follows. Let $\tilde{\mathbf{D}}_{ij}$ represent the encrypted value for attribute A_j in one-hot-coding for DO_i. The AS selects a set of random numbers $R = \{r_k \mid k \in [|domain(A_j)|]\}$ and computes the set $P_{ij} = \{labMult(\tilde{\mathbf{D}}_{ij}[k],$
$labEnc_{pk}(r_k))\}$. Then it sends sets P_{ij} and R to the CSP who validates the record only if $|P_{ij} \cap R| = 1 \, \forall A_j$. Note that since the CSP does not have access to the index information of P_{ij} and R (since they are sets), it cannot learn the value of D_{ij}. Alternatively each data owner can provide a zero knowledge proof for $\forall j, k, \; D_{ij}[k] \in \{0, 1\} \wedge \sum_k D_{ij}[k] = 1$.

2.9.2 Approach 2

Here, we describe the second approach to extend Cryptϵ to account for a malicious adversary. For this we propose the following changes to the implementation of the measurement operator.

Laplace $Lap_{\Delta,\epsilon}(\boldsymbol{c}/\boldsymbol{V})$. Instead of having both the servers, AS and CSP add two separate instances of Laplace noise to the true answer, single instance of the Laplace noise is jointly computed via a SMC protocol [NH12, DKM+06] as follows. First the AS adds a random mask M to the encrypted input $\mathbf{c}$ to generate $\hat{\mathbf{c}}$ and sends it to the CSP. Next the CSP

generates a garbled circuit that 1) inputs two random bit strings S_1 and R_1 from the AS 2) inputs another pair of random strings S_2 and R_2 and a mask M' from the CSP 3) uses S_1 and S_2 to generate an instance of random noise, $\eta \sim Lap(\frac{2 \cdot \Delta}{\epsilon})$ using the fundamental law of transformation of probabilities 4) uses $R_1 \oplus R_2$ as the randomness for generating a Pedersen commitment for $M', Com(M')$ 4) outputs $\tilde{\mathbf{c}}' = \hat{\mathbf{c}} + labEnc_{pk}(\eta) + labEnc_{pk}(M')$, $Com(M')$, and $labEnc_{pk}(r)$ (r is the randomness used for generating $Com(M')$). The CSP sends this circuit to the AS who evaluates the circuit and sends $\tilde{\mathbf{c}}'$, $Com(M')$, and $labEnc_{pk}(r)$ back to the CSP. Now, the CSP decrypts $\tilde{\mathbf{c}}'$ and subtracts the mask M' to return $\tilde{c}' = labDec_{sk}(\tilde{\mathbf{c}}') - M'$ to the AS. Finally the AS can subtract out M to compute the answer $\tilde{c} = \tilde{c}' - M$. Note that one can create an alternative circuit to the one given above which decrypts $\tilde{\mathbf{c}}$ inside the circuit. However, decrypting Pailler ciphertexts inside the garbled circuit is costly. The circuit design given above hence results in a simpler circuit at the cost of an extra round of communication.

NoisyMax $NoisyMax^k_{\epsilon,\Delta}(\cdot)$. The AS sends a masked encrypted vector, $\hat{\boldsymbol{V}}$ to the CSP $\hat{\boldsymbol{V}}[i] = \boldsymbol{V}[i] + M[i], i \in [|\boldsymbol{V}|]$. The CSP generates a garbled circuit that 1) inputs the mask vector M, a vector of random strings S_1, a random number r and its ciphertext $\mathbf{c_r} = labEnc_{pk}(r)$ from the AS 2) inputs the secret key sk and another vector of random strings S_2 the from the CSP 3) checks if $labDec_{sk}(\mathbf{c_r}) == r$, proceed to the next steps only if the check succeeds else return -1 4) uses S_1 and S_2 to generate a vector $\eta[i] \sim Lap(\frac{2 \cdot k \cdot \Delta}{\epsilon})$ using the fundamental law of transformation of probabilities 5) computes $V[i] = labDec_{sk}(\hat{\boldsymbol{V}}[i]) + \eta[i] - M[i]$ 6) finds the indices of the top k highest values of V and outputs them. The CSP sends this circuit to the AS who evaluates it to get the answer. Note that here we are forced to decrypt Paillier ciphertexts inside the circuit because in order to ensure DP in the Noisy-Max algorithm, the noisy intermediate counts cannot be revealed.

Malicious AS. Recall that a Cryptϵ program P consists of a series of transformation operators that transforms the encrypted database $\tilde{\boldsymbol{\mathcal{D}}}$ to a ciphertext $\boldsymbol{c}$ (or a vector of ciphertexts $\boldsymbol{V}$). This is followed by applying a measurement operator on $\boldsymbol{c}$ ($\boldsymbol{V}$). Additionally, as shown in the above discussion, in the very first step of the measurement operators the AS adds a mask to $\boldsymbol{c}$ and sends the masked ciphertext $\hat{\boldsymbol{c}} = \boldsymbol{c} + M$ to the CSP. For a given program P, let P_1 represent first part of the program up till the computation of $\boldsymbol{c}$ ($\boldsymbol{V}$). The zero knowledge proof structure is very similar to the one discussed in the preceding chapter except for the following changes. Now the CSP sends a one-time MAC key k_{AS} to the AS as well and the AS sends the masked ciphertext $\hat{\boldsymbol{c}}$(or $\hat{\boldsymbol{V}}$), along with the commitments and zero knowledge proofs from the data owners and an additional commitment to the one-time MAC on the

mask M, Com_{AS} and a proof for the statement

$$\boldsymbol{c} = P_1(\tilde{\boldsymbol{\mathcal{D}}}_1, \cdots, \tilde{\boldsymbol{\mathcal{D}}}_m) \wedge \forall i \ \text{Open}(Com_i) = MAC_{mk_i}(\tilde{\boldsymbol{\mathcal{D}}}_i)$$
$$\wedge \hat{\boldsymbol{c}} = \boldsymbol{c} + labEnc_{pk}(M) \wedge \ \text{Open}(Com_{AS}) = MAC_{k_{AS}}(M)$$

The CSP proceeds with the rest of the computation only if it can validate the above proof. As long as one of the bit strings (or vectors of bit strings) in $\{S_1, S_2\}$ (and $\{R_2, R_2\}$ in case of the Laplace operator) is generated truly at random (in this case the honest CSP will generate truly random strings), the garbled circuits for the subsequent measurement operators will add the correct Laplace noise. Additionally, the mask M' prevents the AS from cheating in the last round of communication with the CSP in the protocol. It is so because, if the AS does not submit the correct ciphertext to the CSP in the last round, it will get back garbage values (thereby thwarting any privacy leakage). Hence, this prevents a malicious AS from cheating during any Cryptϵ program execution. Note that the construction of the ZKP is similar to the one discussed in the preceding chapter and can be done efficiently via the framework in [CC17].

Malicious CSP. As discussed in Chapter 2.3, the CSP maintains a public ledger with the following information

- total privacy budget ϵ^B which is publicly known
- the privacy budget ϵ used up every time the AS submits a ciphertext for decryption

Since the ledger is public, the AS can verify whether the per program reported privacy budget is correct preventing any disparities in the privacy budget allocation.

Recall that the CSP receives a masked cipher $\hat{\boldsymbol{c}}$ from the AS at the beginning of the measurement operators. The mask M protects the value of $\boldsymbol{c}$ from the CSP. We discuss the setting of a malicious CSP separately for the two measurement operators as follows.

Laplace. In case of the Laplace operator, a malicious CSP can cheat by 1) the generated garbled circuit does not correspond to the correct functionality 2) reports back incorrect decryption results. The correctness of the garbled circuit can be checked by standard mechanisms [WRK17] where the AS specifically checks that a) the circuit functionality is correct b) the circuit uses the correct value for $\hat{\mathbf{c}}$. For the second case, the CSP provides the AS with a zero knowledge proof for the following statement

$$Open(Com(M'), r) = labDec_{sk}(\tilde{\boldsymbol{c}}') - \tilde{c}'$$

NoisyMax. The garbled circuit for the NoisyMax operator is validated similarly by standard mechanisms [WRK17] where the AS checks a) whether the circuit implements the correct

functionality b) the correct value of $\hat{\boldsymbol{V}}$ is used. Note that the equality check of step (3) in the circuit validates if the CSP has provided the correct secret key sk thereby forcing it to decrypt the ciphertexts correctly.

Note that certain operators like CrossProduct, GroupByCount* and CountDistinct the involve interactions with the CSP as well but their validity can also be proven by standard techniques similar to the ones discussed above. Specifically CrossProduct and GroupByCount* can use zero knowledge proof in the framework [CC17] while the garbled circuit in CountDistinct can use [WRK17].

2.10 Related Work

Differential Privacy. Introduced by Dwork et al. [DR14a], differential privacy has enjoyed immense attention from both academia and industry in the last decade. We will discuss the recent directions in two models of differential privacy: the *centralized differential privacy* (CDP), and *local differential privacy* (LDP).

The CDP model assumes the presence of a trusted server which can aggregate all users' data before perturb the query answers. This allows the design of a complex algorithm that releases more accurate query answers than the basic DP mechanisms. For example, an important line of work in the CDP model has been towards proposing "derived" mechanisms" [CDPM18] or "revised algorithms" [BBDS12] from basic DP mechanisms (like exponential mechanism, Laplace mechanism, etc.). The design of these mechanisms leverages on specific properties of the query and the data, resulting in a better utility than the basic mechanisms. One such technique is based on data partition and aggregation [ZCX$^+$, HRMS10b, QYL13b, ACC12, CPS$^+$12, XZX$^+$12, QYL13a, XWG10, CRJ20] and is helpful in answering histogram queries. The privacy guarantees of these mechanisms can be ensured via the composition theorems and the post-processing property of differential privacy [DR14a]. We would like to extend Cryptϵ to support many of these algorithms. Recent work have also extended the applicability of DP from its traditional domain of tabular data to other modalities such as speech [ACFR20], eye-tracking data [LCFK21] and graphical models [CRJ20]. Extending Cryptϵ to support different data modalities is an interesting direction.

The notion of LDP and related ideas has been around for a while [KLN$^+$08, EGS03, War65]. Randomized response proposed by Warner in 1960s [War65] is one of the simplest LDP techniques. The recent LDP research techniques [BS15, EPK14] focus on constructing a frequency oracle that estimates the frequency of any value in the domain. However, when the domain size is large, it might be computationally infeasible to construct the histogram over the entire domain. To tackle this challenge, specialized and efficient algorithms have been proposed to

compute heavy hitters [WLJ17, FPE15], frequent itemsets [QYY+16, WLJ18], and marginal tables [CKS18, ZWL+18]. As the LDP model does not require a trusted data curator, it enjoyed significant industrial adoption, such as Google [EPK14, FPE15], Apple [Gre16], and Samsung [NXY+16].

Recently, it has been showed that augmenting randomized response mechanism with an additional layer of anonymity in the communication channel can improve the privacy guarantees [BEM+17,EFM+18,MCCJ21]. The first work to study this was PROCHLO [BEM+17] implementation by Google. PROCHLO necessitates this intermediary to be trusted, this is implemented via trusted hardware enclaves (Intel's SGX). However, as showcased by recent attacks [VBMW+18], it is notoriously difficult to design a truly secure hardware in practice. Motivated by PROCHLO, the authors in [EFM+18], present a tight upper-bound on the worst-case privacy loss. Formally, they show that any permutation invariant algorithm satisfying ϵ-LDP will satisfy $O(\epsilon\sqrt{\frac{\log(\frac{1}{\delta})}{n}}, \delta)$-CDP, where n is the data size. Cheu et al. [CSU+18] demonstrate privacy amplification by the same factor for 1-bit randomized response by using a mixnet architecture to provide the anonymity. This work also proves another important result that the power of the mixnet model lies strictly between those of the central and local models.

A parallel line of work involves efficient use of cryptographic primitives for differentially private functionalities. Agarwal et al. [AHKM18b] proposed an algorithm for computing histogram over encrypted data. Rastogi et al. [RN10] and Shi et al. [SHCGR+11] proposed algorithms that allow an untrusted aggregator to periodically estimate the sum of n users' values in a privacy preserving fashion. However, both schemes are irresilient to user failures. Chan et al. [CSS12b] tackled this issue by constructing binary interval trees over the users.

Two-Server Model. The two-server model is a popular choice for privacy preserving machine learning techniques. Researchers have proposed privacy preserving ridge regression systems with the help of a cryptographic service provider [NWI+13, GJJ+18, GSB+17a]. While the authors in [GSB+17a] use a hybrid multi-party computation scheme with a secure inner product technique, Nikolaenko et al. propose a hybrid approach in [NWI+13] by combining homomorphic encryptions and Yao's garbled circuits. Gascon et al. [GSB+16] extended the results in [NWI+13] to include vertically partitioned data and the authors in [GJJ+18] solve the problem using just linear homomorphic encryption. Zhang et al. in [MZ17] also propose secure machine learning protocols using a privacy-preserving stochastic gradient descent method. Their main contribution includes developing efficient algorithms for secure arithmetic operations on shared decimal numbers and proposing alternatives to non-linear functions such as sigmoid and softmax tailored for MPC computations. In [NIW+13] and [KKK+16] the authors solve the problem of privacy-preserving matrix factorization.

Both the papers use a hybrid approach combining homomorphic encryptions and Yao's garbled circuits for their solutions.

Homomorphic Encryption. With improvements made in implementation efficiency and new constructions developed in the recent past, there has been a surge in practicable privacy preserving solutions employing homomorphic encryptions. A lot of the aforementioned two-server models employ homomorphic encryption [NWI+13, NIW+13, GJJ+18, KKK+16]. In [HTG17, GBDL+16, CdWM+17] the authors enable neural networks to be applied to homomorphically encrypted data. Linear homomorphic encryption is used in [GJK+18] to enable privacy-preserving machine learning for ensemble methods while uses fully homomorphic encryption to approximate the coefficients of a logistic-regression model. [BCIV17] uses somewhat-homomorphic encryption scheme to compute the forecast prediction of consumer usage for smart grids.

2.11 Conclusions

We have proposed a system and programming framework, Cryptϵ, for differential privacy that achieves the constant accuracy guarantee and algorithmic expressibility of CDP without any trusted server. This is achieved via two non-colluding servers with the assistance of cryptographic primitives, specifically LHE and garbled circuits. Our proposed system Cryptϵ can execute a rich class of programs that can run efficiently by virtue of four optimizations.

Recall that currently the data analyst spells out the explicit Cryptϵ program to the AS. Thus, an interesting future work is constructing a compiler for Cryptϵ that inputs a user specified query in a high-level-language. The compiler should next formalize a Cryptϵ program expressed in terms of Cryptϵ operators with automated sensitivity analysis. Another direction is to support a larger class of programs in Cryptϵ. For example, inclusion of aggregation operators such as sum, median, average is easily achievable. Support for multi-table queries like joins would require protocols for computing sensitivity [JNS17] and data truncation [KTH+19]. Yet another direction is enabling learning algorithms on Cryptϵ; linear regression can be based on [GJJ+18] which also uses LHE and a two-server model. For this, we need to extend Cryptϵ with a new primitive for matrix multiplications. For more involved models like deep learning, DP techniques of [ACG+16] could be combined with the homomorphic encryption techniques of CryptoNet [GBDL+16].

Chapter 3

Strengthening Order Preserving Encryption with Differential Privacy

Frequent mass data breaches [dat16a, dat16b, dat17a, dat17b, dat18, dat19] of sensitive information have exposed the privacy vulnerability of data storage in practice. This has lead to a rapid development of systems that aim to protect the data while enabling statistical analysis on the dataset, both in academia [ABE+13, CLM13, GbF14, KGM+14, PRZB11] and industry [com16c, com16a, IQr16, Sch16, com16b]. Encrypted database systems that allow query computation over the encrypted data is a popular approach in this regard. Typically, such systems rely on *property-preserving encryption* schemes [BBO07, BCLO09] to enable efficient computation. *Order-preserving encryption* (OPE) [Ker15, MRS18, PLZ13a] is one such cryptographic primitive that preserves the numerical order of the plaintexts even after encryption. This allows actions like sorting, ranking, and answering range queries to be performed directly over the encrypted data [AKSX04, GZ07, HILM02, KAK10, LPL+09, LW12, LW13].

However, encrypted databases are vulnerable to inference attacks [BGC+18, DDC16, GLMP18, GLMP19a, GSB+17b, LP15, LMP18, NKW15, KPT20, KPT21, KPT19] that can reveal the plaintexts with good accuracy. Most of these attacks are inherent to any property-preserving encryption scheme – they do not leverage any weakness in the cryptographic security guarantee of the schemes but rather exploit just the *preserved property*. For example, the strongest cryptographic guarantee for OPEs (IND-FA-OCPA, see Chapter 3.2.2) informally states that *only* the order of the plaintexts will be revealed from the ciphertexts. However, inference attacks [GLMP18, GLMP19a, GSB+17b] can be carried out by leveraging only this ordering information. The basic principle of these attacks is to use auxiliary information to estimate the plaintext distribution and then correlate it with the ciphertexts based on the preserved property [FVY+17].

Differential privacy (DP) has emerged as the de-facto standard for data privacy and is an information theoretic guarantee that provides a rigorous guarantee of privacy for individuals

in a dataset regardless of an adversary's auxiliary knowledge [TSD20]. An additional appealing property of DP is that any post-processing computation, such as inference attacks, performed on the noisy output of a DP algorithm does not incur additional privacy loss.

In this work, we ask the following question:

> *Is it possible to leverage the properties of DP for providing a formal security guarantee for OPEs even in the face of inference attacks?*

To this end, we propose a novel *differentially private order preserving encryption* scheme, OPϵ. Recall that standard OPE schemes are designed to reveal nothing but the order of the plaintexts. Our proposed scheme, OPϵ, ensures that this leakage of order is differentially private. In other words, the cryptographic guaratantee of OPEs is strengthened with a layer of DP guarantee (specifically, a relaxed definition of DP as discussed in the following paragraph). As a result, even if the cryptographic security guarantee of standard OPEs proves to be inadequate (in the face of inference attacks), the DP guarantee would continue to hold true. Intuitively, the reason behind is DP's resilience to post-processing computations as discussed above. To our best knowledge, *this is the first work to combine DP with a property-preserving encryption scheme.*

3.1 Brief Overview of Key Ideas

The standard DP guarantee requires any two pairs of input data to be indistinguishable from each other (Chapter 3.2.1) and is generally catered towards answering statistical queries over the entire dataset. However, in our setting we require the output of the DP mechanism to retain some of the ordinal characteristics of its input – the standard DP guarantee is not directly applicable to this case. Hence, we opt for a natural relaxation of DP– only pairs of data points that are "close" to each other should be indistinguishable. Specifically, the privacy guarantee is heterogeneous and degrades linearly with the ℓ_1-distance between a pair of data points. It is denoted by ϵ-dLDP (or ϵ-dDP in the central model of DP, Chapter 3.2). This relaxation is along the lines of d_χ-privacy [CABP13] and is amenable to many practical settings. For instance, consider a dataset of annual sale figures of clothing firms. The information whether a firm is a top selling or a mid-range one is less sensitive than its actual sales figures. Similarly, for an age dataset, whether a person is young or middle-aged is less sensitive than their actual age.

DP guarantee inherently requires randomization – this entails an inevitable loss of utility, i.e., some pairs of output might not preserve the correct order of their respective inputs. In order to reduce the instances of such pairs, OPϵ offers the flexibility of preserving only

a partial order of the plaintexts. Specifically, a (user specified) partition is defined on the input domain and the preserved order is expected at the granularity of this partition. The output domain is defined by a numeric encoding over the intervals of the partition and all the elements belonging to the same interval are mapped to the corresponding encoding for the interval (with high probability). Due to the linear dependence of the DP guarantee (and consequently, the ratio of output probabilities) on the distance between the pair of inputs, lower is the number of intervals in the partition, higher is the probability of outputting the correct encoding in general (Chapter 3.3.2 and Chapter 3.8.2). OPϵ preserves the order over this encoding. The reason why this results in better utility for encrypted databases is illustrated by the following example. The typical usecase for OPE encrypted databases is retrieving a set of records from the outsourced database that belong to a queried range. Suppose a querier asks for a range query $[a, b]$ and let $\mathcal{P}$ be a partition that covers the range with k intervals $\{[s_1, e_1], \cdots, [s_k, e_k]\}$ such that $s_1 < a < e_1$ and $s_k < b < e_k$. A database system encrypted under OPϵ and instantiated with the partition $\mathcal{P}$ will return all the records that are noisily mapped to the range $[s_1, e_k]$ (since the order is preserved at the granularity of $\mathcal{P}$). Thus, the querier has to pay a processing overhead of fetching extra records, i.e., the records that belong to the ranges $\{[s_1, a-1], [b+1, e_k]\}$. However, if $k < b - a$, then with high probability it would receive all the correct records in $[a, b]$ which can be decrypted and verified (Chapter 3.6). To this end, we first propose a new primitive, OPϵc, that enables *order preserving encoding* under ϵ-dLDP. The encryption scheme, OPϵ, is then constructed using the OPϵc primitive and a OPE (Chapter 3.4). Beyond OPEs, the OPϵc primitive can be used as a building block for other secure computation that require ordering, such as secure sorting or order-revealing encryptions (Chapter 3.9). Additionally, OPϵc can be of independent interest for LDP in answering a variety of queries, such as ordinal queries, frequency and mean estimation (Chapter 3.7).

In what follows, we answer some key questions pertinent to our work that a reader might have at this point.

Q1. *What is the advantage of a OPϵ scheme over just OPϵc primitive or a OPE scheme?*
A. OPϵ satisfies a new security guarantee, ϵ-IND-FA-OCPA, (see Chapter 3.4.2) that bolsters the cryptographic guarantee of a OPE scheme (IND-FA-OCPA) with a layer of ϵ-dDP guarantee. As a result, OPϵ enjoys *strictly stronger security* than both OPϵc primitive (ϵ-dDP) and OPE (IND-FA-OCPA).

Q2. *What are the security implications of OPϵ in the face of inference attacks?*
A. In the very least, OPϵ *rigorously limits the accuracy of inference attacks for every record for all adversaries* (Theorem 9, Chapter 3.5). In other words, OPϵ guarantees that none

of the attacks can infer the value of any record beyond a certain accuracy that is allowed by the dLDP guarantee. For instance, for an age dataset and an adversary with real-world auxiliary knowledge, no inference attack in the snapshot model can distinguish between two age values (x, x') such that $|x - x'| \leq 8$ for $\epsilon = 0.1$ (Chapter 3.8.2).

Q3. *How is OPϵ's utility (accuracy of range queries)?*
A. We present a construction for the OPϵc primitive (and hence, OPϵ) and our experimental results on four real-world datasets demonstrate its practicality for real-world use (Chapter 4.6). Specifically, OPϵ misses only 4 in every $10K$ correct records on average for a dataset of size $\sim 732K$ with an attribute of domain size $18K$ and $\epsilon = 1$. The overhead of processing extra records is also low – the average number of extra records returned is just 0.3% of the total dataset size.

Q4. *When to use OPϵ?*
A. As discussed above, OPϵ gives a strictly stronger guarantee than any OPE scheme (even in the face of inference attacks) with almost no extra performance overhead (Chapter 3.6). Additionally, it is backward compatible with any encrypted database that is already using a OPE scheme (satisfying IND-FA-OCPA, see Chapter 3.6). Hence, OPϵ could be used for secure data analytics in settings where (1) the ϵ-dDP guarantee is acceptable, i.e, the main security concern is preventing the distinction between input values close to each other (such as the examples discussed above) and (2) the application can tolerate a small loss in utility. Specifically in such settings, replacing encrypted databases with an encryption under OPϵ would give a strictly stronger security guarantee against all attacks with *nominal change in infrastructure or performance* – a win-win situation.

3.2 Background

3.2.1 Differential Privacy

As mentioned in Chapter 2, there are two popular models of differential privacy, *local* and *central*. The LDP guarantee is formally defined as follows.

Definition 3 (Local Differential Privacy, LDP)**.** *A randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is ϵ-LDP if for any pair of private values $x, x' \in \mathcal{X}$ and any subset of output, $\mathcal{T} \subseteq \mathcal{Y}$*

$$\Pr\big[\mathcal{M}(x) \in \mathcal{T}\big] \leq e^{\epsilon} \cdot \Pr\big[\mathcal{M}(x') \in \mathcal{T}\big] \tag{3.1}$$

ϵ-LDP guarantees the same level of protection for all pairs of private values. However, as discussed in the preceding chapter, in this dissertation we use an extension of LDP which

uses the ℓ_1 distance between a pair of values to customize heterogeneous (different levels of) privacy guarantees for different pairs of private values.

Definition 4 (Distance-based Local Differential Privacy, dLDP)**.** *A randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is ϵ-distance based locally differentially private (or ϵ-dLDP), if for any pair of private values $x, x' \in \mathcal{X}$ and any subset of output $\mathcal{T} \subseteq \mathcal{Y}$,*

$$\Pr[\mathcal{M}(x) \in \mathcal{T}] \leq e^{\epsilon|x-x'|} \cdot \Pr[\mathcal{M}(x') \in \mathcal{T}] \tag{3.2}$$

The above definition is equivalent to the notion of metric-based LDP [ACPP18, CABP13] where the metric used is ℓ_1-norm.

Definition 1 in Chapter 2.2.1 refers to the central differential privacy (CDP) model. We re-iterate it here for the reader's conveience.

Definition 5 (Central Differential Privacy, CDP)**.** *A randomized algorithm $\mathcal{M} : \mathcal{X}^n \mapsto \mathcal{Y}$ satisfies ϵ-differential privacy (ϵ-DP) if for all $\mathcal{T} \subseteq \mathcal{Y}$ and for all adjacent datasets $X, X' \in \mathcal{X}^n$ it holds that*

$$\Pr[\mathcal{M}(X) \in \mathcal{T}] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(X') \in \mathcal{T}] \tag{3.3}$$

The notion of adjacent inputs is application-dependent, and typically means that X and X' differ in a single element (corresponding to a single individual). Particularly in our setting, the equivalent definition of the distance based relaxation of differential privacy in the CDP model is given as follows.

Definition 6 (Distance-based Central Differential Privacy, dDP)**.** *A randomized algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{Y}$ is ϵ-distance based centrally differentially private (or ϵ-dDP), if for any pair of datasets X and X' such that they differ in a single element, x_i and x'_i, and any subset of output $\mathcal{T} \subseteq \mathcal{Y}$,*

$$\Pr\big[\mathcal{M}(X) \in \mathcal{T}\big] \leq e^{\epsilon|x_i-x'_i|} \cdot \Pr\big[\mathcal{M}(X') \in \mathcal{T}\big] \tag{3.4}$$

We define X and X', as described above, to be **t-adjacent** where $t \geq |x_i - x'_i|$, i.e., the differing elements differ by *at most* t. Trivially, any pair of t-adjacent datasets are also t'-adjacent for $t' > t$.
Next, we formalize the resilience of dLDP (and dDP) to post-processing computations.

Theorem 5 (Post-Processing [DR14b])**.** *Let $\mathcal{M} : \mathcal{X} \mapsto \mathcal{Y}$ ($\mathcal{M} : \mathcal{X}^n \mapsto \mathcal{Y}$) be a ϵ-dLDP (dDP) algorithm. Let $g : \mathcal{Y} \mapsto \mathcal{Y}'$ be any randomized mapping. Then $g \circ \mathcal{M}$ is also ϵ-dLDP (dDP).*

3.2.2 Order Preserving Encryption

Here, we discuss the necessary definitions for OPEs.

Definition 7 (Order Preserving Encryption [MRS18])**.** *An order preserving encryption (OPE) scheme $\mathcal{E} = \langle K, E, D \rangle$ is a tuple of probabilistic polynomial time (PPT) algorithms:*

- *Key Generation (K). The key generation algorithm takes as input a security parameter κ and outputs a secret key (or state) S as $S \leftarrow K(1^\kappa)$.*
- *Encryption (E). Let $X = \langle x_1, \cdots, x_n \rangle$ be an input dataset. The encryption algorithm takes as input a secret key S, a plaintext $x \in X$, and an order Γ (any permutation of $\{1, \cdots, n\}$). It outputs a new key S' and a ciphertext y as $(S', y) \leftarrow E(S, x, \Gamma)$.*
- *Decryption (D). Decryption recovers the plaintext x from the ciphertext y using the secret key S, $x \leftarrow D(S, y)$.*

Additionally, we have

- *Correctness Property. $x \leftarrow D\big(E(S, x, \Gamma)\big)$, $\forall S, \forall x, \forall \Gamma$*
- *Order Preserving Property. $x > x' \implies y > y'$, $\forall x, x'$ where y (y') is the ciphertext corresponding to the plaintext x (x')*

The role of Γ in the above definition is discussed later. The strongest formal guarantee for a OPE scheme is *indistinguishability against frequency-analyzing ordered chosen plaintext attacks* (IND-FA-OCPA). We present two definitions in connection to this starting with the notion of *randomized orders* as defined by Kerschbaum [Ker15].

Definition 8. *(Randomized Order [Ker15]) Let $X = \langle x_1, \cdots, x_n \rangle$ be a dataset. An order $\Gamma = \langle \gamma_1, \cdots, \gamma_n \rangle$, where $\gamma_i \in [n]$ and $i \neq j \implies \gamma_i \neq \gamma_j$, for all i, j, of dataset X, is defined to be a randomized order if it holds that*

$$\forall i, j \ (x_i > x_j \implies \gamma_i > \gamma_j) \wedge (\gamma_i > \gamma_j \implies x_i \geq x_j)$$

For a plaintext dataset X of size n, a randomized order, Γ, is a permutation of the plaintext indices $\{1, \cdots, n\}$ such that its inverse, Γ^{-1}, gives a sorted version of X. This is best explained by an example: let $X = \langle 9, 40, 15, 76, 15, 76 \rangle$ be a dataset of size 6. A randomized order for X can be either of $\Gamma_1 = \langle 1, 4, 2, 5, 3, 6 \rangle$, $\Gamma_2 = \langle 1, 4, 3, 5, 2, 6 \rangle$, $\Gamma_3 = \langle 1, 4, 2, 6, 3, 5 \rangle$ and $\Gamma_4 = \langle 1, 4, 3, 6, 2, 5 \rangle$. It is so because the order of the two instances of 76 and 15 does not matter a sorted version of X.

Definition 9 (IND-FA-OCPA [MRS18,Ker15]). *An order-preserving encryption scheme* $\mathcal{E} = (K, E, D)$ *has indistinguishable ciphertexts under frequency-analyzing ordered chosen plaintext attacks if for any* PPT *adversary* $\mathcal{A}_{PPT}$*:*

$$\left|\Pr[\mathcal{G}^{\mathcal{A}_{PPT}}_{\text{FA-OCPA}}(\kappa, 1) = 1] - \Pr[\mathcal{G}^{\mathcal{A}_{PPT}}_{\text{FA-OCPA}}(\kappa, 0) = 1]\right| \leq negl(\kappa) \tag{3.5}$$

where κ *is a security parameter,* $negl(\cdot)$ *denotes a negligible function and* $\mathcal{G}^{\mathcal{A}_{PPT}}_{\text{FA-OCPA}}(\kappa, b)$ *is the random variable denoting* $\mathcal{A}_{PPT}$*'s output for the following game:*

Game $\mathcal{G}^{\mathcal{A}_{PPT}}_{\text{FA-OCPA}}(\kappa, b)$

1. $(X_0, X_1) \leftarrow \mathcal{A}_{PPT}$ *where* $|X_0| = |X_1| = n$ *and* X_0 *and* X_1 *have at least one common randomized order*
2. *Select* Γ^* *uniformly at random from the common randomized orders of* X_0, X_1
3. $S_0 \leftarrow K(1^\kappa)$
4. *For* $\forall i \in [n]$*, run* $(S_i, y_{b,i}) \leftarrow E(S_{i-1}, x_{b,i}, \Gamma^*)$
5. $b' \leftarrow \mathcal{A}_{PPT}(y_{b,1}, \cdots, y_{b,n})$ *where* b' *is* $\mathcal{A}_{PPT}$*'s guess for* b

$\mathcal{A}_{PPT}$ *is said to win the above game iff* $b = b'$*.*

Informally, this guarantee implies that nothing other than the order of the plaintexts, not even the frequency, is revealed from the ciphertexts. Stated otherwise, the ciphertexts only leak a randomized order of the plaintexts (randomized orders do not contain any frequency information since each value always occurs exactly once) which is determined by the input order Γ in Definition 7. In fact, if Γ itself happens to be a randomized order of the input X then, the randomized order leaked by the corresponding ciphertexts is guaranteed to be Γ. For example, for $X = \langle 9, 40, 15, 76, 15, 76\rangle$ and $\Gamma = \langle 1, 4, 2, 5, 3, 6\rangle$, we have $y_1 < y_3 < y_5 < y_2 < y_4 < y_6$ (y_i denotes the corresponding ciphertext for x_i and $\Gamma^{-1} = \langle 1, 3, 5, 2, 4, 6\rangle$). Thus, the IND-FA-OCPA guarantee ensures that two datasets with a common randomized order – but different plaintext frequencies – are indistinguishable. For example, in the aforementioned game $\mathcal{G}^{\mathcal{A}_{\text{PPT}}}_{\text{FA-OCPA}}(\cdot)$, $\mathcal{A}_{\text{PPT}}$ would fail to distinguish between the plaintext datasets $X_0 = \langle 9, 40, 15, 76, 15, 76\rangle$ and $X_1 = \langle 22, 94, 23, 94, 36, 94\rangle$ both of which share the randomized order $\Gamma^* = \langle 1, 4, 2, 5, 3, 6\rangle$.

Note. Although the notion of IND-FA-OCPA was first introduced by Kerschbaum et al. [Ker15], the proposed definition suffered from a subtle flaw which was subsequently rectified by Maffei et al. [MRS18]. The above definition, hence, follows from the one in [MRS18]

(denoted by in IND-FA-OCPA* in [MRS18]). Additionally, Definition 7 in our paper corresponds to the notion of *augmented* order-preserving encryption scheme (denoted by OPE* in [MRS18]) which is crucial for the above security definition. The augmented OPE scheme is in fact a generalization of the standard OPE scheme (the only difference being the encryption algorithm E has an additional input, Γ).

3.3 ϵ-dLDP Order Preserving Encoding (OPϵc)

In this chapter, we discuss our proposed primitive – ϵ-dLDP *order preserving encoding*, OPϵc. First, we define the OPϵc primitive and its construction. Next, we describe how to use the OPϵc primitive to answer queries in the LDP setting.

Notations. Here, we introduce the necessary notations. $[n], n \in \mathbb{N}$ denotes the set $\{1, 2, \cdots, n-1, n\}$. If $\mathcal{X} = [s, e]$ is an input domain, then a k-partition $\mathcal{P}$ on $\mathcal{X}$ denotes a set of k non-overlapping intervals $\mathcal{X}_i = (s_i, e_i]$ [1] , $s_{j+1} = e_j, i \in [k], j \in [k-1]$ such that $\bigcup_{i=1}^{k} \mathcal{X}_i = \mathcal{X}$. For example, for $\mathcal{X} = [1, 100]$, $\mathcal{P} = \{[1, 10], (10, 20], \cdots, (90, 100]\}$ denotes a 10-partition. Let $\hat{\mathcal{X}}$ denote the domain of partitions defined over $\mathcal{X}$. Additionally, let $\mathcal{O} = \{o_1, \cdots, o_k\}, o_i < o_{i+1}, i \in [k-1]$ represent the output domain where o_i is the corresponding encoding for the interval $\mathcal{X}_i$ and let $\mathcal{P}(x) = o_i$ denote that $x \in \mathcal{X}_i$. Referring back to our example, if $\mathcal{O} = \{1, 2, \cdots, 10\}$, then $\mathcal{P}(45) = 5$.

3.3.1 Definition of OPϵc

OPϵc is a randomised mechanism that encodes its input while maintaining some of its ordinality.

