TECHNISCHE UNIVERSITÄT DARMSTADT

Master Thesis

Securely Realizing Output Privacy in MPC

Liang Zhao

July 7, 2022

ENCRYPTO CRYPTOGRAPHY AND PRIVACY ENGINEERING

Cryptography and Privacy Engineering Group
Department of Computer Science
Technische Universität Darmstadt

Supervisors: M.Sc. Helen Möllering
M.Sc. Oleksandr Tkachenko
Prof. Dr.-Ing. Thomas Schneider

Erklärung zur Abschlussarbeit gemäß §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Liang Zhao, die vorliegende Master Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Thesis Statement pursuant to §23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Liang Zhao, have written the submitted Master Thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are identical in content.

Darmstadt, July 7, 2022

Liang Zhao

Abstract

Nowadays, the world has become an information-driven society where the distribution and processing of information is one important economic activity. However, the centralized database may contain sensitive data that would lead to privacy violations if the data or its aggregate statistics are disclosed. Secure Multi-Party Computation (SMPC) enables multiple parties to compute an arbitrary function on their private inputs and reveals no information beyond the computation result. Differential Privacy (DP) is a technique that can preserve the individual's privacy by perturbing the aggregate statistics with random noise. The hybrid approach combining SMPC and DP would provide a robust privacy guarantee and maintain the utility of the aggregate statistics. The theoretical definition of DP assumes precise noise sampling and arithmetic operations under real numbers. However, in the practical implementation of perturbation mechanisms, fixed-point or floating-point numbers are used to represent real numbers that lead to the violation of DP, as Mironov [Mir12] and Jin et al. [JMRO22] showed. This thesis explores the possibilities of *securely* generating distributed random noise in SMPC settings and builds a variety of perturbation mechanisms. Specifically, we evaluate the performance of fixed-point and floating-point arithmetic for noise generation in SMPC and choose the most efficient SMPC protocols to build the perturbation mechanisms.

Contents

1 Introduction

Technologies such as Machine Learning (ML) rely heavily on massive data analysis and pose severe privacy concerns as the individual's information in the highly centralized database may be misused. Privacy violation comes in many forms and is not directly visible. Therefore, it is crucial to provide appropriate privacy protections for user data. Many methods have been explored before to protect the privacy of individuals in the database. Since 2008, cryptographers have proposed Privacy-Preserving Machine Learning (PPML) algorithms to avoid the privacy breach of the training dataset based on SMPC. SMPC enables multiple parties to securely perform distributed computations with parties' private inputs so that only the computation results are revealed.

Let us consider a typical scenario of PPML: Alice wishes to investigate if she has genetic disorders while keeping her genomic data secret. As a service provider, Bob has trained an ML model that can predict genetic disorders given genomic data. However, Bob wants to keep his ML model private as it is his intellectual property that he aims to monetize. One unrealistic solution would be to rely on a trusted third party to analyze Alice's genetic data with Bob's ML model. However, as a trusted third party does rarely exist in practice, Alice and Bob can deploy a SMPC protocol to simulate a trusted third party.

Although SMPC can guarantee the users' computational privacy, an adversary can still infer users' sensitive information from the computation output. Shokri et al. [SSSS17] showed a membership inference attack that can determine if a data record was in the model's training dataset by making an adversarial usage of ML algorithms. One solution to mitigate such an attack is to deploy differentially private mechanisms. The concept of DP was introduced by Dwork et al. [Dwo06; DMNS06] that limits private information disclosure by adding calibrated noise to the revealed output. However, Mironov [Mir12] and Jin et al. [JMRO22] demonstrated a series of attacks against the differentially private mechanisms implemented under floating-point arithmetics.

To the best of our knowledge, most prior works [RN10; SCR+11; ÁC11; CSS12; EKM+14; WHWX16; BRB+17; JWEG18; TBA+19; KVH+21; YSMN21] that combine SMPC and DP do not consider the above security issues. This work attempts to fill this gap by providing *secure* noise generation methods in multi-party settings based on the state-of-the-art SMPC framework MOTION [BDST22]. The start point of this work is the secure noise generation methods and differentially private mechanisms from works [Mir12; Tea20; CKS20]. The fundamental idea of secure noise generation is to generate discrete noise and re-scale it precisely under floating-point implementation to simulate continuous noise such that the noise satisfies the differential privacy requirements. We investigate the potential of applying these noise generation methods in the SMPC. Besides, we aim to achieve DP and maintain the optimal utility of the computation result by adding a minimal amount of noise.

Contributions.

1. The noise is generated in a fully distributed manner that maintains the optimal utility of the aggregate statistics by introducing the minimal amount of noise required to satisfy DP. We consider the outsourcing scenario [KR11], i.e., the data owners first secret share their private inputs to multiple ($N \geq 2$) non-colluding computation parties, and the computation parties execute the SMPC protocols to compute the desired functionality and perturb the result. MOTION [BDST22] supports full-threshold security, and the computation result is secure if at least one computation party is honest and non-collusive. Therefore, the computation parties can jointly generate the shares of a publicly unknown noise with the same magnitude as the noise generated by a single trusted server.

2. We support a variety of differentially private mechanisms such as the (discrete) Laplace mechanism [CSS12; GRS12; DR+14], the (discrete) Gaussian mechanism [DR+14; CKS20], and the snapping mechanism [Mir12] in SMPC.

3. We implement fixed-point and floating-point operations in the binary circuit-based and the arithmetic sharing SMPC protocols and evaluate the performance. We use Single Instruction Multiple Data (SIMD) instructions to eliminate the independent iterations in the sampling algorithms and improve the protocol performance.

Thesis Outline. This thesis is organized as follows: Chapter 2 gives the preliminaries on the concept of secure multiparty computation and differential privacy with motivating examples and formal definitions. Chapter 3 presents a summary and discussion of the related works. Chapter 4 describes the details of the secure noise generation methods, differentially private mechanisms and our modifications. Chapter 5 provides the procedure to combine SMPC protocols and differentially private mechanisms, building blocks, and the SMPC protocols for differentially private mechanisms. Chapter 6 evaluates the performance of fixed-point and floating-point and the differentially private mechanisms. Chapter 7 concludes this work by summarizing the results and pointing out several directions for further research.

2 Preliminaries

In this chapter, we start with the notations used in this thesis § 2.1. Afterwards, we describe the basic knowledge of secure multi-party computation in § 2.2. Finally, we introduce the background knowledge and theory of differential privacy in § 2.3.

2.1 Notations

For $a, b, c \in \mathbb{N}$, (a, b) denotes $\{x \in \mathbb{R} \,|\, a < x < b\}$, and $[a, b]$ denotes $\{x \in \mathbb{R} \,|\, a \leq x \leq b\}$. $\{a, b, c\}$ is a set containing the three numbers. $\mathbb{D}$ denotes the set of floating-point numbers, and $\mathbb{D} \cap (a, b)$ contains floating-point numbers in the interval (a, b).

Let $P_1, \ldots P_N$ denote N computation parties. The value x that is secret shared among N parties are denoted by $\langle x \rangle^{S,D} = \left(\langle x \rangle_1^{S,D}, \ldots, \langle x \rangle_N^{S,D}\right)$, where $\langle x \rangle_i^{S,D}$ is hold by party P_i. Superscript $S \in \{A, B, Y\}$ denotes the sharing type (cf. § 2.2.3): A for arithmetic sharing, B for Boolean sharing with GMW, Y for Yao sharing with BMR. Superscript $D \in \{UINT, INT, FX, FL\}$ indicates the data type: $UINT$ for unsigned integer, INT for signed integer, FX for fixed-point number, and FL for floating-point number. We omit subscript and subscript when it is clear from the context.

Bold symbol $\langle \boldsymbol{x} \rangle$ denotes a vector of ℓ shared bits. We use XOR ($\oplus$), AND ($\wedge$), and NOT ($\neg$) in the logical opeartions. Let $\langle a \rangle^{S,D} \odot \langle b \rangle^{S,D}$ be the arithmetic operations on two shared numbers, where $\odot \in \{+, -, \cdot, \div, >, ==\}$. $\langle a \rangle^B \cdot \langle \boldsymbol{b} \rangle^B$ represents the bitwise AND operations between $\langle a \rangle^B$ and every Boolean sharing bit $\langle b \rangle^B \in \langle \boldsymbol{b} \rangle^B$. Let $\langle a \rangle^{FL} = \Pi^{UINT2FL}\left(\langle a \rangle^{UINT}\right)$ be the conversion from an shared unsigned integer $\langle a \rangle^{UINT}$ to a shared floating-point number $\langle a \rangle^{FL}$. Other data type conversion operations are defined in a similar manner.

2.2 Secure Multi-Party Computation

Secure Multi-Party Computation enables multiply parties to jointly evaluate a function on their private inputs while revealing only the computation result. Yao [Yao82] introduced the concept of secure two-party computation with Yao's Millionaires' problem (i.e., two millionaires wish to know who is richer without revealing their actual wealth) and proposed the garbled circuit protocol [Yao86] as a solution. In the garbled circuit protocol, the target function is represented as a Boolean circuit consisting of connected gates and wires. One party called gabler is responsible for garbling the circuit, and the other party called evaluator evaluates the garbled circuit and outputs the result.

Afterwards, Beaver, Micali and Rogaway (BMR) [BMR90] generalized Yao's garbled circuit protocol to multi-party settings. Goldreich, Micali and Wigderson (GMW) [GMW87] proposed a general solution to SMPC based on secret sharing, where each party splits his data into several shares and sends it to each of the parties. Secret sharing guarantees that any secret shares held by single party leak no information about the parties' private input.

Generally, the execution of MPC protocols is separated into two phases: an offline (or preprocessing) phase and an online phase. In the offline phase, the parties compute everything that does not depend on the private input. In the online phase, the parties compute the input-dependent part.

2.2.1 Security Model

The standard approach to prove the security of cryptographic protocols is to consider adversaries with different capabilities. We describe two types of adversaries: the *semi-honest* adversary and the *malicious* adversary. We refer to [EKR17, Chapter 2] for a formal and detailed description of the security model.

Semi-honest adversaries (also known as passive adversaries) try to infer additional information of other parties from the messages during the protocol execution without attempting to break the protocol. Therefore, it is a weak security model and only prevents the unintentional disclosure of information between parties. The semi-honest protocols are usually very efficient and the first step to design protocols with stronger security guarantees.

Malicious adversaries (also known as active adversaries) may cause corrupted parties to arbitrarily deviate from the protocol specification and attempt to learn information about the other parties' inputs. Protocols against malicious adversaries usually deploy cryptographic mechanisms to ensure that the parties cannot deviate from the protocol specification. Therefore, the protocol is often more expensive than the protocol against semi-honest adversaries.

2.2.2 Cryptographic Primitives for Secure Multi-Party Computation

Oblivious Transfer

Oblivious Transfer (OT) is a cryptographic primitive that enables two parties to obliviously transfer one value out of two values. Specifically, the sender has inputs (x_0, x_1), and the receiver has a choice bit c. Oblivious transfer protocol receives the inputs from the sender and receiver, and outputs x_c to the receiver. It guarantees that the sender does not learn anything about c and the receiver does not learn about x_{1-c}. Impagliazzo and Rudich [IR89] showed that a *black-box* reduction from OT to a one-way function [Isr06, Chapter 2] is as hard as proving $P \neq NP$, which implies that OT requires relatively expensive (than symmetric cryptography) public-key cryptography [RSA78].

Nevertheless, Ishai et al. [IKNP03] proposed OT *extension* techniques that extend a small number of OTs based on public-key cryptography to a large number of OTs with efficient symmetric cryptography. Asharov et al. [ALSZ17] proposed specific OT functionalities for the optimization of SMPC protocols, such as Correlated Oblivious Transfer (C-OT) and Random Oblivious Transfer (R-OT). In C-OT, the sender inputs a correlation function f_Δ (e.g., $f_\Delta(x) = x \oplus \Delta$, where Δ is only known by the sender) and receives random values x_0 and $x_1 = f_\Delta(x_0)$. The receiver inputs a choice bit c and receives x_c. In R-OT, the sender has no inputs and receives random values (x_0, x_1), and the receiver inputs a choice bit c and receives x_c.

