TECHNISCHE
UNIVERSITÄT
DARMSTADT

Master Thesis

Securely Realizing Output Privacy in MPC

Liang Zhao

July 9, 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 9, 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 [Yao86] 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

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 in the multi-party setting [BDST22].

2.2.3 MPC Protocols

We describe the SMPC protocols that are 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)

The arithmetic sharing protocol enables parties to evaluate arithmetic circuits consisting of addition and multiplication gates. In arithmetic sharing, an ℓ-bit value x is shared additively among N parties as $\left(\langle x\rangle_1^A, \ldots, \langle x\rangle_N^A\right) \in \mathbb{Z}_{2^\ell}^N$, 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 computed without parties' interaction. Suppose the parties hold shares $\langle x\rangle_i^A$, $\langle y\rangle_i^A$, and wish to compute $\langle z\rangle = a \cdot \langle x\rangle + \langle y\rangle + b$ with public value a, $b \in \mathbb{Z}_{2^\ell}$. Then, one specific party P_1 computes $\langle z\rangle_1^A = a \cdot \langle x\rangle_1^A + \langle y\rangle_1^A + b$, and the other 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 $c = a \cdot b$. To compute $\langle z\rangle^A = \langle x\rangle^A \cdot \langle y\rangle^A$, the parties first locally 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, each party P_i compute $\langle z\rangle_i^A = \langle c\rangle_i^A + e \cdot \langle x\rangle_i^A + d \cdot \langle y\rangle_i^A - d \cdot e$ locally.

Boolean Sharing with GMW

Boolean GMW protocol [GMW87] uses XOR-based secret sharing and enables multiple parties to evaluate the function represented as a Boolean circuit. 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 GMW can be seen as a special case of arithmetic sharing protocol. 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 analogously as in the 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 settings with BMR [BMR90] protocol. Yao's Garbled Circuit protocol [Yao86] enables two parties, the 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 table $\tilde{g}$ based on the function table of g. For example, gate g is an AND gate and has a function table as Tab. 2.1 shows, the gabler encrypts the keys of wire c and permutes the entries to generate 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 from 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

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

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

to his input. For each of the evaluator's input wire i with corresponding input bit c, the evaluator and the gabler run an 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 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 the garbled circuit $\widetilde{C}$ and the keys of input wires, the evaluator begins to 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. This part presents several prominent optimizations for Yao's Garbled Circuit protocol [Yao86]. Recall that in the evaluation step of Yao's garbled circuit $\widetilde{C}$ protocol, 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 in garbled tables with an additional permutation bit for each wire. Garbled row reduction [NPS99] reduces the number of entries in the garbled 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 protocol [Yao86] 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 25% less communication bits than the two-halves garbling at the cost of more computation.

BMR Protocol. 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 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. Then, each party sends 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 SMPC Framework - MOTION

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

1. Support for SMPC 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 SMPC protocols at the circuit's gate level and asynchronously evaluation, i.e., each gate is separately evaluated once their parent gates become ready.
3. Support for 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.

2.3.1 Probability Distribution and Random Variable Generation

In this section, we introduce essential probability theory and random variable sampling methods as a preparation for differential privacy.

Continuous Probability Distribution

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

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

We use the probability density function to denote the corresponding probability distribution P or a random variable $x \sim P$. For example, $Uni(a, b)$ denotes the continuous uniform distribution with parameters a and b. We sometimes abuse notation and let $Uni(a, b)$ denote a random variable $x \sim Uni(a, b)$.

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

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

The cumulative distribution function of an exponential distribution is defined as:

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

Definition 2.3.3 (Laplace Distribution [DR+14])**.** *The Laplace distribution (centered at 0) 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. It is also called the double exponential distribution because it can be thought of as an exponential distribution assigned with a randomly chosen sign.

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 (also known as the standard normal) distribution with mean μ and standard deviation σ, has the following probability density function:*

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

Discrete Probability Distribution

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

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

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 $x \in \{0,1,2,\ldots,n\}$:*

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

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 $x \in \{0,1,2,\ldots\}$:*

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

Note that the geometric distribution models the number of trials until the first success (each trial has a success probability p). The cumulative distribution function of the geometric distribution is $\Pr(x \leq X \mid p) = 1-(1-p)^{X+1}$.

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:*

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

The discrete Laplace distribution can be generated by reflecting a geometric distribution 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 $x \in \mathbb{Z}$:*

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

Probability Sampling Methods

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

Inverse Sampling from a Bernoulli Distribution. Algorithm 2.1 samples a random variable $x \sim Bern(p)$ based on the comparison result of the generated uniform random variable $u \in (0,1)$ and parameter p.

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

Input: p
Output: $x \sim Bern(p)$
1 : $u \leftarrow\$ (0,1)$
2 : **IF** $u < p$
3 : **RETURN** $x \leftarrow 1$
4 : **ELSE**
5 : **RETURN** $x \leftarrow 0$

Algorithm 2.1: Inverse sampling from a Bernoulli distribution.