Definition 10 (ϵ-dLDP Order Preserving Encoding, OPϵc)**.** *For a given k-partition $\mathcal{P} \in \hat{\mathcal{X}}$, a ϵ-dLDP order preserving encoding scheme, $OP\epsilon c : \mathcal{X} \times \hat{\mathcal{X}} \times \mathbb{R}_{>0} \mapsto \mathcal{O}$ is a randomized mechanism such that*

1. $k = |\mathcal{O}|, k \leq |\mathcal{X}|$
2. *For all $x \in \mathcal{X}$ and $o' \in \mathcal{O} \setminus \mathcal{T}_x$ where* $\mathcal{T}_x = \begin{cases} \{o_1, o_2\} & \text{if } \mathcal{P}(x) = o_1 \\ \{o_{k-1}, o_k\} & \text{if } \mathcal{P}(x) = o_k \\ \{o_{i-1}, o_i, o_{i+1}\} & \text{otherwise} \end{cases}$

   $\exists o \in \mathcal{T}_x$ such that,

$$\Pr\big[OP\epsilon c(x, \mathcal{P}, \epsilon) = o\big] > \Pr\big[OP\epsilon c(x, \mathcal{P}, \epsilon) = o'\big] \tag{3.6}$$

[1] The first interval, $\mathcal{X}_1 = [s_1, e_1]$, is a closed interval.

3. For all $x, x' \in \mathcal{X}, o \in \mathcal{O}$, we have

$$\Pr\big[\mathit{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = o\big] \leq e^{\epsilon|x-x'|} \cdot \Pr\big[\mathit{OP}\epsilon\mathsf{c}(x', \mathcal{P}, \epsilon) = o\big]$$

The first property in the above definition signifies the flexibility of the $\mathsf{OP}\epsilon\mathsf{c}$ primitive to provide only a *partial* ordering guarantee. For instance, in our above example $k = 10 < |\mathcal{X}| = 100$. Thus, $\mathcal{P}$ acts as a utility parameter – it determines the granularity at which the ordering information is maintained by the encoding (this is independent of the privacy-accuracy trade-off arising from the choice of ϵ). For example, for the same value of ϵ and $\mathcal{X} = [1, 100]$, $\mathcal{P} = \{[1, 10], (10, 20], \cdots, (90, 100]\}$ gives better utility than $\mathcal{P}' = \{[1, 33], (33, 66], (66, 100]\}$ since the former preserves the ordering information at a finer granularity. $\mathcal{P} = \mathcal{O} = \mathcal{X}$ denotes the default case where effectively no partition is defined on the input domain and $\mathcal{P}(x) = x, x \in \mathcal{X}$ trivially. We discuss the significance of the parameter $\mathcal{P}$ in Chapter 3.6.

Due to randomization (required for to the dLDP guarantee), $\mathsf{OP}\epsilon\mathsf{c}$ is bound to incur some errors in the resulting numerical ordering of its outputs. To this end, the second property guarantees that the noisy output is most likely to be the either the correct one or the ones immediately next to it. For instance, for the aforementioned example, $\mathsf{OP}\epsilon\mathsf{c}(45, \mathcal{P}, \epsilon)$ is most likely to fall in $\{4, 5, 6\}$. This ensures that the noisy outputs still retain sufficient ordinal characteristics of the corresponding inputs. Note that the actual value of the encodings in $\mathcal{O}$ does not matter at all as long as the ordinal constraint $o_i < o_{i+1}, i \in [k-1]$ is maintained. For instance for $\mathcal{P} = \{[1, 10], (10, 20], \cdots, (90, 100]\}$, $\mathcal{O} = \{1, 2, 3, 4, 5, 6, 7\ , 8, 9, 10\}$, $\mathcal{O}' = \{5, 15, 25, 35, 45, 55, 65, 75, 85, 95\}$ and $\mathcal{O}'' = \{81, 99, 120, 150, 234, 345, 400, 432, 536, 637\}$ are all valid.

Finally, the third property ensures that the primitive satisfies ϵ-dLDP. Note that $\epsilon = \infty$ represents the trivial case $\mathsf{OP}\epsilon\mathsf{c}(X, \mathcal{P}, \infty) = \mathcal{P}(X)$. %

3.3.2 Construction of $\mathsf{OP}\epsilon\mathsf{c}$

In this chapter, we describe a construction for the $\mathsf{OP}\epsilon\mathsf{c}$ primitive (Algorithm 3). The algorithm is divided into two stages. In Stage I (Steps 1-3), it computes the central tendency (a typical value for a distribution) [R.B84], $d_i, i \in [k]$, of each of the intervals of the given k-partition $\mathcal{P}$. Specifically, we use weighted median [CLRS09] as our measure for the central tendency where the weights are determined by a prior on the input data distribution, $\mathcal{D}$. This maximizes the expected number of inputs that are mapped to the correct encoding, i.e., x is mapped to $\mathcal{P}(x)$. In the context of encrypted databases, the data owner has access to the entire dataset in the clear (Chapter 3.6). Hence, they can compute the exact input distribution, $\mathcal{D}$, and use it to instantiate the $\mathsf{OP}\epsilon\mathsf{c}$ primitive (for $\mathsf{OP}\epsilon$). In the LDP setting, $\mathcal{D}$ can be estimated from domain knowledge or auxiliary datasets. In the event such a prior is not available, $\mathcal{D}$ is assumed to be the uniform distribution (d_i is the median).

Algorithm 3: Construction of $\mathsf{OP}\epsilon\mathsf{c}$

Setup Parameters: $\mathcal{D}$ - Prior input distribution over $\mathcal{X}$, its default value is the uniform distribution;
$\mathcal{O}$ - Output domain $\{o_1, \cdots, o_k\}$

Input: x - Number to be encoded via $\mathsf{OP}\epsilon\mathsf{c}$; ϵ - Privacy budget;
$\mathcal{P}$ - A k-partition $\{[s_1, e_1], \cdots, (s_k, e_k]\}$ over $\mathcal{X}$

Output: o - Output encoding;

Stage I: Computation of central tendency for each interval

1: **for** $i \in [k]$
2: d_i = Weighted median of $(s_i, e_i]$ where $\mathcal{D}$ gives the corresponding weights
3: **end for**

Stage II: Computation of the output probability distributions

4: **for** $\mathbf{x} \in \mathcal{X}$:
5: **for** $i \in [k]$
6:

$$p_{\mathbf{x},i} = \frac{e^{-|\mathbf{x}-d_i|\cdot\epsilon/2}}{\sum_{j=1}^{k} e^{-|\mathbf{x}-d_j|\cdot\epsilon/2}} \quad \triangleright\ p_{\mathbf{x},i} = \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(\mathbf{x}, \mathcal{P}, \epsilon) = o_i\big]$$

7: **end for**
8: $p_{\mathbf{x}} = \{p_{\mathbf{x},1}, \cdots, p_{\mathbf{x},k}\}$ ▷ Encoding (output) probability distribution for $\mathbf{x}$
9: **end for**
10: $o \sim p_x$ ▷ Encoding drawn at random from the distribution p_x
11: **Return** o

In Stage II (Steps 4-9), the encoding probability distributions are computed such that the probability of x outputting the i-th encoding, o_i, is inversely proportional to its distance from the i-th central tendency, d_i. Specifically, we use a variant of the classic exponential mechanism [GTT+19, DR14b] (Step 6).

Illustration of Algorithm 3. Here, we illustrate Algorithm 3 with an example. We illustrate the algorithm with the following example. Consider a partition $\mathcal{P} = \{[1, 20], [21, 80], [81, 100]\}$ for the domain $\mathcal{X} = \langle 1, \cdots, 100\rangle$ and let $\mathcal{O} = \{1, 2, 3\}$ denote the set of its corresponding encodings. Let us assume the a uniform prior, $\mathcal{D}$ (default value), on $\mathcal{X}$. Thus, in Stage I, median is our measure from central tendency which gives $d_1 = 10.5, d_2 = 50.5$ and $d_3 = 90.5$.

In Stage II (Steps 4-9), the encoding probability distributions are computed using a variant of the classic exponential mechanism [GTT+19, DR14b] (Step 6). For instance, for the

aforementioned example we have $\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(40,\mathcal{P},\epsilon)=2\big]=p_{40,2}\propto 1/e^{(40-d_2)\epsilon/2}=1/e^{5.25\epsilon}$. The final encoding is then sampled from p_x (Steps 10-11).

Theorem 6. *Algorithm 3 gives a construction for $\mathsf{OP}\epsilon\mathsf{c}$.*

Proof. Here, we need to prove that Algorithm 3 satisfies the Eq. 3.6 and 3 (ϵ-dLDP) from Definition 10. We do this with the help of the following two lemmas.

Lemma 2. *Algorithm 3 satisfies Eq. 3.6 from Definition 10.*

Proof. Let $x\in\mathcal{X}_i, i\in[k]$.
Case I. $d_j, 1\le j<i-1, i\in[2,k]$ In this case, we have $d_j<d_{i-1}$. Thus,

$$\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x,\mathcal{P},\epsilon)=o_{i-1}\big]>\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x,\mathcal{P},\epsilon)=o_j\big] \tag{3.7}$$

Case II. d_j s.t. $i+1<j\le k, i\in[k-2]$
In this case, we have $d_{i+1}<d_j$. Thus,

$$\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x,\mathcal{P},\epsilon)=o_{i+1}\big]>\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x,\mathcal{P},\epsilon)=o_j\big] \tag{3.8}$$

Clearly, this concludes our proof. □

Next, we prove that Algorithm 3 satisfies ϵ-dLDP.

Lemma 3. *Algorithm 3 satisfies ϵ-dLDP.*

Proof. For all $x\in\mathcal{X}$ and $o_i\in\mathcal{O}=\{o_1,\cdots,o_k\}$, we have

$$\begin{aligned}
\frac{\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x,\mathcal{P},\epsilon)=o_i\big]}{\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x+t,\mathcal{P},\epsilon)=o_i\big]} &= \\
&\Big(e^{|x+t-d_i|-|x-d_i|\cdot\epsilon/2}\cdot\frac{\sum_{j=1}^{k}e^{-|x+t-d_j|\cdot\epsilon/2}}{\sum_{j=1}^{k}e^{-|x-d_j|\cdot\epsilon/2}}\Big) \\
&\le e^{t\epsilon/2}\cdot e^{t\epsilon/2} \\
&\big[\because |x-d_j|-t\le|x+t-d_j|\le|x-d_j|+t\big] \\
&= e^{t\epsilon}
\end{aligned} \tag{3.9}$$

Similarly,

$$\frac{\Pr\left[\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = o_i\right]}{\Pr\left[\mathsf{OP}\epsilon\mathsf{c}(x + t, \mathcal{P}, \epsilon) = o_i\right]} \geq e^{-t\epsilon}$$

□

Hence, from Lemmas 9 and 10, we conclude that Algorithm 3 gives a construction for the $\mathsf{OP}\epsilon\mathsf{c}$ primitive.

□

Size of partition $|\mathcal{P}|$. From Step 6, we observe that for every input x, the encoding probability distribution p_x is an exponential distribution centered at $\mathcal{P}(x)$ – its correct encoding. Moreover, the smaller is the size of $\mathcal{P}$ (number of intervals in $\mathcal{P}$), the larger is the probability of outputting $\mathcal{P}(x)$ (or its immediate neighbors). This is demonstrated in Figure 3.1 which plots p_x for $x = 50$ and $\epsilon = 0.1$ under varying equi-length partitioning of the input domain $[100]$.

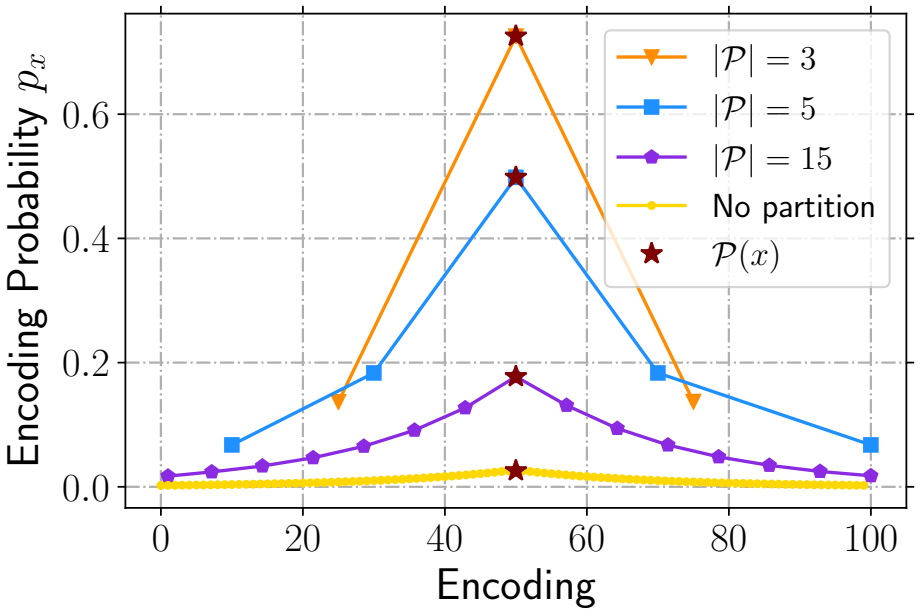


Figure 3.1: Encoding probability distribution for different partition sizes for $x = 50$, $\epsilon = 0.1$ and $\mathcal{X} = [100]$

Remark 1. The ϵ-dLDP guarantee of the $\mathsf{OP}\epsilon\mathsf{c}$ primitive (Theorem 6) does not depend on the partition $\mathcal{P}$. Thus, the partition size could range from $k = |\mathcal{O}| = |\mathcal{X}|$ (no effective partitioning at all) to $k = 2$. Additionally, the dLDP guarantee (and utility) is also independent of the encoding domain, $\mathcal{O}$, as long as the appropriate ordering constraint is valid.

Note that for $k = 1$ is a trivial case which destroys all ordinal information.

3.4 ϵ-dDP Order Preserving Encryption ($\mathsf{OP}\epsilon$)

In this chapter, we describe our proposed ϵ-dDP order preserving encryption scheme, $\mathsf{OP}\epsilon$. First, we define $\mathsf{OP}\epsilon$ followed by a new cryptographic security definition for $\mathsf{OP}\epsilon$ (Chapter 3.4.2).

3.4.1 Definition of $\mathsf{OP}\epsilon$

The *ϵ-dDP order preserving encryption* ($\mathsf{OP}\epsilon$) scheme is an encryption scheme that bolsters the cryptographic guarantee of a OPE scheme with an additional dDP guarantee. Here, we detail how our proposed primitive $\mathsf{OP}\epsilon\mathsf{c}$ can be used in conjunction with a OPE scheme (Definition 7) to form a $\mathsf{OP}\epsilon$ scheme.

Definition 11 (ϵ-dDP Order Preserving Encryption, $\mathsf{OP}\epsilon$)**.** *A ϵ-dDP order preserving encryption scheme, $\mathsf{OP}\epsilon$, is composed of a OPE scheme, $\mathcal{E}$, that satisfies the IND-FA-OCPA guarantee (Definition 12), and the $\mathsf{OP}\epsilon\mathsf{c}$ primitive and is defined by the following algorithms:*

$\mathsf{OP}\epsilon$ *Scheme*

- *Key Generation (K_ϵ). Uses K from the OPE scheme to generate a secret key S.*
- *Encryption (E_ϵ). The encryption algorithm inputs a plaintext $x \in \mathcal{X}$, an order Γ, a partition $\mathcal{P} \in \hat{\mathcal{X}}$, and the privacy parameter ϵ. It outputs $(\mathsf{S}', y) \leftarrow \mathsf{E}(\mathsf{S}, \tilde{o}, \Gamma)$ where $\tilde{o} \leftarrow \mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon/2)$.*
- *Decryption (D_ϵ). The decryption algorithm uses D to get back $\tilde{o} \leftarrow \mathsf{D}(\mathsf{S}, y)$.*

Following the above definition, the encryption of a dataset $X \in \mathcal{X}^n, X = \langle x_1, \cdots, x_n \rangle$ is carried out as follows:

1. Set $\mathsf{S}_0 \leftarrow \mathsf{K}(1^\kappa)$
2. For $\forall i \in [n]$, compute $(\mathsf{S}_i, y_i) \leftarrow \mathsf{E}(\mathsf{S}_{i-1}, \tilde{o}_i, \Gamma)$ where $\tilde{o}_i \leftarrow \mathsf{OP}\epsilon\mathsf{c}(x_i, \mathcal{P}, \epsilon/2)$

Key Idea. A $\mathsf{OP}\epsilon$ scheme works as follows:

- First, obtain an (randomized) encoding for the input using the $\mathsf{OP}\epsilon\mathsf{c}$ primitive (one possible construction is given by Algorithm 3 for any given ϵ and partition $\mathcal{P}$).
- Encrypt the above encoding under a OPE scheme.

Thus, ciphertexts encrypted under $\mathsf{OP}\epsilon$ preserve the order of the corresponding encodings as output by the $\mathsf{OP}\epsilon\mathsf{c}$ primitive. Referring back to our example, if $X = \langle 76, 9, 9, 40, 15, 76, 77\rangle$ and its corresponding encodings are $\widetilde{O} = \{8, 1, 2, 4, 2, 8, 8\}$, then the encryption of X under $\mathsf{OP}\epsilon$ preserves the order of $\widetilde{O}$.

In other words, since a OPE scheme preserves the exact order of its input dataset by definition, the utility of $\mathsf{OP}\epsilon$ (in terms of the preserved ordering information) is determined by the underlying $\mathsf{OP}\epsilon\mathsf{c}$ primitive. This is formalized by the following theorem.

Theorem 7. *[Utility Theorem] If, for a given partition $\mathcal{P} \in \hat{\mathcal{X}}$ and for all $x, x' \in \mathcal{X}$ such that $x > x'$ we have*

$$\Pr\big[\mathit{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) \geq \mathit{OP}\epsilon\mathsf{c}(x', \mathcal{P}, \epsilon)\big] \geq \alpha, \alpha \in [0, 1] \tag{3.10}$$

then for a $\mathit{OP}\epsilon$ scheme instantiated on such a $\mathit{OP}\epsilon\mathsf{c}$ primitive,

$$\Pr\big[\mathsf{E}_\epsilon(x, \mathsf{S}, \Gamma, \mathcal{P}, \epsilon) \geq \mathsf{E}_\epsilon(x', \mathsf{S}, \Gamma, \mathcal{P}, \epsilon)\big] \geq \alpha \tag{3.11}$$

where $\mathsf{S} \leftarrow \mathsf{K}_\epsilon(1^\kappa)$ and any Γ.

The proof follows directly from Definitions 7 and 11.

Lemma 4. *$\mathit{OP}\epsilon$ satisfies $\frac{\epsilon}{2}$-dLDP.*

The proof of the above lemma follows trivially from the post-processing guarantee of dLDP (Theorem 5).

3.4.2 New Security Definition for $\mathsf{OP}\epsilon$

Here, we present a novel securdity guarantee for $\mathsf{OP}\epsilon$, namely *indistinguishable ciphertexts under frequency-analyzing ϵ-dDP ordered chosen plaintext attacks* (ϵ-$\mathsf{IND\text{-}FA\text{-}OCPA}$, Definition 12).

The ϵ-$\mathsf{IND\text{-}FA\text{-}OCPA}$guarantee is associated with a security game, $\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{IND\text{-}FA\text{-}OCPA}_\epsilon}$, where the adversary, $\mathcal{A}_{\mathsf{PPT}}$, first chooses four input dataset of equal length, X_{00}, X_{01}, X_{10} and X_{11}, such that $\mathcal{P}_0(X_{00})$ and $\mathcal{P}_1(X_{10})$ share at least one randomized order where $X_{00}, X_{01} \in$

$\mathcal{X}_0^n$, $X_{10}, X_{11} \in \mathcal{X}_1^n$, $\mathcal{P}_0 \in \hat{\mathcal{X}}_0$ and $\mathcal{P}_1 \in \hat{\mathcal{X}}_1$. Additionally, $\{X_{00}, X_{01}\}$ and $\{X_{10}, X_{11}\}$ are t-adjacent (Definition 6). The challenger then selects two bits $\{b_1, b_2\}$ uniformly at random and returns the corresponding ciphertext for the dataset $X_{b_1 b_2}$. $\mathcal{A}_{\mathsf{PPT}}$ then outputs their guess for the bits and wins the game if they are able to guess either of the bits successfully. The ϵ-IND-FA-OCPA guarantee states that $\mathcal{A}_{\mathsf{PPT}}$ cannot distinguish among the four datasets. In what follows, we first present its formal definition and then, illustrate it using an example.

Definition 12 (ϵ-IND-FA-OCPA)**.** *An encryption scheme $\mathcal{E}_\epsilon = (\mathsf{K}_\epsilon, \mathsf{E}_\epsilon, \mathsf{D}_\epsilon)$ has indistinguishable ciphertexts under frequency-analyzing ϵ-dDP ordered chosen plaintext attacks if for any PPT adversary, $\mathcal{A}_{PPT}$, and security parameter, κ:*

$$\Pr[\mathcal{G}^{\mathcal{A}_{PPT}}_{\mathsf{FA\text{-}OCPA}_\epsilon}(\kappa, b_1, b_2) = (c_1, c_2)] \leq \tag{3.12}$$

$$e^{t\epsilon} \cdot \Pr\big[\mathcal{G}^{\mathcal{A}_{PPT}}_{\mathsf{FA\text{-}OCPA}_\epsilon}(\kappa, b_1', b_2') = (c_1, c_2)\big] + \mathsf{negl}(\kappa) \tag{3.13}$$

where $b_1, b_2, b_1', b_2', c_1, c_2 \in \{0, 1)\}$ and $\mathcal{G}^{\mathcal{A}_{PPT}}_{\mathsf{FA\text{-}OCPA}_\epsilon}(\kappa, b_1, b_2)$ is the random variable indicating the adversary $\mathcal{A}_{PPT}$'s output for following security game:

Game $\mathcal{G}^{\mathcal{A}_{PPT}}_{\mathsf{FA\text{-}OCPA}_\epsilon}(\kappa, b_1, b_2)$

1. $(X_{00}, X_{01}, X_{10}, X_{11}) \leftarrow \mathcal{A}_{PPT}$ *where*
   (a) $X_{00}, X_{01} \in \mathcal{X}_0^n$ *and* $X_{10}, X_{11} \in \mathcal{X}_1^n$
   (b) $\mathcal{P}_0(X_{00})$ *and* $\mathcal{P}_1(X_{10})$ *have at least one common randomized order where* $\mathcal{P}_0 \in \hat{\mathcal{X}}_0$ *and* $\mathcal{P}_1 \in \hat{\mathcal{X}}_1$
   (c) $\{X_{00}, X_{01}\}$ *and* $\{X_{10}, X_{11}\}$ *are t-adjacent (Definition 6)*
2. $\mathsf{S} \leftarrow \mathsf{K}(1^\kappa)$
3. *Compute* $\widetilde{O}_0 \leftarrow \mathsf{OP\epsilon c}(X_{00}, \mathcal{P}_0, \frac{\epsilon}{2})$ *and* $\widetilde{O}_1 \leftarrow \mathsf{OP\epsilon c}(X_{10}, \mathcal{P}_1, \frac{\epsilon}{2})$.
4. *If* $\widetilde{O}_0$ *and* $\widetilde{O}_1$ *do not have any common randomized order, then return* $\perp$. *Else*
   (a) *Select two uniform bits* b_1 *and* b_2 *and a randomized order* Γ^* *common to both* $\widetilde{O}_0$ *and* $\widetilde{O}_1$.
   (b) *If* $b_2 = 0$, *compute* $Y_{b_1 b_2} \leftarrow \mathsf{E}_\epsilon(\widetilde{O}_{b_1}, \mathsf{S}, \Gamma^*, \mathcal{O}_{b_1}, \infty)$[2]. *Else, compute* $Y_{b_1 b_2} \leftarrow \mathsf{E}_\epsilon(X_{b_1 1}, \mathsf{S}, \Gamma^*, \mathcal{P}_{b_1}, \frac{\epsilon}{2})$.

[2]equivalent to running $\mathsf{E}_\epsilon(\mathsf{S}, X_{b_1 0}, \Gamma^*, \mathcal{P}_{b_1}, \epsilon) := \langle \widetilde{O}_{b_1} \leftarrow \mathsf{OP\epsilon c}(X_{b_1 0}, \mathcal{P}_{b_1}, \epsilon/2), \mathsf{E}(\widetilde{O}_{b_1}, \mathsf{S}, \Gamma^*)\rangle$ where $\mathcal{O}_{b_1}$ is the corresponding encoding domain for $\mathcal{P}_{b_1}$

5. $(c_1, c_2) \leftarrow \mathcal{A}_{PPT}(Y_{b_1,b_2})$ *where* $c_1(c_2)$ *is* $\mathcal{A}_{PPT}$*'s guess for* $b_1(b_2)$

$\mathcal{A}_{\mathsf{PPT}}$ is said to win the above game if $b_1 = c_1$ or $b_2 = c_2$.

Example 7. We illustrate the above definition using the following example. Consider $X_{00} = \langle 22, 94, 23, 94, 36, 95\rangle$, $X_{10} = \langle 9, 40, 11, 76, 15, 76\rangle$, $X_{01} = \langle 24, 94, 23, 94, 36, 95\rangle$ and $X_{11} = \langle 9, 40, 8, 76, 15, 76\rangle$ where $\{X_{00}, X_{10}\}$ share a randomized order, $\langle 1, 4, 2, 5, 3, 6\rangle$, and $\{X_{00}, X_{01}\}$ and $\{X_{10}, X_{11}\}$ are 3-adjacent. For the ease of understanding, we consider the default case of $\mathcal{P}_0 = \mathcal{O}_0 = \mathcal{X}_0$ and $\mathcal{P}_1 = \mathcal{O}_1 = \mathcal{X}_1$. This means that $\mathcal{P}_0(X_{00}) = X_{00}$ and so on.

OPϵc. If only OPϵc were to be used to encode the above datasets, then only the pairs $\{X_{00}, X_{01}\}$ and $\{X_{10}, X_{11}\}$ would be indistinguishable to the adversary (albeit an information theoretic one) because of the ϵ-dDP guarantee (Definition 6). However, there would be no formal guarantee on the pairs $\{X_{01}, X_{11}\}, \{X_{01}, X_{10}\}, \{X_{00}, X_{11}\}, \{X_{00}, X_{10}\}$.

OPE. If we were to use just the OPE scheme, then only the pair $\{X_{00}, X_{10}\}$ would be indistinguishable for $\mathcal{A}_{\mathsf{PPT}}$ as the rest of the pairs do not share any randomized order.

OPϵ. Using OPϵ makes all 6 pairs $\{X_{00}, X_{01}\}, \{X_{00}, X_{11}\}, \{X_{00}, X_{10}\}, \{X_{01}, X_{11}\}$, $\{X_{01}, X_{10}\}, \{X_{11}, X_{10}\}$ indistinguishable for $\mathcal{A}_{\mathsf{PPT}}$. This is because OP$\epsilon$ essentially preserves the order of a ϵ-dDP scheme.

Hence, OPϵ enjoys strictly stronger security than both OPϵc and OPE.

Theorem 8. *The proposed encryption scheme, OPϵ satisfies ϵ-IND-FA-OCPA security guarantee.*

Proof. **Intuition.** The intuition of the proof is as follows. Recall that there are four input sequences the adversary has to distinguish among. If the adversary is able to guess bit b_1 correctly (with non-trivial probability), it is akin to breaking the IND-FA-OCPA guarantee of OPEs. Similarly, if the adversary is able to guess bit b_2 correctly, (with non trivial probability) it would imply the violation of the ϵ-dDP guarantee.

The proof is structured as follows. First, we prove that OPϵ satisfies $\epsilon/2$-dDP (or $(\epsilon/2, 0)$-dDP following the notation in Definition 6). The rest of the proof follows directly from this result and the IND-FA-OCPA guarantee of the OPE scheme.

Lemma 5. *Let $\mathcal{M}$ be a mechanism that*

1. *inputs a dataset* $X \in \mathcal{X}^n$

2. *outputs* $\widetilde{O} = \{\tilde{o}_1, \cdots, \tilde{o}_n\}$ *where for all* $i \in [n], \mathcal{P} \in \hat{\mathcal{X}}, \tilde{o}_i \leftarrow \mathit{OP}\epsilon\mathit{c}(x_i, \mathcal{P}, \epsilon/2)$

Then, $\mathcal{M}$ *satisfies* $\epsilon/2$-*dDP.*

Proof. Let $X, X' \in \mathcal{X}^n$ be t-adjacent. Specifically, let $x_i \neq x'_i, i \in [n]$. For brevity, we drop $\mathcal{P}$ and the privacy parameter $\epsilon/2$ from the notation $\mathsf{OP}\epsilon\mathsf{c}(\cdot)$.

$$\begin{aligned}
\frac{\Pr\big[\mathcal{M}(X) = \widetilde{O}\big]}{\Pr\big[\mathcal{M}(X') = \widetilde{O}\big]} &= \frac{\prod_{j=1}^{n} \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x_j) = \tilde{o}_j\big]}{\prod_{j=1}^{n} \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x'_j) = \tilde{o}_j\big]} \\
= \frac{\prod_{j=1, j\neq i}^{n} \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x_j) = \tilde{o}_j\big]}{\prod_{j=1, j\neq i}^{n} \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x_j) = \tilde{o}_j\big]} &\times \frac{\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x_i) = \tilde{o}_i\big]}{\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x'_i) = \tilde{o}_i\big]} \\
&\leq e^{\frac{t\epsilon}{2}} [\text{ From Eq. 3 of Definition 10}]
\end{aligned}$$

This concludes our proof. □

Lemma 6. *OP*ϵ *satisfies* $\epsilon/2$-*dDP.*

This result follows directly from Lemma 5 from Theorem 5.

Now, note that $\widetilde{O}_{b_1} \in \mathcal{O}_{b_1}^n$. Thus, $\mathsf{OP}\epsilon\mathsf{c}(\widetilde{O}_{b_1}, \mathcal{O}_{b_1}, \infty) = \widetilde{O}_{b_1}$ (Chapter 3.3.1) . As a result, $\mathsf{E}_\epsilon(\mathsf{S}, \widetilde{O}_{b_1}, \Gamma^*, \mathcal{O}_{b_1}, \infty)$ (Step 4*b*) is equivalent to running $\mathsf{E}_\epsilon(\mathsf{S}, X_{b_1 0}, \Gamma^*, \mathcal{P}_{b_1}, \epsilon) := \langle \widetilde{O}_{b_1} \leftarrow \mathsf{OP}\epsilon\mathsf{c}(X_{b_1 0}, \mathcal{P}_{b_1}, \epsilon/2), \mathsf{E}(\widetilde{O}_{b_1}, \mathsf{S}, \Gamma^*)\rangle$. Thus, from Definition 9, if $\Big|\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 0, 0) = (c_1, c_2)\big] - \Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 1, 0) = (c_1, c_2)\big]\Big| \geq \mathsf{negl}(\kappa)$, then another PPT adversary, $\mathcal{A}'_{\mathsf{PPT}}$, can use $\mathcal{A}_{\mathsf{PPT}}$ to win the $\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(\cdot)$ game which leads to a contradiction. Hence, we have

$$\begin{aligned}
\Big|&\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 0, 0) = (c_1, c_2)\big] \\
&-\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 1, 0) = (c_1, c_2)\big]\Big| \leq \mathsf{negl}(\kappa)
\end{aligned} \tag{3.14}$$

Without loss of generality, let us assume

$$\begin{aligned}
\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 1, 0) = (c_1, c_2)\big] \leq \\
\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 0, 0) = (c_1, c_2)\big]
\end{aligned} \tag{3.15}$$

Thus, from Eqs. (3.14) and (3.15), we have

$$\begin{aligned}
\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 0, 0) = (c_1, c_2)\big] \leq \\
\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa, 1, 0) = (c_1, c_2)\big] + \mathsf{negl}(\kappa)
\end{aligned} \tag{3.16}$$

From Theorem 5 and Lemma 6,

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,0)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,1)=(c_1,c_2)\big] \tag{3.17}$$

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,1)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,0)=(c_1,c_2)\big] \tag{3.18}$$

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,0)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,1)=(c_1,c_2)\big] \tag{3.19}$$

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,1)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,0)=(c_1,c_2)\big] \tag{3.20}$$

Now from Eqs. (3.16) and (3.19), we have,

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,0)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,1)=(c_1,c_2)\big] + \mathsf{negl}(\kappa) \tag{3.21}$$

Using Eqs. (3.15) and (3.20), we have

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,1)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,0)=(c_1,c_2)\big] \tag{3.22}$$

From Eqs. (3.18) and (3.16), we have

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,1)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,0)=(c_1,c_2)\big] + \mathsf{negl}'(\kappa) \tag{3.23}$$

$[\mathsf{negl}'(\kappa) = e^{\frac{t\epsilon}{2}} \cdot \mathsf{negl}(\kappa)$ which is another negligible function]

Eqs. (3.18) and (3.21) give us

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,1)=(c_1,c_2)\big]| \leq e^{t\epsilon}\Pr[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,1)=(c_1,c_2)] + \mathsf{negl}'(\kappa) \tag{3.24}$$

Using Eqs. (3.15) and (3.17), we have

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,0)=(c_1,c_2)\big] \leq e^{\frac{t\epsilon}{2}}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,1)=(c_1,c_2)\big] \tag{3.25}$$

Finally, Eqs. (3.20) and (3.25) give us

$$\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,1,1)=(c_1,c_2)\big]| \leq e^{t\epsilon}\Pr\big[\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{FA\text{-}OCPA}^\epsilon}(\kappa,0,1)=(c_1,c_2)\big] \tag{3.26}$$

Note that the $\mathcal{G}^{\mathcal{A}_{\mathsf{PPT}}}_{\mathsf{IND\text{-}FA\text{-}OCPA}_\epsilon}$ game can abort sometimes (Step 4, when $\widetilde{O}_0$ and $\widetilde{O}_1$ do not share any randomized order). However, this does not lead to any information leakage to $\mathcal{A}_{\mathsf{PPT}}$ since this step happens before the challenger has chosen any of the bits $\{b_1, b_2\}$. Additionally, the condition $1b$ ensures that the event that the game runs to completion happens with non-zero probability. It is so because if $\mathcal{P}_0(X_{00})$ and $\mathcal{P}_1(X_{10})$ share a randomized order, then $\Pr\big[\widetilde{O}_0$ and $\widetilde{O}_1$ share a randomized order$\big] > 0$.

This concludes our proof. □

Let $\mathcal{N}_{\mathsf{G}}(X) = \{X' | X' \in \mathcal{X}^n$ and $\{X, X'\}$ are indistinguishable to $\mathcal{A}_{\mathsf{PPT}}$ under guarantee $\mathsf{G}\}$. Additionally, we assume $\mathcal{P} = \mathcal{X}$ for the ease of understanding. Thus, in a nutshell, the ϵ-dDP guarantee allows a pair of datasets $\{X, X'\}$ to be indistinguishable[3] only if they are t-adjacent (for relatively small values of t). Referring back to our example, we have $X_{01} \in \mathcal{N}_{\epsilon\text{-dDP}}(X_{00})$ and $X_{11} \in \mathcal{N}_{\epsilon\text{-dDP}}(X_{10})$.

On the other hand, under the IND-FA-OCPA guarantee, $\{X, X'\}$ is indistinguishable[4] to $\mathcal{A}_{\mathsf{PPT}}$ only if they share a common randomized order. For instance, $X_{10} \in \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X_{00})$.

In addition to the above cases, the ϵ-IND-FA-OCPA guarantee allows a pair of datasets $\{X, X'\}$ to be indistinguishable[5] for $\mathcal{A}_{\mathsf{PPT}}$ if $\{X, X'\}$

- do not share a randomized order
- are not adjacent,

but there exists another dataset X'' such that

- $\{X', X''\}$ are adjacent, i.e. $X' \in \mathcal{N}_{\epsilon\text{-dDP}}(X'')$
- $\{X, X''\}$ share a randomized order, i.e., $X'' \in \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X)$.