Multiplication Triples

Multiplication Triples (MTs) were proposed by Beaver [Bea91] that can be precomputed to reduce the online complexity of SMPC protocols by converting expensive operations (e.g., arithmetic multiplication and logical AND) to linear operations (e.g., arithmetic addition and logical XOR).

A multiplication triple has the form $\left(\langle a\rangle^S, \langle b\rangle^S, \langle c\rangle^S\right)$ with $S \in \{B, A\}$. In Boolean sharing with GMW (cf. § 2.2.3), we have $c = a \wedge b$ for Boolean sharing and $c = a \cdot b$ for arithmetic sharing (cf. § 2.2.3). Multiplication triples can be generated using C-OT (cf. § 2.2.2) in the two-party setting [DSZ15] or the multi-party setting [BDST22].

2.2.3 MPC Protocols

We describe the SMPC protocols that is secure against $N-1$ semi-honest corruptions: Arithmetic sharing (cf. § 2.2.3), Boolean sharing with GMW (cf. § 2.2.3), and Yao sharing with BMR (cf. § 2.2.3). We refer to [DSZ15; BDST22] for a formal and detailed description.

Arithmetic Sharing (A)

Arithmetic sharing protocol enables parties to evaluate arithmetic circuits consisting of addition and multiplication gates. For arithmetic sharing, an ℓ-bit value x is shared additively among N parties as $\left(\langle x\rangle_1^A, \dots, \langle x\rangle_N^A\right) \in \mathbb{Z}_{2^\ell}^N$, where $x = \sum_{i=1}^N \langle x\rangle_i^A \mod 2^\ell$ and party P_i holds $\langle x\rangle_i^A$. Value x can be reconstructed by letting each party P_i sends $\langle x\rangle_i^A$ to one specific party who computes $x = \sum_{i=1}^N \langle x\rangle_i^A \mod 2^\ell$. The addition of arithmetic shares can be calculated locally without communication. Suppose the parties holds shares $\langle x\rangle_i^A$, $\langle y\rangle_i^A$, and wish to compute $z = a \cdot x + y + b$ with public value a, $b \in \mathbb{Z}_{2^\ell}$. Then, one specific party P_1 compute $\langle z\rangle_i^A = a \cdot \langle x\rangle_i^A + \langle y\rangle_i^A + b$, and the rest parties compute $\langle z\rangle_i^A = a \cdot \langle x\rangle_i^A + \langle y\rangle_i^A$ locally.

The multiplication of arithmetic shares can be performed using MTs (cf. § 2.2.2). Suppose $\left(\langle a\rangle^A, \langle b\rangle^A, \langle c\rangle^A\right)$ is an MTs in $\mathbb{Z}_{2^\ell}$, where $\langle c\rangle^A = \langle a\rangle^A \cdot \langle b\rangle^A$. To compute $\langle z\rangle_i^A = \langle x\rangle_i^A \cdot \langle y\rangle_i^A$, the parties first compute $\langle d\rangle_i^A = \langle x\rangle_i^A - \langle a\rangle_i^A$ and $\langle e\rangle_i^A = \langle y\rangle_i^A - \langle b\rangle_i^A$, and reconstruct them to get d and e. Finally, the parties compute the addition $\langle z\rangle_i^A = \langle c\rangle_i^A + e \cdot \langle x\rangle_i^A + d \cdot \langle y\rangle_i^A - d \cdot e$.

Boolean Sharing with GMW

Boolean GMW protocol [GMW87] enables multiple parties to evaluate a function represented as a Boolean circuit and uses XOR-based secret sharing. A bit $x \in \{0,1\}$ is shared among N parties as $\left(\langle x\rangle_1^B, \ldots, \langle x\rangle_N^B\right) \in \{0,1\}^N$, where $x = \bigoplus_{i=1}^N \langle x\rangle_i^B$. Boolean Sharing with GMW can be seen as a special case of arithmetic sharing. Operation $\langle x\rangle_i^B \oplus \langle y\rangle_i^B$ and $\langle x\rangle_i^B \wedge \langle y\rangle_i^B$ are computed similaryly as in arithmetic sharing.

Yao Sharing with BMR

We first present Yao's Garbled Circuit protocol [Yao86] following the steps described in work [LP09], and then, extend it to multi-party setting with BMR [BMR90] protocol. Yao's Garbled Circuit protocol [Yao86] enables two parties called garbler and the evaluator to securely evaluate any functionality represented as a Boolean circuit.

1. Circuit Garbling. The garbler converts the jointly decided function f into a Boolean circuit C, and selects a pair of random κ-bit keys $\left(k_0^i, k_1^i\right) \in \{0,1\}^{2\kappa}$ to represent logical value 0 and 1 for each wire. For each gate g in the Boolean circuit C with input wire a and b, and output wire c, the gabler uses the generated random keys $\left(k_0^a, k_1^a\right)$, $\left(k_0^b, k_1^b\right)$, $\left(k_0^c, k_1^c\right)$ to create a garbled gate table $\tilde{g}$ based on the function table of g. For example, suppose gate g is an AND gate and has function table Tab. 2.1, the gabler encrypts the keys of wire c and permutes the entries to generate the garbled table Tab. 2.2. Note that the symmetric encryption function Enc_k uses a secret-key k to encrypt the plaintext, and its decryption function Dec_k decrypts the ciphertext successfully only when the identical secret-key k is given. When all the gates in Boolean circuit C are garbled, the gabler sends the garbled circuit $\widetilde{C}$ that consists of garbled tables of all the gates to the evaluator for evaluation.

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Table 2.1: Function table of AND gate g.

2. Input Encoding. The gabler sends the wire keys corresponding to its input directly to the evaluator. To evaluate the garbled circuit $\widetilde{C}$, the evaluator needs the wire keys corresponding

$\tilde{a}$	$\tilde{b}$	$\tilde{c}$
k_1^a	k_1^b	$\mathrm{Enc}_{k_1^a,k_1^b}\left(k_1^c\right)$
k_0^a	k_1^b	$\mathrm{Enc}_{k_0^a,k_1^b}\left(k_0^c\right)$
k_0^a	k_0^b	$\mathrm{Enc}_{k_0^a,k_0^b}\left(k_0^c\right)$
k_1^a	k_0^b	$\mathrm{Enc}_{k_1^a,k_0^b}\left(k_0^c\right)$

Table 2.2: Garbled table of AND gate g with permuted entries.

to its input. For each of the evaluator's input wire i with corresponding input bit c, the evaluator and the gabler run a $1-out-of-2$-OT, where the gabler acts as a sender with inputs $\left(k_0^i, k_1^i\right)$, and the evaluator acts as a receiver with input c and receives k_c^i. Recall that $1-out-of-2$-OT (cf. § 2.2.2) guarantees that the gabler learns nothing about c and the evaluator learns only k_c^i.

3. Circuit Evaluation. After receiving $\widetilde{C}$ and the keys of input wires, the evaluator can evaluate the garbled circuit $\widetilde{C}$. For each gate g with input wire a and b, output wire c, the evaluator uses the input wire keys $\left(k^a, k^b\right)$ to decrypt the output key k^c. When all the gates in the garbled circuit $\widetilde{C}$ are evaluated, the evaluator obtains the keys for the output wires. To reconstruct the output, either the gabler sends the mapping from output wire keys to plaintext bits to the evaluator, or the evaluator sends the decrypted output wire keys to the gabler.

Optimizations for Yao's Garbled Circuits. In this part, we present several prominent optimizations for Yao's garbled circuit protocol [Yao86]. Note that in the evaluation of Yao's garbled circuit $\widetilde{C}$, the evaluator needs to decrypt at most four entries to obtain the correct key of the output wire. Point and permute [BMR90] technique helps the evaluator to identify the entry that should be decrypted (instead of decrypting four entries) in garbled tables by adding a permutation bit to each wire key. Garbled row reduction [NPS99] reduces the number of entries in the garble table from four to three by fixing the first entry to a constant value. Free-XOR [KS08] allows the parties to evaluate XOR gates without interactions by choosing all the wire key pairs $\left(k_0^i, k_1^i\right)$ with the same fixed distance R (R is kept secret to the evaluator), e.g., k_0^i is chosen at random and k_1^i is set to $R \oplus k_0^i$. Fixed-Key AES garbling [BHKR13] reduces the encryption and decryption workload of Yao's garbled circuit using a block cipher with a fixed key such that the AES key schedule is executed only once. Two-halves garbling [ZRE15] reduces the entry number of each AND gate from three to two by splitting each AND gate into two half-gates at the cost of one more decryption operation of the evaluator. Three-halves garbling [RR21] requires less 25% communication bits than the two-halves garbling at the cost of more computation.

BMR protocol [BMR90] extends Yao's garbled circuit protocol [Yao86] to the multi-party setting. Recall that in Yao's garbled circuit protocol, the circuit is first garbled by one party and evaluated by another party. At a high level, the BMR protocol enables the multi-party computation by having all parties jointly garbling the circuit in the offline phase, and then, each party send the garbled labels that are associated with their private inputs to other parties. Next, each party plays the role of the evaluator and evaluates the garbled circuit locally. Finally, the parties use the received garbled label and the the result of local evaluation to compute the output.

2.2.4 MPC Framework - MOTION

We build upon the recent MPC framework MOTION [BDST22] that provides the following novel features:

1. Support for MPC with N parties, full-threshold security (i.e., tolerating up to $N-1$ passive corruptions) and sharing conversions between ABY.
2. Implementation of primitive operations of MPC protocols at the circuit's gate level and evaluate it asynchronously, i.e., each gate is separately evaluated once their parent gates become ready.
3. Support for Single Instruction Multiple Data (SIMD), i.e., vectors of data are processed instead of single data, that can reduce memory footprint and communication.
4. Integration of HyCC compiler [BDK+18] that can generate efficient circuits for hybrid MPC protocols with functionality described in C programming language.

2.3 Differential Privacy

This section describes the concept of differential privacy in a formal mathematical view. We first introduce basic knowledge of probability distribution and random variable generation methods. Then, we describe traditional privacy preservation techniques and discuss their limitations. Next, we describe the motivation behind differential privacy and formalize its definition. Finally, we describe the differentially private mechanisms for realizing differential privacy.

Continuous Probability Distribution

Definition 2.3.1 (Continuous Uniform Distribution)**.** *The continuous uniform distribution with parameters a and b, has the following probability density function:*

$$Uni\left(x \mid a, b\right) = \begin{cases} \frac{1}{b-a} & for\ a \leq x \leq b \\ 0 & otherwise \end{cases} \tag{2.1}$$

$Uni\left(a, b\right)$ denotes the continuous uniform distribution with parameters a and b. We abuse notation and let $Uni\left(a, b\right)$ denote a random variable $x \sim Uni\left(a, b\right)$.

Definition 2.3.2 (Exponential Distribution). *The exponential distribution with rate parameter $\lambda > 0$ has the following probability density function:*

$$Exp(x \mid \lambda) = \begin{cases} \lambda e^{\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.2}$$

$Exp(\lambda)$ denotes the exponential distribution with parameter λ. $x \sim Exp(b)$ is a exponential random variable. The cumulative distribution function of an exponential distribution is:

$$\Pr(x \mid \lambda) = \begin{cases} 1 - e^{\lambda x} & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{2.3}$$

Definition 2.3.3 (Laplace Distribution [DR+14]). *The Laplace distribution with scale parameter b, has the following probability density function:*

$$Lap(x \mid b) = \frac{1}{2b} e^{\left(-\frac{|x|}{b}\right)} \tag{2.4}$$

The Laplace distribution is a symmetric version of the exponential distribution and is also called the double exponential distribution because it can be thought of as an exponential distribution assigned a randomly chosen sign. We write $Lap(b)$ to denote the Laplace distribution with scale parameter b and $x \sim Lap(b)$ to denote a Laplace random variable.

The cumulative distribution function of the Laplace distribution is defined as:

$$\Pr(x \leq X \mid b) = \begin{cases} \frac{1}{2} e^{\frac{X}{b}} & \text{for } X \leq 0 \\ 1 - \frac{1}{2} e^{-\frac{X}{b}} & \text{for } X > 0 \end{cases} \tag{2.5}$$

Definition 2.3.4 (Gaussian Distribution). *The univariate Gaussian (or standard normal) distribution with mean μ and standard deviation σ, has the following probability density function:*

$$\mathcal{N}(x \mid \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.6}$$

$\mathcal{N}(\mu, \sigma)$ denotes the Gaussian distribution with mean μ and standard deviation σ. $x \sim \mathcal{N}(\mu, \sigma)$ is a Gaussian random variable.