Inverse Sampling from a Laplace Distribution. A Laplace random variable $Y \sim Lap(b)$ can be sampled from an exponential distribuion with cumulative distribution function: $F(x \mid b) = 1 - e^{-\frac{x}{b}}$ as follows [Mir12]:

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

Sampling from a Geometric Distribution. Algorithm 2.2 [Wal74; Tea20] generates a geometric random variable $x \sim Geo(0.5)$ by generating an ℓ-bit random string $r \in \{0,1\}^{\ell}$ (i.e., ℓ Bernoulli trials) and counting its leading zeros (i.e., the number of trials before the first success trial). If there is no 1 bit in the ℓ-bit r, the 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: $x \sim Geo(0.5)$
1 : $x \leftarrow 0$
2 : $r \leftarrow\!\!\$\ \{0,1\}^{\ell}$
3 : $x \leftarrow LeadingZeros(r)$
4 : **RETURN** x

Algorithm 2.2: Sampling from a geometric distribution.

Sampling from a Discrete Laplace Distribution. Algorithm 2.3 [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: Sampling from a discrete Laplace distribution.

2.3.2 Traditional Techniques for Privacy Preservation

Before differential privacy [Dwo06; DMNS06] became the leading method for controlling information disclosure, researchers had proposed approaches for privacy preservation. We briefly introduced several techniques with an adapted example from work [LLV07].

Suppose a fictitious hospital has collected massive data from thousands of patients and wants to make the data available to academic researchers. However, the data contains sensitive information of the patients, such as Zip Code, Age, Nationality, and Health Condition. 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 carry 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 of the patients as Tab. 2.3 shows. The remaining 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. However, the adversary can still 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].

k-anonymity. A typical attack is the re-identification attack [Swe97] that combines the released anonymized data with publicly available information to re-identify individuals. One approach against such re-identification attacks is to deploy privacy preservation methods that satisfy the notion of *k-anonymity* [SS98]. 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 data suppression. The former makes the data less informative by mapping the attribute values to a broader value range, and the latter removes specific attribute values. As Tab. 2.4 shows, the values of Age in the first eight records are replaced by the value ranges such as < 30 and ≥ 40 after generalization. The values of 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., Zip Code: $130**$ and 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].

l-Diversity. Although *k-anonymity* alleviates re-identification attacks, it is still vulnerable to the so-called homogeneity attacks and background knowledge attacks [MKGV07]. Suppose we know one patient about thirty years old has visited the hospital, and his record appears in Tab. 2.4, then we could conclude that he has cancer. Afterward, the notion *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 each equivalent class. As Tab. 2.5 shows, an adversary cannot infer if a man (with Zip Code: $1305*$ and Age 31) has cancer or viral infection, even though it is very unlikely for a man who is less than forty years old to have heart disease.

t-closeness. However, the definition of *l-Diversity* was later proved to suffer from attacks by Li et al. [LLV07], who introduced the concept of *t-closeness* as an enhancement of *l-Diversity*. *t-closeness* requires that the distance between the distribution of the sensitive attributes in each equivalent class and the distribution of the sensitive attributes in the whole table to be less than a given threshold. The distance can be measured by the Kullback-Leibler distance [KL51] or Earth Mover's distance [RTG00]. However, Li et al. [LLV09] showed that *t-closeness* significantly affects the quantity of the valuable information in the released data.

	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, it is non-trivial to build such a privacy-preserving system. 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 may protect privacy to some extend. However, Dinur and Nissim [DN03] demonstrated that the adversary could still 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. Differential privacy [Dwo06; DMNS06] is a robust definition that supports quantitative analysis of the amount of useful statistical information to be released while preserving a desired level of privacy.

2.3.3 Differential Privacy Formalization

Randomized Response

In this part, we introduce differential privacy with a very early differentially private algorithm, the Randomized Response [War65] using an example adapted from [Kam20]. Suppose a psychologist wishes to study the psychological impact of cheating on high school students. The psychologist needs to find out the number of students who have cheated. Undoubtedly, most students would not admit if they had cheated in exams. Suppose there are n students,

and each student has a sensitive information bit $X_i \in \{0,1\}$, where 0 denotes *never cheated* and 1 denotes *cheated*. Every student want to keep their sensitive information X_i secret, but they still need to answer whether they have cheated. Then, each student sends the psychologist an answer Y_i that equals to X_i or a random bit. Finally, the psychologist collects all the answers and estimates the fraction of cheating students $\overline{C} = \frac{1}{n}\sum_{i=1}^{n} X_i$.
The students can apply following strategy:

$$Y_i = \begin{cases} X_i & \textit{with probability } p \\ 1 - X_i & \textit{with probability } 1-p \end{cases} \tag{2.13}$$

where p is the probability that a student honestly answers the question.

Suppose all students use the same strategy and answer 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. Hence, students have to take another strategy by setting $p = \frac{1}{2}$, i.e., each student answers honestly or lies with the same probability. Note that the answer Y_i does not depend on X_i any more and the psychologist could not infer anything about X_i. Further, $\frac{1}{n}\sum_{i=1}^{n} Y_i$ is a binomial random variable $\frac{1}{n}\sum_{i=1}^{n} Y_i \sim \frac{1}{n} \cdot Binomial\left(n, \frac{1}{2}\right)$ and completely independent of $\overline{C}$.