From our aforementioned example, we have $X_{11} \notin \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X_{00})$ and $X_{11} \notin \mathcal{N}_{\epsilon\text{-dDP}}(X_{00})$. But still, $X_{11} \in \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}_\epsilon}(X_{00})$ since $X_{11} \in \mathcal{N}_{\epsilon\text{-dDP}}(X_{10})$ and $X_{10} \in \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X_{00})$. Thus, formally

$$\mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}_\epsilon}(X) = \bigcup_{X'' \in \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X)} \mathcal{N}_{\epsilon\text{-dDP}}(X'') \tag{3.27}$$

[3] the ratio of their output distributions are bounded by $e^{t\epsilon}$, holds against an information theoretic adversary as well

[4] computational indistinguishability [Ode09]

[5] Formally given by Eq. 3.13 which is structurally similar to that of the IND-CDP guarantee [MPRV09] which is a computational differential privacy guarantee.

Since, trivially $X \in \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X)$ and $X \in \mathcal{N}_{\epsilon\text{-}\mathsf{dDP}}(X)$, we have $\mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}_\epsilon}(X) \supseteq \mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}}(X)$ and $\mathcal{N}_{\mathsf{IND\text{-}FA\text{-}OCPA}_\epsilon}(X) \supseteq \mathcal{N}_{\epsilon\text{-}\mathsf{dDP}}(X)$.

Key Insight. The key insight of the ϵ-IND-FA-OCPA security guarantee is that the OPE scheme preserves the order of the outputs of a ϵ-dDP mechanism. As a result, the adversary is now restricted to only an ϵ-dDP order leakage from the ciphertexts. Hence, even if the security guarantee of the OPE layer is completely broken, the outputs of OPϵ would still satisfy ϵ-dDP due to Theorem 5. Referring to Example 1, in the very least input pairs $\{X_{00}, X_{01}\}$ and $\{X_{10}, X_{11}\}$ will remain indistinguishable under all inference attacks. Thus, OPϵ is the first encryption scheme to satisfy a formal security guarantee against all possible inference attacks and still provide some ordering information about the inputs.

Remark 2. The ϵ-IND-FA-OCPA guarantee of the OPϵ scheme is *strictly stronger than both dDP (dLDP) and IND-FA-OCPA (the strongest possible guarantee for any OPE).* Further, it depends only on the dLDP guarantee of the underlying OPϵc primitive which is independent of the partition $\mathcal{P}$ used (as discussed in Chapter 3.3.2). We discuss the role of $\mathcal{P}$ in Chapter 3.6.

3.5 OPϵ and Inference Attacks

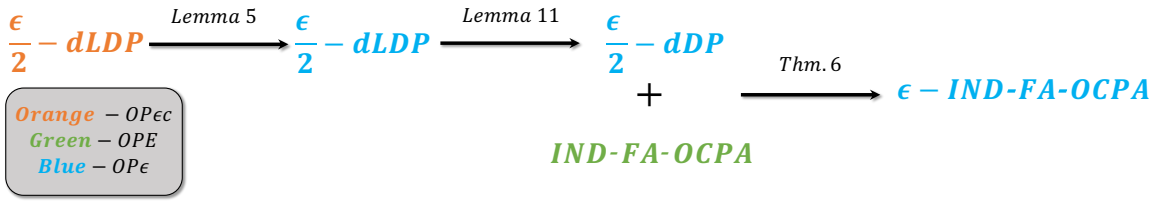


Figure 3.2: Relationships between dLDP, dDP and IND-FA-OCPA guarantees.

In this chapter, we discuss the implications of OPϵ's security guarantee in the face of inference attacks. Specifically, we formalize the protection provided by OPϵ's (relaxed) DP guarantee – this is the worst case guarantee provided by OPϵ.

Recall that the ϵ-IND-FA-OCPA guarantee of a OPϵ bolsters the cryptographic guarantee of a OPE (IND-FA-OCPA) with an additional layer of a (relaxed) DP guarantee. For the rest of the discussion, we focus on the worst case scenario where the OPE scheme provides no protection at all and study what formal guarantee we can achieve from just the (relaxed) DP guarantee. As discussed in Chapter 3.2.1, our proposed distance-based relaxation of DP comes in two flavors – local (dLDP, Definition 4) and central (dDP, Definition 6). Intuitively,

dLDP is a guarantee for each individual data point while dDP is a guarantee for a dataset. As a refresher, Figure 3.2 showcases the relationships between them. The most salient point is that the dLDP is a stronger guarantee than dDP– ϵ-dLDP implies ϵ-dDP. Thus, owing to the dLDP guarantee of the underlying OPϵc primitive, OPϵ trivially satisfies both dLDP (Lemma 4) and dDP (Lemma 6) guarantees.

For our discussion in Chapter 3.4.2, we use the dDP guarantee since the IND-FA-OCPA guarantee of OPEs is also defined on datasets. In what follows, we show how to interpret the protection provided by OPϵ's dLDP guarantee since it is stronger and holds for every data point. We do so with the help of an indistinguishability game, as is traditional for cryptographic security definitions. Let the input be drawn from a discrete domain of size N, i.e., $|\mathcal{X}| = N$. The record indistinguishability game, $\mathcal{G}^{\mathcal{A}}_{\beta-RI}$, is characterized by a precision parameter $\beta \in [\frac{1}{N}, 1]$. In this game, the adversary has to distinguish among a single record (data point) x and set of values $Q(x)$ that differ from x by at most $\lceil \beta N \rceil$. For instance, for $x = 3$, $N = 10$ and $\beta = 1/5$, the adversary has to distinguish among the values 3 and $Q(x) = \{1, 2, 4, 5\}$ ($\lceil \beta N \rceil = 2$). Let y_i denote the ciphertext for x_i after encryption under OPϵ. The game is formally defined as follows:

Game $\mathcal{G}^{\mathcal{A}}_{\beta-RI}(p)$

1. $x_0 \leftarrow \mathcal{A}$
2. $Q(x) = \{x_1, \cdots, x_q\}$ where $x_i \in \mathcal{X}, i \in [q]$ s.t. $|x_0 - x_i| \leq \lceil \beta N \rceil$ and $x_i \neq x_0$
3. Select $p \in \{0, 1, \cdots, q\}$ uniformly at random
4. $p' \leftarrow \mathcal{A}(y_p)$

$\mathcal{A}$ is said to win the above game if $p' = p$. Let *rand* be a random variable indicating the output of the baseline strategy where the adversary just performs random guessing.

Theorem 9. *For a OPϵ scheme satisfying $\frac{\epsilon}{2}$-dLDP, we have*

$$\Big|\Pr\big[p' = p\big] - \Pr\big[rand = p\big]\Big| \leq \frac{e^{\epsilon^*}}{q + e^{\epsilon^*}} - \frac{1}{q+1} \tag{3.28}$$

where $\epsilon^* = \epsilon\lceil \beta N \rceil$ and $q = |Q(x_0)|$ (Step (2) of game $\mathcal{G}^{\mathcal{A}}_{\beta-RI}$).

Proof. Let y denote the output ciphertext (Step 4) observable to the adversary $\mathcal{A}$. Note that the game itself satisfies $\epsilon/2$-dLDP. Let d be the probability that the adversary $\mathcal{A}$ wins the game, i.e., $d := \Pr\big[p' = p\big]$. Clearly, this cannot be greater than $\Pr\big[\mathsf{OP}\epsilon(\mathsf{S}, \epsilon/2, x_p) = y\big]$ (we use this shorthand to refer to the encryption as defined in Definition 11) – the probability

that encrypting x_p under $\mathsf{OP}\epsilon$ actually outputs y. Let $\mathcal{I} = \{i | i \in \{0, \cdots, q\}, i \neq p\}$. Since $|x_i - x_j| \leq 2\lceil \beta N \rceil, i, j \in \{0, \cdots, q\}$, from the $\epsilon/2$-dLDP guarantee we have

$$\begin{gathered} \forall i \in \mathcal{I} \\ \Pr\big[\mathsf{OP}\epsilon(\mathsf{S}, \epsilon/2, x_p) = y\big] \leq e^{\epsilon^*} \Pr\big[\mathsf{OP}\epsilon(\mathsf{S}, \epsilon/2, x_i) = y\big] \\ d \leq e^{\epsilon^*} \Pr\big[\mathsf{OP}\epsilon(\mathsf{S}, \epsilon/2, x_i) = y\big] \end{gathered}$$

Summing the equations for all $i \in \mathcal{I}$, we have

$$\begin{aligned} q \cdot d &\leq e^{\epsilon^*} \sum_{i \in \mathcal{I}} \Pr\big[\mathsf{OP}\epsilon(\mathsf{S}, \epsilon/2, x_i) = y\big] \\ &\Rightarrow q \cdot d \leq e^{\epsilon^*}(1 - d) \\ &\Rightarrow d \leq \frac{e^{\epsilon^*}}{q + e^{\epsilon^*}} \end{aligned} \tag{3.29}$$

Clearly,

$$\Pr\big[rand = p\big] = \frac{1}{q+1} \tag{3.30}$$

Hence, from Eqs. 3.29 and 3.30, we have

$$\Big|\Pr\big[p' = p\big] - \Pr\big[rand = p\big]\Big| \leq \frac{e^{\epsilon^*}}{q + e^{\epsilon^*}} - \frac{1}{q+1} \tag{3.31}$$

□

From the above theorem, observe that for low values of ϵ^* (i.e., low ϵ and β) the R.H.S of the Eq. (3.28) is low. This means that *for reasonably low values of ϵ (high privacy), with very high probability an adversary cannot distinguish among input values that are close to each other (small β) any better than just random guessing.* Now, recall that owing to the dLDP guarantee of the underlying $\mathsf{OP}\epsilon\mathsf{c}$ primitive, every data point encrypted under $\mathsf{OP}\epsilon$ is also protected by the dLDP guarantee (Lemma 4). This implies that, for any dataset X, the above indistinguishability result holds for every individual data point (record) simultaneously. In other words, *the dLDP guarantee rigorously limits the accuracy of any inference attack for every record of a dataset.*

As a concrete example, let us look at the binomial attack on OPE schemes satisfying $\mathsf{IND\text{-}FA\text{-}OCPA}$ proposed by Grubbs et al. [GSB+17b]. The attack uses a biased coin model to locate the range of ciphertexts corresponding to a particular plaintext. Experimental results on a dataset of first names show that the attack can recover records corresponding to certain high frequency plaintexts (such as first name 'Michael') with high accuracy. In this context, the implications of the above result is as follows. Consider a dataset with plaintext records

corresponding to first names 'Michael' and 'Michele'. For $\mathsf{OP}\epsilon$, the recovery rate for either would not be better than the random guessing baseline since both the values are close to each other in alphabetic order.

Note that the above result is information-theoretic and holds for *any* adversary – active or passive, both in the persistent (access to volume/access-pattern/search-pattern leakage) and snapshot attack models (access to a single snapshot of the encrypted data) [FVY+17].

Remark 4. In the very least, $\mathsf{OP}\epsilon$ *rigorously limits the accuracy of any inference attack for every record of a dataset for all adversaries.* (Theorem 9).

3.6 $\mathsf{OP}\epsilon$ for Encrypted Databases

In this chapter, we describe how to use a $\mathsf{OP}\epsilon$ scheme in practice in the context of encrypted databases. Specifically, we discuss how we can leverage the partition parameter, $\mathcal{P}$, of the underlying $\mathsf{OP}\epsilon\mathsf{c}$ primitive for improved utility.

Problem Setting. For encrypted databases, typically a data owner has access to the entire database in the clear and encrypts it before outsourcing it to an untrusted server. The queriers of the encrypted databases are authorized entities with access to the secret keys. In fact, in many practical settings the data owner themselves is the querier [FVY+17].

The most popular use case for databases encrypted under OPEs is retrieving the set of records belonging to a queried range. However, due to randomization, encryption under $\mathsf{OP}\epsilon$ leads to loss in utility. Specifically in the context of range queries, it might miss some of the correct data records and return some incorrect ones. For the former, constraining $\mathsf{OP}\epsilon$ to maintain only a partial order is found to be helpful. As discussed in Chapter 3.3.2, the more coarse grained the partition is (the lesser the number of intervals), the larger is the probability for $\mathsf{OP}\epsilon\mathsf{c}$ to output the correct encoding. Hence, if any given range $[a, b]$ is covered by a relatively small number of intervals in $\mathcal{P}$, then with high probability the set of records corresponding to the encodings $\{\tilde{o} | \tilde{o} \in O \wedge \mathcal{P}(a) \leq \tilde{o} \leq \mathcal{P}(b)]\}$ will contain most of the correct records. This results in better accuracy for the subsequent $\mathsf{OP}\epsilon$ scheme since it's accuracy is determined by the underlying $\mathsf{OP}\epsilon\mathsf{c}$ primitive (Theorem 7).

The problem of returning incorrect records can be mitigated by piggybacking every ciphertext encrypted under $\mathsf{OP}\epsilon$ with another ciphertext that is obtained from encrypting the corresponding plaintext under a standard authenticated encryption scheme [enc], $\overline{\mathcal{E}} := \langle \overline{\mathsf{K}}, \overline{\mathsf{E}}, \overline{\mathsf{D}} \rangle$.

We refer to this as the augmented $\mathsf{OP}\epsilon$[6] scheme and it works as follows:
Augmented $\mathsf{OP}\epsilon$, $\mathcal{E}^\dagger$

- *Key Generation* ($\mathsf{K}_\epsilon^\dagger$). This algorithm generates a pair of keys (S, K) where $\mathsf{S} \leftarrow \mathsf{K}_\epsilon(\kappa)$ and $K \leftarrow \overline{\mathsf{K}}(\kappa)$
- *Encryption* ($\mathsf{E}_\epsilon^\dagger$). This algorithm generates (S', y_0, y_1) where $\tilde{o} \leftarrow \mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon/2)$, $(\mathsf{S}', y_0) \leftarrow \mathsf{E}(\mathsf{S}, \tilde{o}, \Gamma)$, $y_1 \leftarrow \overline{\mathsf{E}}(K, x)$
- *Decryption* ($\mathsf{D}_\epsilon^\dagger$). The decryption algorithm uses S and K to decrypt both the ciphertexts, $(x, \tilde{o})$ as $\tilde{o} \leftarrow \mathsf{D}_\epsilon(\mathsf{S}, y_0)$ and $x \leftarrow \overline{\mathsf{D}}(K, y_1)$.

After receiving the returned records from the server, the querier can decrypt $\{y_{i1}\}$ and discard the irrelevant ones. The cost of this optimization for the querier is the processing overhead for the extra records (see discussion later).

The data owner (who has access to the dataset in the clear) can decide on the partition based on the dataset before encrypting and outsourcing it. For most input distributions, an equi-depth partitioning strategy works well (as demonstrated by our experimental results in Chapter 3.8.2). Nevertheless, the partition can be update dynamically as well (see Chapter 3.9).

> **Remark 5.** The partitioning of the input domain ($\mathcal{P}$) has no bearing on the formal security guarantee. It is performed completely from an utilitarian perspective in the context of encrypted databases – it results in an accuracy-overhead trade-off (accuracy - number of correct records retrieved; overhead - number of extra records processed).

Range Query Protocol. The end-to-end range query protocol is described in Algorithm 4. Before detailing it, we will briefly discuss the protocol for answering range queries for a OPE scheme, $\mathcal{E}$, that satisfies the IND-FA-OCPA guarantee (see [Ker15] for details). Recall that every ciphertext is unique for such a OPE scheme. Hence, a querier has to maintain some state information for every plaintext. Specifically, if $Y = \{y_1, \cdots, y_n\}$ denotes the corresponding ciphertexts for an input set $X = \{x_1, \cdots, x_n\}$, then the querier stores the maximum and minimum ciphertext in Y that corresponds to the plaintext x_i, denoted by $\max_{\mathcal{E}}(x_i)$ and $\min_{\mathcal{E}}(x_i)$, respectively. For answering a given range query $[a, b]$, the querier asks for all the records in Y that belong to $\big[\min_{\mathcal{E}}(a), \max_{\mathcal{E}}(b)\big]$. Recall that in $\mathsf{OP}\epsilon$, the OPE scheme is applied to the output (encodings) of the $\mathsf{OP}\epsilon\mathsf{c}$ primitive. So now for answering $[a, b]$, the querier has to retreive records corresponding to $[\mathcal{P}(a), \mathcal{P}(b)]$ instead where $\mathcal{P}$ is

[6]The augmented $\mathsf{OP}\epsilon$ scheme still upholds the ϵ-IND-FA-OCPA guarantee owing to the semantic security of the encryption scheme $\overline{\mathcal{E}}$.

the partition for the encoding. Hence, the querier first maintains the state information for the encodings (Steps 1-6, Algorithm 4). Note that since the size of the encoding space is smaller than the input domain $\mathcal{X}$, the amount of state information to be stored for a $\mathsf{OP}\epsilon$ is less than that for a OPE (see Chapter 3.9). Next, the querier asks for all the encrypted records in the set $Y' = \left\{\langle y'_{i0}, y'_{i1}\rangle | i \in [n] \text{ and } y'_{i0} \in \left[\min_{\mathcal{E}^\dagger}(\mathcal{P}(a)), \max_{\mathcal{E}^\dagger}(\mathcal{P}(b)\right]\right\}$ from the server (Steps 7-10). On receiving them, the querier only retains those records that fall in the queried range (Steps 11-18).

There are two ways the utility can be further improved. The first is including records from some of the intervals preceding $\mathcal{P}(a)$ and following $\mathcal{P}(b)$. The querier can ask for the records in $[\max(o_1, o_{a-l}), \min(o_{b+l}, o_k))], l \in \mathbb{Z}_{\geq 0}$ where $o_a := \mathcal{P}(a)$ and $o_b := \mathcal{P}(b)$. However, the cost is increase in extra records.

Another optimization is to answer a workload of range queries at a time. Under $\mathsf{OP}\epsilon$, queries can be made only at the granularity of the partition. Thus, if a queried range $[a, b]$ is much smaller than $[\mathcal{P}(a), \mathcal{P}(b)]$, then the querier has to pay the overhead of processing extra records. This cost can be reduced in the case of a workload of range queries where multiple queries fall within $[\mathcal{P}(a), \mathcal{P}(b)]$ (records that are irrelevant for one query might be relevant for some other in the workload). Additionally, the number of missing records for the query $[a, b]$ is also reduced if records from the neighboring intervals of $[\mathcal{P}(a), \mathcal{P}(b)]$ are also included in the response (owing to the other queries in the workload).

Discussion. As described above, the server side interface for range query protocols is the same for both $\mathsf{OP}\epsilon$ and a OPE scheme with the $\mathsf{IND\text{-}FA\text{-}OCPA}$ guarantee (with a nominal change to accommodate the extra ciphertexts $\{y_{i1}\}$). The cost is the extra storage for $\{y_{i1}\}$. However, in this age of cloud services, outsourced storage ceases to be a bottleneck [sto].

The querier, on the other hand, needs to decrypt all the returned records (specifically, $\{y_{i1}\}$). However, decryption is in general an efficient operation. For instance, the decryption of 1 million ciphertexts encrypted under AES-256 GCM requires < 3 minutes in our experimental setup. Thus, on the overall there is no tangible overhead in adopting $\mathsf{OP}\epsilon$.

Remark 6. $\mathsf{OP}\epsilon$ could be used for secure data analytics in settings where (1) the ϵ-dDP guarantee is acceptable, i.e, the main security concern is preventing the distinction between input values close to each other and, (2) the application can tolerate a small loss in utility. Specifically in such settings, replacing encrypted databases that are already deploying OPE schemes (satisfying $\mathsf{IND\text{-}FA\text{-}OCPA}$)with a $\mathsf{OP}\epsilon$ scheme would give a strictly stronger security guarantee against all attacks with *nominal change in infrastructure or performance* - a win-win situation.

Algorithm 4: Range Query Protocol

Notations: Z - Input dataset with n records (r_i, x_i) where $x_i \in \mathcal{X}$ denotes the sensitive attribute to be encrypted under $\mathsf{OP}\epsilon$ and r_i denotes the rest of associated data (other attributes could be encrypted too);
S - Secret key for the $\mathsf{OP}\epsilon$ scheme; $\mathcal{P}$- Partition used for $\mathsf{OP}\epsilon$;
K- Secret key for the authenticated encryption scheme $\overline{\mathcal{E}}$;

Input: Range Query $[a, b], a, b \in \mathcal{X}$

Output: Set of records $V = \{r_i | (r_i, x_i) \in Z, x_i \in [a, b]\}$[7]

Initialization: Querier

$X = \langle x_1, \cdots, x_n \rangle$

$Y = \mathcal{E}^{\dagger}(X, \mathsf{S}, K, \Gamma, \mathcal{P}, \frac{\epsilon}{2})$ ▷ Contains encrypted attributes $\{(y_{i0}, y_{i1})\}$

for $o \in \mathcal{O}$

  $\max_{\mathcal{E}^{\dagger}}(o) = \max\{y_{i0} | (y_{i0}, y_{i1}) \in Y$ and y_{i0} decrypts to $o\}$

  $\min_{\mathcal{E}^{\dagger}}(o) = \min\{y_{i0} | (y_{i0}, y_{i1}) \in Y$ and y_{i0} decrypts to $o\}$

end for ▷ Querier maintains state information

Range Query Protocol: Querier

$C = \{\min_{\mathcal{E}^{\dagger}}(\mathcal{P}(a)), \max_{\mathcal{E}^{\dagger}}(\mathcal{P}(b)\}$ ▷ Transformed range query based on state information

Querier $\xrightarrow{C}$ *Server*

Server

$Y' = \left\{(y_{i0}, y_{i1}) | i \in [n] \text{ and } y_{i0} \in \left[\min_{\mathcal{E}^{\dagger}}\left(\mathcal{P}(a)\right), \max_{\mathcal{E}^{\dagger}}\left(\mathcal{P}(b)\right)\right]\right\}$

▷ Server returns the set of records matching the query

Server $\xrightarrow{Y'}$ *Querier*

Querier

$V = \phi$

for $y_{i1} \in Y'$

  $x'_i \leftarrow \overline{\mathsf{D}}(K, y'_{i1})$

  if $(x'_i \in [a, b])$ ▷ Verifying whether record falls in $[a, b]$

    $V = V \cup r_i$

  end if

end for

Return V

3.7 LDP Mechanisms using OPϵc

The $\mathsf{OP}\epsilon\mathsf{c}$ primitive can be of independent interest in the LDP setting. Depending on the choice of the partition $\mathcal{P}$ over the input domain $\mathcal{X}$, $\mathsf{OP}\epsilon\mathsf{c}$ can be used to answer different

[7] V has a small utility loss as explained before.

types of queries with high utility. In this section, we describe how to use $\mathsf{OP}\epsilon\mathsf{c}$ to answer two such queries.

Problem Setting. We assume the standard LDP setting with n data owners, $\mathsf{DO}_i, i \in [n]$ each with a private data x_i.

3.7.1 Ordinal Queries

$\mathsf{OP}\epsilon\mathsf{c}$ can be used to answer queries in the LDP setting that require the individual noisy outputs to retain some of the ordinal characteristics of their corresponding inputs. One class of such queries include identifying which q-quantile does each data point belong to. This constitutes a popular class of queries for domains such as annual employee salaries, annual sales figures of commercial firms and student test scores. For example, suppose the dataset consist of the annual sales figures of different clothing firms and the goal is to group them according to their respective deciles. Here, partition $\mathcal{P}$ is defined by dividing the input domain into $q = 10$ equi-depth intervals using an estimate of the input distribution, $\mathcal{D}$. In the case such an estimate is not available, a part of the privacy budget can be first used to compute this directly from the data [LWLZ$^+$20]. For another class of queries, the partition can be defined directly on the input domain based on its semantics. Consider an example where the goal is to group a dataset of audiences of TV shows based on their age demographic – the domain of age can be divided into intervals $\{[1, 20], [21, 40], [41, 60], [61, 100]\}$ based on categories like "youth", "senior citizens". Once the partition is defined, each data owner uses the $\mathsf{OP}\epsilon\mathsf{c}$ primitive to report their noisy encoding. Note that the dLDP privacy guarantee is amenable to these cases, as one would want to report the intervals correctly but the adversary should not be able to distinguish between values belonging to the same interval. The full mechanism is outline in Algorithm 5.

Algorithm 5: Answering Ordinal Queries

Parameter $\mathcal{P}$ - Partition defined over the input domain as specified by the query;
ϵ - Privacy parameter

1: **for** $i \in [n]$
2: DO_i computes $o_i = \mathsf{OP}\epsilon\mathsf{c}(x_i, \mathcal{P}, \epsilon)$ and sends it to the data aggregator
3: **end for**
4: **Return** $O = \{o_1, \cdots, o_n\}$

3.7.2 Frequency Estimation

Here, we discuss the default case of the $\mathsf{OP}\epsilon\mathsf{c}$ primitive where the partition is same as the input domain, i.e., $\mathcal{P} = \mathcal{O} = \mathcal{X}$. Under this assumption, we can construct a mechanism for obtaining a frequency oracle in the LDP setting under the dLDP guarantee. The mechanism is outlined in Algorithm 6 and described below. Given a privacy parameter, ϵ, each data owner, $\mathsf{DO}_i, i \in [n]$, reports $\tilde{o}_i = \mathsf{OP}\epsilon\mathsf{c}(x_i, \mathcal{X}, \epsilon)$ to the untrusted data aggregator (Steps 1-4). Next, the data aggregator performs non-negative least squares (NNLS) as a post-processing inferencing step on the noisy data to compute the final frequency estimations (Steps 5-6). NNLS is a type of constrained least squares optimizations problem where the coefficients are not allowed to become negative. That is, given a matrix $\mathbf{A}$ and a (column) vector of response variables $\mathbf{Y}$, the goal is to find $\mathbf{X}$ such that

$$\arg\min_{\mathbf{X}} \|\mathbf{A} \cdot \mathbf{X} - \mathbf{Y}\|_2, \text{subject to } \mathbf{X} \geq 0$$

where $||\cdot||_2$ denotes Euclidean norm. The rationale behind this inferencing step (Step 5) is discussed below.

Lemma 7. *W.l.o.g let $\mathcal{X} = \{1, \cdots, m\}$ and let $\mathbf{Y}$ be the vector such that $\mathbf{Y}(i), i \in [m]$ indicates the count of value i in the set $\{\tilde{o}_1, \cdots, \tilde{o}_n\}$ where $\tilde{o}_i = \mathsf{OP}\epsilon\mathsf{c}(i, \mathcal{X}, \epsilon)$. Given,*

$$\mathbf{A}(i,j) = \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(i, \mathcal{X}, \epsilon) = j\big], i, j \in [m] \tag{3.32}$$

the solution $\mathbf{X}$ of $\mathbf{A} \cdot \mathbf{X} = \mathbf{Y}$ gives an unbiased frequency estimator ($\mathbf{X}(i)$ is the unbiased estimator for value i).

Proof. Let $\mathbf{X}'$ be a vector such that $\mathbf{X}'(i)$ represents the true count of the value $i \in [m]$. Thus, we have

$$\begin{aligned}
\mathbb{E}\big[X(i)\big] &= \mathbb{E}\big[\sum_{j=1}^{n} \mathbf{A}^{-1}(i,j) \cdot \mathbf{Y}(j)\big] \\
&= \sum_{j=1}^{n} \mathbf{A}^{-1}(i,j) \cdot \mathbb{E}\big[\mathbf{Y}(j))\big] \\
&= \sum_{j=1}^{n} \mathbf{A}^{-1}(i,j) \cdot \big(\sum_{k=1}^{n} \mathbf{X}'(k) \cdot \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(k, \mathcal{X}, \epsilon) = j\big]\big) \\
&= \sum_{j=1}^{n} \mathbf{A}^{-1}(i,j) \cdot \big(\sum_{k=1}^{n} \mathbf{X}'(k) \cdot \mathbf{A}(j,k)\big) \\
&= \sum_{j=1}^{n} \mathbf{X}'(j) \cdot (\sum_{k=1}^{n} \mathbf{A}^{-1}(i,k) \cdot \mathbf{A}(k,j)) \\
&= \mathbf{X}'(i)
\end{aligned}$$

This concludes the proof. □

Thus by the above lemma, $\mathbf{X}$ is an unbiased frequency estimator. However, it is important to note that the solution $\mathbf{X}$ is not guaranteed to be non-negative. But, given our problem setting, the count estimates are constrained to be non-negative. Hence, we opt for an NNLS inferencing. When the exact solution $\mathbf{X} = \mathbf{A}^{-1} \cdot \mathbf{Y}$ is itself non-negative, the estimator obtained from the NNLS optimization is identical to the exact solution. Otherwise, the NNLS optimization gives a biased non-negative estimator that results in minimal least square error. The resulting frequency oracle can be used to answer other queries like mean estimation and range queries[8].

Algorithm 6: Frequency Estimation

Input: X - Input dataset $\langle x_1, \cdots, x_n \rangle$; ϵ - Privacy parameter

Output: $\mathbf{X}$ - Estimated frequency

Data Owner

1: Set $\mathcal{P} = \mathcal{X}$

2: **for** $i \in [n]$

3: DO_i computes $\tilde{o}_i = \mathsf{OP}\epsilon\mathsf{c}(x_i, \mathcal{X}, \epsilon)$ and sends it to the aggregator

4: **end for**

Data Aggregator

5: Data aggregator performs NNLS optimization as follows

$$\mathbf{A} \cdot \mathbf{X} = \mathbf{Y} \text{ where}$$
$$\mathbf{A}(i,j) = \Pr[\mathsf{OP}\epsilon\mathsf{c}(i, \mathcal{X}, \epsilon) = j], i, j \in [m]$$
$$\mathbf{Y}(i) = \text{Count of value } i \text{ in } \{\tilde{o}_1, \cdots, \tilde{o}_n\}$$

6: **Return X**

Utility Analysis for Frequency Estimation Using OPϵc

Here, we present a formal utility analysis of the frequency oracle. Let $p_{ij}, i, j \in \mathcal{X}$ denote the $\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(i, \mathcal{X}, \epsilon)\big] = j$ and let $\mathbf{X}'[i]$ denote the true count for i. Additionally, let $\mathbb{I}_{i,j}$ be an indicator variable for the event $\mathsf{OP}\epsilon\mathsf{c}(i, \mathcal{X}, \epsilon) = j$.

[8] In the LDP setting, this refers to statistical range query, i.e., the count of the records that belong to a queried range.

Theorem 10. *The variance of count estimation* $\mathbf{X}[i]$ *is given by*

$$\mathit{Var}(\mathbf{X}[i]) = \sum_{j=1}^{n}\sum_{k=1}^{n}\big(p_{k,j}(1-p_{k,j})\cdot(\mathbf{X}'[k]\cdot\mathbf{A}^{-1}[i,j])^2\big) - \sum_{k=1}^{n}\sum_{1\le j_1<j_2\le n}\Big(\mathbf{X}'[k]^2\cdot\mathbf{A}^{-1}[i,j_1]\cdot\mathbf{A}^{-1}[i,j_2]\cdot p_{k,j_1}\cdot p_{k,j_2}\Big)$$

Proof. Variance of the indicator variable is given by,

$$\mathsf{Var}(\mathbb{I}_{j,i}) = p_{j,i}\cdot q_{j,i}$$

Additionally, we so have

$$\mathsf{Cov}(\mathbb{I}_{j,i},\mathbb{I}_{j,k}) = -p_{j,i}p_{j,k}\cdot$$
$$\mathsf{Cov}(\mathbb{I}_{j,i},\mathbb{I}_{k,l}) = 0$$

Using this we have,

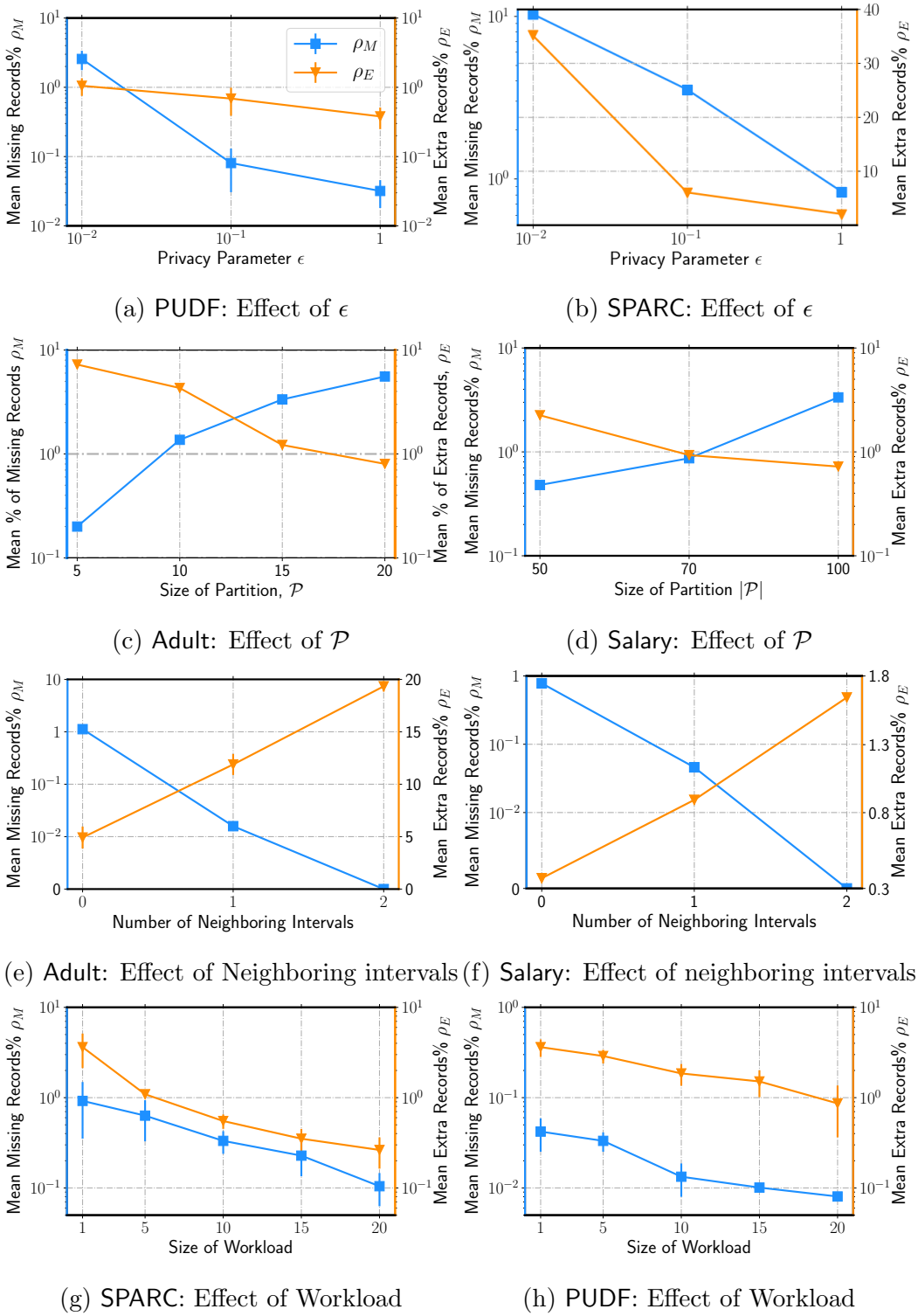
$$\mathsf{Var}(\mathbf{X}[i]) = \mathsf{Var}\Big(\sum_{j=1}^{n}\mathbf{A}^{-1}[i,j])\cdot\mathbf{Y}[j]\Big)$$
$$= \mathsf{Var}\Big(\sum_{j=1}^{n}\Big(\mathbf{A}^{-1}[i,j]\cdot\sum_{k=1}^{n}\mathbf{X}'[k]\cdot\mathbb{I}_{k,j}\Big)\Big)$$
$$= \mathsf{Var}\Big(\sum_{j=1}^{n}\sum_{k=1}^{n}\big(\mathbb{I}_{k,j}\cdot\mathbf{X}'[k]\cdot\mathbf{A}^{-1}[i,j]\big)\Big)$$
$$= \sum_{j=1}^{n}\sum_{k=1}^{n}\big(\mathsf{Var}(\mathbb{I}_{k,j})\cdot(\mathbf{X}'[k]\cdot\mathbf{A}^{-1}[i,j])^2\big) + \sum_{k=1}^{n}\sum_{1\le j_1<j_2\le n}\Big(\mathbf{X}'[k]^2\cdot\mathbf{A}^{-1}[i,j_1]\cdot\mathbf{A}^{-1}[i,j_2]\cdot\mathsf{Cov}(\mathbb{I}_{k,j_1},\mathbb{I}_{k,j_2})\Big)$$

$$= \sum_{j=1}^{n}\sum_{k=1}^{n}\big(p_{k,j}(1-p_{k,j})\cdot(\mathbf{X}'[k]\cdot\mathbf{A}^{-1}[i,j])^2\big) - \sum_{k=1}^{n}\sum_{1\le j_1<j_2\le n}\Big(\mathbf{X}'[k]^2\cdot\mathbf{A}^{-1}[i,j_1]\cdot\mathbf{A}^{-1}[i,j_2]\cdot p_{k,j_1}\cdot p_{k,j_2}\Big)$$

□

3.8 Experimental Evaluation

In this chapter, we present our evaluation results for the proposed primitives, $\mathsf{OP}\epsilon$ and $\mathsf{OP}\epsilon\mathsf{c}$. Specifically, we answer the following three questions:



(a) PUDF: Effect of ϵ

(b) SPARC: Effect of ϵ

(c) Adult: Effect of $\mathcal{P}$

(d) Salary: Effect of $\mathcal{P}$

(e) Adult: Effect of Neighboring intervals

(f) Salary: Effect of neighboring intervals

(g) SPARC: Effect of Workload

(h) PUDF: Effect of Workload

Figure 3.3: Accuracy Analysis of OPϵ in the Context of Encrypted Databases

- **Q1:** Does OPEϵ retrieve the queried records with high accuracy?
- **Q2:** Is the processing overhead of OPEϵ reasonable?
- **Q3:** Can OPEϵc answer statistical queries in the LDP setting with high accuracy?