Discrete Probability Distribution

Definition 2.3.5 (Bernoulli distribution). *The Bernoulli distribution with parameter $p \in [0,1]$ has the following probability mass function for $x \in \{0,1\}$:*

$$Bern(x \mid p) = \begin{cases} p & \text{for } x = 1 \\ 1-p & \text{for } x = 0 \end{cases} \tag{2.7}$$

$Bern(p)$ denotes the Bernoulli distribution with parameter p. $x \sim Bern(p)$ is a Bernoulli random variable.

Definition 2.3.6 (Binomial Distribution). *The binomial distribution with parameters $n \in \mathbb{N}$ and $p \in [0,1]$ has the following probability mass function for $x \in \{0,1,2,\ldots,n\}$:*

$$Bino(x \mid n,p) = \frac{n!}{x!\,(n-x)!} p^x (1-p)^{n-x} \tag{2.8}$$

$Bino(n,p)$ denotes the binomial distribution with parameters n and p. $x \sim Bino(n,p)$ is a binomial random variable. Note that the distribution $Bino(n,p=0.5) - \frac{n}{2}$ is symmetric about the y-axis, which we denote by $SymmBino(n,p=0.5)$.

Definition 2.3.7 (Geometric Distribution). *The geometric distribution with parameter $p \in [0,1]$ has the following probability mass function for $x \in \{0,1,2,\ldots\}$:*

$$Geo(x \mid p) = (1-p)^x p \tag{2.9}$$

$Geo(p)$ denotes the geometric distribution with parameter p. $x \sim Geo(p)$ is a geometric random variable. Note that the geometric distribution counts the number of failures until the first success (each trial with success probability p). The cumulative distribution function of the geometric distribution is $\Pr(x \le X \mid p) = 1-(1-p)^{X+1}$.

Definition 2.3.8 (Discrete Laplace Distribution [CKS20]). *The discrete Laplace distribution (also known as the two-side geometric distribution [GRS12]) with parameter $t > 0$ and $x \in \mathbb{Z}$ has the following probability mass function:*

$$DLap(x \mid t) = \frac{e^{\frac{1}{t}} - 1}{e^{\frac{1}{t}} + 1} \cdot e^{-\frac{|x|}{t}} \tag{2.10}$$

$DLap(t)$ denotes the discrete Laplace distribution with parameter t. $x \sim DLap(t)$ is a discrete Laplace random variable. $DLap(t)$ can be generated by reflecting the $Geo(p)$ across the y-axis and rescaling it such that its cumulative probability in the interval $(-\infty,\infty)$ equals to one.

Definition 2.3.9 (Discrete Gaussian Distribution [CKS20]). *The discrete Gaussian distribution with mean μ and standard deviation σ, has the following probability mass function for $x \in \mathbb{Z}$:*

$$DGau(x \mid \mu, \sigma) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sum_{y \in \mathbb{Z}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}} \tag{2.11}$$

$DGau(\mu, \sigma)$ denotes the discrete Gaussian distribution with mean μ and standard deviation σ. $x \sim DGau(\mu, \sigma)$ is a discrete Gaussian random variable.

Probability Sampling Methods

Inverse Transform Sampling Method Inverse transform sampling method is a common method to generate random variables from certain distribution f using its inverted cumulative distribution F^{-1}.

Theorem 1 (Inverse Transform Sampling Method [Ste87, Theorem 2.1]). *Let F be a continuous distribution function on $\mathbb{R}$ with inverse F^{-1} defined as follows:*

$$F^{-1}(u) = \inf\{x : F(x) = u, 0 < u < 1\} \tag{2.12}$$

If $U \sim Uni(0,1)$ (cf. 2.3.1), then $F^{-1}(U)$ is a distribution with cumulative function F. Also, if X has cumulative function F, then $F(X)$ is uniformly distributed in the interval $[0,1]$.

Inverse Transform-Based Laplace Sampling Method For example, we can sample a Laplace random variable $Y \sim Lap(b)$ from an exponential distribuion with cumulative function: $F(x \mid b) = 1 - e^{-\frac{x}{b}}$ as follows [Knu14, Chapter 3.4]:

1. Sample $U \sim Uni(0,1) \setminus 1$ and $Z \sim Bern(0.5)$
2. $F^{-1}(U) = -b \cdot \ln(1-U)$ is a geometric random variable.
3. Transform $F^{-1}(U)$ to Laplace random variable Y with $Y \leftarrow (2Z-1) \cdot b \ln(1-U)$

Sampling from a Bernoulli Distribution $Algo^{Bern}(p)$ [KVH+21] samples a random variable $x \sim Bern(p)$ based on the comparison result between the generated uniform random variable $u \in (0,1)$ and parameter p.

Algorithm: $Algo^{Bern}(p)$

Input: p
Output: $x \sim Bern(p)$

```
u ←$ (0,1)
IF u < p
    RETURN x ← 1
ELSE
    RETURN x ← 0
```

Algorithm 2.1: Algorithm for sampling from Bernoulli distribution.

Sampling from a Geometric Distribution $Algo^{Geo}(0.5)$ [Wal74; Tea20] generates a geometric random variable $x \sim Geo(0.5)$ by first generating a ℓ-bit random string $r \in \{0,1\}^{\ell}$ (i.e., ℓ Bernoulli trials) and counting its leading zeros (i.e., number of trials before the first success). If there is no 1 bit in r (i.e., $LeadingZeros(r) = \ell$), the sampling algorithm fails. However, we can decrease the failure probability (0.5^{ℓ}) by increasing the length of the random string r.

Algorithm: $Algo^{Geo}(0.5)$

Input: 0.5
Output: $x \sim Geo(0.5)$

```
x ← 0
r ←$ {0,1}^ℓ
x ← LeadingZeros(r)
RETURN x
```

Algorithm 2.2: Algorithm for sampling from geometric distribution.

Sampling from a Discrete Laplace Distribution $Algo^{DLap\_EKMPP}(t)$ [EKM+14] generates a discrete Laplace random variable $x \sim DLap(t)$ by transforming two independent uniform random variables $u1, u2 \in (0,1)$ as follows:

Algorithm: $Algo^{DLap\_EKMPP}(t)$

Input: t

Output: $x \sim DLap(t)$

1 : $u_1 \leftarrow Uni(0,1)$

2 : $u_2 \leftarrow Uni(0,1)$

3 : **RETURN** $x \leftarrow \lfloor -t \cdot \ln(u_1) \rfloor - \lfloor -t \cdot \ln(u_2) \rfloor$

Algorithm 2.3: Algorithm for sampling from discrete Laplace distribution.

Number Representation

In this work, we rely on fixed-point and floating-point to perform the arithmetic operations.

Floating-Point Representation. Double-precision floating-point numbers occupy 64 bits (1 bit for sign, 11 bits for exponent, 52 bits for mantissa/significant) and are represented as follows:

$$(-1)^S (1.d_1 \dots d_{52})_2 \times 2^{(e_1 \dots e_{11})_2 - 1023}, \tag{2.13}$$

where $S \in \{0,1\}$ denotes the sign, $d_1 \dots d_{52} \in \{0,1\}^{52}$ denotes the mantissa (with implicit value of 1 at the first bit), $e_1 \dots e_{11} \in \{0,1\}^{11}$ denotes the binary representation of exponent (with 1023 as an offset).

Fixed-Point Representation. Fixed-point numbers are rational numbers represented as k-bit digits with an e-bit integer part (including sign bit s) and a f-bit fraction part as follows:

$$s \cdot \left(d_{e-2} \dots d_0 . d_{-1} \dots d_{-f}\right), \tag{2.14}$$

where $s \in \{-1,1\}$ and $e = k - f$.

2.3.1 Traditional Techniques for Privacy Preservation

revised based on feedback Suppose a fictitious hospital has collected massive data from thousands of patients and wants to make the data available to academic researchers such as data analysts. However, the data contains sensitive information of the patients, e.g., $ZipCode$, Age, $Nationality$ and $HealthCondition$. Because the hospital has an obligation, e.g., due to the EU General Data Protection Regulation (GDPR) [VV17], to preserve the privacy of the patients, it must take specific privacy preservation measures before releasing the data to academic researchers.

Let us assume that the released data is already anonymized by removing the identifying features such as the name and social security number (SSN) of the patients. Tab. 2.3 shows the anonymized medical records from the fictitious hospital. The attributes are divided into two groups: the non-sensitive attributes and the sensitive attribute. The value of the sensitive attributes must be kept secret for each individual in the records. We want to guarantee that no attacker can identify the patient and discover his $Condition$ by combining the records with other publicly available information.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

Table 2.3: Inpatient microdata [MKGV07].

A typical attack is the re-identification attack [Swe97] that combines the released anonymized data with publicly available information to re-identify individuals. One traditional approach against the re-identification attacks is to deploy privacy preservation methods that satisfy the notion of $k-anonymity$ [SS98] to anonymize the data and prevent the data subjects from being re-identified. More specifically, the $k-anonymity$ requires that for all individuals whose information appears in the dataset, each individual's information cannot be distinguished from at least $k-1$ other individuals.

Samarati et al. [SS98] introduced two techniques to achieve $k-anonymity$: data generalization and suppression. The former method makes the data less informative by mapping specific attribute values to a broader value range, and the latter method removes specific attribute values. As Tab. 2.4 shows, the values of attribute Age in the first eight records are replaced by value ranges such as < 30 and ≥ 40 after generalization. The values of attribute $Nationality$ are suppressed by being replaced with $*$. Finally, the records in Tab. 2.4 satisfy the $4-anonymity$ requirement. For example, given one patient's non-sensitive attribute values (e.g., $ZipCode$: $130**$, Age: < 30), there are at least three other patients with the same non-sensitive attribute values.

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	$130**$	< 30	*	Heart Disease
2	$130**$	< 30	*	Heart Disease
3	$130**$	< 30	*	Viral Infection
4	$130**$	< 30	*	Viral Infection
5	$1485*$	≥ 40	*	Cancer
6	$1485*$	≥ 40	*	Heart Disease
7	$1485*$	≥ 40	*	Viral Infection
8	$1485*$	≥ 40	*	Viral Infection
9	$130**$	$3*$	*	Cancer
10	$130**$	$3*$	*	Cancer
11	$130**$	$3*$	*	Cancer
12	$130**$	$3*$	*	Cancer

Table 2.4: $4-anonymous$ inpatient microdata [MKGV07].

$k-anonymity$ alleviates re-identification attacks but is still vulnerable to the so-called homogeneity attacks and background knowledge attacks [MKGV07]. One example for the background knowledge attack is that, suppose we know one patient who is about thirty years old, has visited the hospital and his record is in Tab. 2.4, then we could conclude that he has cancer. Afterward, $l-Diversity$ [MKGV07] was proposed to overcome the shortcoming of $k-anonymity$ by preventing the homogeneity of sensitive attributes in the equivalent classes. Specifically, $l-Diversity$ requires that there exist at least l different values for the sensitive attribute in every equivalent class as Tab. 2.5 shows. However, the definition of $l-Diversity$ is proved to suffer from other attacks [LLV07]. Then, in 2007, Li et al. [LLV07] introduced the concept of $t-closeness$ as an enhancement of $l-diversity$. $t-closeness$ that requires the distance (e.g., Kullback-Leibler distance [KL51] or Earth Mover's distance [RTG00]) between the distribution of the sensitive attributes in each equivalent class differs from the distribution of the sensitive attributes in the whole table to be less than the given threshold t. However, $t-closeness$ was later showed to significantly affect the quantity of valuable information the released data contains by Li et al. [LLV09].

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305$*$	≤ 40	*	Heart Disease
4	1305$*$	≤ 40	*	Viral Infection
9	1305$*$	≤ 40	*	Cancer
10	1305$*$	≤ 40	*	Cancer
5	1485$*$	> 40	*	Cancer
6	1485$*$	> 40	*	Heart Disease
7	1485$*$	> 40	*	Viral Infection
8	1485$*$	> 40	*	Viral Infection
2	1306$*$	≤ 40	*	Heart Disease
3	1306$*$	≤ 40	*	Viral Infection
11	1306$*$	≤ 40	*	Cancer
12	1306$*$	≤ 40	*	Cancer

Table 2.5: $3-diverse$ inpatient microdata [MKGV07].

Instead of releasing anonymized data, a more promising method is to limit the data analyst's access by deploying a curator who manages all the individual's data in a database. The curator answers the data analysts' queries, protects each individual's privacy, and ensures that the database can provide statistically useful information. However, protecting privacy in such a system is nontrivial. For instance, the curator must prohibit queries targeting a specific individual, such as "Does Bob suffers from heart disease?". In addition, a single query that seems not to target individuals may still leak sensitive information when several such queries are combined. Instead of releasing the actual query result, releasing approximate statistics could prevent the above attack. However, Dinur et al. [DN03] showed that the adversary could reconstruct the entire database when sufficient queries were allowed and the approximate statistics error was bound to a certain level. Therefore, there are fundamental limits between what privacy protection can achieve and what useful statistical information the queries can provide. Finally, the problem turns into finding a theory that can interpret the relation between preserving privacy and providing valuable statistical information. Differential privacy [Dwo06] is a robust definition that can support quantitative analysis of how much useful statistical information should be released while preserving a desired level of privacy.

2.3.2 Differential Privacy Formalization

Randomized Response

In this part, we introduce differential privacy and start with a very early differentially private algorithm, the Randomized Response [DN03].

We adapt an example from [Kam20] to illustrate the basic idea of Randomized Response. Suppose a psychologist wishes to study the psychological impact of cheating on high school students. The psychologist first needs to find out the number of students who have cheated. Undoubtedly, most students would not admit honestly if they had cheated in exams. More precisely, suppose there are n students, and each student has a sensitive information bit $X_i \in \{0, 1\}$, where 0 denotes $nevercheated$ and 1 denotes $havecheated$. Every student want to keep their sensitive information X_i secret, but they need to answer whether they have cheated. Then, each student sends the psychologist an answer Y_i which may be equal to X_i or a random bit. Finally, the psychologist collects all the answers and tries to get an accurate estimation of the fraction of cheating students $CheatFraction = \frac{1}{n}\sum_{i=1}^{n} X_i$.
The strategy of students can be expressed with following formulas:

$$Y_i = \begin{cases} X_i & \text{with probability } p \\ 1 - X_i & \text{with probability } 1 - p \end{cases} \tag{2.15}$$

Where p is the probability that student i honestly answers the question.

Suppose all students take the same strategy to answer the question either honestly ($p = 1$) or dishonestly ($p = 0$). Then, the psychologist could infer their sensitive information bit exactly since he knows if they are all lying or not. To protect the sensitive information bit X_i, the students have to take another strategy by setting $p = \frac{1}{2}$, i.e., each student either answers honestly or lies but with equal probability. In this way, the answer Y_i does not depend on X_i any more and the psychologist could not infer anything about X_i through Y_i. Therefore, $\frac{1}{n}\sum_{i=1}^{n} Y_i$ is distributed as a binomial random variable $bino \sim \frac{1}{n}Binomial\left(n, \frac{1}{2}\right)$ and completely independent of $CheatFraction$.

So far, we have explored two strategies: the first strategy ($p = 0$, 1) leads to a completely accurate answer but is not privacy preserving, and the second strategy ($p = \frac{1}{2}$) is perfectly private but not accurate. A more practical strategy is to find the trade-off between two strategies by setting $p = \frac{1}{2} + \gamma$, where $\gamma \in \left[0, \frac{1}{2}\right]$. $\gamma = \frac{1}{2}$ corresponds to the first strategy where all students are honest, and $\gamma = 0$ corresponds to the second strategy where everyone answers randomly. Therefore, the students can increase their privacy protection level by setting $\gamma \to 0$ or provide more accurate result by setting $\gamma \to \frac{1}{2}$. To measure the accuracy of this strategy, we start with the Y_i's expectation $\mathbb{E}[Y_i] = 2\gamma X_i + \frac{1}{2} - \gamma$, thus $\mathbb{E}\left[\frac{1}{2\gamma}\left(Y_i - \frac{1}{2} + \gamma\right)\right] = X_i$. For sample mean $\tilde{C} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2\gamma}\left(Y_i - \frac{1}{2} + \gamma\right)\right]$, we have $\mathbb{E}\left[\tilde{C}\right] = CheatFraction$. The variance of $\tilde{C}$ is

$$Var\left[\tilde{C}\right] = Var\left[\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2\gamma}\left(Y_i - \frac{1}{2} + \gamma\right)\right]\right] = \frac{1}{4\gamma^2 n^2}\sum_{i=1}^{n} Var\left[Y_i\right]. \tag{2.16}$$

Since Y_i is a Bernoulli random variable, we have $Var\left[Y_i\right] = p\left(1 - p\right) \le \frac{1}{4}$ and

$$\begin{aligned}\frac{1}{4\gamma^2 n^2}\sum_{i=1}^{n} Var\left[Y_i\right] &= \frac{1}{4\gamma^2 n} Var\left[Y_i\right] \\ &\leq \frac{1}{16\gamma^2 n}.\end{aligned} \tag{2.17}$$

With Chebyshev's inequality: For any real random variable Z with expectation μ and variance σ^2,

$$\Pr(|X-\mu| \geq t) \leq \frac{\sigma^2}{t^2}, \tag{2.18}$$

For $t = O\left(\frac{1}{\gamma\sqrt{n}}\right)$, we have

$$\begin{aligned}\Pr\left(\left|\tilde{C} - CheatFraction\right| \geq O\left(\frac{1}{\gamma\sqrt{n}}\right)\right) \leq O(1) \\ \Pr\left(\left|\tilde{C} - CheatFraction\right| \leq O\left(\frac{1}{\gamma\sqrt{n}}\right)\right) \geq O(1),\end{aligned} \tag{2.19}$$

and $\left|\tilde{C} - CheatFraction\right| \leq O\left(\frac{1}{\gamma\sqrt{n}}\right)$ with high probability. The error term $\left|\tilde{C} - CheatFraction\right| \to 0$ as $n \to \infty$ with high probability. The conclusion is that the error increases as the privacy protection level increases $\gamma \to 0$. To maintain accuracy, more data $n \to \infty$ is needed. To further quantify the privacy and accuracy, we need to define differential privacy.

For the formalization of differential privacy, we adapted the terms and definitions from [DR+14].

Terms and Definitions

Database. The database D consists of n entries of data from a data universe $\mathcal{X}$ and is denoted by $D \in \mathcal{X}^n$. In the following, we will use the words database and dataset interchangeably.

Take Tab. 2.6 as an example. The database contains the names and exam scores of five students. The database is represented by its rows. The data universe $\mathcal{X}$ contains all the combinations of student names and exam scores.

Name	Score
Alice	80
Bob	100
Charlie	95
David	88
Evy	70

Table 2.6: Database example.

Data Curator. A data curator is trusted to manage and organize the database, and its primary goal is to ensure that the database can be reused reliably. In terms of differential privacy, the data curator is responsible for preserving the privacy of individuals represented in the database. The curator can also be replaced by cryptographic protocols such as secure multiparty protocols [GMW87].

Adversary. The adversary plays the role of a data analyst interested in learning sensitive information about the individuals in the database. In differential privacy, any legitimate data analyst of the database can be an adversary.

Definition 2.3.10 (Privacy Mechanism [DR$^+$14])**.** *A privacy mechanism $M : \mathcal{X}^n \times \mathcal{Q} \to \mathcal{Y}$ is an algorithm that takes databases, queries as input and produces an output string, where $\mathcal{Q}$ is the query space and $\mathcal{Y}$ is the output space of M.*

The query process is as Fig. 2.1 shows, a data curator manages the database and provides an interface that deploys a privacy mechanism for a data analyst/adversary to query. After the querying, the data analyst/adversary receives an output.

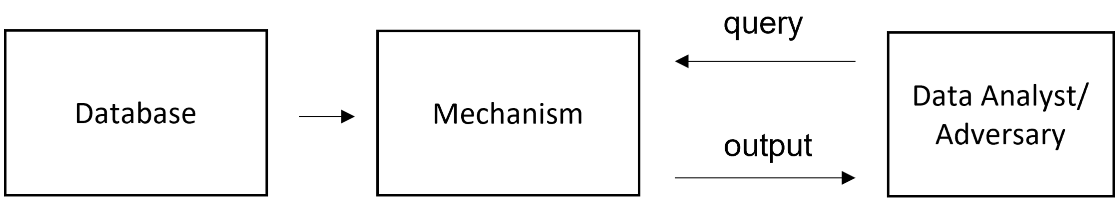

Figure 2.1: DP setting.

Definition 2.3.11 (Neighboring Databases [DR$^+$14])**.** *Two databases D_0, $D_1 \in \mathcal{X}^n$ are called neighboring if they differ in exact one entry. This is expressed as $D_0 \sim D_1$.*

Definition 2.3.12 (Differential Privacy [DR$^+$14])**.** *A privacy mechanism $M : \mathcal{X}^n \times \mathcal{Q} \to \mathcal{Y}$ is (ε, δ)-differential privacy if for any two neighboring databases D_1, $D_1 \in \mathcal{X}^n$, and for all $T \subseteq \mathcal{Y}$, we have $\Pr[M(D_0) \in T] \leq e^{\varepsilon} \cdot \Pr[M(D_1) \in T] + \delta$, where the randomness is over the choices made by M.*

Roughly, the differential privacy implies that the distribution of M's output for all neighboring databases is similar. M is called ε-DP (or pure DP) when $\delta = 0$, and (ε, δ)-DP (or approximate DP) when $\delta \neq 0$.

Definition 2.3.13 (L_1 norm)**.** *The L_1 norm of a vector $\vec{X} = (x_1, x_2, \ldots, x_n)^T$ measures the sum of the magnitudes of the vectors $\vec{X}$ and is denoted by $\left\|\vec{X}\right\|_1 = \sum_{i=1}^{n} |x_i|$.*

Definition 2.3.14 (L_2 norm)**.** *The L_2 norm of a vector $\vec{X} = (x_1, x_2, \ldots, x_n)^T$ measures the shortest distance of $\vec{X}$ to origin point and is denoted by $\left\|\vec{X}\right\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$.*

Definition 2.3.15 (ℓ_t-sensitivity [DR+14])**.** *The ℓ_t-sensitivity of a query $f : \mathcal{X}^n \to \mathbb{R}^k$ is defined as $\Delta_t^{(f)} = \max_{D_0, D_1} \|f(D_0) - f(D_1)\|_t$, where D_0, D_1 are neighboring databases and $t \in \{1, 2\}$.*

Recall the Differential Privacy Definition 2.3.12 attempts to *blur* the contribution of any individual in the database using the notion of neighboring databases. Therefore, the sensitivity is a natural quantity when considering differential privacy since it calculates the upper bound of how much f can change when modifying a single entry.

Motivating Example of Differential Privacy

The previous example about randomized response § 2.3.2 indicates that we need DP to solve the trade-off problem between learning useful statistics and preserving the individuals' privacy. In other words, the psychologist wants to find the fraction of students who have cheated in the exam while guaranteeing that no students suffer from privacy leakage by participating in the questionnaire. To illustrate how DP solves such problems, we adapt the example from [Zum15]. Consider a game as Prot. 2.1 shows,

- A challenger implements a function M that can calculate useful statistical information. An adversary proposes two data sets D_0 and D_1 that differ by only one entry and a test set Q.
- Given $M(D_0)$, $M(D_1)$ in a random order, the adversary aims to differentiate D_0 and D_1. If the adversary succeeds, privacy is violated.
- The challenger's goal is to choose M such that $M(D_0)$ and $M(D_1)$ *look similar* to prevent them from being distinguished by the adversary.
- M is called ε-differentially private iff: $\left|\frac{\Pr[M(D_0) \in Q]}{\Pr[M(D_1) \in Q]}\right| \leq e^{\varepsilon}$.

Challenger C		**Adversary** A
input: M		**input**: D_0, D_1, Q
	$\xleftarrow{D_0,\, D_1}$	
$b \leftarrow\!\$\ \{0,1\}$ $M(D_b)$, $M(D_{1-b})$		
	$\xrightarrow{M(D_b),\, M(D_{1-b})}$	
		$b' = 0$, **if** $M(D_{1-b}) \in Q$ $b' = 1$, otherwise **if** $b == b'$, A wins.