Evaluation Highlights

- OPEϵ retrieves almost all the records of the queried range. For instance, OPEϵ only misses around 4 in every $10K$ correct records on average for a dataset of size $\sim 732K$ with an attribute of domain size $\sim 18K$ and $\epsilon = 1$.
- The overhead of processing extra records for OPE is low. For example, for the above dataset, the number of extra records processed is just 0.3% of the dataset size for $\epsilon = 1$.
- We give an illustration of OPEϵ's protection against inference attacks. For an age dataset and an adversary with real-world auxiliary knowledge, no inference attack in the snapshot attack model can distinguish between two age values (x, x') such that $|x - x'| \leq 8$ for $\epsilon = 0.1$.
- OPEϵc can answer several queries in the LDP setting with high utility. For instance, OPEϵc can answer ordinal queries with 94.5% accuracy for a dataset of size $\sim 38K$, an attribute of domain size $\sim 240K$ and $\epsilon = 1$. Additionally, OPEϵc achieves $6\times$ lower error than the state-of-the-art ϵ-LDP technique for frequency estimation for $\epsilon = 0.1$.

3.8.1 Experimental Setup

Datasets. We use the following datasets:

- *PUDF* [PUD]. This is a hospital discharge data from Texas. We use the 2013 PUDF data and the attribute PAT_ZIP (7,31,188 records of patient's 5-digit zipcode from the domain $[70601, 88415]$).
- *Statewide Planning and Research Cooperative System (SPARCS)* [NYC]. This is a hospital inpatient discharge dataset from the state of New York. This dataset has 25,31,896 records and we use the length_of_stay (domain $[1, 120]$) attribute for our experiments.
- *Salary* [sal15]. This dataset is obtained from the Kaggle repository and contains the compensation for San Francisco city employees. We use the attribute BasePay (domain $[1000, 230000]$) from the years 2011 ($35,707$ records) and 2014 ($38,122$ records).

- *Adult* [AN10]. This dataset is obtained from the UCI repository and is derived from the 1994 Census. The dataset has $32,561$ records and we use the attribute Age (domain $[1, 100]$) for our experiments.

- *Population* [cen]. This is a US Census dataset of annual estimates of the resident population by age and sex. We use the data for male Puerto Ricans for 2011 and 2019.

Datasets Adult and SPARCS have small and dense domains while PUDF and Salary have larger and sparse domains.

Metrics. We use the following metrics for our experiments. For evaluating **Q1**, we use the relative percentage of missing records, $\rho_M = \frac{\#\text{missing records}}{\#\text{correct records}}\%$. Note that ρ_M essentially captures false negatives which is the only type of error encountered – the querier can remove all cases false positives as discussed in Chapter 3.6. We evaluate **Q2** via the percentage of extra records processed relative to the dataset size, $\rho_E = \frac{\#\text{extra records}}{\#\text{records in dataset}}\%$. A key advantage of outsourcing is that the querier doesn't have to store/process the entire database. ρ_E measures this – low ρ_E implies that the (relative) count of extra records is low and it is still advantageous to outsource. In other words, low ρ_E implies that the client's processing overhead is low (relative to the alternative of processing the whole dataset). We believe this is a good metric for assessing the overhead in our setting because:

- For clients, the decryption of extra records doesn't result in a tangible time overhead (1 million records take < 3 minutes, see Chapter 3.6).

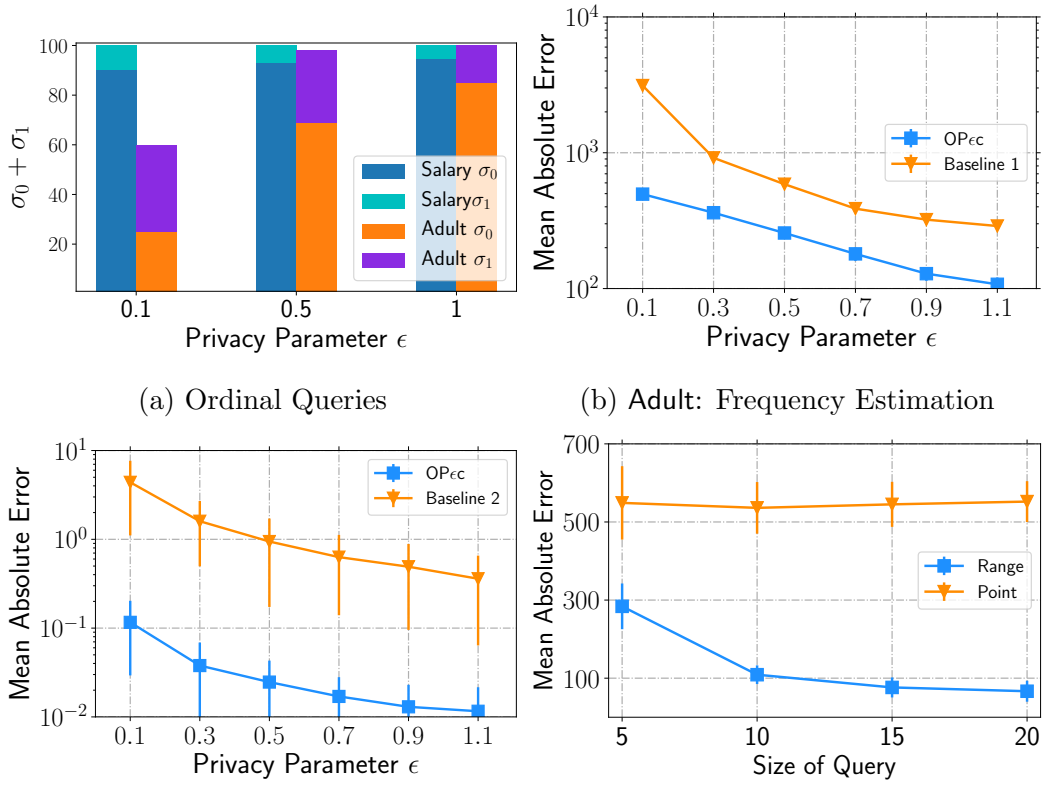
- OPϵ has no impact on the server since its interface (functionality) is the same as that for OPE.

For evaluating ordinal queries (Figure 3.4a), we use $\sigma_k = \%$ of points with $|\mathcal{P}(x) - \tilde{o}_x| = k$ where $\mathcal{P}(x)$ and $\tilde{o}_x$ denote the correct and noisy encoding for x, respectively. For instance, $\sigma_0 = 90$ means that 90% of the input data points were mapped to the correct bins. For frequency and mean estimation (Figures 3.4b and 3.4c), we measure the absolute error $|c - \tilde{c}|$ where c is the true answer and $\tilde{c}$ is the noisy output. For Figure 3.4d, we use the error metric $|c - \tilde{c}|/k$ where k is the size of the query. We report the mean and s.t.d of error values over 100 repetitions for every experiment.

Configuration. All experiments were conducted on a Macbook with i5, 8GB RAM and OS X Mojave (v10.14.6). We used Python 3.7.6. The reported privacy parameter ϵ refers to the ϵ-IND-FA-OCPA guarantee, and implies $\frac{\epsilon}{2}$-dDP and $\frac{\epsilon}{2}$-dLDP for OPϵ (Figure 3.2). We instantiate the OPϵc primitive using Alg. 3. Due to lack of space, we present the results

for all only two datasets, 1 dense (Adult, SPARCS) and 1 sparse (PUDF, Salary), in Figure 3.3. The default settings are $\epsilon = 1$, equi-depth partitioning of sizes $|\mathcal{P}| = 122$ for PUDF, $|\mathcal{P}| = 8$ for SPARC, $|\mathcal{P}| = 10$ for Adult, and $|\mathcal{P}| = 70$ for Salary. The range queries are chosen uniformly at random. We use the data from Salary for 2011 as an auxiliary dataset for Figure 3.4a.

3.8.2 Experimental Results



(a) Ordinal Queries

(b) Adult: Frequency Estimation

(c) Adult: Mean Estimation

(d) Adult: Range and Point Queries

Figure 3.4: Accuracy Analysis of OPϵc in the LDP setting

Utility and Overhead of OPϵ

Here, we evaluate **Q1** and **Q2** by computing the efficacy of OPϵ in retrieving the queried records of a range query. Recall, that a OPE scheme preserves the exact order of the plaintexts. Thus, the loss in accuracy (ordering information) arises solely from OPϵ's use of the OPϵc primitive. Hence first, we study the effect of the parameters of OPϵc.

We start with the privacy parameter, ϵ (Figs. 3.3a and 3.3b). We observe that OPϵ achieves high utility even at high levels of privacy. For example, OPϵ misses only about 2% of the correct records (i.e., $\rho_M = 2\%$) for PUDF (Figure 3.3a) on average even for a low value of

$\epsilon = 0.01$ (i.e., the ratio of the output distributions of two datasets that are 100-adjacent is bounded by e). The associated processing overhead is also reasonable – the size of the extra records retrieved, ρ_E, is around 1% of the total dataset size on average. Next, we observe that as the value of ϵ increases, both the number of missing and extra records drop. For instance, for $\epsilon = 1$ we have $\rho_M = 0.04\%$, i.e., only 4 in every $10K$ correct records are missed on average. Additionally, the number of extra records processed is just 0.3% of the total dataset size on average. We observe similar trends for SPARC (Figure 3.3b) as well. However, the utility for SPARC is lower than that for PUDF. For instance, $\rho_M = 10\%$ and $\rho_E = 35\%$ for $\epsilon = 0.01$ for SPARC. This is so because the ratio of the domain size (120) and partition size (8) for SPARC is smaller than that for PUDF (domain size $\sim 18K$, $|\mathcal{P}| = 122$). As a result, the individual intervals for SPARC are relatively small which results in lower utility.

Next, we study the effect of the size of the partition (number of intervals) on OPϵ's utility. As expected, we observe that for Adult, decreasing the partition size from 20 to 5 decreases ρ_M from 5% to 0.2% (Figure 3.3c). However, this increases the number of extra records processed – ρ_E increases from 0.8% to 7%. Similar trends are observed for Salary (Figure 3.3d).

Next, we study the effect of including neighboring intervals (Chapter 3.6) in Figures 3.3e and 3.3f. For instance, for Salary, including records from 2 extra neighboring intervals drops ρ_M from 1% to 0%. However, ρ_E increases from 0.4% to 1.7%. The increase in ρ_E is more significant for Adult. The reason is that the domain size for Adult is small and the dataset is dense. On the other hand, Salary has a larger and sparse domain.

Another way for improving utility is to answer a workload of range queries at a time (Chapter 3.6). We present the empirical results for this in Figures 3.3g and 3.3h. For SPARC, we observe that ρ_M and ρ_E drop from 0.9% to 0.1% and 4% to 0.2%, respectively as the size of the workload is increased from 1 to 20. Further, we note that this effect is more pronounced for SPARC than for PUDF. This is because, the domain of PUDF is larger and hence, the probability that the queried ranges in the workload are close to each other is reduced.

Utility of OPϵc in the LDP Setting

In what follows, we evaluate **Q3** by studying the utility of the OPϵc primitive in the LDP setting.

First, we consider ordinal queries. For Adult, we define an equi-length partition $\mathcal{P} = \{[1, 10], \cdots, [91, 100]\}$ over the domain and our query of interest is: *Which age group (as defined by $\mathcal{P}$) does each data point belong to?* For Salary, we define an equi-depth partition

of size 10 over the domain and our query of interest is: *Which decile does each data point belong to?* Our results are reported in Figure 3.4a. The first observation is that OPϵc reports the correct encodings with good accuracy. For instance, for $\epsilon = 1$, $\sigma_0 = 94.5\%$ and $\sigma_1 = 5.5\%$ for the Salary dataset. Another interesting observation is that for low values of ϵ, the accuracy for Salary is significantly higher than that for Adult. Specifically, for $\epsilon = 0.1$, $\sigma_0 = 90\%$ and $\sigma_0 = 35\%$ for Salary and Adult, respectively. The reason for this is two fold. Firstly, we use an auxiliary dataset for Salary to compute the weighted medians for the central tendencies. On the other hand, we do not use any auxiliary dataset for Adult and use the median of each interval as our measure for central tendency. Secondly, the domain size of Salary ($230K$) is relatively large compared to the number of intervals (10) which results in higher utility (as explained in Chapter 3.3.2).

Figure 3.4b shows our results for using OPϵc for frequency estimation. Baseline1 denotes the state-of-the art ϵ-LDP frequency oracle [WBLJ17b]. We observe that OPϵc achieves significantly lower error than Baseline1. For instance, the error of OPϵc is $6\times$ lower than that of Baseline1 for $\epsilon = 0.1$. This gain in accuracy is due to OPϵc's relaxed ϵ-dLDP guarantee.

From Figure 3.4c, we observe that the frequency oracle designed via OPϵc can be used for mean estimation with high utility. Here Baseline2 refers to the state-of-the-art protocol [DKY17] for mean estimation for ϵ-LDP. We observe that for $\epsilon = 0.1$, OPϵc achieves $\sim 40\times$ lower error than Baseline2.

Another interesting observation is that OPϵc's frequency oracle gives better accuracy for a range query of size k than k individual point queries (Figure 3.4d). For instance, a range query of size 20 gives $5\times$ lower error than 20 point queries. The reason behind this is that the output distribution of OPϵc is the exponential distribution centered at the input x. Hence, with high probability x either gets mapped to itself or some other point in its proximity. Thus, the probability for accounting for most copies of x is higher for the case of answering a range query $x \in [a, b]$ than for answering a point estimation for x.

An Illustration of OPϵ's Protection

Here, we give an illustration of OPϵ's protection against inference attacks on a real-world dataset. We use the "snapshot" attack model (the adversary obtains a onetime copy or snapshot of the encrypted data [FVY$^+$17]) for the ease of exposition. Our analysis is based on a formal model that captures a generic inference attack in the snapshot model – we create a bitwise leakage profile for the plaintexts from the revealed order and adversary's auxiliary knowledge as described below.

Model Description. We assume the input domain to be discrete, and finite and w.l.o.g denote it as $\mathcal{X} = [0, 2^{m-1}]$. Additionally, let $\mathcal{D}$ represent the true input distribution and $X = \{x_1, \cdots, x_n\}$ be a dataset of size n with each data point sampled i.i.d from $\mathcal{D}$. We model our adversary, $\mathcal{A}_{\mathsf{PPT}}$, to have access to (1) auxiliary knowledge about a distribution, $\mathcal{D}'$, over the input domain, $\mathcal{X}$ and (2) the ciphertexts, $\mathcal{C}$, corresponding to X which represent the snapshot of the encrypted data store. The adversary's goal is to recover as many bits of the plaintexts as possible. Let $X(i), i \in [n]$ represent the plaintext in X with rank [ran] i and let $X(i,j), j \in [m]$ represent the j-th bit for $X(i)$. Additionally, let $b(i,j)$ represent the adversary's guess for $X(i,j)$. Let $\mathcal{L}$ be a $n \times m$ matrix where $\mathcal{L}(i,j) = \Pr\Big[X(i,j) = b(i,j) \big| \mathcal{D}, \mathcal{D}'\Big]$ represent the probability that $\mathcal{A}_{\mathsf{PPT}}$ correctly identifies the j-th bit of the plaintext with rank i. Hence, $\mathcal{L}$ allows analysis of bitwise information leakage from $\mathcal{C}$ to $\mathcal{A}_{\mathsf{PPT}}$. The rationale behind using this model is that it *captures a generic inference attack in the snapshot model* allowing analysis at the granularity of bits.

Adversary's Approach. $\mathcal{A}_{\mathsf{PPT}}$'s goal is to produce their best guess for $X(i,j)$. Given $\mathcal{A}_{\mathsf{PPT}}$'s auxiliary knowledge about the distribution, $\mathcal{D}'$, the strategy with the least probabilistic error is as follows:

$$\begin{aligned} b(i,j) &= \arg\max_{b \in \{0,1\}} \big\{\Pr_{\mathcal{D}'}\big[X(i,j) = b\big]\big\}, i \in [n], j \in [m] \\ &= \begin{cases} 0 \text{ if } \mathbb{E}_{\mathcal{D}'}\big[X(i,j)\big] \leq 1/2 \\ 1 \text{ if } \mathbb{E}_{\mathcal{D}'}\big[X(i,j)\big] > 1/2 \end{cases} \end{aligned} \tag{3.33}$$

Next, we formalize $\mathcal{L}$ when X is encrypted under $\mathsf{OP}\epsilon$.

Theorem 11. *If X is encrypted under $\mathsf{OP}\epsilon$, then for all $i \in [n], j \in [m]$ we have*

$$\begin{aligned} \mathcal{L}(i,j) = \sum_{s \in S^j_{b(i,j)}} \Pr_{\mathcal{D}}\big[x = s\big] \Big(\sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \cdot \\ \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(s, \mathcal{P}, \epsilon) = v\big] / \Pr_{\mathcal{D}^*}\big[o = v\big] \Big) + \mathsf{negl}(\kappa) \end{aligned} \tag{3.34}$$

where $\widetilde{O}(r)$ denotes the encoding with rank $r, r \in [n]$, $\mathcal{P} \in \hat{\mathcal{X}}, o \sim \mathcal{D}^*$, and $\mathcal{D}^* : \mathcal{X} \mapsto \mathcal{O}$ represents the distribution of the encoding $\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon)$, $x \sim \mathcal{D}$ which is given as

$$\Pr_{\mathcal{D}^*}\big[v\big] = \sum_{x \in \mathcal{X}} \Pr_{\mathcal{D}}\big[x\big] \cdot \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = v\big], v \in \mathcal{O}$$

Proof. **Fact 1.** If $\mathcal{D}$ represents an input distribution and $X = \{x_1, \cdots, x_n\}$ denotes a dataset of size n with each data point sampled i.i.d from $\mathcal{D}$, then we have:

$$\Pr_{\mathcal{D}}\big[X(i,j) = b\big] = \sum_{s \in \mathcal{S}^j_b} \Pr_{\mathcal{D}}\big[X(i) = s\big]$$

where $i \in [n], j \in [m], b \in \{0,1\}$, and $\mathcal{S}_b^j = \{s | s \in \mathcal{X}$ and its j-th bit $s^j = b\}$.

Let $\mathcal{C}(i)$ represent the ciphertext with rank i in $\mathcal{C}$. Additionally, let $X'(i)$ represent the corresponding plaintext for $\mathcal{C}(i)$. From the IND-FA-OCPA guarantee, we observe that the rank of a ciphertext $y \in Y$ is equal to the rank of its corresponding plaintext in X, i.e, $X'(i) = X(i)$. Thus, we have this, we have

$$\begin{aligned}
\mathcal{L}(i,j) &= \Pr_{\mathcal{D}}\big[X'(i,j) = b(i,j)\big] + \mathsf{negl}(\kappa) \\
&\text{[The term } \mathsf{negl}(\kappa) \text{ accounts for the corresponding term in Eq. 3.5} \\
&\text{for the IND-OCPA guarantee of the OPE scheme.]} \\
&= \Pr_{\mathcal{D}}\big[X(i,j) = b(i,j)\big] \\
&= \sum_{s \in \mathcal{S}^j_{b(i,j)}} \Pr_{\mathcal{D}}\big[X(i) = s\big] \text{ [From Fact 1]} \qquad (3.35)
\end{aligned}$$

□

Eqs. 3.33 and 3.35 can be numerically computed using the following lemma.

Lemma 8. *If $\mathcal{D}$ represents an input distribution and $X = \{x_1, \cdots, x_n\}$ denotes a dataset of size n with each data point sampled i.i.d from $\mathcal{D}$, then we have:*

$$\Pr_{\mathcal{D}}\big[X(i) = s\big] = \begin{cases}
\sum_{j=n-i+1}^{n} \binom{n}{j} \cdot \Pr_{\mathcal{D}}\big[x < s\big]^{n-j} \cdot \Pr_{\mathcal{D}}\big[x = s\big]^{j} \\
\qquad \textbf{\textit{if}}\ \Pr_{\mathcal{D}}\big[x > s\big] = 0 \\
\sum_{j=i}^{n} \binom{n}{j} \cdot \Pr_{\mathcal{D}}\big[x = s\big]^{j} \cdot \Pr_{\mathcal{D}}\big[x > s\big]^{n-j} \\
\qquad \textbf{\textit{if}}\ \Pr_{\mathcal{D}}\big[x < s\big] = 0 \\
\sum_{j=1}^{n} \Bigg(\sum_{k=\max\{1,i-j+1\}}^{\min\{i,n-j+1\}} \Big(\binom{n}{k-1,j,n-k-i+1} \cdot \\
\qquad \Pr_{\mathcal{D}}\big[x < s\big]^{k-1} \cdot \Pr_{\mathcal{D}}\big[x = s\big]^{j} \cdot \\
\Pr_{\mathcal{D}}\big[x > s\big]^{n-k-j+1} \Big) \Bigg) \quad \textbf{\textit{otherwise}}
\end{cases}$$

where $x \sim \mathcal{D}, i \in [n]$ and $s \in \mathcal{X}$.

Proof. Let X_{sort} denote the sorted version of X. Additionally, let r_s^f and r_s^l denote the positions of the first and last occurrences of the value s in X_{sort}, respectively. Let cnt_s denote the count of data points with value s in X. Thus, clearly $cnt_s = r_s^l - r_s^f + 1$

Case I: $\Pr_{\mathcal{D}}\big[x > s\big] = 0$

In this case, we have

$$X(i) = s \implies X(r) = s, \forall r \text{ s.t } i \leq r \leq n$$

Thus, $r_s^l = n$ and $n - i + 1 \leq cnt_s \leq n$ and

$$\begin{aligned}\Pr_{\mathcal{D}}\big[X(i) = s\big] &= \sum_{j=n-i+1}^{n} \Pr_{\mathcal{D}}\big[X(i) = s | cnt_s = j\big] \\ &= \sum_{j=n-i+1}^{n} \binom{n}{j} \cdot \Pr_{\mathcal{D}}\big[x < s\big]^{n-j} \cdot \Pr_{\mathcal{D}}\big[x = s\big]^{j}\end{aligned}$$

Case II: $\Pr_{\mathcal{D}}\big[x < s\big] = 0$
In this case, we have

$$X(i) = s \implies X(r) = s, \forall r \text{ s.t } 1 \leq r \leq i$$

Thus, $r_s^f = 1$ and therefore $i \leq cnt_s \leq n$ and

$$\begin{aligned}\Pr_{\mathcal{D}}\big[X(i) = s\big] &= \sum_{j=i}^{n} \Pr_{\mathcal{D}}\big[X(i) = s | cnt_s = j\big] \\ &= \sum_{j=i}^{n} \binom{n}{j} \cdot \Pr_{\mathcal{D}}\big[x = s\big]^{j} \cdot \Pr_{\mathcal{D}}\big[x > s\big]^{n-j}\end{aligned}$$

Case III: Otherwise
For all other cases, if $cnt_s = j, j \in [n]$, then we must have $\max\{1, i - j + 1\} \leq r_s^l \leq \min\{i, n - j + 1\}$. Thus, we have

$$\begin{aligned}\Pr_{\mathcal{D}}\big[X(i) = s\big] &= \sum_{j=1}^{n} \Pr_{\mathcal{D}}\big[X(i) = s | cnt_s = j\big] \\ = \sum_{j=1}^{n} \Bigg(\sum_{k=\max\{1,i-j+1\}}^{\min\{i,n-j+1\}} \Big(\binom{n}{k-1, j, n-k-i+1} \cdot & \\ \Pr_{\mathcal{D}}\big[x < s\big]^{k-1} \cdot \Pr_{\mathcal{D}}\big[x = s\big]^{j} \cdot \Pr_{\mathcal{D}}\big[x > s\big]^{n-k-j+1} \Big) \Bigg) &\end{aligned}$$

□

Next, we formalize $\mathcal{L}$ when X is encrypted under a OPE scheme.

Theorem 12. *If X is encrypted under an OPE scheme that satisfies the IND-FA-OCPA guarantee, then for all $i \in [n], j \in [m]$ we have*

$$\mathcal{L}(i, j) = \sum_{s \in \mathcal{S}^j_{b(i,j)}} \Pr_{\mathcal{D}}\big[X(i) = s\big] + negl(\kappa) \tag{3.36}$$

where $x \sim \mathcal{D}$, and $\mathcal{S}_b^j = \{s | s \in \mathcal{X}$ and its j-th bit $s^j = b\}$.

Proof. The above theorem formalizes what $\mathcal{A}_{\mathsf{PPT}}$ can learn from just the order of the plaintexts (that is leaked by $\mathcal{C}$ by definition). Recall that in $\mathsf{OP}\epsilon$, the OPE scheme is applied to the encodings obtained from the $\mathsf{OP}\epsilon\mathsf{c}$ primitive. Thus, in this case, the ciphertexts $\mathcal{C}$ preserve the rank of the encodings of $\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon), x \in X$. Let $\widetilde{O}$ represent this set of encodings. Additionally, let $\widetilde{O}(i)$ be the encoding in $\widetilde{O}$ with rank i. Let $X''(i)$ represent the corresponding plaintext for the encoding $\widetilde{O}(i)$. Thus, for $s \in \mathcal{X}, x \sim \mathcal{D}$ and $o \sim \mathcal{D}^*$, we have

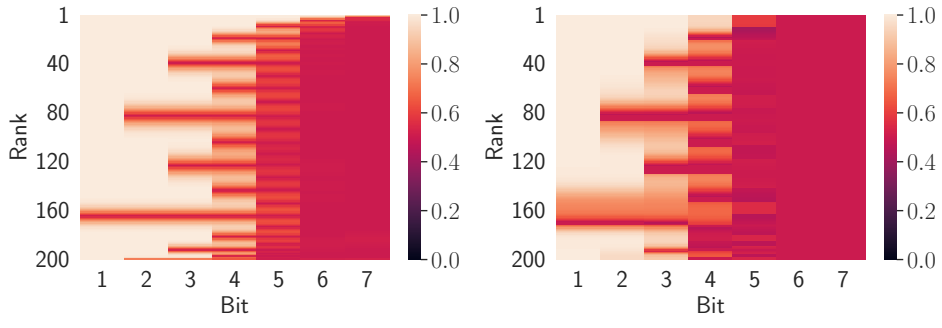
$$
\begin{aligned}
&\Pr_{\mathcal{D}}\big[X''(i) = s\big] = \\
&\qquad \sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \cdot \Pr_{\mathcal{D}}\big[X''(i) = s | \widetilde{O}(i) = v\big] \\
&= \sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \cdot \Pr_{\mathcal{D}}\big[x = s | \mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = v\big] \\
&= \sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \cdot \\
&\qquad \frac{\Pr_{\mathcal{D}}\big[\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = v | x = s\big] \cdot \Pr_{\mathcal{D}}\big[x = s\big]}{\Pr_{\mathcal{D}}\big[\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = v\big]} \\
&= \sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \cdot \\
&\qquad \frac{\Pr_{\mathcal{D}}\big[\mathsf{OP}\epsilon\mathsf{c}(x, \mathcal{P}, \epsilon) = v | x = s\big] \cdot \Pr_{\mathcal{D}}\big[x = s\big]}{\Pr_{\mathcal{D}^*}\big[o = v\big]} \\
&= \Pr_{\mathcal{D}}\big[x = s\big] \sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \frac{\Pr\big[\mathsf{OP}\epsilon\mathsf{c}(s, \mathcal{P}, \epsilon) = v\big]}{\Pr_{\mathcal{D}^*}\big[o = v\big]}
\end{aligned}
$$

Thus finally,

$$
\begin{aligned}
\mathcal{L}(i, j) &= \Pr\big[X''(i, j) = b(i, j)\big] + \mathsf{negl}(\kappa) \\
&\text{[The term } \mathsf{negl}(\kappa) \text{ accounts for the corresponding term in Eq. 3.13} \\
&\text{for the } \epsilon\text{-IND-FA-OCPA guarantee of the } \mathsf{OP}\epsilon \text{ scheme.]} \\
&= \sum_{s \in S^j_{b(i,j)}} \Pr\big[X''(i) = s\big] + \mathsf{negl}(\kappa) \\
&= \sum_{s \in S^j_{b(i,j)}} \Pr_{\mathcal{D}}\big[x = s\big] \Big(\sum_{v \in \mathcal{O}} \Pr_{\mathcal{D}^*}\big[\widetilde{O}(i) = v\big] \cdot \\
&\qquad \Pr\big[\mathsf{OP}\epsilon\mathsf{c}(s, \mathcal{P}, \epsilon) = v\big] / \Pr_{\mathcal{D}^*}\big[o = v\big] \Big) + \mathsf{negl}(\kappa)
\end{aligned}
$$

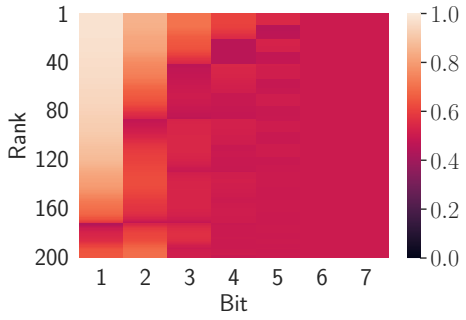
□

Remark. The bitwise leakage matrix, $\mathcal{L}$, captures the efficacy of *a generic inference attack in the snapshot model* at the granularity of the plaintext bits. We present this analysis to provide an *intuitive insight* into $\mathsf{OP}\epsilon$'s improved protection against inference attacks (given formally by Theorem 9).



(a) $\mathcal{L}$ for OPE

(b) $\mathcal{L}$ for OPϵ for $\epsilon = 1$



(c) $\mathcal{L}$ for OPϵ for $\epsilon = 0.1$

Figure 3.5: Numerical Analysis of the Bitwise Leakage Matrix, $\mathcal{L}$

Evaluation Results. Using these theorems, we analytically compute $\mathcal{L}$ for a real-world dataset. We use the age data from the Population dataset for the year 2019 as the true input distribution, $\mathcal{D}$, and the data from the year 2011 is considered to be the adversary's auxiliary distribution, $\mathcal{D}'$. We consider the dataset size to be 200 and the number of bits considered is 7 (domain of age is $[1, 100]$). Additionally, the partition $\mathcal{P}$ for the OPϵc primitive is set to be $\mathcal{P} = \mathcal{O} = \mathcal{X} = [1, 100]$. The reported privacy parameter ϵ refers to the ϵ-IND-FA-OCPA guarantee, and implies $\frac{\epsilon}{2}$-dDP and $\frac{\epsilon}{2}$-dLDP for OPϵ (Figure 3.2). As shown in Figure 3.5[9], we observe that the probability of successfully recovering the plaintext bits is significantly lower for OPϵ as compared to that of a OPE. Moreover, the probability of recovering the lower-order bits (bits in the right half) is lower than that of higher-order bits – the probability of recovering bits 5-7 is ≈ 0.5 which is the random guessing baseline. Recall that Thm. 9 implies that the adversary would not be able to distinguish between pairs of inputs that are close to each other. Hence, the above observation is expected since values that are close to each other are most likely to differ only in the lower-order bits. Additionally, as expected, the probability of the adversary's success decreases with decreasing value of ϵ. For instance, the average probability of success for the adversary for bit 4 reduces from 0.77 in the case of OPEs (Figure 3.5a) to 0.62 and 0.51 for $\epsilon = 1$ (Figure 3.5b) and $\epsilon = 0.1$

[9]For Figure 3.5, we omit the negl(κ) term from Eqs. 3.34 and 3.36

(Figure 3.5c), respectively, for $\mathsf{OP}\epsilon$. Concretely, *no inference attack in the snapshot model that uses the given auxiliary knowledge can distinguish between two age values* (x, x') *such that* $|x - x'| \leq 8$ *for* $\epsilon = 0.1$.

3.9 Discussion

$\mathsf{OP}\epsilon$ is the first step towards integrating OPEs and DP. Here, we discuss several avenues for future research.

Extension to Other Related Cryptographic Security Guarantees. The current scheme can be trivially extended to the IND-OCPA security guarantee [BCLO09, PLZ13b] for OPEs by replacing Definition 11 with a OPE scheme that satisfies IND-OCPA guarantee instead. Extension to order-revealing encryptions (ORE) [BLR$^+$15, LW16, CLWW16] is also straightforward – replacing the OPE by an ORE in the construction would work. The security guarantee again follows from the post-processing resilience of DP. Exploring connections with modular OPEs [MCO$^+$15, BCO11] is also an interesting future direction. The property of partial order preserving can provide protection against certain inference attacks. For example, some attacks require access patterns for uniformly random range queries [GLMP19b] or knowledge about the volume of every range query [GLMP18]. This is clearly not possible with $\mathsf{OP}\epsilon$ as only queries at the granularity of the chosen partition are permitted. Hence, another future direction could be formalizing this security gain parameterized on the choice of the partition. A related path to explore here could be studying connections with the existing notion of partially order preserving encoding POPE proposed by Roche et. al [RACY16]. A recent line of work has focused on providing formal guarantees against some specific types of attacks in the context of encrypted databases [GKL$^+$20, LP18, KT19, AHKM18b]. Our model is distinct from all the above mentioned approaches. Additionally, since the dDP guarantee holds regardless of the type of inference attacks, it would be interesting to see if it can be combined with the above approaches for a stronger formal guarantee or better efficiency.

Beyond OPEs, secure ordering could be required in a distributed setting where n mutually untrusting parties, each holding a data point, want to compute a sorted order over their data (generalization of the classic Yao's millionaires' problem [Yao86, JKU11]). OPEs are ill-suited for this setting because (1) currently OPEs are defined only in private key cryptography which means that a single malicious agent posing as a data owner can compromise the protocol (2) OPEs (satisfying ϵ-IND-FA-OCPAand IND-OCPA) are stateful and mutable [KT19, PLZ13b] which is not feasible in a distributed setting. This requires the use of multi party computation (MPC) techniques. A straightforward way to extend is to compute over the outputs of the $\mathsf{OP}\epsilon\mathsf{c}$ primitive. Proposing techniques for improved utility is an important future work.

Compromised Querier. In the context of a database encrypted under a OPE scheme, a querier has access only to the records that precisely belong to the queried range. However, in our setting the querier might know the values of some records that fall outside the queried range (Chapter 3.6). This might lead to additional leakage, when compared to the case of a OPE encrypted database, in the event the querier is compromised. One way to prevent this is to use an attribute-based encryption scheme [BSW07] for $\overline{\mathcal{E}}$ where the decryption is possible only if the record belongs to the queried range.