Protocol 2.1: A motivating example of differential privacy.

Suppose the adversary A has chosen two data sets:

- $D_0 = \{0,0,0,\dots,0\}$ (100 zeros)
- $D_1 = \{1,0,0,\dots,0\}$ (0 one and 99 zeroes).

The testing set Q is an interval $[T,1]$, where the threshold T is chosen by the adversary. The threshold T is chosen such that when the adversary has $T < M(D) < 1$, he knows M has input $D = D_1$ (or $D = D_0$, when $0 < M(S) \leq T$) .

The Deterministic Case. Suppose the challenger wants to calculate the mean value of data sets and chooses $M(D) = mean(D)$. Since $M(D_0) = 0$ and $M(D_1) = 0.01$, the adversary can set $Q = [0.005, 1]$ and identify precisely the database D used in $M(D)$ every time they play the game. In Fig. 2.2, the blue line represents the distribution of $M(D_0)$, whereas the orange line represents the distribution of $M(D_1)$ (plotted upside down for clarity). The vertical dotted line represents the threshold $T = 0.005$ which separates D_0 and D_1 perfectly.

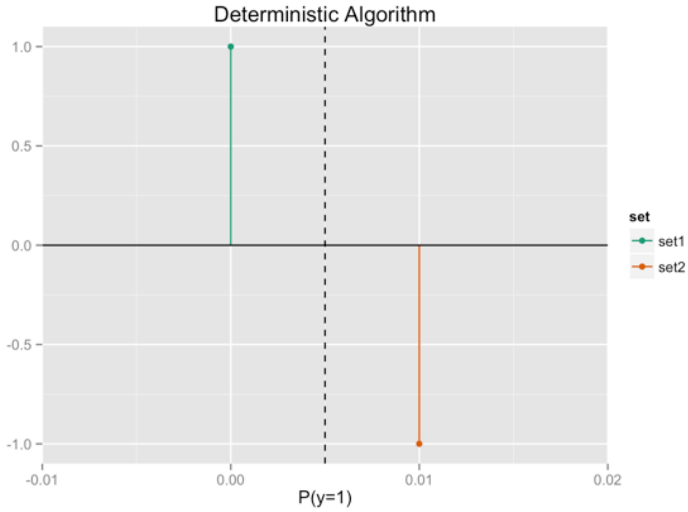

Figure 2.2: Deterministic algorithm.

The Indeterministic Case. The challenger needs to take some measures to *blur* the difference between $M(D_0)$ and $M(D_1)$. Suppose the challenger decides to add Laplace noise $lap \sim Laplace(b = 0.05)$ to the result of $M(D)$ as Fig. 2.3 shows. The shaded blue region is the chance that $M(D_0)$ would return a value greater than the adversary's threshold T. In other words, the probability that the adversary would mistake D_0 for D_1. In contrast, the shaded orange area is the probability that the adversary identify D as D_1. The challenger can decrease the adversary's probability of winning by adding more noise as Fig. 2.4 shows, where the shaded blue and orange areas are almost of the same size. Comparing $M(D)$ with T is no longer reliable to distinguish D_0 and D_1. In fact, we have $\varepsilon = \log\left(\frac{\text{blue area}}{\text{orange area}}\right)$, where ε expresses the degree of differential privacy and a smaller ε guarantees a stronger privacy protection. Although the challenger can add more noise to decrease the adversary's success probability, the mean estimation accuracy is also decreased.

TODO: need reproduce following figures

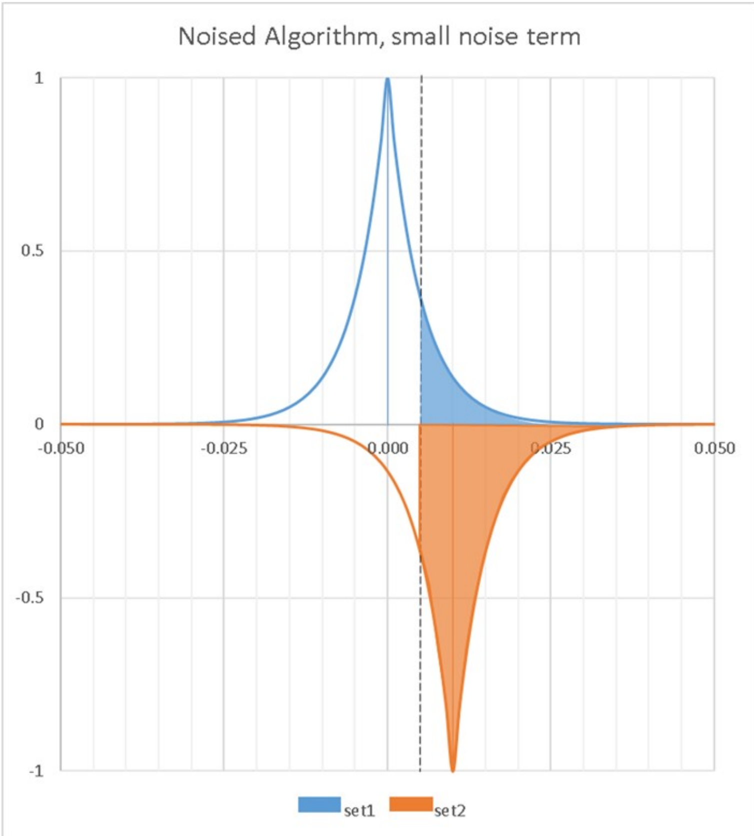

Figure 2.3: Indeterministic algorithm with small noise ($b = 0.005$).

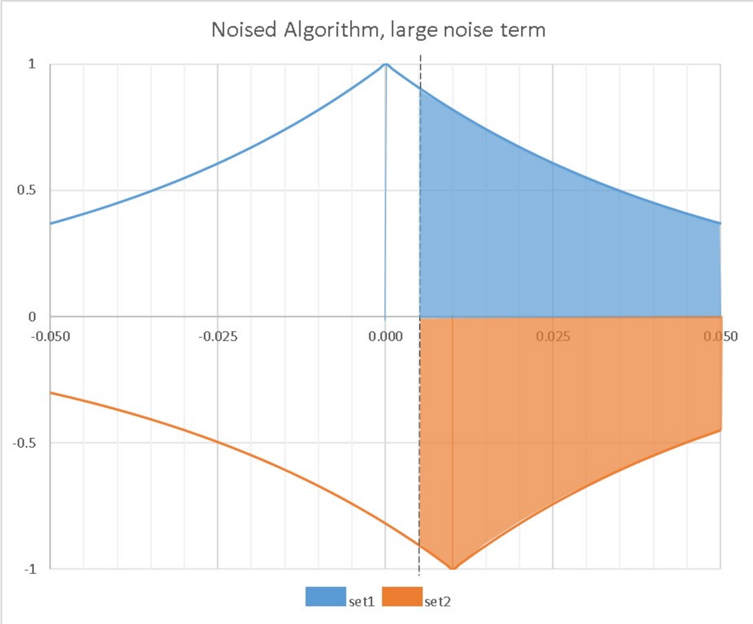

Figure 2.4: Indeterministic algorithm with large noise ($b = 0.05$).

Properties of Differential Privacy

Post-Processing

Theorem 2. *Let $M : \mathcal{X}^n \to \mathcal{Y}$ be (ε, δ)-DP mechanism, and let $F : \mathcal{Y} \to \mathcal{Z}$ be an arbitrary randomized mapping. Then $F \circ M$ is $(\varepsilon, \delta = 0)$-DP [DR+14].*

The Post-Porcessing properties implies the fact that once a database is privatized, it is still differentially private after further processing.

Group Privacy

Theorem 3. *Let $M : \mathcal{X}^n \to \mathcal{Y}$ be (ε, δ)-DP mechanism. For all $T \subseteq \mathcal{Y}$, we have $\Pr[M(D_0) + T] \leq e^{k\varepsilon} \cdot \Pr[M(D_1) \in T] + \delta$, where D_0, $D_1 \in \mathcal{X}^n$ are two databases that differ in exactly k entries [DR+14].*

Differential privacy can also be defined when considering two databases with more than one entry differences. The larger privacy decay rate $e^{k\varepsilon}$ implies a smaller ε, where *more* noise are necessary to guarantee the same level of privacy.

Basic Composition

Theorem 4. *Suppose $M = (M_1 \dots M_k)$ is a sequence of $(\varepsilon_i, \delta_i)$-differentially private mechanisms, where M_i is chosen sequentially and adaptively. Then M is $\left(\sum_{i=1}^{n} \varepsilon_i, \sum_{i=1}^{n} \delta_i\right)$-DP [DR+14].*

Basic Composition provides a way to evaluate the overall privacy when k privacy mechanisms are applied on the same dataset and the results are released.

Differentially Private Mechanisms

Differential privacy is a formal framework to quantify the trade-off between privacy and the accuracy of query results. In this part, we introduce two common differentially private mechanisms.

ε-Differential Privacy

Definition 2.3.16 (Laplace Mechanism [DR+14])**.** *Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The Laplace mechanism is defined as $M_{Lap}(X) = f(X) + (Y_1, \dots, Y_k)$, where the Y_i are independent Laplace random variables drawn from a Laplace distribution $Lap(Y_i \mid b) = \frac{1}{2b} e^{\left(-\frac{|Y_i|}{b}\right)}$ with $b = \frac{\Delta_1^{(f)}}{\varepsilon}$.*

Theorem 5. *Laplace Mechanism preservers ε-DP [DR+14].*

(ε, δ)-Differential Privacy ε-DP has strong privacy requirement which leads to adding too much noise and affecting the accuracy of the queries. We introduce an relaxation of ε-DP, (ε, δ)-DP.

Definition 2.3.17 (Gaussian Mechanism [DR+14])**.** *Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The Gaussian mechanism is defined as $M(X) = f(X) + (Y_1, \dots, Y_k)$, where the Y_i are independent Gaussian random variables drawn from distribution $\mathcal{N}\left(Y_i | \mu, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{Y_i - \mu}{\sigma}\right)^2}$ with $\mu = 0$, $\sigma^2 = 2\ln\left(\frac{1.25}{\delta} \cdot \left(\frac{\Delta_2^{(f)})^2}{\varepsilon^2}\right)\right)$.*

Gaussian mechanism is proved to satisfy (ε, δ)-DP [DR+14].

Discussion about Differential Privacy

Local and Central Differential Privacy Differential privacy is a definition that can be realized in many ways. Two common modes of DP are centralized differential privacy [DR+14] and local differential privacy [DN03].

In centralized DP, all data is stored centrally and managed by a trusted curator before the differentially private mechanism is applied. As Fig. 2.5 shows, the raw data from clients is first collected in a centralized database, then, the curator applies the privacy mechanism and answers the queries $f(x)$ with $f'(x)$. The local DP mode is, as Fig. 2.6 shows, where the clients first apply a privacy mechanism on their data, and send the perturbed data to the curator. An advantage of local DP mode is that no trusted central curator is needed since the data is perturbed independently before sending to the curator. However, the disadvantage is that the collected data contains redundant noise and may decrease the utility.

TODO: reproduce following figures

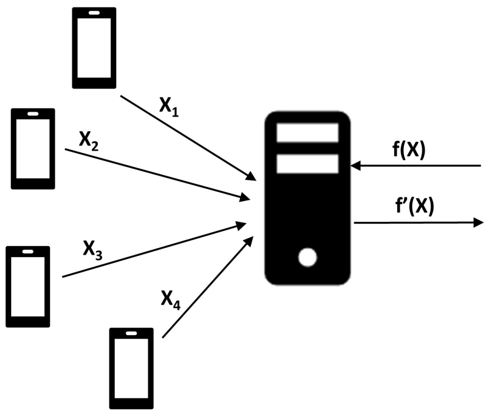

Figure 2.5: Centralized DP mode.

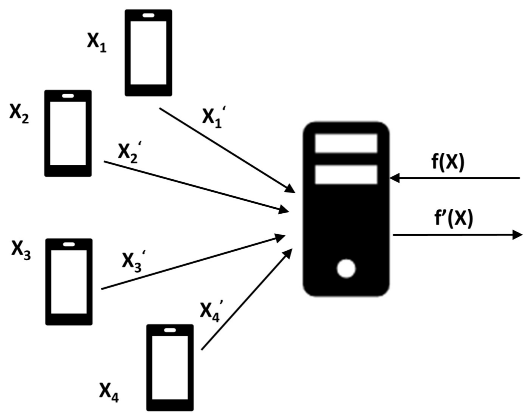

Figure 2.6: Local DP mode.

Advantages of Differential Privacy From the example § 2.3.2, we found that DP can still protect privacy even if the adversary has the knowledge of the database. Generally speaking, DP ensures privacy protection by making no assumption about the adversary's auxiliary information (even when the adversary is the data provider) or computational strategy (regarding the complexity of modern cryptography) [Vad17]. In addition, DP provides a quantitive theory about safely releasing data and maintaining certain level of accuracy.

Challenges of Differential Privacy DP provides a method to guarantee and quantify individual privacy at the theoretical level. However, it faces a series of practical challenges.

Sensitivity Calculation. For certain types of data, the sensitivity is not difficult to calculate. Take a database with human ages as an example, the ages should be bounded between 0 and 150 (longest human lifespan is 122 years and 164 days according to [Whi97]). However, the data with an unbounded value range brings great challenges. A common solution is to roughly estimate the value range and limit the data within that range. For example, if the value range estimation is $[a, b]$, then all values smaller than a are replaced by a and all values bigger than b are replaced by b. Finally, the sensitivity of query function that outputs ages is $b - a$. If value range $[a, b]$ is chosen too wide, the utility is potentially destroyed because of the large magnitude of the noise. If the value range $[a, b]$ is chosen too narrow, the utility is also potentially decreased because too many values beyond $[a, b]$ are truncated.

Implementation of DP Mechanisms. The theory of DP is built upon the real number arithmetic. The practical implementation of differentially private mechanisms relies on floating-point or fixed-point arithmetic only provides an approximation of the mathematical abstractions. Mironov [Mir12] showed that the irregularities of floating-point implementations and porouse distribution of the Laplace mechanism with textbook sampling algorithm lead to the breach of differential privacy. Further, Gazeau et al. [GMP16] proved that any differentially private mechanism that perturbs data by adding noise with a finite precision can resulting secret disclosure regardless of the actual implementation.

3 Related Work

3.1 Distributed Differential Privacy (DDP)

The fundamental idea of differential privacy [Dwo06] is to perturb the query on a database such that influence of each individual record in the database is bounded. The original definition of DP assumes the existence of a centralized and trusted server that manages the database and process the queries. Subsequent works [DKM+06] extends DP to a distributed setting, where the central server is replaced by several mutually distrustful and potentially malicious computation parties. To the best of our knowledge, Dwork et al. [DKM+06] was the first to consider deploying malicious SMPC to aggregate and perturb data with the shares of random noise. Generally, the works that attemepes to realizied Distributed Differential Privacy (DDP) can be categorized into two groups: central DP model and local DP model. In the central DP model, a trusted, central server responsible for computing the aggregate statistics and perturbation, is simulated by several semi-honest (or malicious) computation parties with SMPC. It is typically assumed that the majority of computation parties are not colluding. However, SMPC incurs high computation and communication overhead that reduces the efficiency and scalability. In the local DP model, the server is not trusted anymore. The users applies randomness to privatized their data before sending to the server. Therefore, the accuracy of the aggregate statistics is limited as the randomization is applied multiple times. Both models face series of open challenges and we discuss the details of them below.

3.1.1 Local DP Model

The existing solution to local DP model [RN10; SCR+11; ÁC11; CSS12; BRB+17; SCRS17; TBA+19] relied on homomorphic encryption, SMPC and the infinite divisibility of certain probability distribution (e.g., Laplace distribution [KKP01] and Gaussian distribution). Specifically, each user perturbs their data independently and encrypts it with homomorphic encryption scheme such that the server can aggregate the encrypted data and only reveal the noisy aggregated result. Instead of using homomorphic encryption, other works [BP20; GKM+21; BBGN20] had the users to mask the local perturbed data with additional noise and send the masked data to the server. After the aggregation of the server, the additional noise is canceled out and the noisy result that satisfied DP is revealed. However, the existing works of local DP model faces two major challenges. The first challenge is that the collusion users can substract their noise term from the revealed result and reduce the DP-guarantee. Therefore, in order to achieve the required DP guarantee, each user has to add a larger amount of noise, that leads a reduced utility of the aggregated result. The second challenge is that the users have to pay significant amount of computation effort, that makes the local DP model less practical for devices that has limited computation power.

3.1.2 Central DP Model

For the central DP model, prior works [DKM+06; EKM+14; WHWX16; JWEG18; KVH+21; YSMN21; EIKN21] porposed variety of methods to satisfy DP by generating distributed noise in SMPC. Dwork et al. [DKM+06] supported noise from two distributions : Gaussian distribution (approximated with binomial distribution), discrete Laplace distribution (approximated with Possion distribution). To satisfy (ε, δ)-DP, it needs to process $n \geq 64 \log_2(2/\delta)/\varepsilon^2$ (e.g., $\varepsilon = 0.01, \delta = 0.0001 \Rightarrow n \approx 2^{23}$) uniform random bits in SMPC to generate binomial noise, that would leads to high SMPC overhead. For discrete Laplace noise, the protocol requires securely evaluating a circuit in SMPC to generating biased bits. However, the evaluation of the circuit fails with non-zero probability and requires multiple iterations to make the failure probability negligible. Eigner et al. [EKM+14] proposed an architecture called PrivaDA, that combined DP and SMPC, and generated Laplace noise and discrete Laplace noise in SMPC protocols. However, the generated Laplace noise suffers from the floating-point attack [Mir12]. The discrete Laplace noise is susceptible to similar floating-point attack because its generation procedure is similar to the Laplace noise. Wu et al. [WHWX16] described methods for generating Bernoulli noise, Laplace noise, and Gaussian noise in SMPC setting. The Laplace noise is generated basd on the central limit theory [AL06, Example 10.3.2], i.e., the aggregation of n of Bernoulli random variable $Bern(0.5)$ approximates a normal random variable $\mathcal{N}\left(0, \frac{1}{4}\right)$ because of $(\sqrt{n}\left(\frac{\sum_{i=1}^{n} Bern(0.5)}{n} - \mu\right) \approx \mathcal{N}\left(0, \frac{1}{4}\right))$. However, the central limit theory holds when $n \to \infty$, and there is no discussion about whether the choice of n would affect DP guarantee in the work of Wu et al. [WHWX16].

Jayaraman et al. [JWEG18], Knott et al. [KVH+21] and Yuan et al. [YSMN21] presented distributed learning approaches that combine SMPC and DP by generating distributed Laplace noise and Gaussian noise with SMPC protocols. The protocols for Laplace noise are similar to the work of Eigner et al. [EKM+14], and the protocol for Gaussian noise are all based on the Box-Muller sampling algorithm [BOX58]. However, Jin et al. [JMRO22] had demonetrated an floating-point attack against the Box-Muller method.

Eriguchi et al. [EIKN21] provided SMPC-based protocols to generate two types of noise: Finite Discrete Laplace (FDL) noise and binomial noise. In contrast to discrete Laplace distribution that can sample arbitrarily large integers with low probability, FDL can only generate integers in a given range $[-N, N]$. The protocol for binomial noise deploys pseudorandom secret-sharing [CDI05] for generating shares of uniform random variables non-interactively and use the binomial mechanism [ASY+18] to satisfy DP guarantee. However, the binomial mechanism only satisfy computational differential privacy [MPRV09], that is an relaxation of the standard differential privacy definintion [DR+14] and only secure against computational bounded adversary.

Our work provides an alternative solution to the central DP model by *securely* generating distributed noise that are not affected by the attacks [Mir12; JMRO22].

3.2 Arithmetic Operations in SMPC

Generally, most SMPC protocols that support arithemtic operations in SMPC are based on binary circuit approach or Linear Secret Sharing Scheme (LSSS). In binary circuit based approach, the arithmetic operations is represented as a Boolean circuit and evaluated with Yao's Garbled circuit protocol [Yao86] (BMR [BMR90] for multi-party setting) or Boolean GMW protocol [GMW87]. By contrast, in the LSSS-based approach [CCD88; BGW88], the parties divide their secret values into shares over a field $\mathbb{F}_q$ (or a ring $\mathbb{Z}_{2^\ell}$) and send it to each of the parties. Next, we explain the SMPC protocols for arithemtic operations of these two types in details.

3.2.1 LSSS-Based SMPC

In order to guarantee the high efficiency of the SMPC protocols, much prior works [CS10; Lie12; HLOW16; AS19; LFH$^+$20] deployed fixed-point arithmetic to represent real number operations. Catrina and Saxena [CS10] built a series of fixed-point operations (e.g., additioin, subtraction, multiplication and division) by representing a fixed-point $x = \overline{x} \cdot 2^{-f}$, where $\overline{x}$ is an integer in field $\mathbb{F}_q$ and f is the length of oth fraction bits. The works [Lie12; HLOW16; AS19; LFH$^+$20] proposed protocols for fixed-point operations such as exponential, square root, natural logarithm, and trigonometric functions with polynomial approximation [Har78] or Goldschmidt approximation [Mar04].

Another line of works [ABZS12; KW14; KW15; RBS$^+$22] focused on floating-point operations.

Aliasgari et al. [ABZS12] used a quadruple (v, p, z, s) to represent the floating-point number $u = (1-2s)\cdot(1-z)\cdot v\cdot 2^p$, where v (mantissa), p (exponent), z (zero bit), and s (sign bit) $\in \mathbb{F}_q$. Aliasgari et al. [ABZS12] also provided SMPC protocols for operations such as addition, subtraction, multiplication, divisibility, square root, logarithm, and expoentiation. The subsequent works [KW14; KW15; RBS$^+$22] applied similar representation form of the floating-point numbers.

Truex et al. [TBA$^+$19] proposed a hybrid method combing fixed-point and floating-point arithmetic, i.e., representing the mantissa of a floating-point number as fixed-point number, and used LSSS-based fixed-point arithmetic when the mantissa is involved in the floating-point arithmetic. However, the fixed-point arithmetic is prone to overflow or underflow that requires additioinal SMPC protocols to fixed the computation result that decrease the overall protocol performance.

Kamm and Willemson et al. [KW15] provided SMPC protocols for square root, natural exponentiation, and error function approximated with Taylor series expansion or Chebyshev polynomials.

Rathee et al. [RBS+22] built a precise and efficient 32-bit floating-point operation library (SecFloat) for secure two-party computation. One highlight is the use of the mixed-bitwidth computation technique, i.e., use low bitwidth to represent numbers as much as possible. The conversion operations between different bitwidth are performed with specialized zero-extension and truncation Two-Party Computation (2PC) protocols. The second highlight is the use of low-degree polynomials to improve accuracy and efficiency. One common method to compute function like $log_2 x$ is polynomial approximation, where high-degree polynomials yields more accurate result but incur more computation effort. Rathee et al. [RBS+22] replaced the high-degree polynomials with low-degree piecewise polynomials without decrease accuracy. In specifically, for input $x \in (a, b)$, they approximated $log_2 x$ using different low-degree polynomials for k subintervals $((a, a_1), (a_1, a_2), \ldots, (a_{k-1}, b))$. To determine the active interval of x, they deployed the Lookup Table (LUT) protocol [DKS+17] to compute the correct polynomial coefficients. To explore if efficiency of SecFloat [RBS+22] still preserves in multi-party setting, we implement its building blocks (conversion operations between low-bitwidth and high-bitwidth, multi-party lookup table protocol [KOR+17], MSNZB [RRG+21]) in the MITION [BDST22] framework. After benchmarking (cf. **??**), we found the benefit brought by mixed-bitwidth becomes negligible when extending it to a multi-party setting. The reason is as follows:

1. The conversion operations between low-bitwidth and high-bitwidth in SecFloat rely on 2PC comparison protocol based on OT, and it can not be directly extended to multi-party setting.
2. In the two-party setting, when we take the value of two ℓ-bit arithmetic shares $\langle a \rangle_0^A$, $\langle a \rangle_1^A$ as plaintext value and compute the addition result, we need an $\ell + 1$-bit integer $a = \langle a \rangle_0^A + \langle a \rangle_1^A$ to hold the addition result without overflow. The most significant bit of a is used during the conversion operation. For $N \geq 3$ parties, the addition result of N ℓ-bit arithmetic value needs a $\lceil \log_2 N \rceil + l$-bit integer to hold. The $\lceil \log_2 N \rceil$ most significant bits are used for the conversion. As the number of parties grows, the complexity of conversion operations also increases.