Support for Non-ordinal Data. Currently, ϵ-dLDP (equivalently dDP) provides a semantically useful privacy guarantee only for data domains that have a naturally defined order. A possible future direction can be exploring how to extend this guarantee for non-ordinal domains (like categorical data). One such way could be associating the categories of the non-ordinal domain with some ordinal features like popularity [GTT+19] and defining the guarantee w.r.t to these ordinal features instead.

Extension of LDP Mechanisms. The performance of the algorithms presented in Chapter 3.7 could be improved by borrowing techniques from the existing literature in LDP. For example, the partition for OPϵc could be learnt from the workload factorization mechanism from [MMMM20]. In another example, a B-ary tree could be constructed over the input domain using OPϵc for answering range queries [Kul19].

Less State Information for Clients. For OPϵ, in fact the clients need to store less state information than for OPEs satisfying IND-FA-OCPA as discussed below. Clients for any OPE scheme, $\mathcal{E}$, (satisfying the IND-FA-OCPA guarantee) need to store two pieces of state information for each unique value of $\mathcal{X}$ that appears in the dataset X to be encrypted (see [MRS18]). For example, for input domain $\mathcal{X} = [100]$ and a dataset $X = \{42, 45, 45, 50, 88, 67, 67, 77, 90, 98, 98, 98, 98\}$ drawn from this domain, the client needs to store two information $\{\max_{\mathcal{E}}(x), \min_{\mathcal{E}}(x)\}$ for $x \in \{42, 45, 50, 88, 67, 77, 90, 98\}$. Recall that OP$\epsilon$ applies OPE to the output of the OPϵc primitive. This implies that for OPϵ, $\mathcal{E}^{\dagger}$, the client needs to store the state information only for each encoding in $\mathcal{O}$ of the underlying OPϵc primitive. For the above mentioned example, consider a partition $\mathcal{P} = \{[1, 20], [21, 40], [41, 60], [61, 80], [81, 100]\}$ with corresponding encodings $\mathcal{O} = \{10, 30, 50, 70, 90\}$. Here, the client needs to store $\{\max_{\mathcal{E}^{\dagger}}(o), \min_{\mathcal{E}^{\dagger}}(o)\}$ only for $o \in \mathcal{O} = \{10, 30, 50, 70, 90\}$. This means that clients now need to store less state information for OPϵ than for OPE.

Encrypting Multiple Columns. For encrypting records with multiple columns, we can encrypt each column individually under the OPϵ scheme (satisfying ϵ-dLDP). Then, from

the composition theorem of dLDP, we would still enjoy $c \cdot \epsilon$-dLDP guarantee over the entire dataset where c is the total number of columns.

Extension of OPϵc. The OPϵc primitive can be extended to generic metric space along the lines of previous literature [ACPP18, CABP13]. This would support arbitrary partition instead of just non-overlapping intervals. For this, first sort the input domain $\mathcal{X}$ according to the metric $d(\cdot)$. Then divide the sorted domain, $\mathcal{X}_S$, into non-overlapping intervals which determines the partition, $\mathcal{P}$, for the OPϵc primitive. Algorithm 3 can now be defined on $\mathcal{P}$ with metric $d(\cdot)$.

Recent work in database theory community has explored efficient k-top query answering mechanisms [DK18, DHK20, DHK21a, DHK21b, DK21]. OPϵc can be used in conjunction with these mechanisms for guaranteeing data privacy.

Choice of Partition. as long as the encoding domain $\mathcal{O}$ of the underlying OPϵc primitive has enough wiggle room. For instance, for input domain $\mathcal{X} = [100]$, let the initial partition be $\mathcal{P} = \{[1, 20], [21, 40],$
$[41, 60], [61, 80], [81, 100]\}$ over a input domain $[100]$. Let the corresponding encodings be $\mathcal{O} = \{1, 21, 41, 61, 81\}$. Now, if in the future the interval $[1, 40]$ needs to be further partitioned into $\{[1, 10], [11, 20], [21, 30], [31, 40]\}$, it can be performed as follows:

1. retrieve and delete all records from the database in the range $[1, 40]$ (this step might incur some loss in accuracy)

2. assign the encodings $\{1, 11, 21, 31\}$ for the aforementioned sub-partition

3. insert back the records encrypted under the new encoding

However, the cost here is that every update consumes an additional ϵ-dLDP privacy budget for the updated records.

3.10 Related Work

The dLDP guarantee is equivalent to the notion of metric-based LDP [ACPP18] where the metric used is ℓ_1-norm. Further, metric-LDP is a generic form of Blowfish [HMD14] and d_χ-privacy [CABP13] adapted to LDP. Other works [BCSZ18, XDHZ19, ABCP13, CEP17, GTT+19, WNW+17] have also modelled the data domain as a metric space and scaled the privacy parameter between pairs of elements by their distance. A recent work [ABK+19] propose context-aware framework of LDP that allows a privacy designer to incorporate the application's context into the privacy definition.

A growing number of work has been exploring the association between differential privacy and cryptography [WHMM20]. Mironov et al. [MPRV09] introduced the notion of computational differential privacy where the privacy guarantee holds against a PPT adversary. A line of work [BEM+17, CSU+19] has used cryptographic primitives for achieving anonymity for privacy amplification in the LDP setting. Mazroom et al. [MG18b] have proposed techniques for secure computation with DP access pattern leakage. Bater et al. [BHT+18] combine differential privacy with secure computation for query performance optimization in private data federations. Groce et al. [GRR19b] show that allowing differentially private leakage can significantly improve the performance of private set intersection protocols. Vuvuzela [vdHLZZ15] is an anonymous communication system that uses differential privacy to enable scalability and privacy of the messages. Differential privacy has also been used in the context of ORAMs [CCMS19b, WCM18]. A parallel line of work involves efficient use of cryptographic primitives for differentially private functionalities. Agarwal et al. [AHKM18a] design encrypted databases that support differentially-private statistical queries, specifically private histogram queries. Rastogi et al. [RN10] and Shi et al. [SHCGR+11] proposed algorithms that allow an untrusted aggregator to periodically estimate the sum of n users' values in a privacy preserving fashion. However, both schemes are irresilient to user failures. Chan et al. [CSS12b] tackled this issue by constructing binary interval trees over the users. Böhler et al. [WLJ17] solves the problem of differentially private heavy hitter estimation in the distributed setting using secure computation. Recently, Humphries at al. [HMVK21] have proposed a solution for computing differentially private statistics over key-value data using secure computation. in the combined Additionally, recent works have combined DP and cryptography in the setting of distributed learning [KLS21a, ASY+18, CCDD+21].

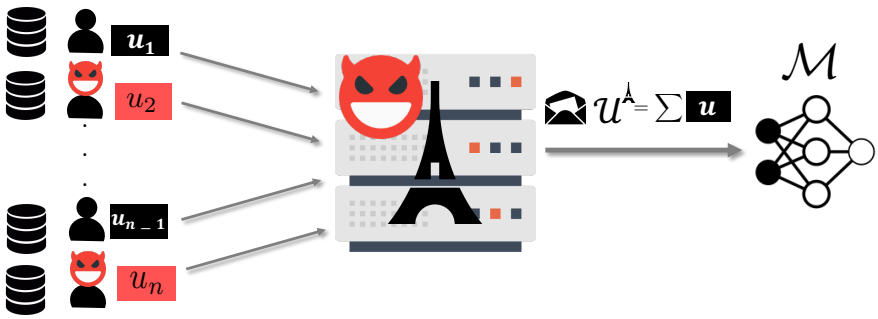
3.11 Conclusion

We have proposed a novel ϵ-dDP order preserving encryption scheme, OPϵ. OPϵ enjoys a formal guarantee of ϵ-dDP, in the least, even in the face of inference attacks. To the best of our knowledge, this is the first work to combine DP with a property-preserving encryption scheme. Additionally, OPϵ is based on a novel ϵ-dLDP order preserving encoding scheme, OPϵc, that can be of independent interest in the LDP setting. Our experimental results show that OPϵc and OPϵ achieve high utility on real-world datasets.

Chapter 4

EIFFeL: Ensuring Integrity for Federated Learning

Federated learning (FL; [MMR+17]) is a learning paradigm for decentralized data in which multiple clients collaborate with a server to train a machine-learning (ML) model. Each client computes an update on its *local* training data and shares it with the server; the server aggregates the local updates into a *global* model update. This allows clients to contribute to model training without sharing their private data. However, the local updates can still reveal information about a client's private data [MSDCS19, BDF+18, ZLH19, YMV+21, NSH19]. FL addresses this by using *secure aggregation*: clients mask the updates they share, and the server can *only* recover the final aggregate in the clear. A major challenge in FL is



Security Goal	Cryptographic Primitive
Input Privacy	Shamir's Threshold Secret Sharing Scheme [Sha79]
Input Integrity	Secret-Shared Non-Interactive Proof [CGB17] Verifiable Secret Shares [Fel87]

Figure 4.1: OPϵc performs secure aggregation of *verified* inputs in FL. The table lists its security goals and the cryptographic primitives we adopt to achieve them.

that it is vulnerable to Byzantine errors. In particular, malicious clients can inject poisoned updates into the learner with the goal of reducing the global model accuracy [BNL12, MZ15,

FCJG20, BCMC19, KMA+19] or implanting back doors in the model that can be exploited later [CLL+17, BVH+18, XHCL20]. Even a single malformed update can significantly alter the trained model [BMGS17b]. Thus, ensuring the well-formedness of the updates, *i.e.*, upholding their *integrity*, is essential for ensuring robustness in FL. This problem is especially challenging in the context of secure aggregation as the individual updates are masked from the server, which prevents audits on them.

These challenges in FL lead to the research question: *How can a federated learner efficiently verify the integrity of clients' updates without violating their privacy?*

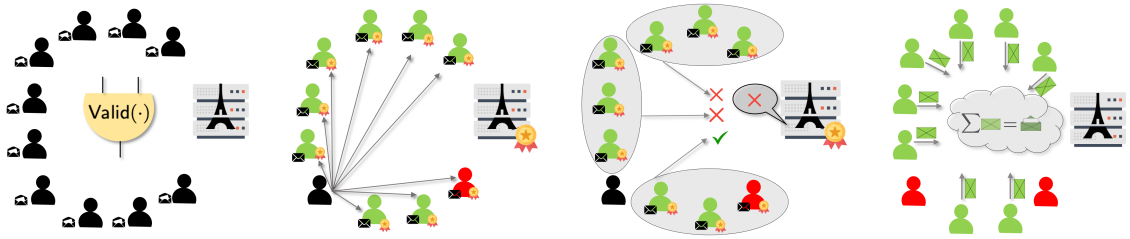
We formalize this problem by proposing *secure aggregation of verified inputs* (SAVI) protocols that: (1) securely verify the integrity of each local update, (2) aggregate *only* well-formed updates, and (3) release only the final aggregate in the clear. A SAVI protocol allows for checking the well-formedness of updates *without observing them*, thereby ensuring *both* the privacy and integrity of updates.

In order to demonstrate the feasibility of SAVI, we propose OPϵc: a system that instantiates a SAVI protocol that can perform *any integrity check that can be expressed as an arithmetic circuit with public parameters*. This provides OPϵc the flexibility to implement a plethora of modern ML approaches that ensure robustness to Byzantine errors by checking the integrity of per-client updates before aggregating them [SKSM19b, SKL17, XKG20, XKG19, LCW+20, DMG+18, BVH+18, SH21]. OPϵc is a general framework that empowers a federated learner to deploy (multiple) *arbitrary* integrity checks of their choosing on the "masked" updates.

OPϵc uses secret-shared non-interactive proofs (SNIP; [CGB17]) which are a type of zero-knowledge proofs that are optimized for the client-server setting. SNIP, however, requires multiple honest verifiers to check the proof. OPϵc extends SNIP to a *malicious* threat model by carefully *co-designing its architectural and cryptographic components*. Moreover, we develop a suite of optimizations that improve OPϵc's performance by at least 2.3×. Our empirical evaluation of OPϵc demonstrates its practicality for real-world usage. For instance, with 100 clients and a poisoning rate of 10%, OPϵc can train an MNIST classification model to the same accuracy as that of a non-poisoned federated learner in just 2.4 seconds per iteration.

4.1 Problem Overview

In this chapter, we introduce the problem setting, followed by its threat analysis and an overview of our solution.



(a) EIFFeL consists of multiple clients $\mathcal{C}$ and a server $\mathcal{S}$ with a public validation predicate $\mathsf{Valid}(\cdot)$ that defines the integrity check. A client $\mathcal{C}_i$ needs to provide a proof π_i for $\mathsf{Valid}(u_i) = 1$ (Round 1).

(b) For checking the proof π_i, all other clients $\mathcal{C}_{\setminus i}$ act as the verifiers under the supervision of $\mathcal{S}$. C_i splits its update u_i and proof π_i using Shamir's scheme with threshold $m+1$ and shares it with $\mathcal{C}_{\setminus i}$ (Round 2).

(c) Conceptually, any set of $m+1$ clients in $\mathcal{C}_{\setminus i}$ can emulate the SNIP verification protocol. The server uses this redundancy to *robustly* verify the proof (Round 3).

(d) The clients only aggregate the shares of well-formed updates and the resulting aggregate is revealed to the server (Round 4).

Figure 4.2: High-level overview of EIFFeL. See Chapter 4.1.4 for key ideas, and Chapter 4.3.3 for a detailed description of the system.

4.1.1 Problem Setting

In FL, multiple parties with distributed data jointly train a *global model*, $\mathcal{M}$, without explicitly disclosing their data to each other. FL has two types of actors:

- **Clients.** There are n clients where each client, $\mathcal{C}_i, i \in [n]$, owns a private dataset, D_i. The raw data is never shared, instead, every client computes a local update for $\mathcal{M}$, such as the average gradient, over the private dataset D_i.

- **Server.** There is a single *untrusted* server, $\mathcal{S}$, who coordinates the updates from different clients to train $\mathcal{M}$.

A single training iteration in FL consists of the following steps:

- **Broadcast.** The server broadcasts the current parameters of the model $\mathcal{M}$ to all the clients.

- **Local computation.** Each client $\mathcal{C}_i$ locally computes an update, u_i, on its dataset D_i.

- **Aggregation.** The server $\mathcal{S}$ collects the client updates and aggregates them, $\mathcal{U} = \sum_{i \in [n]} u_i$.
- **Global model update.** The server $\mathcal{S}$ updates the global model $\mathcal{M}$ based on the aggregated update $\mathcal{U}$.

In settings where there is a large number of clients, it is typical to subsample a small subset of clients to participate in a given iteration. We denote by n the number of clients that participate in each iteration and $\mathcal{C}$ denotes this subset of n clients, which the server announces at the beginning of the iteration.

4.1.2 Security Goals

- **Input Privacy (Client's Goal).** The first goal is to ensure privacy for all *honest* clients. That is, no party should be able learn anything about the raw input (update) u_i of an honest client C_i, other than what can be learned from the final aggregate $\mathcal{U}$.
- **Input Integrity (Server's Goal).** The server $\mathcal{S}$ is motivated to ensure that the individual updates from each client are well-formed. Specifically, the server has a *public* validation predicate, $\mathsf{Valid}(\cdot)$, that defines a syntax for the inputs (updates). An input (update) u is considered valid and, hence, passes the integrity check iff $\mathsf{Valid}(u) = 1$. For instance, any per-client update check, such as Zeno++ [XKG20], can be a good candidate for $\mathsf{Valid}(\cdot)$ (we evaluate four such validation predicates in Chapter 4.6.2).

We assume that the honest clients, denoted by $\mathcal{C}_H$: (1) follow the protocol correctly, *and* (2) have well-formed inputs. We require the second condition because, in case the input of an honest client does not pass the integrity check (which can be verified locally since $\mathsf{Valid}(\cdot)$ is public), the client has no incentive to participate in the training iteration.

4.1.3 Threat Model

We consider a *malicious adversary* threat model:

- **Malicious Server.** We consider a malicious server that can deviate from the protocol arbitrarily with the aim of recovering the raw updates u_i for $i \in [n]$.
- **Malicious Clients.** We also consider a set of m malicious clients, $\mathcal{C}_M$. Malicious clients can arbitrarily deviate from the protocol with the goals of: (1) sending malformed inputs to the server and compromising the final aggregate; (2) failing the integrity check of an honest client that submits well-formed updates; (3) violating the privacy of an honest client, potentially in collusion with the server.

4.1.4 Solution Overview

Prior work on FL has mostly focused on ensuring input privacy via secure aggregation, *i.e.*, securely computing the aggregate $\mathcal{U} = \sum_{\mathcal{C}_i \in \mathcal{C}} u_i$. Motivated by the above problem setting and threat analysis, we introduce a new type of FL protocol, called *secure aggregation with verified inputs* (SAVI), that ensures *both* input privacy and integrity. The goal of a SAVI protocol is to securely aggregate *only* well-informed inputs.

In order to demonstrate the feasibility of SAVI, we propose EIFFeL: a system that instantiates a SAVI protocol for any Valid$(\cdot)$ that can be expressed as an arithmetic circuit with public parameters (Figure 4.1). EIFFeL ensures input privacy by using Shamir's threshold secret sharing scheme [Sha79] (Chapter 4.3.1). Input integrity is guaranteed via SNIP and verifiable secret shares (VSS) which validates the correctness of the secret shares (Chapter 4.3.1). The key ideas are:

- SNIP requires multiple honest verifiers. EIFFeL enables this in a single-server setting by having the clients act as the verifiers for each other under the supervision of the server (in Figure 4.2b, verifiers are marked by 🏅).
- EIFFeL extends SNIP to the malicious threat model to account for the malicious clients. Our key observation is that using a threshold secret sharing scheme creates multiple subsets of clients that can emulate the SNIP verification protocol. The server uses this redundancy to robustly verify the proofs and aggregate updates with verified proofs *only* (Figure 4.2c and 4.2d).

4.2 Secure Aggregation with Verified Inputs

Below, we provide the formal definition of a *secure aggregation with verified inputs* (SAVI) protocol.

Definition 13. *Given a public validation predicate* $\mathsf{Valid}(\cdot)$ *and security parameter* κ*, a protocol* $\Pi(u_1, \cdots, u_n)$ *is a secure aggregation with verified inputs (SAVI) protocol if:*

- ***Integrity.*** *The output of the protocol,* out*, returns the aggregate of a subset of clients,* $\mathcal{C}_{\mathsf{Valid}}$*, such that all clients in* $\mathcal{C}_{\mathsf{Valid}}$ *have well-formed inputs.*

$$\Pr\left[\mathsf{out} = \mathcal{U}_{\mathsf{Valid}}\right] \geq 1 - \mathrm{negl}(\kappa) \text{ where } \mathcal{U}_{\mathsf{Valid}} = \sum_{\mathcal{C}_i \in \mathcal{C}_{\mathsf{Valid}}} u_i$$

$$\text{for all } \mathcal{C}_i \in \mathcal{C}_{\mathsf{Valid}} \text{ we have } \mathsf{Valid}(u_i) = 1, \mathcal{C}_H \subseteq \mathcal{C}_{\mathsf{Valid}} \subseteq \mathcal{C} \tag{4.1}$$

- ***Privacy.*** *For a set of malicious clients $\mathcal{C}_M$ and a malicious server $\mathcal{S}$, there exists a probabilistic polynomial-time (P.P.T.) simulator* $\mathrm{Sim}(\cdot)$ *such that:*

$$\mathrm{Real}_{\Pi}\big(\{u_{\mathcal{C}_H}\}, \Omega_{\mathcal{C}_M \cup \mathcal{S}}\big) \equiv_C \mathrm{Sim}\big(\Omega_{\mathcal{C}_M \cup \mathcal{S}}, \mathcal{U}_H, \mathcal{C}_H\big) \text{ where } \mathcal{U}_H = \sum_{\mathcal{C}_i \in \mathcal{C}_H} u_i. \tag{4.2}$$

  $\{u_{\mathcal{C}_H}\}$ *denotes the input of all the honest clients,* Real_{Π} *denotes a random variable representing the joint view of all the parties in* Π*'s execution,* $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$ *indicates a polynomial-time algorithm implementing the "next-message" function of the parties in* $\mathcal{C}_M \cup \mathcal{S}$ *(see Chapter 4.4), and* $\equiv_C$ *denotes computational indistinguishability.*

From Definition 13, the output of a SAVI protocol is of the form:

$$\mathcal{U}_v = \underbrace{\mathcal{U}_H}_{\substack{\text{well-formed updates of} \\ \textit{all} \text{ honest clients } \mathcal{C}_H}} + \underbrace{\sum_{\mathcal{C}_i \in \mathcal{C}_{\mathsf{Valid}} \setminus \mathcal{C}_H} u_i}_{\substack{\text{well-formed updates of} \\ \text{some malicious clients}}} \tag{4.3}$$

The clients in $\mathcal{C}_{\mathsf{Valid}} \setminus \mathcal{C}_H$ are clients who have submitted well-formed inputs but can behave maliciously otherwise (*e.g.*, by violating input privacy/integrity of honest clients).

The privacy constraint of the SAVI protocol means that a simulator Sim can generate the views of all parties with just access to the list of the honest clients $\mathcal{C}_H$ and their aggregate $\mathcal{U}_H$. Note that Sim takes $\mathcal{U}_H$ as an input instead of the protocol output $\mathcal{U}_{\mathsf{Valid}}$. This is because the clients in $\mathcal{C}_{\mathsf{Valid}} \setminus \mathcal{C}_H$, by virtue of being malicious, can behave arbitrarily and announce their updates to reveal $\mathcal{U}_H = \mathcal{U}_{\mathsf{Valid}} - \sum_{\mathcal{C}_i \in \mathcal{C}_{\mathsf{Valid}} \setminus \mathcal{C}_H} u_i$. Thus, SAVI ensures that nothing can be learned about the input u_i of an honest client $\mathcal{C}_i \in \mathcal{C}_H$ except:

- that u_i is well-formed, *i.e.*, $\mathsf{Valid}(u_i) = 1$,
- anything that can be learned from the aggregate $\mathcal{U}_H$.

Remark 1. The integrity constraint of SAVI requires the protocol to detect *and* remove *all* malformed inputs before computing the final aggregate. Note that there is a fundamental difference between the design choice of just detection of a malformed input versus detection *and* removal. In the former, the server can only abort the current round even when a *single* malformed input is detected. This allows an adversary to stage a denial-of-service attack that renders the server unable to train the model. When the protocol can both detect and remove malformed inputs, such denial-of-service attacks are impossible as the server can train the model using just the valid updates.

4.3 EIFFeL System Description

This section introduces EIFFeL: the system we propose to perform secure aggregation of verified inputs.

4.3.1 Cryptographic Building Blocks

Arithmetic Circuit. An arithmetic circuit, $\mathcal{C} : \mathbb{F}^k \mapsto \mathbb{F}$, represents a computation over a finite field $\mathbb{F}$. It can be represented by a directed acyclic graph (DAG) consisting of three types of nodes: (1) inputs, (2) gates and (3) outputs. Input nodes have in-degree zero and out-degree one: the k input nodes return input variables $\{x_1, \cdots, x_k\}$ with $x_i \in \mathbb{F}$. Gate nodes have in-degree two and out-degree one; they perform either the $+$ operation (addition gate) or the $\times$ operation (multiplication gate). Every circuit has a single output node with out-degree zero. A circuit is evaluated by traversing the DAG, starting from the inputs, and assigning a value in $\mathbb{F}$ to every wire until the output node is evaluated.

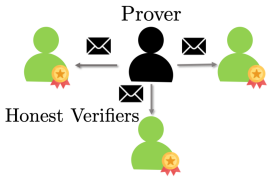
Shamir's t-out-of-n Secret Sharing Scheme [Sha79] allows distributing a secret s among n parties such that: (1) the complete secret can be reconstructed from any combination of t shares; and (2) any set of $t-1$ or fewer shares reveals no information about s. Herein, t is the *threshold* of the secret sharing scheme. The scheme is parameterized over a finite field $\mathbb{F}$ and consists of a tuple of two algorithms:

- *Construct shares* ($\mathsf{SS.share}$). Given a secret $s \in \mathbb{F}$, a set of n unique field elements $P \in \mathbb{F}^n$ and a threshold t with $t \leq n$, this algorithm constructs n shares, $\{(i, s_i)\}_{i \in P} \xleftarrow{\$} \mathsf{SS.share}(s, P, t)$. The algorithm chooses a random polynomial $c \in \mathbb{F}[X]$ such that $c(0) = s$ and generates the shares as $(i, c(i)), i \in P$.

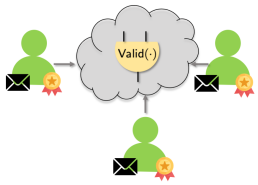
- *Reconstruct secret* ($\mathsf{SS.recon}$). Given the shares corresponding to a subset $Q \subseteq P, |Q| \geq t$, the reconstruction algorithm recovers the secret, $s \leftarrow \mathsf{SS.recon}(\{(i, s_i)_{i \in Q}\})$.

Shamir's secret sharing scheme is *linear*, which means a party can *locally* perform: (1) addition of two shares, (2) addition of a constant, and (3) multiplication by a constant.
Shamir's secret sharing scheme is closely related to Reed-Solomon error correcting codes [LC04], which is a group of polynomial-based error correcting codes. The share generation is similar to (non-systemic) message encoding in these codes which can successfully recover a message even in the presence of errors and erasures (message dropouts). Consequently, we can leverage Reed-Solomon decoding algorithms for robust reconstruction of Shamir's secret shares:

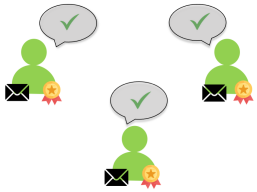
- *Robust Reconstruction* ($\mathsf{SS.robustRecon}$). Shamir's secret sharing scheme results in a $[n, t, n-t+1]$ Reed-Solomon code that can tolerate up to q errors and e erasures



(a) Prover sends secret shares of its input and the SNIP proof to multiple verifiers.



(b) The verifiers gossip among themselves and check the proof.



(c) The check passes successfully if all parties are honest.

Figure 4.3: High-level overview of a secret-shared non-interactive proof (SNIP; [CGB17]).

(message dropouts) such that $2q + e < n - t + 1$. Given any subset of $n - e$ shares $Q \subseteq P, |Q| \geq n - e$ with up to q errors, any standard Reed Solomon decoding algorithm [Bla83] can robustly reconstruct $s \leftarrow \mathsf{SS.robustRecon}(\{(i, s_i)\}_{i \in Q})$. EIFFeL uses Gao's decoding algorithm [Gao03].

Verifiable secret sharing scheme is a related concept where the scheme has an additional property of *verifiability*. Given a share of the secret, a party must be able to check whether it is indeed a valid share. If a share is valid, then there exists a unique secret which will be the output of the reconstruction algorithm when run on any t distinct valid shares. Formally:

- *Verify shares* (SS.verify). The verification algorithm inputs a share and a check string Ψ_s such that

$$\exists s \in \mathbb{F}, \forall V \subset \mathbb{F} \times \mathbb{F} \text{ where } |V| = t, \text{ s.t.}$$
$$(\forall (i, v) \in V, \mathsf{SS.verify}((i, v), \Psi_s) = 1) \implies \mathsf{SS.recon}(V) = s$$

  The share construction algorithm is augmented to output the check string, $(\{(i, s_i)_{i \in P}\}, \Psi_s) \leftarrow \mathsf{SS.share}(s, P, t)$.

For EIFFeL, we use the non-interactive verification scheme by Feldman [Fel87]. Let $c(x) = c_0 + c_1 x + \cdots c_{t-1} x^{t-1}$ denote the polynomial used in generating the shares where $c_0 = s$ is the secret. The check string are the commitments to the coefficients given by

$$\psi_i = g^{c_i}, i \in \{0, \cdots, t-1\} \tag{4.4}$$

where g denotes a generator of $\mathbb{F}$. All arithmetic is taken modulo q such that $(p|q-1)$ where p is the prime of $\mathbb{F}$. For verifiying a share (j, s_j), a party needs to check whether $g^{s_j} = \prod_{i=0}^{t-1} \psi_i^{j^i}$. The privacy of the secret $s = c_0$ is implied by the the intractability of computing discrete logarithms [Fel87].

Key Agreement Protocol. A key agreement protocol consists of a tuple of the following three algorithms:

- *Parameter Generation* (KA.param). The parameter generation algorithm samples a set of public parameters pp that have security parameter κ, $(pp) \xleftarrow{\$} \mathsf{KA.param}(1^\kappa)$.
- *Key Generation* (KA.Gen). The key generation algorithm samples a public/secret key pair from the public parameters, $(pk, sk) \xleftarrow{\$} \mathsf{KA.gen}(pp)$.
- *Key Agreement* (KA.agree). The key agreement protocol receives a public key pk_i and a secret key sk_j as input and generates the shared key, $sk_{ij} \leftarrow \mathsf{KA.agree}(pk_i, sk_j)$.

Authenticated Encryption combines confidentiality and integrity guarantees for messages exchanged between two parties. It consists of a tuple of three algorithms as follows:

- *Key Generation* (AE.gen). The key generation algorithm that outputs a private key k where κ is the security parameter, $k \xleftarrow{\$} \mathsf{AE.gen}(1^\kappa)$.
- *Encryption* (AE.enc). The encryption algorithm takes as input a key k and a message x, and outputs a ciphertext, $\overline{x} \xleftarrow{\$} \mathsf{AE.enc}(k, x)$.
- *Decryption* (AE.dec). The decryption algorithm takes as input a ciphertext and a key and outputs either the original plaintext, or a special error symbol $\perp$ on failure, $x \leftarrow \mathsf{AE.dec}(k, \overline{x})$.

Security relies on indistinguishability under a chosen ciphertext attack (IND-CCA) [KL14b].

Secret-shared Non-interactive Proofs. The secret-shared non-interactive proof (SNIP) [CGB17] is an information-theoretic zero-knowledge proof for distributed data (Figure 4.3). SNIP is designed for a multi-verifier setting where the private data is distributed or secret-shared among the verifiers. Specifically, SNIP relies on an additive secret sharing scheme over a field $\mathbb{F}$ as described below. A secret $s \in \mathbb{F}$ is split into k random shares $([s]_1, \cdots, [s]_k)$ such that $\sum_{i=1}^{k} [s]_i = s$. A subset of up to $k-1$ shares reveals *no* information about the secret s. The additive secret-sharing scheme is linear as well.

SNIP Setting. SNIP considers a setting with $k > 2$ verifiers $\{\mathcal{V}_i\}, i \in [k]$ and a prover $\mathcal{P}$ with a private vector $x \in \mathbb{F}^d$. Additionally, all parties hold a *public* arithmetic circuit representing a validation predicate $\mathsf{Valid} : \mathbb{F}^d \mapsto \mathbb{F}$. Let M be the number of multiplication gates in $\mathsf{Valid}(\cdot)$. $\mathbb{F}$ is chosen such that $2\mathsf{M} \ll |\mathbb{F}|$. The prover $\mathcal{P}$ splits x into k shares $\{[x_1], \cdots, [x_k]\}$. Next, they generate k proof strings $\pi_i, i \in [k]$ based on $\mathsf{Valid}(\cdot)$ and shares $([x_i], \pi_i)$ with every verifier $\mathcal{V}_i$.

The prover's goal is to convince the verifiers that, indeed, $\mathsf{Valid}(x) = 1$. The prover does so via proof strings $\pi_i, i \in [k]$, that do not reveal anything else about x. After receiving the proof, the verifiers gossip with each other to conclude either that $\mathsf{Valid}(x) = 1$ (the verifiers "Accept x") or not ("Reject x"). Formally, SNIP satisfies the following properties:

- *Completeness.* If all parties are honest and $\mathsf{Valid}(x) = 1$, then the verifiers will accept x.

$$\forall x \in \mathbb{F} \text{ s.t. } \mathsf{Valid}(x) = 1 : \Pr_{\pi}[\mathsf{Accept}\ x] = 1.$$

- *Soundness.* If all verifiers are honest, and if $\mathsf{Valid}(x) = 0$, then for all malicious provers, the verifiers will reject x with overwhelming probability.

$$\forall x \in \mathbb{F} \text{ s.t. } \mathsf{Valid}(x) = 0 : \ \Pr_{\pi}\big[\mathsf{Reject}\ x\big] \geq 1 - {}^{2|M|-2}/_{|\mathbb{F}|}.$$

- *Zero knowledge.* If the prover and at least one verifier are honest, then the verifiers learn nothing about x, except that $\mathsf{Valid}(x) = 1$. Formally, when $\mathsf{Valid}(x) = 1$, there exists a simulator $\mathrm{Sim}(\cdot)$ that can simulate the view of the protocol execution for every proper subset of verifiers:

$$\forall x \text{ s.t. } \mathsf{Valid}(x) = 1 \text{ and } \forall \bar{\mathcal{V}} \subset \bigcup_{i=1}^{k} \mathcal{V}_i \text{ we have}$$
$$\mathrm{Sim}_{\pi}\big(\mathsf{Valid}(\cdot), \{([x]_i, \pi_i)\}_{\mathcal{V}_i \in \bar{\mathcal{V}}}\big) \equiv \mathrm{View}_{\pi,\bar{\mathcal{V}}}(\mathsf{Valid}(\cdot), x)$$

SNIP works in two stages as follows:
(1) *Generation of Proof.* For generating the proof, the prover $\mathcal{P}$ first evaluates the circuit $\mathsf{Valid}(\cdot)$ on its input x to obtain the value of every wire in the arithmetic circuit corresponding to the computation of $\mathsf{Valid}(x)$. Using these wire values, $\mathcal{P}$ constructs three polynomials f, g, and h such that $h = f \cdot g$ and $f(i), g(i)$ and $h(i), i \in \big[|\mathsf{M}|\big]$ encode the values of the two input wires and one output wire of the i-th multiplication gate, respectively. $\mathcal{P}$ also samples a single set of Beaver's multiplication triples [Bea92]: $(a, b, c) \in \mathbb{F}^3$ such that $a \cdot b = c \in \mathbb{F}$. Finally, it generates the shares of the proof, $[\pi]_i = \big([h]_i, ([a]_i, [b]_i, [c]_i)\big)$, which consists of:

- shares of the coefficients of the polynomial h, denoted by $[h]_i$, and
- shares of the Beaver's triples, $([a]_i, [b]_i, [c]_i) \in \mathbb{F}^3$.

The prover then sends the respective shares of the input and the proof $([x]_i, [\pi]_i)$ to each of the verifiers $\mathcal{V}_i$.

(2) *Verification of Proof.* To verify that $\mathsf{Valid}(x) = 1$ and hence, accept the input x, the verifiers need to check two things:

- check that the value of final output wire of the computation, $\mathsf{Valid}(x)$, denoted by w^{out} is indeed 1, and

- check the consistency of $\mathcal{P}$'s computation of $\mathsf{Valid}(x)$.

To this end, each verifier $\mathcal{V}_i$ *locally* constructs the shares of every wire in $\mathsf{Valid}(x)$. This can be done via affine operations on the shares of the private input $[x]_i$ and $[h]_i$ (see [CGB17] for details). Next, $\mathcal{V}_i$ broadcasts a summary $\sigma_i = ([w^{out}]_i, [\lambda]_i)$, where $[w^{out}]_i$ is $\mathcal{V}_i$'s share of the output wire and $[\lambda]_i$ is a share of a random digest that the verifier computes from the shares of the other wire values and the proof π_i. Using these broadcasted summaries, the verifiers check the proof as follows:

- For checking the output wire, the verifiers can reconstruct its exact value from all the broadcasted shares $w^{out} = \sum_{i=1}^{k} [w^{out}]_i$ and check whether $w^{out} = 1$.