Protocol $MSNZB\left(\langle x \rangle^A\right)$ [ABZS12; RRG+21] computes the most significant non-zero bit index of the input $\langle x \rangle^A \in \mathbb{Z}_{2^\ell}$ in arithmetic share. We implement two types of $MSNZB$ [ABZS12; RRG+21] protocols in MOTION [BDST22] and evaluate the performance.

$MSNZB$ [ABZS12] used operations such as bit-decomposition, Boolean sharing to arithmetic sharing conversion and arithemtic sharing addition. By contrast, $MSNZB$ [RRG+21] first decomposed the input $\langle x \rangle^A \in \mathbb{Z}_{2^\ell}$ into $\frac{\ell}{8}$ low-bitwidth arithmetic shares $\langle x_1 \rangle^A$, $\langle x_2 \rangle^A$, $\ldots$, $\left\langle x_{\ell/8} \right\rangle^A \in \mathbb{Z}_{2^8}$. Then, each low-bitwidth arithmetic share is used as a input to the lookup table protocol to compute $MSNZB\left(\langle x_1 \rangle^A\right)$, $\ldots$, $MSNZB\left(\left\langle x_{\ell/8} \right\rangle^A\right)$. Finally, $MSNZB\left(\langle x_1 \rangle^A\right)$, $\ldots$, $MSNZB\left(\left\langle x_{\ell/8} \right\rangle^A\right)$ were combined together to compute $MSNZB\left(\langle x \rangle^A\right)$. We can see in

Tab. 3.1 that the use of mixed-bitwidth and lookup table does not bring efficiency improvement in the LAN (10Gbit/s Bandwidth, 1ms RTT) and WAN (100Mbit/s Bandwidth, 100ms RTT) networks. Details of benchmark environment are in **??**.

Table 3.1: Online run-times in milliseconds (ms) for operation MSNZB for the GMW (A). We take the average over 10 protocol runs in the LAN and WAN environments.

	LAN		WAN	
Operation	N=3	N=5	N=3	N=5
$MSNZB$ [ABZS12]	18.41	80.12	886.06	1 036.23
$MSNZB$ [RBS+22]	359.76	567.14	5 436.82	6 355.90

3.2.2 Binary Circuit Based SMPC

The binary circuits are typically composed of XOR, NOT and AND gates, where the AND gates cause the major cost. Therefore, the binary circuits should be optimized based on the cost metric of SMPC protocols. A common method to generate low AND-depth and low AND-size binary circuit is to use CBMC-GC [BHWK16] circuit compiler that derives circuit design with C program. Pullonen and Siim [PS15] used CBMC-GC [BHWK16] circuit compiler to generate size-optimized circuit for IEEE 754 floating-point operations with C library [Hau18; Fel21]. Archer et al. [AAS21] deployed use CBMC-GC [BHWK16] to generate depth-optimized circuit for IEEE 754 floating-point operations. In this works, we also use CBMC-GC [AAS21] to generate depth-optimized circuit and size-optimized circuits for arithemtic operations with different C librarys [Hau18; Fel21] or available optimized circuits from work [DSZ15].

List of Figures

List of Tables

List of Abbreviations

SMPC Secure Multi-Party Computation

DP Differential Privacy

DDP Distributed Differential Privacy

PPML Privacy-Preserving Machine Learning

ML Machine Learning

BMR Beaver, Micali and Rogaway

GMW Goldreich, Micali and Wigderson

OT Oblivious Transfer

C-OT Correlated Oblivious Transfer

R-OT Random Oblivious Transfer

MTs Multiplication Triples

FDL Finite Discrete Laplace

LSSS Linear Secret Sharing Scheme

2PC Two-Party Computation

LUT Lookup Table

SIMD Single Instruction Multiple Data

Bibliography

[AAS21] D. W. Archer, S. Atapoor, N. P. Smart. **"The Cost of IEEE Arithmetic in Secure Computation"**. In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2021, pp. 431–452.

[ABZS12] M. Aliasgari, M. Blanton, Y. Zhang, A. Steele. **"Secure computation on floating point numbers"**. In: *Cryptology ePrint Archive* (2012).

[ÁC11] G. Ács, C. Castelluccia. **"I have a dream!(differentially private smart metering)"**. In: *International Workshop on Information Hiding*. Springer. 2011, pp. 118–132.

[AL06] K. B. Athreya, S. N. Lahiri. **"Measure theory and probability theory"**. Vol. 19. Springer, 2006.

[ALSZ17] G. Asharov, Y. Lindell, T. Schneider, M. Zohner. **"More efficient oblivious transfer extensions"**. In: *Journal of Cryptology* 30.3 (2017), pp. 805–858.

[AS19] A. Aly, N. P. Smart. **"Benchmarking privacy preserving scientific operations"**. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2019, pp. 509–529.

[ASY+18] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, B. McMahan. **"cpSGD: Communication-efficient and differentially-private distributed SGD"**. In: *Advances in Neural Information Processing Systems* 31 (2018).

[BBGN20] B. Balle, J. Bell, A. Gascón, K. Nissim. **"Private summation in the multi-message shuffle model"**. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 657–676.

[BDK+18] N. Büscher, D. Demmler, S. Katzenbeisser, D. Kretzmer, T. Schneider. **"HyCC: Compilation of hybrid protocols for practical secure computation"**. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 847–861.

[BDST22] L. Braun, D. Demmler, T. Schneider, O. Tkachenko. **"MOTION–A Framework for Mixed-Protocol Multi-Party Computation"**. In: *ACM Transactions on Privacy and Security* 25.2 (2022), pp. 1–35.

[Bea91] D. Beaver. **"Efficient multiparty protocols using circuit randomization"**. In: *Annual International Cryptology Conference*. Springer. 1991, pp. 420–432.

[BGW88] M. BEN-OR, S. GOLDWASSER, A. WIGDERSON. **"Completeness theorems for non-cryptographic fault-tolerant distributed computation"**. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988, pp. 1–10.

[BHKR13] M. BELLARE, V. T. HOANG, S. KEELVEEDHI, P. ROGAWAY. **"Efficient garbling from a fixed-key blockcipher"**. In: *2013 IEEE Symposium on Security and Privacy*. IEEE. 2013, pp. 478–492.

[BHWK16] N. BÜSCHER, A. HOLZER, A. WEBER, S. KATZENBEISSER. **"Compiling low depth circuits for practical secure computation"**. In: *European Symposium on Research in Computer Security*. Springer. 2016, pp. 80–98.

[BMR90] D. BEAVER, S. MICALI, P. ROGAWAY. **"The round complexity of secure protocols"**. In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990, pp. 503–513.

[BOX58] G. BOX. **"A Note on the Generation of Random Normal Deviates"**. In: *Annals of Mathematical Statistics* 29 (1958), pp. 610–611.

[BP20] D. BYRD, A. POLYCHRONIADOU. **"Differentially private secure multi-party computation for federated learning in financial applications"**. In: *Proceedings of the First ACM International Conference on AI in Finance*. 2020, pp. 1–9.

[BRB+17] V. BINDSCHAEDLER, S. RANE, A. E. BRITO, V. RAO, E. UZUN. **"Achieving differential privacy in secure multiparty data aggregation protocols on star networks"**. In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 2017, pp. 115–125.

[CCD88] D. CHAUM, C. CRÉPEAU, I. DAMGARD. **"Multiparty unconditionally secure protocols"**. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988, pp. 11–19.

[CDI05] R. CRAMER, I. DAMGÅRD, Y. ISHAI. **"Share conversion, pseudorandom secret-sharing and applications to secure computation"**. In: *Theory of Cryptography Conference*. Springer. 2005, pp. 342–362.

[CKS20] C. L. CANONNE, G. KAMATH, T. STEINKE. **"The discrete gaussian for differential privacy"**. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15676–15688.

[CS10] O. CATRINA, A. SAXENA. **"Secure computation with fixed-point numbers"**. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2010, pp. 35–50.

[CSS12] T.-H. H. CHAN, E. SHI, D. SONG. **"Privacy-preserving stream aggregation with fault tolerance"**. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2012, pp. 200–214.

[DKM+06] C. DWORK, K. KENTHAPADI, F. MCSHERRY, I. MIRONOV, M. NAOR. **"Our data, ourselves: Privacy via distributed noise generation"**. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2006, pp. 486–503.

[DKS+17] G. DESSOUKY, F. KOUSHANFAR, A.-R. SADEGHI, T. SCHNEIDER, S. ZEITOUNI, M. ZOHNER. **"Pushing the Communication Barrier in Secure Computation using Lookup Tables"**. In: *NDSS* (2017).

[DMNS06] C. DWORK, F. MCSHERRY, K. NISSIM, A. SMITH. **"Calibrating noise to sensitivity in private data analysis"**. In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284.

[DN03] I. DINUR, K. NISSIM. **"Revealing information while preserving privacy"**. In: *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2003, pp. 202–210.

[DR+14] C. DWORK, A. ROTH. **"The algorithmic foundations of differential privacy"**. In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[DSZ15] D. DEMMLER, T. SCHNEIDER, M. ZOHNER. **"ABY-A framework for efficient mixed-protocol secure two-party computation."** In: *NDSS*. 2015.

[Dwo06] C. DWORK. **"Differential privacy"**. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2006, pp. 1–12.

[EIKN21] R. ERIGUCHI, A. ICHIKAWA, N. KUNIHIRO, K. NUIDA. **"Efficient Noise Generation to Achieve Differential Privacy with Applications to Secure Multiparty Computation"**. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2021, pp. 271–290.

[EKM+14] F. EIGNER, A. KATE, M. MAFFEI, F. PAMPALONI, I. PRYVALOV. **"Differentially private data aggregation with optimal utility"**. In: *Proceedings of the 30th Annual Computer Security Applications Conference*. 2014, pp. 316–325.

[EKR17] D. EVANS, V. KOLESNIKOV, M. ROSULEK. **"A pragmatic introduction to secure multi-party computation"**. In: *Foundations and Trends® in Privacy and Security* 2.2-3 (2017).

[Fel21] R. FELKER. **"musl libc"**. 2021.

[GKM+21] B. GHAZI, R. KUMAR, P. MANURANGSI, R. PAGH, A. SINHA. **"Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message"**. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 3692–3701.

[GMP16] I. GAZEAU, D. MILLER, C. PALAMIDESSI. **"Preserving differential privacy under finite-precision semantics"**. In: *Theoretical Computer Science* 655 (2016), pp. 92–108.

[GMW87] O. Goldreich, S. Micali, A. Wigderson. **“How to play ANY mental game”**. In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 218–229.

[GRS12] A. Ghosh, T. Roughgarden, M. Sundararajan. **“Universally utility-maximizing privacy mechanisms”**. In: *SIAM Journal on Computing* 41.6 (2012), pp. 1673–1693.

[Har78] J. F. Hart. **“Computer approximations”**. Krieger Publishing Co., Inc., 1978.

[Hau18] J. Hauser. **“Berkeley SoftFloat”**. In: (2018).

[HLOW16] B. Hemenway, S. Lu, R. Ostrovsky, W. Welser Iv. **“High-precision secure computation of satellite collision probabilities”**. In: *International Conference on Security and Cryptography for Networks*. Springer. 2016, pp. 169–187.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, E. Petrank. **“Extending oblivious transfers efficiently”**. In: *Annual International Cryptology Conference*. Springer. 2003, pp. 145–161.

[IR89] R. Impagliazzo, S. Rudich. **“Limits on the provable consequences of one-way permutations”**. In: *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. 1989, pp. 44–61.

[Isr06] O. (I. O. S. G. (Israel)). **“Foundations of Cryptography: Volume 1, Basic Tools”**. Cambridge University Press, 2006.

[JLL+19] K. Järvinen, H. Leppäkoski, E.-S. Lohan, P. Richter, T. Schneider, O. Tkachenko, Z. Yang. **“PILOT: Practical privacy-preserving indoor localization using outsourcing”**. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2019, pp. 448–463.