- The circuit consistency check is more involved and is performed using the random digest λ. First, $\mathcal{V}_i$ *locally* computes the shares of the polynomials f and g (denoted as $[f]_i$ and $[g]_i$) (see [CGB17]). To verify the circuit consistency, the verifiers need to check that the shares $[h]_i$ sent by the prover $\mathcal{P}$ are of the correct polynomial, *i.e.*, confirm that $f \cdot g = h$. For this, SNIP uses the Schwartz-Zippel polynomial identity test [Sch80, Zip79]. Specifically, verifiers test whether $\lambda = f(r) \cdot g(r) - h(r) = 0$ on a randomly selected $r \in \mathbb{F}$. The computation of the random digest λ uses the shares of the Beaver's triples $([a]_i, [b]_i, [c]_i)$. This multiplication requires one round of communication between the verifiers.

4.3.2 System Building Blocks

Public Validation Predicate. EIFFeL requires a public validation predicate $\mathsf{Valid}(\cdot)$, expressed by an arithmetic circuit, that captures the notion of update well-formedness. In principle, any per-client update robustness test [SKSM19b, SKL17, XKG20, LCW+20, DMG+18, BVH+18, SH21] from the ML literature can be a suitable candidate. The parameters of the test (for instance, the threshold ρ for some bound $\psi(u) \leq \rho$) can be computed from a clean, public dataset $\mathcal{D}_P$ that is available to the server $\mathcal{S}$. This assumption of a clean, public dataset is common in both ML [XKG20, CFLG21, KMA+19] as well as privacy literature [LVS+21, BCM+20, BKN+20]. The dataset can be small and obtained by manual labeling [MR17].

Public Bulletin Board. EIFFeL assumes the availability of a public bulletin board $\mathcal{B}$ that is accessible to all the parties. This is similar to prior work [RNFH19, BIK+17, KMA+19]. In practice, the bulletin $\mathcal{B}$ can be implemented as an append-only log hosted at a public web

address where every message and its sender is visible. Every party in EIFFeL has read/write access to it. We use the bulletin $\mathcal{B}$ as a tool for broadcasting [BT85, CW09].

4.3.3 EIFFeL Workflow

The goal of EIFFeL is to instantiate a secure aggregation with verified inputs (SAVI) protocol in FL. For a given public validation predicate $\mathsf{Valid}(\cdot)$, EIFFeL checks the integrity of every client update using SNIP and outputs the aggregate of *only* well-formed updates, *i.e.*, $\mathsf{Valid}(u) = 1$. To implement the SNIP for our setting, EIFFeL introduces two key ideas:

Key Ideas.

- In EIFFeL, the clients act as verifiers for each other. Specifically, for every client $\mathcal{C}_i, i \in \mathcal{C}$, all of the other $n-1$ clients, $\mathcal{C}_{\setminus i}$, and the server $\mathcal{S}$ jointly acts as the verifiers. This is different from systems like Prio [CGB17] (the original deployment setting for SNIP) that use multiple *honest servers* to perform verification.
- In EIFFeL, verification can be performed even in the presence of malicious verifiers. This is essential in our setting since we have m malicious clients (*i.e.*, verifiers). For this, EIFFeL uses Shamir's t-out-of-n secret scheme. This allows any cohort of t verifiers to reconstruct a secret and, hence, instantiate a SNIP. If $t < n$, we have multiple such instantiations and can use the redundancy to perform the integrity check even in the presence of malicious verifiers.

EIFFeL is carefully designed such that (1) it can efficiently distribute the computational burden of verifying the integrity checks between the clients and the server, and (2) all protocol interactions can integrate seamlessly with the FL architecture.

The full protocol is presented in Figure 4.4. The protocol involves a setup phase followed by four rounds.

Setup Phase. In the setup phase, all parties are initialized with the system-wide parameters, namely the security parameter κ, the number of clients n out of which *only* $m < \lfloor \frac{n-1}{3} \rfloor$ can be malicious, public parameters for the key agreement protocol $pp \xleftarrow{\$} \mathsf{KA.param}(\kappa)$, and a field $\mathbb{F}$ where $|\mathbb{F}| \geq 2^{\kappa}$. EIFFeL works in a synchronous protocol between the server $\mathcal{S}$ and the n clients in four rounds. To prevent the server from simulating an arbitrary number of clients, the clients register themselves with a specific user ID on the public bulletin board $\mathcal{B}$ and are authenticated with the help of standard public key infrastructure (PKI). The bulletin board $\mathcal{B}$ allows parties to register IDs only for themselves, preventing impersonation. More concretely, the PKI enables the clients to register identities (public keys), and sign messages using their identity (associated secret keys), such that others can verify this signature, but

cannot impersonate them [KL14b]. We omit this detail for the ease of exposition. For notational simplicity, we assume that each client $\mathcal{C}_i$ is assigned a unique logical ID in the form of an integer i in $[n]$. Each client holds as input a d-dimensional vector $u_i \in \mathbb{F}^d$ representing its local update. All clients have a private, authenticated communication channel with the server $\mathcal{S}$. Additionally, every party (clients and server) has read and write access to the public bulletin $\mathcal{B}$ via authenticated channels. For every client $\mathcal{C}_i$, the server $\mathcal{S}$ maintains a list, $\mathsf{Flag}[i]$, of all clients that have flagged $\mathcal{C}_i$ as malicious. All $\mathsf{Flag}[i]$ lists are initialized to be empty lists.

Round 1 (Announcing Public Information). In the first round, all the parties announce their public information relevant to the protocol on the public bulletin $\mathcal{B}$. Specifically, each client $\mathcal{C}_i$ generates its key pair $(pk_i, sk_i) \xleftarrow{\$} \mathsf{KA.gen}(pp)$ and advertises the public key pk_i on the public bulletin $\mathcal{B}$. The server $\mathcal{S}$ publishes the validation predicate $\mathsf{Valid}(\cdot)$ on $\mathcal{B}$.

Round 2 (Generate and Distribute Proofs). Every client generates shares of its private update u_i and the proof π_i, and distributes these shares to the other clients $\mathcal{C}_{\setminus i}$. First, client $\mathcal{C}_i$ generates a common pairwise encryption key sk_{ij} for every other client $\mathcal{C}_j \in \mathcal{C}_{\setminus i}$ using the key agreement protocol, $sk_{ij} \leftarrow \mathsf{KA.agree}(sk_i, pk_j)$. Next, the client generates the secret shares of its private update $\{(1, u_{i1}), \cdots, (n, u_{in}), \Psi_i^u\} \xleftarrow{\$} \mathsf{SS.share}(u, [n], m+1)$. The sharing of u_i is performed dimension-wise; we abuse notations and denote the j-th such share by $(j, u_{ij}), j \in [n]$. Note that the client $\mathcal{C}_i$ generates a share (i, u_{ii}) for *itself* as well which will be used later in the protocol. Next, the client $\mathcal{C}_i$ generates the proof for the computation $\mathsf{Valid}(u_i) = 1$. Specifically, it computes the polynomials f_i, g_i, and $h_i = f_i \cdot g_i$ and samples a set of Beaver's multiplication triples $(a_i, b_i, c_i) \in \mathbb{F}^3, a_i \cdot b_i = c_i \in \mathbb{F}$. Since the other clients will verify the proof, client $\mathcal{C}_i$ then splits the proof to generate shares $\pi_{ij} = \big((j, h_{ij}), (j, a_{ij}), (j, b_{ij}), (j, c_{ij})\big)$ for every other client $\mathcal{C}_j \in \mathcal{C}_{\setminus i}$. Herein, the shares themselves are generated via $\{(1, h_{i1}), \cdots, (i-1, h_{i(i-1)}), (i+1, h_{i(i+1)}), \cdots, (n, h_{in}), \Psi_i^h\} \xleftarrow{\$} \mathsf{SS.share}(h_i, [n] \setminus i, m+1)$, and so on. Finally, the client encrypts the proof strings (shares of the update u_i and the proof π_i) using the corresponding pairwise secret key, $\overline{(j, u_{ij})||(j, \pi_{ij})} \xleftarrow{\$} \mathsf{AE.enc}\big(sk_{ij}, (j, u_{ij})||(j, \pi_{ij})\big)$, and publishes the encrypted proof strings on the public bulletin $\mathcal{B}$. The client also publishes check strings Ψ_i^u and $\Psi^{\pi_i} = (\Psi_i^h, \Psi_i^a, \Psi_i^b, \Psi_i^c)$ for verifying the validity of the shares of u_i and π_i, respectively.

Round 3 (Verify Proof). In this round, every client $\mathcal{C}_i$ partakes in the verification of the proofs π_j of all other clients $\mathcal{C}_j \in \mathcal{C}_{\setminus i}$, under the supervision of the server $\mathcal{S}$. The goal of the server is to identify the malicious clients, $\mathcal{C}_M$. To this end, the server maintains a (partial) list, $\mathcal{C}^*$ (initialized as an empty list), of clients it has so far identified as malicious. The proof-verification round consists of three phases as follows:

(*i*) *Verifying the validity of the secret shares.* First, every client $\mathcal{C}_i$ downloads and decrypts their shares $(i, u_{ji})||(i, \pi_{ji}) \leftarrow \mathsf{AE.dec}\big(sk_{ij}, \overline{(i, u_{ji})||(i, \pi_{ji})}\big), \forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}$ from the bulletin $\mathcal{B}$. Additionally, $\mathcal{C}_i$ downloads the check strings (Ψ_i^u, Ψ_i^π) and verifies the validity of the shares. If the shares from any client $\mathcal{C}_j$:

- fail to be decrypted, *i.e.*, $\mathsf{AE.dec}(\cdot)$ outputs $\perp$, OR
- fail to pass the verification, *i.e.*, $\mathsf{SS.verify}(\cdot)$ returns 0,

$\mathcal{C}_i$ flags $\mathcal{C}_j$ on the bulletin $\mathcal{B}$. Every time a client $\mathcal{C}_i$ flags another client $\mathcal{C}_j$, the server updates the corresponding list $\mathsf{Flag}[j] \leftarrow \mathsf{Flag}[j] \cup \mathcal{C}_i$. If $|\mathsf{Flag}[j]| \geq m + 1$, the server $\mathcal{S}$ marks $\mathcal{C}_j$ as malicious: $\mathcal{C}^* \leftarrow \mathcal{C}^* \cup \mathcal{C}_j$. The server can do so because the pigeon hole principle implies that $\mathcal{C}_j$ must have sent an invalid share to at least one honest client; hence, the correctness of the value recovered from that client's shares cannot be guaranteed. In case $1 \leq |\mathsf{Flag}[j]| \leq m$, the server supervises the following actions. Suppose client $\mathcal{C}_i$ has flagged client $\mathcal{C}_j$. Client $\mathcal{C}_j$ then reveals the shares for $\mathcal{C}_i$, $\big((i, u_{ji}), (i, \pi_{ji})\big)$ in the clear (on bulletin $\mathcal{B}$) for the server $\mathcal{S}$ (or anyone else) to verify using $\mathsf{SS.Verify}(\cdot)$. If that verification passes, $\mathcal{C}_i$ is instructed by the server to use the released shares for its computations. Otherwise, $\mathcal{C}_j$ is marked as malicious by the server $\mathcal{S}$. Note that this does not lead to privacy violation for an honest client since at most m shares corresponding to the m malicious clients would be revealed (see Chapter 4.4). If a client $\mathcal{C}_i$ flags $\geq m + 1$ other clients, $\mathcal{S}$ marks $\mathcal{C}_i$ as malicious. Thus, at this point every client on the list $\mathcal{C}^*$ has either:

- provided invalid shares to at least one honest client, OR
- flagged an honest client.

In other words, every client who is *not* in $\mathcal{C}^*$, $\mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*$, is guaranteed to have submitted at least $n-m-1$ valid shares for the honest clients in $\mathcal{C}_H \setminus \mathcal{C}_i$ (see Chapter 4.4 for details). Additionally, the server cannot be tricked into marking an honest client as malicious, *i.e.*, EIFFeL ensures $\mathcal{C}^* \cap \mathcal{C}_H = \varnothing$ (see Chapter 4.4). The server $\mathcal{S}$ publishes $\mathcal{C}^*$ on the bulletin $\mathcal{B}$.

(*ii*) *Computation of proof summaries by clients.* For this phase, the server $\mathcal{S}$ advertises a random value $r \in \mathbb{F}$ on the bulletin $\mathcal{B}$. Next, a client $\mathcal{C}_i$ proceeds to distill the proof strings of all clients *not* in $\mathcal{C}^*$ to generate summaries for the server $\mathcal{S}$. Specifically, client $\mathcal{C}_i$ prepares a proof summary $\sigma_{ji} = \big((i, w_{ji}^{out}), (i, \lambda_{ji})\big)$ for $\forall \mathcal{C}_j \in \mathcal{C} \setminus (\mathcal{C}^* \cup \mathcal{C}_i)$ as per the description in the previous section, and publishes it on $\mathcal{B}$.

(*iii*) *Verification of proof summaries by the server.* Next, the server moves to the last step of verifying the proof summaries $\sigma_i = (w_i^{out}, \lambda_i)$ for all clients not in $\mathcal{C}^*$. Recall from the discussion in Chapter 4.3.1 that this involves recovering the values w_i^{out} and λ_i from the

shares of σ_i and checking whether $w_i^{out} = 1$ and $\lambda_i = 0$. However, we cannot simply use the naive share reconstruction algorithm from Chapter 4.3.1 since some of the shares might be incorrect (submitted by the malicious clients). To address this issue, EIFFeL performs a *robust reconstruction* of the shares as follows. A naive strategy would be sampling multiple subsets of $m+1$ shares (each subset can emulate a SNIP setting), reconstructing the secret for each subset, and taking the majority vote. However, we can do much better by exploiting the connections between Shamir's secret shares and Reed-Solomon error correcting codes (Chapter 4.3.1). Specifically, the Shamir's secret sharing scheme used by EIFFeL is a $[n-1, m+1, n-m]$ Reed-Solomon code that can correct up to q errors and e erasures (message dropouts) where $2q + e < n-m-1$. The server $\mathcal{S}$ can, therefore, use $\mathsf{SS.robustRecon}(\cdot)$ to reconstruct the secret when $m < \lfloor \frac{n-1}{3} \rfloor$.

After the robust reconstruction of the proof summaries, the server $\mathcal{S}$ verifies them and updates the list $\mathcal{C}^*$ with *all* malicious clients with malformed updates. Specifically:

$$\forall \mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*,$$
$$\Big(\mathsf{SS.robustRecon}(\{(j, w_{ij}^{out})\}_{\mathcal{C}_j \in \mathcal{C} \setminus \{\mathcal{C}^* \cup \mathcal{C}_i\}}) \neq 1 \ \vee \mathsf{SS.robustRecon}(\{(j, \lambda_{ij})\}_{\mathcal{C}_j \in \mathcal{C} \setminus \{\mathcal{C}^* \cup \mathcal{C}_i\}}) \neq 0\Big)$$
$$\implies \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \mathcal{C}_i.$$

Additionally, if a client $\mathcal{C}_i$ withholds some of the shares of the proof summaries for other clients, $\mathcal{C}_i$ is marked as malicious as well by the server. Thus, in addition to the malicious clients listed above, the list $\mathcal{C}^*$ now has all clients that have either:

- failed the proof verification, *i.e.*, provided malformed updates, OR
- withheld shares of proof summaries of other clients (malicious message dropout).

To conclude the round, the server publishes the updated list $\mathcal{C}^*$ on the public bulletin $\mathcal{B}$.

Round 4 (Compute Aggregate). This is the final round of EIFFeL where the aggregate of the well-formed updates is computed. If a client $\mathcal{C}_i$ is on $\mathcal{C}^*$ wrongfully, it can dispute its malicious status by showing the other clients the transcript of the robust reconstruction from all the shares of σ_i (publicly available on bulletin $\mathcal{B}$). If any client $\mathcal{C}_i \in \mathcal{C}$ successfully raises a dispute, all clients abort the protocol because they conclude that the server $\mathcal{S}$ has acted maliciously by trying to withhold a verified well-formed update from the aggregation. If no client raises a successful dispute, every client $\mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*$ generates its share of the aggregate, $(i, \mathcal{U}_i)$ with $\mathcal{U}_i = \sum_{\mathcal{C}_j \in \mathcal{C} \setminus \mathcal{C}^*} u_{ji}$, and sends that share to the server $\mathcal{S}$. Note that, herein, $\mathcal{C}_i$ uses its own share of the update, (i, u_{ii}), as well.

The server recovers the aggregate $\mathcal{U} = \sum_{\mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*} \mathcal{U}_j$ using robust reconstruction: $\mathcal{U} \leftarrow \mathsf{SS.robustRecon}(\{(i, \mathcal{U}_i)\}_{\mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*})$.

Remark 2. The protocol described above corresponds to a single iteration of model training in FL. The server $\mathcal{S}$ can choose a different $\mathsf{Valid}(\cdot)$ for every iteration. Additionally, $\mathcal{S}$ can hold multiple $\mathsf{Valid}_1(\cdot), \cdots, \mathsf{Valid}_k(\cdot)$ and want to check whether the client's update passes them all. For this, we have $\mathsf{Valid}_i(\cdot)$ return zero (instead of one) on success. If $w^{out,i}$ is the value on the output wire of the circuit $\mathsf{Valid}_i(\cdot)$, the server chooses random values $(l_1, \cdots, l_k) \in \mathbb{F}^k$ and recovers the sum $\sum_{i=1}^{k} l_i \cdot w^{out,i}$ in Round 3. If any $w^{out,i} = 0$, then the sum will be non-zero with high probability and $\mathcal{S}$ will reject.

4.3.4 Complexity Analysis

We present the complexity analysis of EIFFeL in terms of the number of clients n, number of malicious clients m and data dimension d (Table 4.1).

Computation Cost. Each client $\mathcal{C}_i$'s computation cost can be broken into six components: (1) performing $n-1$ key agreements – $O(n)$; (2) generating proof π_i for $\mathsf{Valid}(u_i) = 1$ – $O(|\mathsf{Valid}| + |\mathsf{M}| \log |\mathsf{M}|)$[1]; (3) creating secret shares of the update u_i and the proof π_i – $O(mn(d + |\mathsf{M}|))$[2]; (4) verifying the validity of the received shares – $O(mn(d + |\mathsf{M}|)$; (5) generating proof digest for all other clients – $O(n|\mathsf{Valid}|)$; and (6) generating shares of the final aggregate – $O(nd)$. Assuming $|\mathsf{Valid}|$ is of the order of $O(d)$, the overall computation complexity of each client $\mathcal{C}_i$ is $O(mnd)$.

The server $\mathcal{S}$'s computation costs can be divided into three parts: (1) verifying the validity of the flagged shares – $O(md \min(n, m^2))$; (2) verifying the proof digest for all clients – $O(n^2 \log^2 n \log\log n)$; and (3) computing the final aggregate – $O(dn \log^2 n \log\log n)$. Therefore, the total computation complexity of the server is $O\big((n+d)n \log^2 n \log\log n + md \min(n, m^2)\big)$.

Communication Cost. The communication cost of each client $\mathcal{C}_i$ has seven components: (1) exchanging keys with all other clients – $O(n)$; (2) receiving $\mathsf{Valid}(\cdot)$ – $O(|\mathsf{Valid}|)$; (3) sending encrypted secret shares and check strings for all other clients – $O(n(d + |\mathsf{M}|) + md)$; (4) receiving encrypted secret shares and check strings from all other clients – $O(n(d + |\mathsf{M}|) + mnd)$; (5) sending proof digests for every other client – $O(n)$; (6) receiving the list of corrupt clients $\mathcal{C}$ – $O(m)$; and (7) sending the final aggregate – $O(d)$. Thus, the communication complexity for every client is $O(mnd)$.

The servers communication costs include: (1) sending the validation predicate – $O(|\mathsf{Valid}|)$; (2) receiving check strings and secret shares from flagged clients – $O(md \min(n, m^2))$; (3) receiving proof digests – $O(n^2)$; (4) sending the list of malicious clients – $O(m)$; and (5)

[1] We use standard discrete FFT for all polynomial operations [GG13].

[2] This uses the fact that the Lagrange coefficients can be pre-computed [Mat].

- **Setup Phase.**
  - All parties are given the security parameter κ, the number of clients n out of which at most $m < \lfloor \frac{n-1}{3} \rfloor$ are malicious, honestly generated $pp \xleftarrow{\$} \mathsf{KA.gen}(\kappa)$ and a field $\mathbb{F}$ to be used for secret sharing. Server initializes lists $\mathsf{Flag}[i] = \varnothing, i \in [n]$ and $\mathcal{C}^* = \varnothing$.
- **Round 1 (Announcing Public Information).**
  Client: Each client $\mathcal{C}_i$
  - Generates its key pair and announces the public key. $(pk_i, sk_i) \xleftarrow{\$} \mathsf{KA.gen}(pp)$, $\mathcal{C}_i \xrightarrow{pk_i} \mathcal{B}$.

  Server:
  - Publishes the validation predicate $\mathsf{Valid}(\cdot)$. $\mathcal{S} \xrightarrow{\mathsf{Valid}(\cdot)} \mathcal{B}$
- **Round 2 (Generate and Distribute Proof).**
  Client: Each client C_i
  - Computes $n-1$ pairwise keys. $\forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}, sk_{ij} \leftarrow \mathsf{KA.agree}(pk_j, sk_i)$
  - Generates proof $\pi_i = \big(h_i, (a_i, b_i, c_i)\big), h_i \in \mathbb{F}[X]$, $(a_i, b_i, c_i) \in \mathbb{F}^3$, $a_i \cdot b_i = c_i$ for the statement $\mathsf{Valid}(u_i) = 1$.
  - Generates shares of the input $u_i \in \mathbb{F}^d$. $\{(1, u_{i1}), \cdots, (n, u_{in}), \Psi_i^u\} \xleftarrow{\$} \mathsf{SS.share}(u_i, [n], m+1)$
  - Generates shares of the proof π_i.
    $$\{(1, h_{i1}), \cdots, (n, h_{in}), \Psi_i^h\} \xleftarrow{\$} \mathsf{SS.share}(h_i, [n] \setminus i, m+1), \{(1, a_{i1}), \cdots, (n, a_{in}), \Psi_i^a\} \xleftarrow{\$} \mathsf{SS.share}(a_i, [n] \setminus i, m+1),$$
    $$\{(1, b_{i1}), \cdots, (n, b_{in}), \Psi_i^b\} \xleftarrow{\$} \mathsf{SS.share}(b_i, [n] \setminus i, m+1), \{(1, c_{i1}), \cdots, (n, c_{in}), \Psi_i^c\} \xleftarrow{\$} \mathsf{SS.share}(c_i, [n] \setminus i, m+1)$$
  - Encrypts proof strings for all other clients. $\forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}, \overline{(j, u_{ij})||(j, \pi_{ij})} \xleftarrow{\$} \mathsf{AE.enc}\big(sk_{ij}, (j, u_{ij})||(j, \pi_{ij})\big), \pi_{ij} = h_{ij}||a_{ij}||b_{ij}||c_{ij}$.
  - Publishes check strings and the encrypted proof strings on the bulletin. $\forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}, \mathcal{C}_i \xrightarrow{\overline{(j,u_{ij})||(j,\pi_{ij})}} \mathcal{B}; \mathcal{C}_i \xrightarrow{\Psi_i^u, \Psi_i^\pi} \mathcal{B}$
- **Round 3 (Verify Proof)**.
  (i) *Verifying validity of secret shares*:
  Client: Each client $\mathcal{C}_i$
  - Downloads and decrypts proof strings for all other clients from the public bulletin. Flags a client in case their decryption fails.
    $$\forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}, \mathcal{C}_i \xleftarrow{\overline{(i,u_{ji})||(i,\pi_{ji})}, \Psi_j^u, \Psi_j^\pi} \mathcal{B}, (i, u_{ji})||(i, \pi_{ji}) \leftarrow \mathsf{AE.dec}\big(sk_{ij}, \overline{(i, u_{ji})||(i, \pi_{ji})}\big)$$
    $$\perp \leftarrow \mathsf{AE.dec}\big(sk_{ij}, \overline{(i, u_{ji})||(i, \pi_{ji})}\big) \implies C_i \xrightarrow{\text{Flag } \mathcal{C}_j} \mathcal{B}$$
  - Verifies the shares $u_{ji}(\pi_{ji})$ using checkstrings $\Psi_j^u(\Psi_j^\pi)$ and flags all clients with invalid shares.
    $$\forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}, 0 \leftarrow \big(\mathsf{SS.verify}((i, u_{ji}), \Psi_j^u) \wedge \mathsf{SS.verify}((i, \pi_{ji}), \Psi_j^\pi)\big) \implies \mathcal{C}_i \xrightarrow{\text{Flag } \mathcal{C}_j} \mathcal{B}$$

  Server:
  - If client $\mathcal{C}_i$ flags client $\mathcal{C}_j$, the server updates $\mathsf{Flag}[j] = \mathsf{Flag}[j] \cup \mathcal{C}_i$
  - Updates the list of malicious client $\mathcal{C}^*$ as follows:
    - Adds all clients who have flagged $\geq m+1$ other clients. $\forall \mathcal{C}_i$ s. t. $Z = \{j | \mathcal{C}_i \in \mathsf{Flag}[j]\}, |Z| \geq m+1 \implies \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \mathcal{C}_i$
    - Adds all clients with more than $m+1$ flag reports. $|\mathsf{Flag}[i]| \geq m+1 \implies \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \mathcal{C}_i$
    - For clients with less flag reports, the server obtains the corresponding shares in the clear, verifies them and updates $\mathcal{C}^*$ accordingly. $\forall \mathcal{C}_j$ s.t $1 \leq |\mathsf{Flag}[j]| \leq m, \forall \mathcal{C}_i$ s.t. $\mathcal{C}_i$ has flagged $\mathcal{C}_j$
      - $\mathcal{C}_j \xrightarrow{(i,u_{ji}),(i,\pi_{ji})} \mathcal{B}$
      - if $\big(\mathsf{SS.verify}((i, u_{ji}), \Psi_j^u) \wedge \mathsf{SS.verify}((i, \pi_{ji}), \Psi_j^\pi)\big) = 0 \implies \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \mathcal{C}_j$, otherwise, $\mathcal{C}_i$ uses the verified shares to compute its proof summary σ_{ji}
  - Publishes $\mathcal{C}^*$ on the bulletin. $\mathcal{S} \xrightarrow{\mathcal{C}^*} \mathcal{B}$

  (ii) *Generation of proof summaries by the clients.*
  Server:
  - Server announces a random number $r \in \mathbb{F}$. $\mathcal{S} \xrightarrow{r} \mathcal{B}$

  Client: Each client $\mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*$
  - Generates a summary σ_{ji} of the proof string π_{ji} based on r, $\forall \mathcal{C}_j \in \mathcal{C} \setminus (\mathcal{C}^* \cup \mathcal{C}_i), \mathcal{C}_i \xleftarrow{r} \mathcal{B}, \sigma_{ji} = \big((i, w_{ji}^{out}), (i, \lambda_{ji})\big), \mathcal{C}_i \xrightarrow{\sigma_{ji}} \mathcal{B}$

  (iii) *Verification of proof summaries by the server.*
  Server:
  - Downloads and verifies the proof for all clients not on $\mathcal{C}^*$ via robust reconstruction of the digests and updates $\mathcal{C}^*$ accordingly. $\forall \mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*, \mathcal{S} \xleftarrow{\sigma_{ij}} \mathcal{B}, \big(\mathsf{SS.robustRecon}(\{(j, w_{ij}^{out})\}_{\mathcal{C}_j \in \mathcal{C} \setminus (\mathcal{C}^* \cup \mathcal{C}_i)}) \neq 1 \vee \mathsf{SS.robustRecon}(\{(j, \lambda_{ij})\}_{\mathcal{C}_j \in \mathcal{C} \setminus (\mathcal{C}^* \cup \mathcal{C}_i)}) \neq 0\big) \implies \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \mathcal{C}_i$
  - Publishes the updated list $\mathcal{C}^*$ on the bulletin. $\mathcal{S} \xrightarrow{\mathcal{C}^*} \mathcal{B}$
- **Round 4 (Compute Aggregate).**
  Client: Each client $\mathcal{C}_i$
  - If $\mathcal{C}_i$ is on $\mathcal{C}^*$, $\mathcal{C}_i$ raises a dispute by sending the transcript of the reconstruction of σ_i that shows $\lambda_i = 0 \wedge w_j^{out} = 1$ and aborts, OR $\forall \mathcal{C}_j \in \mathcal{C}_{\setminus i}, \mathcal{C}_i \xleftarrow{\sigma_{ij}} \mathcal{B}, \mathcal{C}_i \xrightarrow{\text{Transcript of } \mathsf{SS.robustRecon}(\{(j, \sigma_{ij})\}_{\mathcal{C}_j \in \mathcal{C} \setminus (\mathcal{C}^* \cup \mathcal{C}_i)})} \mathcal{B}$
  - Aborts protocol if it sees any other client on $\mathcal{C}^*$ successfully raise a dispute, OR
  - If no client has raised a dispute and $\mathcal{C}_i$ is not on $\mathcal{C}^*$, sends the aggregate of the shares of clients in $\mathcal{C} \setminus \mathcal{C}^*$ to the server. $\mathcal{U}_i = \sum_{\mathcal{C}_j \in \mathcal{C} \setminus \mathcal{C}^*} u_{ji}, \mathcal{C}_i \xrightarrow{\mathcal{U}_i} \mathcal{S}$

  Server:
  - Reconstructs the final aggregate. $\mathcal{U} \leftarrow \mathsf{SS.robustRecon}(\{(i, \mathcal{U}_i)\}_{\mathcal{C}_i \in \mathcal{C} \setminus \mathcal{C}^*})$

Figure 4.4: EIFFeL: Description of the secure aggregation with verified inputs protocol.

	Computation	Communication
Client	$O(mnd)$	$O(mnd)$
Server	$O\big((n+d)n\log^2 n\log\log n + md\min(n,m^2)\big)$	$O\big(n^2 + md\min(n,m^2)\big)$

Table 4.1: Computational and communication complexity of EIFFeL for the server and an individual client.

receiving the shares of the final aggregate – $O(nd)$. Hence, the overall communication complexity of the server is $O(n^2 + md\min(n, m^2))$.

4.4 Security Analysis

In this chapter, we formally analyze the security of EIFFeL.

Theorem 13. *For any public validation predicate $\mathsf{Valid}(\cdot)$ that can be expressed by an arithmetic circuit, EIFFeL is a SAVI protocol (Definition 13) for $|\mathcal{C}_M| < \lfloor\frac{n-1}{3}\rfloor$ and $\mathcal{C}_{\mathsf{Valid}} = \mathcal{C} \setminus \mathcal{C}^*$.*

Proof. The proof relies on the following two facts.

Fact 1. *Any set of m or less shares in EIFFeL reveals nothing about the secret.*

Fact 2. *A $(n, m+1, n-m)$ Reed-Solomon error correcting code can correctly construct the message with up to q errors and e erasures (message dropout), where $2q+e < n-m+1$. In EIFFeL, we have $q+e=m$ where q is the number of malicious clients that provide erroneous shares and e is the number of clients that withhold a message or are barred from participation (*i.e.*, are in $\mathcal{C}^*$).*

Integrity. We prove that EIFFeL satisfies the integrity constraint of the SAVI protocol using the following three lemmas.

Lemma 9. *EIFFeL accepts the update of every honest client.*

$$\forall \mathcal{C}_i \in \mathcal{C}_H, \Pr_{\mathit{EIFFeL}}[\mathit{Accept}\ u_i] = 1. \tag{4.5}$$

Proof. By definition, client $\mathcal{C}_i \in \mathcal{C}_H$ has well-formed inputs, that is, $\mathsf{Valid}(u_i)=1$. Additionally, $\mathcal{C}_i$, by virtue of being honest, submits valid shares. Hence, at least $n-m-1$ other honest clients $\mathcal{C}_H \setminus \mathcal{C}_i$ will produce correct shares of the proof summary $\sigma_i = (w_i^{out}, \lambda_i)$. Using Fact 2, the server $\mathcal{S}$ is able to correctly reconstruct the value of σ_i. Eq. 4.5 is now implied by the completeness property of SNIP. □

Lemma 10. *All updates accepted by EIFFeL are well-formed with probability $1 - \mathrm{negl}(\kappa)$.*

$$\forall \mathcal{C}_i \in \mathcal{C}, \Pr_{\mathit{EIFFeL}}\big[\mathit{Valid}(u_i) = 1 \big| \mathit{Accept}\ u_i\big] = 1 - \mathrm{negl}(\kappa). \tag{4.6}$$

Proof. In Round 3, the proof corresponding to a client $\mathcal{C}_i$ is verified iff it has submitted valid shares for the $n-m-1$ honest clients $\mathcal{C}_H \setminus \mathcal{C}_i$. This is clearly true if $\mathcal{C}_i$ is honest. If $\mathcal{C}_i$ is malicious, *i.e.*, it submitted at least one invalid share:

- *Case 1:* $|\mathsf{Flag}[i]| \geq m+1$. It is clear that $\mathcal{C}_i$ has submitted an invalid share to at least one honest client and, hence, is removed from the rest of the protocol.

- *Case 2:* $|\mathsf{Flag}[i]| \leq m$. All honest clients in $\mathcal{C}_H$ will be flagging $\mathcal{C}_i$. Hence, $\mathcal{C}_i$ either has to submit the corresponding valid shares or be removed from the protocol.

Given $n-m-1$ valid shares, using Fact 2, we know that EIFFeL reconstructs the proof summary for $\mathcal{C}_i$ correctly. Eq. 4.6 then follows from the soundness property of SNIP. □

Corollary 2. *EIFFeL rejects all malformed updates with probability* $1 - \text{negl}(\kappa)$.

Based on the above lemmas, at the end of Round 3, $\mathcal{C} \setminus \mathcal{C}^*$ (set of clients whose updates have been accepted) must contain *all* honest clients $\mathcal{C}_H$. Additionally, it may contain some clients $\mathcal{C}_i$ who have submitted well-formed updates with at least $n-m-1$ valid shares for $\mathcal{C}_H$, but may act maliciously for other steps of the protocol (for instance, give incorrect shares of proof summary for other clients or give incorrect shares of the final aggregate). This is acceptable provided that EIFFeL is able to reconstruct the final aggregate containing *only* well-formed updates which is guaranteed by the following lemma.

Lemma 11. *The aggregate* $\mathcal{U}$ *must contain the updates of all honest clients or the protocol is aborted.*

$$\mathcal{U} = \mathcal{U}_H + \sum_{\mathcal{C}_i \in \bar{\mathcal{C}}} u_i \text{ where } \mathcal{U}_H = \sum_{\mathcal{C}_i \in \mathcal{C}_H} u_i$$
$$\bar{\mathcal{C}} \subseteq \mathcal{C} \setminus \{\mathcal{C}^* \cup \mathcal{C}_H\} \tag{4.7}$$

Proof. If the server $\mathcal{S}$ acts maliciously and publishes a list $\mathcal{C}^*$ such that $\mathcal{C}^* \cap \mathcal{C}_H \neq \varnothing$, an honest client $\mathcal{C}_i \in \mathcal{C}^* \cap \mathcal{C}_H$ publicly raises a dispute. This is possible since all the shares of σ_i are publicly logged on $\mathcal{B}$. If the dispute is successful, all honest clients will abort the protocol. Note that a malicious client with malformed updates cannot force the protocol to abort in this way since it will not be able to produce a successful transcript with high probability (Lemma 10). If no clients raise a successful dispute, Eq. 4.7 follows directly from Fact 2. Here $\bar{C}$ represents a set of malicious clients with well-formed updates which corresponds to $\mathcal{C}_{\mathsf{Valid}} \setminus \mathcal{C}_H$ in Eq. 4.3. □

Privacy. First, we outline a proof sketch for intuition. *Proof Sketch.*Recall, the privacy constraint of SAVI requires that nothing should be revealed about a private update u_i for an honest client $\mathcal{C}_i$, except:

- u_i passes the integrity check, *i.e.*, $\mathsf{Valid}(u_i) = 1$
- anything that can be learned from the aggregate of honest clients, $\mathcal{U}_H$.

We prove that EIFFeL satisfies this privacy constraint with the help of the following two helper lemmas.

Lemma 12. *In Rounds 1-3, for an honest client $\mathcal{C}_i \in \mathcal{C}_H$, EIFFeL reveals nothing about u_i except $\mathsf{Valid}(u_i) = 1$.*

Proof. In Round 2, observe that the shares $(j, u_{ij}), (j, \pi_{ij})$ for each client $\mathcal{C}_j \in \mathcal{C}_{\setminus i}$ are encrypted with the pairwise secret key and distributed. Hence, a collusion of m malicious clients (and the server $\mathcal{S}$)[3] can access *at most* m shares of any honest client $\mathcal{C}_i \in \mathcal{C}_H$. This is true even in Round 3 where:

- A malicious client might falsely flag $\mathcal{C}_i$.
- No honest client in $\mathcal{C}_H \setminus \mathcal{C}_i$ will flag $\mathcal{C}_i$ since they would be receiving valid shares (and their encryptions) from $\mathcal{C}_i$.
- $\mathcal{S}$ cannot lie about who flagged who, since everything is logged publicly on the bulletin $\mathcal{B}$.

Thus, only m shares of $\mathcal{C}_i$ can be revealed which correspond to the m malicious clients. Since at least $m + 1$ shares are required to recover the secret, any instantiation of the SNIP verification protocol (*i.e.*, reconstruction of the values of $\sigma_i = (w_i^{out}, \lambda_i)$) requires at least one *honest* client to act as the verifier. Hence, at the end of Round 3, from Fact 1 and the zero-knowledge property of SNIP, the only information revealed is that $\mathsf{Valid}(u_i) = 1$. □

Lemma 13. *In Round 4, for an honest client $\mathcal{C}_i \in \mathcal{C}_H$, EIFFeL reveals nothing about u_i except whatever can be learned from the aggregate.*

Proof. In Round 4, from Lemma 11 and Fact 2, the information revealed is either the aggregate or $\perp$. □

[3]The server does not have access to any share of its own in EIFFeL.

Formal Proof. We prove the theorem by a standard hybrid argument. Let $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$ indicate the polynomial-time algorithm that denotes the "next-message" function of parties in $\mathcal{C}_M \cup \mathcal{S}$. That is, given a party identifier $c \in \mathcal{C}_M \cup \mathcal{S}$, a round index i, a transcript T of all messages sent and received so far by all parties in $\mathcal{C}_M \cup \mathcal{S}$, joint randomness $r_{\mathcal{C}_M \cup \mathcal{S}}$ for the corrupt parties' execution, and access to random oracle O, $\Omega_{\mathcal{C}_M \cup \mathcal{S}}(c, i, T, r_{\mathcal{C}_M \cup \mathcal{S}})$ outputs the message for party c in round i (possibly making several queries to O along the way). We note that $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$ is thus effectively choosing the inputs for all corrupt users.

We will define a simulator Sim through a series of (polynomially many) subsequent modifications to the real execution $\text{Real}_{\mathsf{EIFFeL}}$, so that the views of $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$ in any two subsequent executions are computationally indistinguishable.

1. Hyb_0. This random variable is distributed exactly as the view of $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$ in $\text{Real}_{\mathsf{EIFFeL}}$, the joint view of the parties $\mathcal{C}_M \cup \mathcal{S}$ in a real execution of the protocol.

2. Hyb_1. In this hybrid, for any pair of honest clients $\mathcal{C}_i, \mathcal{C}_j \in \mathcal{C}_H$, the simulator changes the key from $\mathsf{KA.agree}(pk_j, sk_i)$ to a uniformly random key. We use Diffie-Hellman key exchange protocol in EIFFeL. The DDH assumption [DH76] guarantees that this hybrid is indistinguishable from the previous one. also be able to break the DDH.

3. Hyb_2. This hybrid is identical to Hyb_1 , except additionally, Sim will abort if $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$ succeeds to deliver, in round 2, a message to an honest client $\mathcal{C}_i$ on behalf of another honest client $\mathcal{C}_j$, such that (1) the message is different from that of Sim, and (2) the message does not cause the decryption to fail. Such a message would directly violate the IND-CCA security of the encryption scheme.

4. Hyb_3. In this round, for every honest party in $\mathcal{C}_H$, Sim samples $s_i \in \mathbb{F}$ such that $\mathsf{Valid}(s_i) = 1$ and replaces all the shares and the check strings accordingly. This allows the server to compute the $\sigma_i = (w_i^{out}, \lambda_i)$ such that $w_i^{out} = 1 \wedge \lambda_i = 0$ for all honest clients in the same way as in the previous hybrid. An adversary noticing any difference would break (1) the computational discrete logarithm assumption used by the VSS [Fel87], OR (2) the $\mathsf{IND\text{-}CCA}$ guarantee of the encryption scheme, OR (3) the information theoretic perfect secrecy of Shamir's secret sharing scheme with threshold $m + 1$, OR (4) zero-knowledge property of SNIP.

5. Hyb_4. In this hybrid, Sim uses $\mathcal{U}_H$ to compute the following polynomial. Let (j, S_j) represent the share of $\sum_{i \in \mathcal{C}_H} s_i$ for a malicious client $\mathcal{C}_j \in \mathcal{C} \setminus \mathcal{C}_H$ where s_i denotes the random input Sim had sampled for $\mathcal{C}_i \in \mathcal{C}_H$ in Hyb_3. Sim performs polynomial interpolation to find the $m + 1$-degree polynomial $p*$ that satisfies $p * (0) = \mathcal{U}_H$ and $p(j) = S_j$. Next, for all honest client, Sim computes the share for $\mathcal{U} = \mathcal{U}_H + \sum_{\mathcal{C}_j \in \bar{\mathcal{C}}} u_j$

(Eq. 4.7) by using the polynomial $p*$ and the relevant messages from $\Omega_{\mathcal{C}_M \cup \mathcal{S}}$. Clearly, this hybrid is indistinguishable from the previous one by the perfect secrecy of Shamir's secret shares. This concludes our proof.

□

4.5 EIFFeL Optimizations

We propose several optimizations to improve the performance of EIFFeL.

4.5.1 Probabilistic Reconstruction

The Gao's decoding algorithm alongside the use of verifiable secret sharing guarantees that the correct secret will be recovered (with probability one). However, we can improve performance at the cost of a small probability of failure.

Verifying Secret Shares. As discussed in Chapter 4.3.4, verifying the validity of the secret shares is the dominating cost for client-side computation. To reduce this cost, we propose an optimization where the validation of the shares corresponding to the proof $\pi_i = \big(h_i, (a_i, b_i, c_i)\big)$ can be eliminated. Specifically, we propose the following changes to Round 3:

- Each client $\mathcal{C}_i$ skips verifying the validity of the shares (i, π_{ji}) for $\mathcal{C}_j \in \mathcal{C}_{\setminus i}$.
- Let $e = |\mathcal{C}^*|$. The server $\mathcal{S}$ samples two sets of clients P_1, P_2 from $\mathcal{C} \setminus \{\mathcal{C}_i \cup \mathcal{C}^*\}$ of size at least $3m - 2e + 1$ (P_1, P_2 can be overlapping) and performs Gao's decoding on both the sets to obtain polynomials p_1 and p_2. The server accepts the w_i^{out} (λ_i) only iff $p_1 = p_2$ and $p_1(0) = p_1(0) = 1 (p_1(0) = p_1(0) = 0)$. The cost of this step is $O(n^2 \log^2 n \log\log n)$ which is less than verifying the shares of π_i when $m < n \ll d$ (improves runtime by 2.3×, see Table 4.2).

Note that a $[n, k, n-k+1]$ Reed-Solomon error correcting code can correct up to $\lfloor \frac{n-k-l}{2} \rfloor$ errors with l erasures. Thus, with $m-e$ malicious clients, only $3m-2e+1$ shares are sufficient to correctly reconstruct the secret for honest clients. Since, the random sets P_1 and P_2 are not known, a malicious client with more than $m-e$ invalid shares can cheat only with probability at most $1/\binom{n-e}{3m-2e+2}$. We cannot extend this technique for the secret shares of the update u, because, unlike the value of the digests ($w^{out} = 1, \lambda = 0$), the value of the final aggregate is unknown and needs to be reconstructed from the shares.

Robust Reconstruction. In case $m \leq \sqrt{n} - 2$, the robust reconstruction mechanism can be optimized as follows. Let $q = m - |\mathcal{C}^*|$ be the number of malicious clients that remain undetected. The server $\mathcal{S}$ partitions the set of clients in $\mathcal{C} \setminus \mathcal{C}^*$ into at least $q + 2$ disjoint partitions, $P = \{P_1, \cdots, P_{q+2}\}$ each of size $m + 1$. Let $p_j(x) = c_{j,0} + c_{j,1}x + c_{j,2}x^2 + \cdots +$

$c_{j,m}x^m$ represent the polynomial corresponding to the $m+1$ shares of partition P_j. Recall that recovering just $p_j(0) = c_{j,0}$ suffices for a typical Shamir secret share reconstruction. However, now, the server $\mathcal{S}$ recovers the entire polynomial p_j, *i.e.*, all of its coefficients $\{c_{j,0}, c_{j,1}, \cdots, c_{j,q}\}$ for all $q+2$ partitions. Based on the pigeon hole principle, it can be argued that at least two of the partitions ($P_l, P_k \in P$) will consist of *honest* clients only. Hence, we must have at least two polynomials p_l and p_k that match and the value of the secret is their constant coefficient $p_l(0)$. Note that the above mentioned optimization of skipping verifying the shares of the proof can be applied here as well. A malicious client can cheat (*i.e.*, make the server $\mathcal{S}$ accept even when $w_i^{out} \neq 1 \vee \lambda_i \neq 0$ or reject the proof for an honest client) only if they can manipulate the shares of at least two partitions which must contain at least $2(m+1)-q$ honest clients. Since the random partition P is not known to the clients, this can happen only with probability $1/\binom{n-m-1}{2(m+1)-q}$.

4.5.2 Crypto-Engineering Optimizations

We propose the following crypto-engineering optimizations.

Equality Checks. The equality operator $=$ is relatively complicated to implement in an arithmetic circuit. To circumvent this issue, we replace any validation check of the form $\Phi(u) = c_1 \vee \Phi(u) = c_2 \vee \cdots \vee \Phi(u) = c_k$ in the output nodes of $\mathsf{Valid}(\cdot)$, where $\Phi(\cdot)$ is some arithmetic function, by an output of the form $(\Phi(u) - c_1) \times \cdots \times (\Phi(u) - c_k)$. Recall that in EIFFeL, the honest clients have well-formed inputs that satisfy $\mathsf{Valid}(\cdot)$ by definition. Hence, this optimization does not violate the privacy of honest, which is our security goal.

Proof Summary Computation. In addition to being a linear secret sharing scheme, Shamir's scheme is also multiplicative: given the shares of two secrets (i, z_i) and (i, v_i), a party can locally compute (i, s_i) with $s = z \cdot v$. However, if the original shares correspond to a polynomial of degree t, the new shares represent a polynomial of degree $2t$. Hence, we do not rely on this property for the multiplication gates of $\mathsf{Valid}(\cdot)$ as it would support only limited number of multiplications. However, if $m < \frac{n-1}{4}$, we can still leverage the multiplicative property to generate shares of the random digest $\lambda_i = f_i(r) \cdot g_i(r) = h_i(r)$ locally (instead of using Beaver's triples). Specifically, each client can locally multiply the shares (j, f_{ij}) and (j, g_{ij}) to generate $(i, (f_j \cdot g_j)_i)$. In order to make the shares consistent, $\mathcal{C}_i$ multiplies the share of (i, h_{ji}) with (i, z_i) where $z = 1$ (these can be generated and shared by the server $\mathcal{S}$ in the clear). In this way, $\mathcal{C}_j$ can locally generate a share of the digest (j, d_{ij}) that correspond to a polynomial of degree $2m$. Since $m < \frac{n-1}{4}$, this optimization is still compatible with robust reconstruction. This saves a round of communication and reduces the number of robust reconstructions for λ_i from three to just one (see Chapter 4.3.1).

Random Projection. As shown in Table 4.1, both communication and computation grows linearly with the data dimension d. Hence, we rely on the random projection [Nel] technique for reducing the dimension of the updates. Specifically, we use the fast random projection using Walsh-Hadamard transforms [AC06].

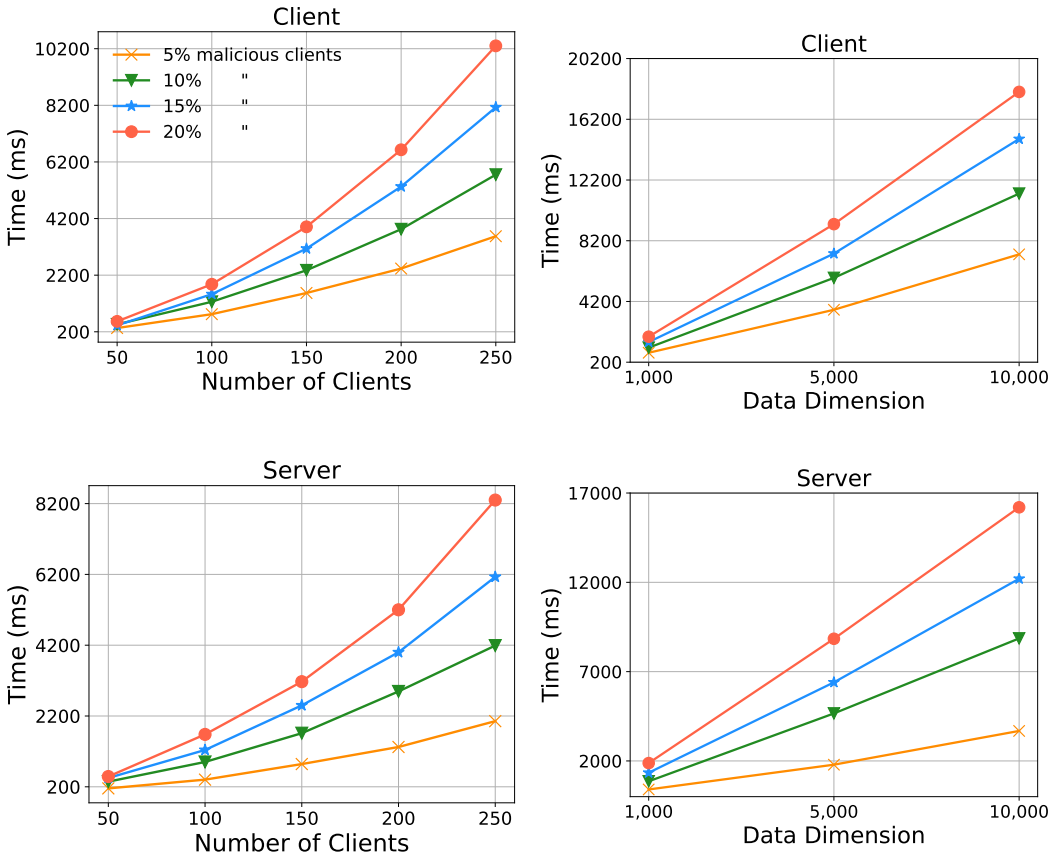


Figure 4.5: Computation cost analysis of EIFFeL. The **left** two plots show the runtime of a single client client in milliseconds as a function of: (left) the number of clients n and (right) dimensionality of the updates d. The **right** two plots show the runtime of the server as a function of the same variables. The results demonstrate that performance decays quadratically in n, and linearly in d.

4.6 Experimental Evaluation

We perform experiments to evaluate the practical performance of EIFFeL.

4.6.1 Performance Evaluation.

In this section, we analyze the performance of EIFFeL.

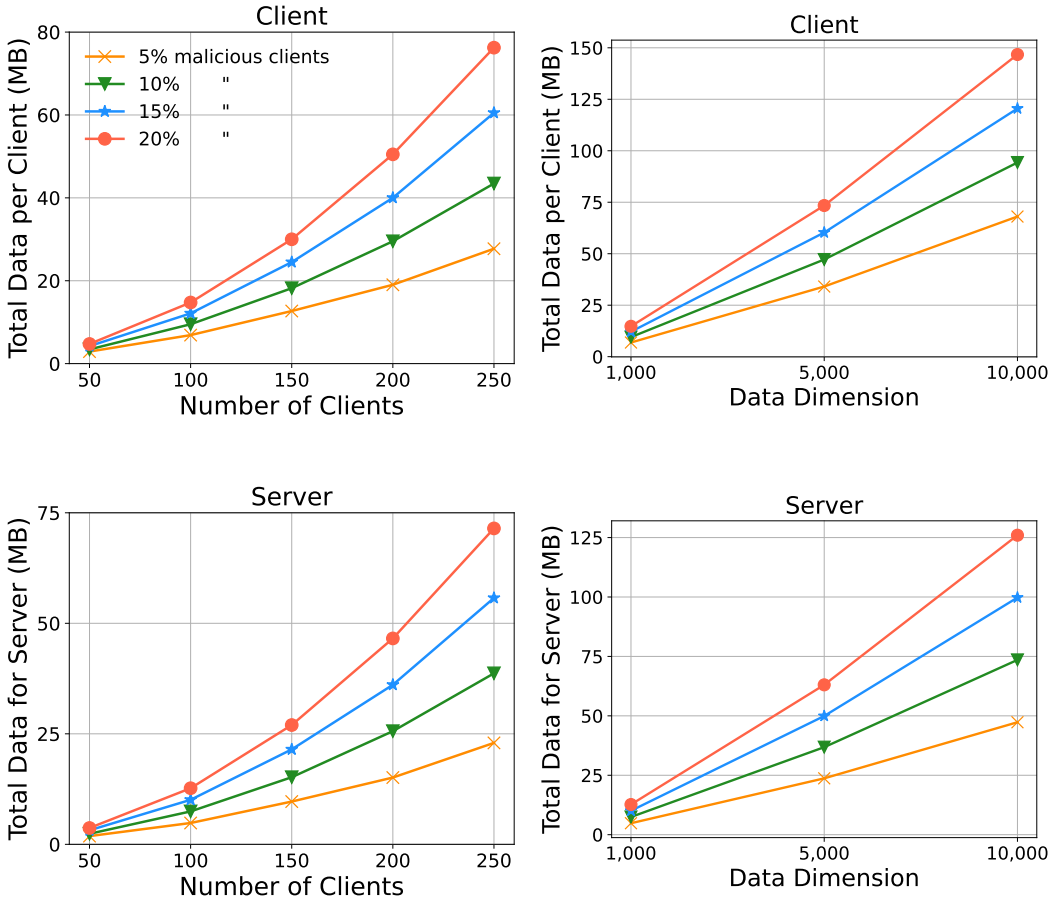


Figure 4.6: Communication cost analysis of EIFFeL. The **left** two plots show the amount of communication (in MB) for each client as a function of: (left) the number of clients n and (right) dimensionality of the updates d. The **right** two plots show the the amount of communication (in MB) for the server as a function of the same variables. The results show communication increases quadratically in n, and linearly in d.

Configuration. We run experiments on two Amazon EC2 c5.9large instances with Intel Xeon Platinum 8000 processors. To emulate server-client communication, we use two instances in the US East (Ohio) and US West (Oregon) regions, with a round trip time of 21 ms. We implemented EIFFeL in Python and C++ using NTL library [NTL]. We use AES-GCM for encryption and a 44-bit prime field $\mathbb{F}$. For key agreement, we use elliptic curve Diffie-Hellman [DH76] over the NIST P-256 curve. Unless otherwise specified, the default settings are $d = 1000$, $n = 100$, $m = 10\%$ and $|\mathsf{Valid}(\cdot)| \approx 4d$. We report the mean of 10 runs for each experiment.

Computation Costs. Figure 4.5 presents EIFFeL's runtime. We vary the number of malicious clients between 5%-20% of the number of clients. We observe that per-client runtime of EIFFeL is low: it is 1.3 seconds if $m = 10\%$, $d = 1000$, and $n = 100$. The runtime scales quadratically in n because a client has $O(mnd)$ computation complexity (see Table 4.1) and m is a linear function of n. As expected, the runtime increases linearly with d. A client takes around 11 seconds when $d = 10,000$, $n = 100$, and $m = 10\%$. The runtime for the server is also low: the server completes its computation in about 1 second for $n = 100$, $d = 1K$, and $m = 10\%$. The server's runtime also scales quadratically in n due to the $O(mnd)$ computation complexity (Table 4.1). The runtime increases linearly with d.

In Figure 4.7, we break down the runtime per round. We observe that: Round 1 (announcing public information) incurs negligible cost for both clients and the server; and Round 3 (verify proof) is the costliest round for both clients and the server where the dominating cost is verifying the validity of the shares (Chapter 4.3.4). Note that the server has no runtime cost for Round 2 since the proof generation only involves clients.

Table 4.2 presents our end-to-end performance which contains the runtimes of a client, the server and the communication latencies. For instance, the end-to-end runtime for $n = 100$, $d = 1,000$ and $m = 10\%$ is $\sim 2.4s$. We also present the impact of one of our key optimizations – eliminating the verification of the secrets shares of the proof – which cuts down the costliest step in EIFFeL and improves the performance by 2.3×. Additionally, we compare EIFFeL's performance with BREA [SGA20], which is a Byzantine-robust secure aggregator. EIFFeL differs from BREA in two key ways: (1) EIFFeL is a general framework for per-client update integrity checks whereas BREA implements the multi-Krum aggregation algorithm [BMGS17b] that considers the entire dataset to determine the malicious updates (computes all the pairwise distances between the clients and then, detects the outliers), and (2) BREA has an additional privacy leakage as it reveals the values of all the pairwise distances between clients. Nevertheless, we choose BREA as our baseline because, to the best of our knowledge, this is the only prior work that: (1) detects and removes malformed updates, and (2) works in the malicious threat model for the general FL setting (see Chapter 4.9). We observe that EIFFeL outperforms BREA and that the improvement increases with n. For instance, for $n = 250$, EIFFeL is 18.5× more performant than BREA. This is due to BREA's complexity of $O(n^3 \log^2 n \log\log n + mnd)$, where the $O(n^3)$ factor is due to each client partaking in the computation of the $O(n^2)$ pairwise distances.

Communication Cost. Figure 4.6 depicts the total data transferred by a client and the server in EIFFeL. The communication complexity is $O(mnd)$ for a single client and for the server. Hence, the total communication increases quadratically with n and linearly with d, respectively. We observe that EIFFeL has acceptable communication cost. For instance,

# Clients (n)	Time (ms)	Improvement over	
		Unoptimized EIFFeL	BREA [SGA20]
50	1,072	2.3×	2.5×
100	2,367	2.3×	5.2×
150	4,326	2.3×	7.8×
200	6,996	2.3×	12.8×
250	10,389	2.3×	18.5×

Table 4.2: End-to-end time for a single iteration of EIFFeL with $d = 1000$ and $m = 10\%$ malicious clients, as a function of the number of clients, n. We also compare it with a variant of EIFFeL without optimizations, and with BREA [SGA20].

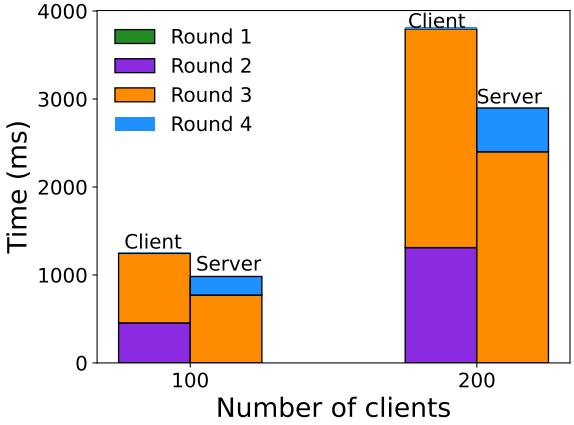


Figure 4.7: Computation cost per round in EIFFeL.

the total data consumed by a client is 94MB for the configuration $n = 100, d = 10K, m = 10\%$. This is equivalent to streaming a full-HD video for $20s$ [dat]. Since most clients partake in FL training iterations infrequently, this amount of communication is acceptable.

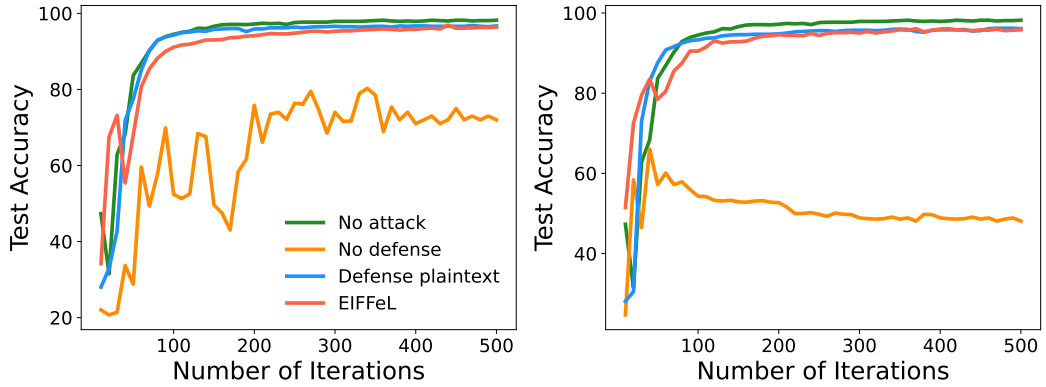
4.6.2 Integrity Guarantee Evaluation

In this section, we evaluate EIFFeL's efficacy in ensuring update integrity on real-world datasets.

Datasets. We evaluate EIFFeL on three image datasets:

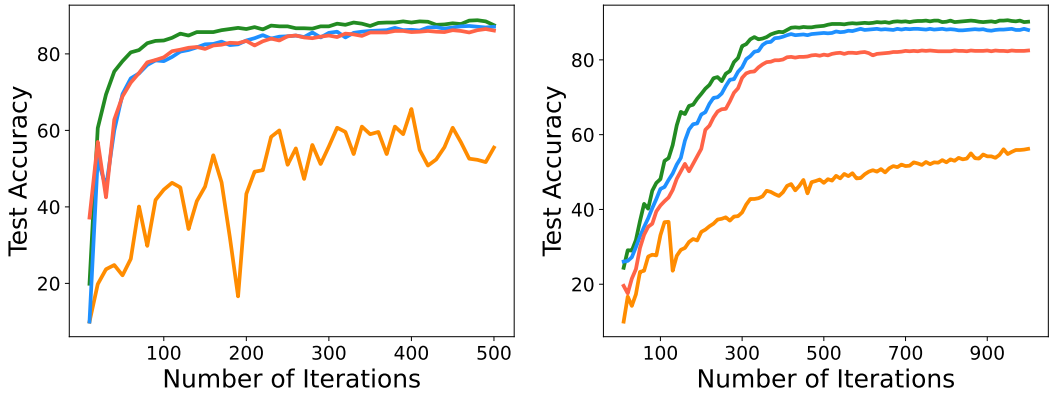
- *MNIST* [LCB] is a handwritten digit dataset of 60,000 training images and 10,000 test images with ten classes (each digit is its own class).
- *FMNIST* [Zal] is a dataset of display clothing items that is identical to MNIST in terms number of classes, and number of training and test images.
- *CIFAR-10* [Kri] contains RGB images with ten object classes. It has 50,000 training and 10,000 test images.

Models. We test EIFFeL with three classification models:



(a) MNIST: Sign flip attack with norm ball validation predicate (defense).

(b) MNIST: Scaling attack and cosine similarity validation predicate.



(c) FMNIST: Additive noise attack with Zeno++ validation predicate.

(d) CIFAR-10: Scaling attack with norm bound validation predicate.

Figure 4.8: Accuracy analysis of EIFFeL. Test accuracy is shown as a function of the FL iteration for different datasets and attacks.

- *LeNet-5* [LBBH98] is one of the first successful convolutional network architectures. It has five layers and 60,000 parameters. We use LeNet-5 to experiment on MNIST.
- For FMNIST, we use a five-layer convolutional network with 70,000 parameters and a similar architecture as LeNet-5.
- *ResNet-20* [HZRS16] is a more modern convolutional network with 20 layers and 273,000 parameters. We use it for our experiments on the CIFAR-10 dataset.

Validation Predicates. To demonstrate the flexibility of EIFFeL, we evaluate four validations predicates as follows:

- *Norm Bound* [SKSM19a]. This method checks whether the ℓ_2-norm of a client update is bounded: $\mathsf{Valid}(u) = \mathbb{I}[\|u\|_2 < \rho]$ where $\mathbb{I}[\cdot]$ is the indicator function and the threshold ρ is computed from the public dataset $\mathcal{D}_P$.

- *Norm Ball* [SKL17]. This method checks whether a client update is within a spherical radius from v which is the gradient update computed from the clean public dataset $\mathcal{D}_P$: $\mathsf{Valid}(u) = \mathbb{I}\big[\|u - v\|_2 \leq \rho\big]$ where radius ρ is also computed from $\mathcal{D}_P$.

- *Zeno++* [Xie19] compares the client update u with a loss gradient v that is computed on public dataset $\mathcal{D}_P$: $\mathsf{Valid}(u) = \mathbb{I}[\gamma\langle v, u\rangle - \rho||u||_2 \geq -\gamma\epsilon]$ where γ, ρ and ϵ are threshold parameters also computed from $\mathcal{D}_P$ and u is ℓ_2-normalized to have the same norm as v.

- *Cosine Similarity* [CFLG21, BVH+18]. This method compares the cosine similarity between the client update u and the global model update of the last iteration u': $\mathsf{Valid}(u) = \mathbb{I}\big[\frac{\langle u,u'\rangle}{\|u\|_2\|u'\|_2} < \rho\big]$ where ρ is computed from $\mathcal{D}_P$ and u is ℓ_2-normalized to have the same norm as u'.

Poisoning Attacks. To test the efficacy of EIFFeL's implementations of the four validation predicates introduced above, we test it against three poisoning attacks:

- *Sign Flip Attack* [DMG+18]. In this attack, the malicious clients flip the sign of their local update: $\hat{u} = -c \cdot u, c \in \mathbb{R}_+$.

- *Scaling Attack* [BCMC19] scales a local update to increase its influence on the global update: $\hat{u} = c \cdot u, c \in \mathbb{R}_+$.

- *Additive Noise Attack* [LXC+19] adds Gaussian noise to the local update: $\hat{u} = u + \eta, \eta \sim \mathcal{N}(\sigma, \mu)$.

Configuration. We use the same configuration as before. We implement the image-classification models in PyTorch. We randomly select 10,000 samples from each training set as the public dataset $\mathcal{D}_P$ and train on the remaining samples. The training set is divided into 5,000 subsets to create the local dataset for each client. For each training iteration, we sample the required number of data subsets out of these 5,000 subsets.

Results. Figure 4.8 reports the accuracy of training different image-classification models in EIFFeL. We set $n = 100$ and $m = 10\%$, and use random projection to project the updates to a dimension d of 1,000 (MNIST), 5,000 (FMNIST), or 10,000 (CIFAR-10). Our experiment assesses how the random projection affects the efficacy of the integrity checks. We observe that for MNIST (Figures 4.8a and 4.8b) and FMNIST (Figure 4.8c), EIFFeL achieves performance comparable to a baseline that applies the defense (validation predicate) on the

plaintext. In most cases, the defenses retain their efficacy even after random projection. This is because they rely on computing inner products and norms of the update; these operations preserve their relative values after the projection with high probability [Nel]. We do observe a drop in accuracy ($\sim 7\%$) on CIFAR-10 as updates for ResNet-20 with 273,000 parameters are projected to 10,000. The end-to-end per-iteration time for MNIST, FMNIST, and CIFAR-10 is $2.4s$ (Table 4.2), $10.7s$, and $20.5s$, respectively. The associated communication costs for the client are 9.5MB, 47MB, and 94MB (Figure 4.6).

4.7 Extension to Differential Privacy

EIFFeL is the first step toward designing aggregation protocols for federated learning that ensures both input privacy and integrity. In this chapter, we discuss how to extend EIFFeL to support differential privacy.

Recall that the goal of secure aggregation is to protect the individual client updates and reveal only the final aggregate. However, privacy violations can arise even from this aggregate [ASY+18]. The common approach to tackle this is to add noise to the revealed aggregate to ensure DP. EIFFeL outputs an aggregate with an additional property – all the aggregated updates are well-formed. EIFFeL can be used as a building block and easily extended to support DP along the lines of prior work [ASY+18, KLS21b]. Specifically, we need to introduce the following two changes to the protocol. First, DP requires the client updates to be clipped in order to bound the sensitivity. This can be enforced by introducing an additional check in the validation predicate $\mathsf{Valid}(\cdot)$. Next, in Round 4, the clients add shares of discretized Gaussian noise to the shares of the aggregate. The privacy budget (ϵ) across multiple training iterations can be controlled using standard privacy accounting techniques [BS16, DR16, WBK19, Mir17b].

Prior work has shown that DP provides robustness guarantees as well [SKSM19b, NHC21]. Hence, DP can enhance both goals of EIFFeL – ensuring privacy for clients and Byzantine robustness for the federated learner.

4.8 Discussion

In this chapter, we discuss several possible avenues for future research.

Handling Higher Fraction of Malicious Clients. For $\lfloor \frac{n-1}{3} \rfloor < m < \lfloor \frac{n-1}{2} \rfloor$ (honest majority), the current implementation of EIFFeL can detect but not remove malformed inputs (Gao's decoding algorithm returns $\perp$ if $m > \lfloor \frac{n-1}{3} \rfloor$). To robustly reconstruct in this case as

well, we could use techniques such as Guruswami-Sudan list decoder [McE03]. We do not do so in EIFFeL because the reconstruction might fail sometimes.

Handling Client Dropouts. In practice, clients might have only sporadic access to connectivity and so, the protocol must be robust to clients dropping out. EIFFeL can already accommodate malicious client dropping out – it is straightforward to extend this for the case of honest clients as well.

Reducing Client's Computation. Currently, verifying the validity of the secret shares is the dominant cost for clients. This task can be offloaded to the server $\mathcal{S}$ by using a publicly verifiable secret sharing scheme (PVSS) [Sch99, Sta96, TPH] where the validity of a secret share can be verified by any party. However, typically PVSS employs public key cryptography (which is costlier than symmetric cryptography) which might increase the end-to-end running time.

Private Validation Predicate. If $\mathsf{Valid}(\cdot)$ contains some secrets of the server $\mathcal{S}$, we can employ multiple servers where the computation of $\mathsf{Valid}(u)$ is done at the servers [CGB17]. We leave a single-server solution of this problem for future work.

Identifying All Malicious Clients. Currently, EIFFeL identifies a partial list of malicious clients. To detect all malicious clients, one can use: (1) PVSS to identify all clients who have submitted at least one invalid share, and (2) decoding algorithms such as Berlekamp-Welch [Bla83] that can detect the location of the errors from the reconstruction. We do not use them in EIFFeL as they have higher computation cost.

Byzantine-Robust Aggregation. In EIFFeL, the integrity check is done individually on each client update, independent of all other clients. An alternative approach to compare the local model updates of *all* the clients (via pairwise distance/ cosine similarity) [BMGS17a, BMGS17b, CFLG21, FCJG20] and remove statistical outliers before using them to update the global model. A general framework to support secure Byzantine-robust aggregations rules, such as above, is an interesting future direction.

$\mathsf{Valid}(\cdot)$ Structure. If $\mathsf{Valid}(\cdot)$ contains repeated structures, the G-gate technique [BBCG$^+$19] can improve efficiency.

Scaling EIFFeL. Our experimental results in Chapter 4.6 show that EIFFeL has reasonable performance for clients sizes up to 250. One way of scaling EIFFeL for larger client sizes can be by dividing the clients into smaller subsets of size ~ 250 and then running EIFFeL for each of these subsets [BEG$^+$19].

Revealing Malicious Clients. In our current implementation, EIFFeL publishes the (partial) list of malicious clients $\mathcal{C}^*$. To hide the identity of malicious clients, we could include an equal number of honest clients in the list before publishing it, thereby providing those clients plausible deniability. We leave more advanced cryptographic solutions as a future direction.

Complex Aggregation Rules. EIFFeL can be used for more complex aggregation rules, such as mode, by extending SNIP with affine-aggregatable encodings (AFE) [CGB17].