[JMRO22] J. Jin, E. McMurtry, B. Rubinstein, O. Ohrimenko. **“Are We There Yet? Timing and Floating-Point Attacks on Differential Privacy Systems”**. In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society. 2022, pp. 1547–1547.

[JWEG18] B. Jayaraman, L. Wang, D. Evans, Q. Gu. **“Distributed learning without distress: Privacy-preserving empirical risk minimization”**. In: *Advances in Neural Information Processing Systems* 31 (2018).

[Kam20] G. Kamath. **“Lecture 3 - Intro to Differential Privacy”**. 2020.

[KKP01] S. Kotz, T. Kozubowski, K. Podgórski. **“The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance”**. 183. Springer Science & Business Media, 2001.

[KL51] S. Kullback, R. A. Leibler. **“On information and sufficiency”**. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[Knu14] D. E. Knuth. **“Art of computer programming, volume 2: Seminumerical algorithms”**. Addison-Wesley Professional, 2014.

[KOR+17] M. Keller, E. Orsini, D. Rotaru, P. Scholl, E. Soria-Vazquez, S. Vivek. **"Faster secure multi-party computation of AES and DES using lookup tables"**. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2017, pp. 229–249.

[KR11] S. Kamara, M. Raykova. **"Secure outsourced computation in a multi-tenant cloud"**. In: *IBM Workshop on Cryptography and Security in Clouds*. 2011, pp. 15–16.

[KS08] V. Kolesnikov, T. Schneider. **"Improved garbled circuit: Free XOR gates and applications"**. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2008, pp. 486–498.

[KVH+21] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, L. v. d. Maaten. **"Crypten: Secure multi-party computation meets machine learning"**. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4961–4973.

[KW14] T. Krips, J. Willemson. **"Hybrid model of fixed and floating point numbers in secure multiparty computations"**. In: *International Conference on Information Security*. Springer. 2014, pp. 179–197.

[KW15] L. Kamm, J. Willemson. **"Secure floating point arithmetic and private satellite collision analysis"**. In: *International Journal of Information Security* 14.6 (2015), pp. 531–548.

[LFH+20] W.-j. Lu, Y. Fang, Z. Huang, C. Hong, C. Chen, H. Qu, Y. Zhou, K. Ren. **"Faster secure multiparty computation of adaptive gradient descent"**. In: *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*. 2020, pp. 47–49.

[Lie12] M. Liedel. **"Secure distributed computation of the square root and applications"**. In: *International Conference on Information Security Practice and Experience*. Springer. 2012, pp. 277–288.

[LLV07] N. Li, T. Li, S. Venkatasubramanian. **"t-closeness: Privacy beyond k-anonymity and l-diversity"**. In: *2007 IEEE 23rd International Conference on Data Engineering*. IEEE. 2007, pp. 106–115.

[LLV09] N. Li, T. Li, S. Venkatasubramanian. **"Closeness: A new privacy measure for data publishing"**. In: *IEEE Transactions on Knowledge and Data Engineering* 22.7 (2009), pp. 943–956.

[LP09] Y. Lindell, B. Pinkas. **"A proof of security of Yao's protocol for two-party computation"**. In: *Journal of cryptology* 22.2 (2009), pp. 161–188.

[Mar04] P. Markstein. **"Software division and square root using Goldschmidt's algorithms"**. In: *Proceedings of the 6th Conference on Real Numbers and Computers (RNC'6)*. Vol. 123. 2004, pp. 146–157.

[Mir12] I. Mironov. **"On significance of the least significant bits for differential privacy"**. In: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, pp. 650–661.

[MKGV07] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam. **"l-diversity: Privacy beyond k-anonymity"**. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), 3–es.

[MPRV09] I. Mironov, O. Pandey, O. Reingold, S. Vadhan. **"Computational differential privacy"**. In: *Annual International Cryptology Conference*. Springer. 2009, pp. 126–142.

[MRT20] P. Mohassel, M. Rosulek, N. Trieu. **"Practical Privacy-Preserving K-means Clustering"**. In: *Proceedings on Privacy Enhancing Technologies* 2020.4 (2020), pp. 414–433.

[NPS99] M. Naor, B. Pinkas, R. Sumner. **"Privacy preserving auctions and mechanism design"**. In: *Proceedings of the 1st ACM Conference on Electronic Commerce*. 1999, pp. 129–139.

[PS15] P. Pullonen, S. Siim. **"Combining secret sharing and garbled circuits for efficient private IEEE 754 floating-point computations"**. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2015, pp. 172–183.

[RBS+22] D. Rathee, A. Bhattacharya, R. Sharma, D. Gupta, N. Chandran, A. Rastogi. **"SecFloat: Accurate Floating-Point meets Secure 2-Party Computation"**. In: *Cryptology ePrint Archive* (2022).

[RN10] V. Rastogi, S. 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*. 2010, pp. 735–746.

[RR21] M. Rosulek, L. Roy. **"Three halves make a whole? beating the half-gates lower bound for garbled circuits"**. In: *Annual International Cryptology Conference*. Springer. 2021, pp. 94–124.

[RRG+21] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, A. Rastogi. **"SiRnn: A math library for secure RNN inference"**. In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 1003–1020.

[RSA78] R. L. Rivest, A. Shamir, L. Adleman. **"A method for obtaining digital signatures and public-key cryptosystems"**. In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[RTG00] Y. Rubner, C. Tomasi, L. J. Guibas. **"The earth mover's distance as a metric for image retrieval"**. In: *International journal of computer vision* 40.2 (2000), pp. 99–121.

[SCR+11] E. Shi, T.-H. Chan, E. Rieffel, R. Chow, D. Song. **"Privacy-Preserving Aggregation of Time-Series Data"**. In: vol. 2. 2011.

[SCRS17] E. Shi, T.-H. H. Chan, E. Rieffel, D. Song. **"Distributed private data analysis: Lower bounds and practical constructions"**. In: *ACM Transactions on Algorithms (TALG)* 13.4 (2017), pp. 1–38.

[SS98] P. SAMARATI, L. SWEENEY. **"Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression"**. In: (1998).

[SSSS17] R. SHOKRI, M. STRONATI, C. SONG, V. SHMATIKOV. **"Membership inference attacks against machine learning models"**. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 3–18.

[Ste87] J. M. STEELE. **"Non-Uniform Random Variate Generation (Luc Devroye)"**. 1987.

[Swe97] L. SWEENEY. **"Weaving technology and policy together to maintain confidentiality"**. In: *The Journal of Law, Medicine & Ethics* 25.2-3 (1997), pp. 98–110.

[TBA+19] S. TRUEX, N. BARACALDO, A. ANWAR, T. STEINKE, H. LUDWIG, R. ZHANG, Y. ZHOU. **"A hybrid approach to privacy-preserving federated learning"**. In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 2019, pp. 1–11.

[Tea20] G. D. P. TEAM. **"Secure Noise Generation"**. 2020.

[Vad17] S. VADHAN. "The complexity of differential privacy". In: *Tutorials on the Foundations of Cryptography*. Springer, 2017, pp. 347–450.

[VV17] P. VOIGT, A. VON DEM BUSSCHE. **"The eu general data protection regulation (gdpr)"**. In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676 (2017), pp. 10–5555.

[Wal74] A. J. WALKER. **"Fast generation of uniformly distributed pseudorandom numbers with floating-point representation"**. In: *Electronics Letters* 10.25 (1974), pp. 533–534.

[Whi97] C. R. WHITNEY. **"Jeanne Calment, World's Elder, Dies at 122"**. 1997.

[WHWX16] G. WU, Y. HE, J. WU, X. XIA. **"Inherit differential privacy in distributed setting: Multiparty randomized function computation"**. In: *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE. 2016, pp. 921–928.

[Yao82] A. C. YAO. **"Protocols for secure computations"**. In: *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE. 1982, pp. 160–164.

[Yao86] A. C.-C. YAO. **"How to generate and exchange secrets"**. In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE. 1986, pp. 162–167.

[YSMN21] S. YUAN, M. SHEN, I. MIRONOV, A. C. NASCIMENTO. **"Practical, label private deep learning training based on secure multiparty computation and differential privacy"**. In: *Cryptology ePrint Archive* (2021).

[ZRE15] S. ZAHUR, M. ROSULEK, D. EVANS. **"Two halves make a whole"**. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 220–250.

[Zum15] N. Zumel. **“A Simpler Explanation of Differential Privacy”**. 2015.

A Appendix

A.1 MPC Protocols

Oblivious Array Access $\Pi^{ObliviousSelection}$ is inspired by the works [JLL+19; MRT20]. $\Pi^{ObliviousSelection}$ uses the inverted binary tree, i.e., the leaves represent input elements, and the root represents the output element. The tree has $\log_2 \ell$-depth for input array of length ℓ and each node hold two shares: $\langle c \rangle^B$ and $\langle \boldsymbol{y} \rangle^B$. Fig. A.1 shows an example of the inverted binary tree. For $i \in [0, \ell - 1]$ and $j \in \left[\log_2 \ell\right]$, the values of node ($\left(\left\langle c_{a,b} \right\rangle, \left\langle \boldsymbol{y_{a,b}} \right\rangle^B\right)$) in the j-th layer are computed with the value of two nodes (with value $\left(\langle c_a \rangle, \langle \boldsymbol{y_a} \rangle^B\right)$ and $\left(\langle c_b \rangle, \langle \boldsymbol{y_b} \rangle^B\right)$) in the $j-1$-th layer as follows:

$$\left(\left\langle c_{a,b} \right\rangle, \left\langle \boldsymbol{y_{a,b}} \right\rangle^B\right) = \begin{cases} \left(\langle c_a \rangle, \langle \boldsymbol{y_a} \rangle^B\right), & \text{if} \langle c_a \rangle == 1 \\ \left(\langle c_b \rangle, \langle \boldsymbol{y_b} \rangle^B\right), & \text{if} \langle c_a \rangle == 0 \wedge \langle c_b \rangle == 1 \\ (0, 0) & \text{if} \langle c_a \rangle == 0 \wedge \langle c_b \rangle == 0, \end{cases} \tag{A.20}$$

which is equivalent to

$$\left\langle c_{a,b} \right\rangle = \langle c_a \rangle \oplus \langle c_b \rangle \oplus (\langle c_a \rangle \wedge \langle c_b \rangle) \tag{A.21}$$

$$\begin{aligned} \left\langle \boldsymbol{y_{a,b}} \right\rangle = \quad & (\langle c_a \rangle \oplus \langle c_b \rangle) \cdot \left(\langle \boldsymbol{y_a} \rangle^{B,UINT} \cdot \langle c_a \rangle \oplus \langle \boldsymbol{y_b} \rangle^{B,UINT} \cdot \langle c_b \rangle\right) \\ & \oplus (\langle c_a \rangle \wedge \langle c_b \rangle) \cdot \langle \boldsymbol{y_a} \rangle^{B,UINT} \end{aligned} \tag{A.22}$$

The nodes is evaluated from the 1th layer until the root.

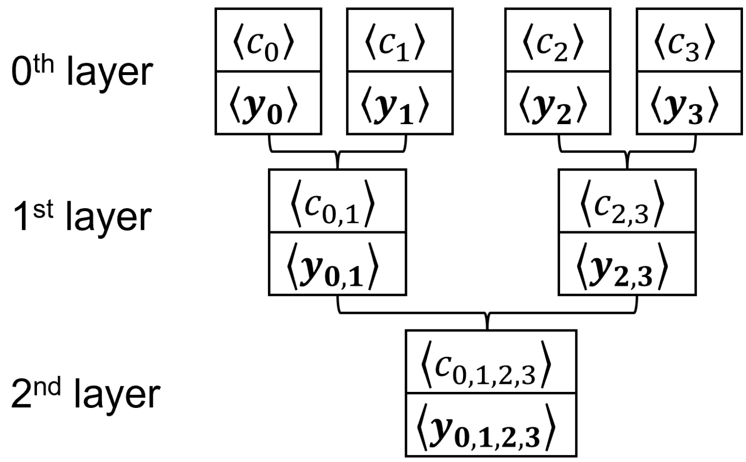

Figure A.1: Example inverted binary tree for $\Pi^{ObliviousSelection}$.