4.9 Related Work

In this chapter, we discuss the relevant prior literature on FL.

Secure Aggregation. Prior work has addressed the problem of (non-Byzantine) secure aggregation in FL [BIK+17, BBG+20, AC11, SGA21]. A popular approach is to use pairwise random masking to protect the local updates [BIK+17, AC11]. Recent approaches have improved the communication overhead, by training in a smaller parameter space [KMY+16], autotuning the parameters [BSK+19], or via coding techniques [SGA21].

Robust Machine Learning. A large number of studies have explored methods to make machine learners robust to Byzantine failures [BVH+18, BCMC19, KMA+19]. Many of these robust machine-learning methods require the learned to have full access to the training data or to fully control the training process [CSL+08, GDGG17, LDGG18, SS19, SKL17, WYS+19] which is infeasible in FL. Another line of work has focused on the development of estimators that are inherently robust to Byzantine errors [BMGS17b, CWCP18, PZW+20, RWCP19, YCKB19]. In our work, we target a set of methods that provides robustness by checking per-client updates [BMGS17b, FYB18, STS16].

Verifying Data Integrity in Secure Aggregation. There is limited prior work that seeks to develop cryptographic protocols for data-integrity verification in secure aggregation. Most similar to our work is RoFL [BLV+21], which uses range proofs to check update integrity. There are three key differences between RoFL and EIFFeL: (1) RoFL supports only range checks with ℓ_2 or ℓ_∞ norms. By contrast, EIFFeL is a general framework that supports arbitrary validation predicates. (2) RoFL is susceptible to DoS attacks because it *only* detects malformed updates and aborts if it finds one. By contrast, EIFFeL is a SAVI protocol that detects and removes malformed updates in every round. (3) RoFL assumes an honest-but-curious server, whereas EIFFeL considers a malicious threat model. BREA [SGA20] also removes outlying updates but, unlike EIFFeL, it leaks pairwise distances between inputs. Alternative solutions [NRY+21, HKJ20] for distance-based Byzantine-robust aggregation uses

two non-colluding servers in the semi-honest threat model, which is incompatible with centralized FL. Other work [VXK21] randomly clusters clients, reveals inputs from the clusters and then robustly aggregates them; but doing this requires small clusters, which affects input privacy.

4.10 Conclusion

Practical federated learning settings need to ensure both the privacy and integrity of model updates provided by clients. In this paper, we have formalized these goals in a new protocol, SAVI, that securely aggregates *only* well-formed inputs (*i.e.*, updates). To demonstrate the feasibility of SAVI, we have proposed EIFFeL: a system that efficiently instantiates a SAVI protocol. OPϵc works under a malicious threat model and ensures both privacy for honest clients and Byzantine robustness for the federated learner. Our empirical evaluation has shown that EIFFeL is practical for real-world usage.

Chapter 5

Conclusion

Be it online shopping on Instagram, streaming the newest Netflix series or doomscrolling our Twitter feed, today we spend more than quarter of our lives online – and our *every* movement is documented. Private corporations hold large swathes of personal data about every individual. Without proper regulation, this could be easily harnessed to manipulate our online lives and lead to violation of personal autonomy. Hence, data privacy can no longer be limited to just an academic pursuit – privacy has to be concomitant with every click we make.

In this dissertation, we have provided a way forward. We have shown that it is possible to build large scale privacy-first systems by *combining techniques from differential privacy and cryptography*. In fact, this synergy is multi-faceted: (1) cryptography can push the frontiers of deployment of DP, as demonstrated by Cryptϵ, (2) DP can aid in making cryptographic systems robust to inference attacks, as demonstrated by OPϵ, and (3) privacy-preserving decentralized learning requires co-designing both DP and cryptography, as demonstrated by EIFFeL.

Modern technology must do a better job of upholding the privacy of our personal data. I am hopeful of a future where we as individuals are entitled to and empowered with our data sovereignty.

LIST OF REFERENCES

[ABCP13] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 901–914, New York, NY, USA, 2013. Association for Computing Machinery.

[ABE+13] Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, and Ramaratnam Venkatesan. Orthogonal security with cipherbase. In *Proc. of the 6th CIDR, Asilomar, CA*, 2013.

[ABFMO08] Esma Aïmeur, Gilles Brassard, José M. Fernandez, and Flavien Serge Mani Onana. Alambic: a privacy-preserving recommender system for electronic commerce. *International Journal of Information Security*, 7(5), Oct 2008.

[ABK+19] Jayadev Acharya, Keith Bonawitz, Peter Kairouz, Daniel Ramage, and Ziteng Sun. Context-aware local differential privacy, 2019.

[ABY] https://github.com/encryptogroup/aby.

[AC06] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, page 557–563, New York, NY, USA, 2006. Association for Computing Machinery.

[AC11] Gergely Ács and Claude Castelluccia. I have a dream! differentially private smart metering. In *Proceedings of the 13th International Conference on Information Hiding*, IH'11, page 118–132, Berlin, Heidelberg, 2011. Springer-Verlag.

[ACC12] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *2012 IEEE 12th International Conference on Data Mining*, pages 1–10, Dec 2012.

[ACFR20] Shimaa Ahmed, Amrita Roy Chowdhury, Kassem Fawaz, and Parmesh Ramanathan. Preech: A system for Privacy-Preserving speech transcription. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2703–2720. USENIX Association, August 2020.

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM.

[ACPP18] M. Alvim, K. Chatzikokolakis, C. Palamidessi, and A. Pazii. Invited paper: Local differential privacy on metric spaces: Optimizing the trade-off with utility. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 262–267, 2018.

[AHKM18a] A. Agarwal, M. Herlihy, S. Kamara, and Tarik Moataz. Encrypted databases for differential privacy. *Proceedings on Privacy Enhancing Technologies*, 2019:170–190, 2018.

[AHKM18b] Archita Agarwal, Maurice Herlihy, Seny Kamara, and Tarik Moataz. Encrypted databases for differential privacy, 01 2018. `https://eprint.iacr.org/2018/860`.

[AKL+09] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, abhi shelat, and Ivan Visconti. Collusion-free multiparty computation in the mediated model. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 524–540, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 563–574, New York, NY, USA, 2004. Association for Computing Machinery.

[AN10] A.Asuncion and D. Newman. Uci machine learning repository, 2010.

[AS19] John M. Abowd and Ian M. Schmutte. An economic analysis of privacy protection and statistical accuracy as social choices. *American Economic Review*, 109(1):171–202, January 2019.

[ASY+18] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[BBCG+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In *CRYPTO*, 2019.

[BBDS12] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, Oct 2012.

[BBG+20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1253–1269, New York, NY, USA, 2020. Association for Computing Machinery.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 535–552, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[BCIV17] Joppe W. Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Privacy-friendly forecasting for the smart grid using homomorphic encryption and the group method of data handling. In Marc Joye and Abderrahmane Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017*, pages 184–201, Cham, 2017. Springer International Publishing.

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology - EUROCRYPT 2009 - Volume 5479*, page 224–241, Berlin, Heidelberg, 2009. Springer-Verlag.

[BCM+20] Raef Bassily, Albert Cheu, Shay Moran, Aleksandar Nikolov, Jonathan Ullman, and Zhiwei Steven Wu. Private query release assisted by public data. In *ICML*, 2020.

[BCMC19] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *Proceedings of the International Conference on Machine Learning*, pages 634–643, 2019.

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Proceedings of the 31st Annual Conference on Advances in Cryptology*, CRYPTO'11, page 578–595, Berlin, Heidelberg, 2011. Springer-Verlag.

[BCSZ18] C. Borgs, J. Chayes, A. Smith, and I. Zadik. Revealing network structure, confidentially: Improved rates for node-private graphon estimation. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 533–543, 2018.

[BDF+18] Abhishek Bhowmick, John C. Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan M. Rogers. Protection against reconstruction and its applications in private federated learning. *ArXiv*, abs/1812.00984, 2018.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 420–432, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, pages 97–109, Berlin, Heidelberg, 1995. Springer-Verlag.

[BEG+19] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design, 2019.

[BEM+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 441–459, New York, NY, USA, 2017. ACM.

[BGC+18] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. The tao of inference in privacy-protected databases. *Proc. VLDB Endow.*, 11(11):1715–1728, July 2018.

[BGP+19] Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, Kurt Rohloff, and Vinod Vaikuntanathan. Optimized homomorphic encryption solution for secure genome-wide association studies. *IACR Cryptology ePrint Archive*, 2019:223, 2019.

[BHT+18] Johes Bater, Xi He, S Yu Tendryakova, Ashwin Machanavajjhala, and Jennie Duggan. Shrinkwrap: Differentially-private query processing in private data federations. *CoRR*, abs/1810.01816(3):307–320, November 2018.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[BKN+20] Amos Beimel, Aleksandra Korolova, Kobbi Nissim, Or Sheffet, and Uri Stemmer. The power of synergy in differential privacy: Combining a small curator with local randomizers. In *ITC*, 2020.

[Bla83] Richard E. Blahut. Theory and practice of error control codes. 1983.

[BLR+15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Proceedings of EUROCRYPT, 2015.

[BLV+21] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Attestable robustness for secure federated learning. In *arXiv:2107.03311*, 2021.

[BMGS17a] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer. Byzantine-tolerant machine learning. In *arXiv:1703.02757*, 2017.

[BMGS17b] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 118–128, 2017.

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the International Coference on International Conference on Machine Learning*, pages 1467–1474, 2012.

[BNO08] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology*, CRYPTO 2008, pages 451–468, Berlin, Heidelberg, 2008. Springer-Verlag.

[BNO11] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: On simultaneously solving how and what. *CoRR*, abs/1103.2626, 2011.

[BNS14] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. *CoRR*, abs/1407.2674, 2014.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006*, pages 409–426, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[BS15] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 127–135, New York, NY, USA, 2015. ACM.

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds, 2016.

[BSK+19] Keith Bonawitz, Fariborz Salehi, Jakub Konecný, H. Brendan McMahan, and Marco Gruteser. Federated learning with autotuned communication-efficient secure aggregation. *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1222–1226, 2019.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, 2007.

[BT85] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, oct 1985.

[BVH+18] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *arXiv:1807.00459*, 2018.

[CABP13] Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. Broadening the scope of differential privacy using metrics. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies*, pages 82–102, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[CC17] Pyrros Chaidos and Geoffroy Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. *IACR Cryptology ePrint Archive*, 2017:1029, 2017.

[CCDD+21] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. Ca{pc} learning: Confidential and private collaborative learning. In *International Conference on Learning Representations*, 2021.

[CCMS19a] T-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2448–2467, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics.

[CCMS19b] TH Hubert Chan, Kai-Min Chung, Bruce M Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2448–2467. SIAM, 2019.

[ccp] California consumer privacy act (ccpa). `https://oag.ca.gov/privacy/ccpa`.

[CDJ+21] Amrita Roy Chowdhury, Bolin Ding, Somesh Jha, Weiran Liu, and Jingren Zhou. Strengthening order preserving encryption with differential privacy, 2021.

[CDPM18] Thee Chanyaswad, Alex Dytso, H. Vincent Poor, and Prateek Mittal. Mvg mechanism: Differential privacy under matrix-valued query. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 230–246, New York, NY, USA, 2018. ACM.

[CdWM+17] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptology ePrint Archive*, 2017:35, 2017.

[cen] National population by characteristics: 2010-2019. `https://www.census.gov/data/tables/time-series/demo/popest/2010s-national-detail.html/`.

[Cen20] Disclosure avoidance and the 2020 census. `https://www.census.gov/about/policies/privacy/statistical_safeguards/disclosure-avoidance-2020-census.html/`, 2020.

[CEP17] Kostas Chatzikokolakis, Ehab Elsalamouny, and Catuscia Palamidessi. Efficient utility improvement for location privacy. *Proceedings on Privacy Enhancing Technologies*, 2017, 10 2017.

[CFLG21] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. 2021.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2017.

[CGJvdM21] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. Eiffel: Ensuring integrity for federated learning, 2021.

[CKS18] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 131–146, New York, NY, USA, 2018. ACM.

[CLL+17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. In *arXiv:1712.05526*, 2017.

[CLM13] Ellick M. Chan, Peifung E. Lam, and John C. Mitchell. Understanding the challenges with medical data segmentation for privacy. In *Proceedings of the 2013 USENIX Conference on Safety, Security, Privacy and Interoperability of Health Information Technologies*, HealthTech'13, page 2, USA, 2013. USENIX Association.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[CLWW16] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. Practical order-revealing encryption with limited leakage. In *Revised Selected Papers of the 23rd International Conference on Fast Software Encryption - Volume 9783*, FSE 2016, page 474–493, Berlin, Heidelberg, 2016. Springer-Verlag.

[CM99] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 107–122, Berlin, Heidelberg, 1999. Springer-Verlag.

[com16a] Ciphercloud. `http://www.ciphercloud.com/`, 2016.

[com16b] Microsoft, always encrypted (database engine). `https://msdn.microsoft.com/en-us/library/mt163865.aspx/`, 2016.

[com16c] Perspecsys: A blue coat company. `http://perspecsys.com/`, 2016.

[CPS+12] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. *2012 IEEE 28th International Conference on Data Engineering*, Apr 2012.

[CRJ20] Amrita Roy Chowdhury, Theodoros Rekatsinas, and Somesh Jha. Data-dependent differentially private parameter learning for directed graphical models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1939–1951. PMLR, 13–18 Jul 2020.

[CSL+08] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE Symposium on Security and Privacy (SP)*, pages 81–95, 2008.

[CSS12a] T-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *Proceedings of the 20th Annual European Conference on Algorithms*, ESA'12, pages 277–288, Berlin, Heidelberg, 2012. Springer-Verlag.

[CSS12b] T. H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security*, pages 200–214, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[CSU+18] Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via mixnets. *CoRR*, abs/1808.01394, 2018.

[CSU+19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 375–403, Cham, 2019. Springer International Publishing.

[CW09] Scott A. Crosby and Dan S. Wallach. Efficient data structures for tamper-evident logging. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, page 317–334, USA, 2009. USENIX Association.

[CWCP18] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *Proceedings of the International Conference on Machine Learning*, 2018.

[dat] Youtube system requirements. `https://support.google.com/youtube/answer/78358?hl=en`.

[dat16a] Anthem. anthem data breach. `https://www.anthemfacts.com/`, 2016.

[dat16b] Yahoo data breach. `https://money.cnn.com/2016/09/22/technology/yahoo-data-breach/`, 2016.

[dat17a] Wikipedia. sony pictures entertainment hack. `https://en.wikipedia.org/wiki/Sony_Pictures_Entertainment_hack/`, 2017.

[dat17b] Wikipedia. target customer privacy. `https://en.wikipedia.org/wiki/Target_Corporation#Customer_privacy/`, 2017.

[dat18] Wikipedia. facebook–cambridge analytica data scandal. `https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scanda/`, 2018.

[dat19] Facebook data breach. `https://www.forbes.com/sites/daveywinder/2019/09/05/facebook-security-snafu-exposes-419-million-user-phone-numbers/`, 2019.

[DDC16] F. Betül Durak, Thomas M. DuBuisson, and David Cash. What else is revealed by order-revealing encryption? In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1155–1166, New York, NY, USA, 2016. Association for Computing Machinery.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHK20] Shaleen Deep, Xiao Hu, and Paraschos Koutris. Join project query evaluation using matrix multiplication. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2020.

[DHK21a] Shaleen Deep, Xiao Hu, and Paraschos Koutris. Enumeration algorithms for conjunctive queries with projection. In *24th International Conference on Database Theory (ICDT 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[DHK21b] Shaleen Deep, Xiao Hu, and Paraschos Koutris. Space-time tradeoffs for answering boolean conjunctive queries. *arXiv preprint arXiv:2109.10889*, 2021.

[DJW13] 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.

[DK18] Shaleen Deep and Paraschos Koutris. Compressed representations of conjunctive query results. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 307–322, 2018.

[DK21] Shaleen Deep and Paraschos Koutris. Ranked enumeration of conjunctive query results. In *24th International Conference on Database Theory (ICDT 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pages 486–503, Berlin, Heidelberg, 2006. Springer-Verlag.

[DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3571–3580. Curran Associates, Inc., 2017.

[DMG+18] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. Asynchronous byzantine machine learning (the case of sgd). In *ICML*, 2018.

[DR14a] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, August 2014.

[DR14b] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[DR16] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy, 2016.

[DSZ15] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby - a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[EFM+18] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. *CoRR*, abs/1811.12469, 2018.

[EGS03] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 211–222, New York, NY, USA, 2003. ACM.

[EMP] https://github.com/emp-toolkit.

[enc] Nist, block cipher techniques. `https://csrc.nist.gov/projects/block-cipher-techniques/bcm/modes-development/`.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[ES15] Hamid Ebadi and David Sands. Featherweight pinq, 2015.

[FCJG20] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438, 1987.

[FPE15] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries, 2015.

[FVY+17] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham. Sok: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 172–191, May 2017.

[FYB18] Clement Fung, Chris J.M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. In *arXiv:1808.04866*, 2018.

[Gao03] Shuhong Gao. *A New Algorithm for Decoding Reed-Solomon Codes*, pages 55–68. Springer US, Boston, MA, 2003.

[GBDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Robert Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016.

[GbF14] Carl Gunter, Mike berry, and Martin French. Decision support for data segmentation (ds2): application to pull architectures for hie. 2014.

[GCP] Google cloud platform. `https://cloud.google.com`.

[GDGG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *arXiv:1708.06733*, 2017.

[gdp] General data protection regulation gdpr. `https://gdpr-info.eu/`.

[GG13] Joachim von zur Gathen and Jrgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, USA, 3rd edition, 2013.

[GHIM19] Chang Ge, Xi He, Ihab F. Ilyas, and Ashwin Machanavajjhala. Apex: Accuracy-aware differentially private data exploration. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 177–194, New York, NY, USA, 2019. ACM.

[GJJ+18] Irene Giacomelli, Somesh Jha, Marc Joye, C. David Page, and Kyonghwan Yoon. Privacy-preserving ridge regression with only linearly-homomorphic encryption. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security*, pages 243–261, Cham, 2018. Springer International Publishing.

[GJK+18] Irene Giacomelli, Somesh Jha, Ross Kleiman, David Page, and Kyonghwan Yoon. Privacy-preserving collaborative prediction using random forests, 2018.

[GKL+20] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Rachit Li, Lucy Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores, 2020.

[GLMP18] Paul Grubbs, Marie-Sarah Lacharite, Brice Minaud, and Kenneth G. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 315–331, New York, NY, USA, 2018. Association for Computing Machinery.

[GLMP19a] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1067–1083, 2019.

[GLMP19b] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1067–1083, 2019.

[Gre16] Andy Greenberg. Apple's 'differential privacy' is about collecting your data—but not *your* data. *Wired*, Jun 13 2016.

[GRR19a] Adam Groce, Peter Rindal, and Mike Rosulek. Cheaper private set intersection via differentially private leakage. Cryptology ePrint Archive, Report 2019/239, 2019. https://eprint.iacr.org/2019/239.

[GRR19b] Adam Groce, Peter Rindal, and Mike Rosulek. Cheaper private set intersection via differentially private leakage. *Proceedings on Privacy Enhancing Technologies*, 2019:25–6, 2019.

[GSB+16] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptology ePrint Archive*, 2016:892, 2016.

[GSB+17a] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *PoPETs*, 2017:345–364, 2017.

[GSB+17b] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 655–672, 2017.

[GTT+19] Mehmet Emre Gursoy, Acar Tamersoy, Stacey Truex, Wenqi Wei, and Ling Liu. Secure and utility-aware data collection with condensed local differential privacy. *ArXiv*, abs/1905.06361, 2019.

[GZ07] T. Ge and S. Zdonik. Fast, secure encryption for indexing in a column-oriented dbms. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 676–685, 2007.

[HGH+14] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C. Pierce, and Aaron Roth. Differential privacy: An economic method for choosing epsilon. *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410, 2014.

[HILM02] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, page 216–227, New York, NY, USA, 2002. Association for Computing Machinery.

[HKJ20] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. Secure byzantine-robust machine learning, 2020.

[HMD14] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, page 1447–1458, New York, NY, USA, 2014. Association for Computing Machinery.

[HMFS17] Xi He, Ashwin Machanavajjhala, Cheryl Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1389–1406, New York, NY, USA, 2017. ACM.

[HMVK21] Thomas Humphries, Rasoul Akhavan Mahdavi, Shannon Veitch, and Florian Kerschbaum. Selective MPC: distributed computation of differentially private key value statistics. *CoRR*, abs/2107.12407, 2021.

[HRMS10a] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, September 2010.

[HRMS10b] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, Sep 2010.

[HTG17] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data, 2017.

[HZRS16] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[IQr16] IQrypt. Iqrypt: Encrypt and query your database, 2016.

[JKU11] Kristján Valur Jónsson, Gunnar Kreitz, and Misbah Uddin. Secure multiparty sorting and applications. Cryptology ePrint Archive, Report 2011/122, 2011. `https://eprint.iacr.org/2011/122`.

[JNS17] Noah M. Johnson, Joseph P. Near, and Dawn Xiaodong Song. Practical differential privacy for SQL queries using elastic sensitivity. *CoRR*, abs/1706.09479, 2017.

[JNS18] Noah Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proc. VLDB Endow.*, 11(5):526–539, January 2018.

[KAK10] Hasan Kadhem, Toshiyuki Amagasa, and Hiroyuki Kitagawa. A secure and efficient order preserving encryption scheme for relational databases. In *KMIS*, 2010.

[Kea98] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, November 1998.

[Ker15] Florian Kerschbaum. Frequency-hiding order-preserving encryption. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 656–667, New York, NY, USA, 2015. ACM.

[KGM+14] J. Kepner, V. Gadepally, P. Michaleas, N. Schear, M. Varia, A. Yerukhimovich, and R. K. Cunningham. Computing on masked data: a high performance method for improving big data veracity. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6, 2014.

[KKK+16] Sungwook Kim, Jinsu Kim, Dongyoung Koo, Yuna Kim, Hyunsoo Yoon, and Junbum Shin. Efficient privacy-preserving matrix factorization via fully homomorphic encryption: Extended abstract. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pages 617–628, New York, NY, USA, 2016. ACM.

[KL14a] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2014.

[KL14b] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2014.

[KLN+08] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, Oct 2008.

[KLS21a] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation, 2021.

[KLS21b] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. *ArXiv*, abs/2102.06387, 2021.

[KMA+19] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G.L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adria Gascon, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecny, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Ozgur, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramer, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. In *arXiv:1912.04977*, 2019.

[KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multiparty computation. Cryptology ePrint Archive, Report 2011/272, 2011. `https://eprint.iacr.org/2011/272`.

[KMY+16] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *ArXiv*, abs/1610.05492, 2016.

[KPT19] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Data recovery on encrypted databases with k-nearest neighbor query leakage. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1033–1050, 2019.

[KPT20] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1223–1240, 2020.

[KPT21] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks. Cryptology ePrint Archive, Report 2021/093, 2021. `https://eprint.iacr.org/2021/093`.

[Kri] Alex Krizhevsky. The cifar-10 dataset.

[KT19] F. Kerschbaum and A. Tueno. An efficiently searchable encrypted data structure for range queries. In *In: Sako K., Schneider S., Ryan P. (eds) Computer Security – ESORICS 2019 ESORICS 2019. Lecture Notes in Computer Science, vol 11736. Springer, Cham*, 2019.

[KTH+19] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: A differentially private sql query engine. *Proc. VLDB Endow.*, 12(11):1371–1384, July 2019.

[Kul19] Tejas Kulkarni. Answering range queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 1832–1834, New York, NY, USA, 2019. Association for Computing Machinery.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LC04] Shu Lin and Daniel J. Costello. *Error control coding: fundamentals and applications*. Pearson/Prentice Hall, Upper Saddle River, NJ, 2004.

[LC11] Jaewoo Lee and Chris Clifton. How much is enough? choosing ϵ for differential privacy. In *Proceedings of the 14th International Conference on Information Security*, ISC'11, pages 325–340, Berlin, Heidelberg, 2011. Springer-Verlag.

[LCB] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits.

[LCFK21] Jingjie Li, Amrita Roy Chowdhury, Kassem Fawaz, and Younghyun Kim. Kalεido: Real-Time privacy control for Eye-Tracking systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1793–1810. USENIX Association, August 2021.

[LCW+20] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *CoRR*, abs/2002.00211, 2020.

[LDGG18] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. pages 273–294, 2018.

[LLSY16] N. Li, M. Lyu, D. Su, and W. Yang. *Differential Privacy: From Theory to Practice*. Morgan and Claypool, 2016.

[LMP18] M. Lacharité, B. Minaud, and K. G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 297–314, 2018.

[Lov19] Ben Lovejoy. We now spend more than a quarter of our lives online, shows digital 2019 report. `https://9to5mac.com/2019/01/31/digital-2019/`, 2019.

[LP09a] Yehuda Lindell and Benny Pinkas. A proof of security of yao’s protocol for two-party computation. *J. Cryptol.*, 22(2):161–188, April 2009.

[LP09b] Yehuda Lindell and Benny Pinkas. A proof of security of yao’s protocol for two-party computation. *J. Cryptol.*, 22(2):161–188, April 2009.

[LP15] Marie-Sarah Lacharit´e and Kenneth G Paterson. A note on the optimality of frequency analysis vs. l_p-optimization, 2015.

[LP18] Marie-Sarah Lacharité and Kenneth G. Paterson. Frequency-smoothing encryption: preventing snapshot attacks on deterministically encrypted data. *IACR Transactions on Symmetric Cryptology*, 2018(1):277–313, Mar. 2018.

[LPL+09] Seungmin Lee, Tae-Jun Park, Donghyeok Lee, Taekyong Nam, and Sehun Kim. Chaotic order preserving encryption for efficient and secure queries on databases. *IEICE Transactions*, 92-D:2207–2217, 11 2009.

[LSL17] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *PVLDB*, 10:637–648, 2017.

[LVS+21] Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Zhiwei Steven Wu. Leveraging public data for practical private query release, 2021.

[LW12] Dongxi Liu and Shenlu Wang. Programmable order-preserving secure index for encrypted database query. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, page 502–509, USA, 2012. IEEE Computer Society.

[LW13] Dongxi Liu and Shenlu Wang. Nonlinear order preserving index for encrypted database query in service cloud environments. *Concurr. Comput. Pract. Exp.*, 25:1967–1984, 2013.

[LW16] Kevin Lewi and David J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1167–1178, New York, NY, USA, 2016. Association for Computing Machinery.

[LWLZ+20] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. Estimating numerical distributions under local differential privacy. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 621–635, New York, NY, USA, 2020. Association for Computing Machinery.

[LXC+19] Liping Li, Wei Xu, Tianyi Chen, Georgios Giannakis, and Qing Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1544–1551, 07 2019.

[Mat] Wolfram Mathworld. Lagrange interpolating polynomial. `https://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html`.

[MCCJ21] Casey Meehan, Amrita Roy Chowdhury, Kamalika Chaudhuri, and Somesh Jha. A shuffling framework for local differential privacy, 2021.

[McE03] R. J. McEliece. The guruswami–sudan decoding algorithm for reed–solomon codes, 2003.

[MCO+15] Charalampos Mavroforakis, Nathan Chenette, Adam O'Neill, George Kollios, and Ran Canetti. Modular order-preserving encryption, revisited. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 763–777, New York, NY, USA, 2015. Association for Computing Machinery.

[McS09] Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM.

[MG18a] Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 490–507, New York, NY, USA, 2018. ACM.

[MG18b] Sahar Mazloom and S. Dov 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.

[Mir17a] Ilya Mironov. Renyi differential privacy. *CoRR*, abs/1702.07476, 2017.

[Mir17b] Ilya Mironov. Rényi differential privacy. *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, Aug 2017.

[MKA+08] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *2008 IEEE 24th International Conference on Data Engineering*, pages 277–286, 2008.

[MMMM20] Ryan McKenna, Raj Kumar Maity, Arya Mazumdar, and Gerome Miklau. A workload-adaptive mechanism for linear queries under local differential privacy, 2020.

[MMR+17] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2017.

[MPC] http://www.multipartycomputation.com/mpc-software.

[MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 126–142, Berlin, Heidelberg, 16–20 August 2009. Springer-Verlag, Springer Berlin Heidelberg.

[MR17] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data, 2017.

[MRS18] Matteo Maffei, Manuel Reinert, and Dominique Schröder. On the security of frequency-hiding order-preserving encryption. In Srdjan Capkun and Sherman S. M. Chow, editors, *Cryptology and Network Security*, pages 51–70, Cham, 2018. Springer International Publishing.

[MSDCS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2019.

[MZ15] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2871–2877, 2015.

[MZ17] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, May 2017.

[Nel] Jelani Nelson. Sketching algorithms.

[NH12] Arjun Narayan and Andreas Haeberlen. Djoin: Differentially private join queries over distributed databases. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pages 149–162, Berkeley, CA, USA, 2012. USENIX Association.

[NHC21] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. Local and central differential privacy for robustness and privacy in federated learning, 2021.

[NIW+13] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 801–812, New York, NY, USA, 2013. ACM.

[NKW15] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 644–655, New York, NY, USA, 2015. Association for Computing Machinery.

[NRY+21] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Ahmad-Reza Sadeghi, Thomas Schneider, and Shaza Zeitouni. Flguard: Secure and private federated learning, 2021.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019.

[NTL] `https://libntl.org/`.

[NWI+13] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348, May 2013.

[NXY+16] Thông T. Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. Collecting and analyzing data from smart device users with local differential privacy. *CoRR*, abs/1606.05053, 2016.

[NYC] Hospital inpatient discharges. `https://health.data.ny.gov/Health/Hospital-Inpatient-Discharges-SPARCS-De-Identified/u4ud-w55t/`.

[Ode09] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

[PLZ13a] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*, pages 463–477, 2013.

[PLZ13b] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*, pages 463–477, 2013.

[PRZB11] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, page 85–100, New York, NY, USA, 2011. Association for Computing Machinery.

[PUD] Hospital discharge data public use data file. `http://www.dshs.state.tx.us/THCIC/Hospitals/Download.shtm/`.

[PZW+20] Xudong Pan, Mi Zhang, Duocai Wu, Qifan Xiao, Shouling Ji, and Zhemin Yang. Justinian's GAAvernor: Robust distributed learning with gradient aggregation agent. In *USENIX Security*, pages 1641–1658, 2020.

[QYL13a] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 757–768, April 2013.

[QYL13b] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. *Proc. VLDB Endow.*, 6(14):1954–1965, September 2013.

[QYY+16] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 192–203, New York, NY, USA, 2016. ACM.

[RACY16] Daniel S. Roche, Daniel Apon, Seung Geol Choi, and Arkady Yerukhimovich. Pope: Partial order preserving encoding. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1131–1142, New York, NY, USA, 2016. Association for Computing Machinery.

[ran] Ranking. `https://en.wikipedia.org/wiki/Ranking/`.

[R.B84] Williams R.B.G. *Measures of Central Tendency.* 1984.

[RCWH+20] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Cryptϵ: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 603–619, New York, NY, USA, 2020. Association for Computing Machinery.

[RN10] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 735–746, New York, NY, USA, 2010. ACM.

[RNFH19] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. Honeycrisp: Large-scale differentially private aggregation without a trusted core. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 196–210, New York, NY, USA, 2019. Association for Computing Machinery.

[RWCP19] Shashank Rajput, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Detox: A redundancy-based framework for faster and more robust gradient aggregation. 2019.

[sal15] Sf salaries, kaggle. `https://www.kaggle.com/kaggle/sf-salaries/`, 2015.

[SC17] Fundamental right to privacy. `https://www.scobserver.in/reports/k-s-puttaswamy-right-to-privacy-judgment-of-the-court-in-plain-english-i/`, 2017.

[Sca] https://github.com/kuleuven-cosic/scale-mamba.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *In CRYPTO*, pages 148–164. Springer-Verlag, 1999.

[Sch16] Andreas Schaad. Sap seeed project, 2016.

[SGA20] Jinhyun So, Basak Guler, and A. Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal in Selected Areas in Communications: Machine Learning in Communications and Networks*, 2020.

[SGA21] Jinhyun So, Basak Guler, and A. Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning, 2021.

[SH21] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[SHCGR+11] Elaine Shi, T.-H Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. volume 2, 01 2011.

[SKL17] Jacob Steinhardt, Pang Wei W. Koh, and Percy S. Liang. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3517–3529, 2017.

[SKSM19a] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? *ArXiv*, abs/1911.07963, 2019.

[SKSM19b] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? In *arXiv:1911.07963*, 2019.

[SS19] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning (ICML)*, pages 5739–5748, 2019.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. pages 190–199. Springer-Verlag, 1996.

[sto] Aws pricing. `https://aws.amazon.com/s3/pricing/`.

[STS16] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *ACM ACSAC*, pages 508–519, 2016.

[TPH] Chunming Tang, Dingyi Pei, and Zhuojun Liu Yong He. Non-interactive and information-theoretic secure publicly verifiable secret sharing.

[TSD20] M. C. Tschantz, S. Sen, and A. Datta. Sok: Differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 354–371, 2020.

[VBMW+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel sgx kingdom with transient out-of-order execution. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pages 991–1008, Berkeley, CA, USA, 2018. USENIX Association.

[vdHLZZ15] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, page 137–152, New York, NY, USA, 2015. Association for Computing Machinery.

[VSA17] Lars Vilhuber, Ian M. Schmutte, and John M. Abowd. Proceedings from the 2016 NSF–Sloan workshop on practical privacy, Jan 2017.

[VXK21] Raj Kiriti Velicheti, Derek Xia, and Oluwasanmi Koyejo. Secure byzantine-robust distributed learning via clustering, 2021.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60 60, no. 309:63–69, 1965.

[WBK19] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *AISTATS*, 2019.

[WBLJ17a] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, pages 729–745, Berkeley, CA, USA, 2017. USENIX Association.

[WBLJ17b] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *Proceedings of the 26th USENIX Conference on Security Symposium*, pages 729–745, Berkeley, CA, USA, 2017. USENIX Association.

[WCM18] Sameer Wagh, Paul Cuff, and Prateek Mittal. Differentially private oblivious ram. *Proceedings on Privacy Enhancing Technologies*, 2018(4):64–84, 2018.

[WHMM20] Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. Dp-cryptography: Marrying differential privacy and cryptography in emerging applications, 2020.

[WLJ17] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private heavy hitter identification, August 2017.

[WLJ18] T. Wang, N. Li, and S. Jha. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 127–143, May 2018.

[WNW+17] S. Wang, Y. Nie, P. Wang, H. Xu, W. Yang, and L. Huang. Local private ordinal data distribution estimation. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[WRK17] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 21–37, New York, NY, USA, 2017. ACM.

[WYS+19] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019.

[XDHZ19] Zhuolun Xiang, B. Ding, X. He, and Jingren Zhou. Linear and range counting under metric-based local differential privacy. *arXiv: Cryptography and Security*, 2019.

[XHCL20] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *ICLR*, 2020.

[Xie19] Cong Xie. Zeno++: robust asynchronous SGD with arbitrary number of byzantine workers. *CoRR*, abs/1903.07020, 2019.

[XKG19] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *Proceedings of the International Conference on Machine Learning*, 2019.

[XKG20] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno++: Robust fully asynchronous SGD. In *Proceedings of the International Conference on Machine Learning*, 2020.

[XWG10] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, 2010.

[XZX+12] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In *2012 IEEE 28th International Conference on Data Engineering*, pages 32–43, April 2012.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, Oct 1986.

[YCKB19] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning (ICML)*, 2019.

[YMV+21] Hongxu Yin, Arun Mallya, Arash Vahdat, José Manuel Álvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16332–16341, 2021.

[Zal] Zalando. Fashion mnist.

[ZCX+] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 587–595.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*, EUROSAM '79, page 216–226, Berlin, Heidelberg, 1979. Springer-Verlag.

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *NeurIPS*, 2019.

[ZMK+18] Dan Zhang, Ryan McKenna, Ios Kotsogiannis, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. EKTELO: A framework for defining differentially-private computations. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 115–130, 2018.

[ZWL+18] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 212–229, New York, NY, USA, 2018. ACM.