So far, we have explored two strategies: the first strategy ($p = \{0,1\}$) leads to an accurate answer but is not privacy-preserving, while the second strategy ($p = \frac{1}{2}$) is perfectly private but delivers useless information. 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. The students can increase their privacy protection level by setting $\gamma \to 0$ or provide more accurate result by increasing $\gamma \to \frac{1}{2}$.

To measure the accuracy of the strategy, we start with the Y_i's expectation $\mathbb{E}[Y_i] = 2\gamma X_i + \frac{1}{2} - \gamma$, then we obtain $\mathbb{E}\left[\frac{1}{2\gamma}\left(Y_i - \frac{1}{2} + \gamma\right)\right] = X_i$. Next, we compute the sample mean $\tilde{C} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2\gamma}\left(Y_i - \frac{1}{2} + \gamma\right)\right]$, where $\mathbb{E}\left[\tilde{C}\right] = \overline{C}$. The variance of $\tilde{C}$ is computed as follows:

$$Var\left[\tilde{C}\right] = Var\left[\frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2\gamma}\left(Y_i - \frac{1}{2} + \gamma\right)\right]\right] = \frac{1}{4\gamma^2 n^2}\sum_{i=1}^{n} Var[Y_i]. \tag{2.14}$$

As $Y_i \sim Bern(p)$, we have $Var[Y_i] = p(1-p) \leq \frac{1}{4}$ and

$$\begin{aligned} \frac{1}{4\gamma^2 n^2}\sum_{i=1}^{n} Var[Y_i] &= \frac{1}{4\gamma^2 n} Var[Y_i] \\ &\leq \frac{1}{16\gamma^2 n}. \end{aligned} \tag{2.15}$$

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

$$\Pr(|X-\mu| \geq t) \leq \frac{\sigma^2}{t^2}, \tag{2.16}$$

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

$$\begin{aligned} \Pr\left(\left|\tilde{C}-\overline{C}\right| \geq O\left(\frac{1}{\gamma\sqrt{n}}\right)\right) &\leq O(1), \\ \Pr\left(\left|\tilde{C}-\overline{C}\right| \leq O\left(\frac{1}{\gamma\sqrt{n}}\right)\right) &\geq O(1), \end{aligned} \tag{2.17}$$

and $\left|\tilde{C}-\overline{C}\right| \leq O\left(\frac{1}{\gamma\sqrt{n}}\right)$ with high probability. Note that the error $\left|\tilde{C}-\overline{C}\right| \to 0$ as $n \to \infty$ with high probability. Therefore, to reduce the error, we need to either decrease the privacy protection level $\gamma \to 0.5$ or increase the number of students n.

Terms and Definitions

We adapted the terms and definitions from [DR+14] to formalize the definition of differential privacy.

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. The database in Tab. 2.6 contains five students' names and exam scores. The database is represented by its rows, and 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 of five students.

Data Curator. A data curator is trusted to manage and organize the database. Its primary goal is to ensure that the database can provide useful information and preserve privacy after multiply queries. The curator can be simulated with cryptographic protocols such as SMPC [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 (Mechanism [DR$^+$14]). *A 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 mechanism for the data analyst/adversary to query. After the querying, the data analyst/adversary receives an output.

Figure 2.1: Query process of a database.

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

Definition 2.3.12 (Differential Privacy [DR$^+$14]). *A mechanism $M : \mathcal{X}^n \times \mathcal{Q} \to \mathcal{Y}$ is said to satisfy (ε, δ)-differential privacy if for any two neighboring databases D_1, $D_1 \in \mathcal{X}^n$, and for all $T \subseteq \mathcal{Y}$,*

$$\Pr[M(D_0) \in T] \leq e^{\varepsilon} \cdot \Pr[M(D_1) \in T] + \delta,$$

where the randomness is over the choices made by M.

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

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 function $f : \mathcal{X}^n \to \mathbb{R}^k$ is defined as $\Delta_t^{(f)} = \max_{D_0, D_1} \|f(D_0) - f(D_1)\|_t$ for $t \in \{1, 2\}$, where D_0, D_1 are neighboring databases.*

Generally, the sensitivity calculates the upper bound of how much a query function f can change when modifying a single entry using the notion of neighboring databases.

Motivating Example of Differential Privacy

The previous example about randomized response § 2.3.3 indicates that we can use DP to solve the trade-off problem between learning useful statistics and preserving the individuals' privacy. To illustrate how DP solves such problems, we adapt an example [1] shown in Prot. 2.1:

- An adversary proposes two datasets D_0 and D_1 that differ by exactly one entry and a test set Q. A challenger implements a mechanism M that can compute useful statistical information with databases D_0 and D_1.
- The challenger outputs $M(D_0)$, $M(D_1)$ to the adversary in a random order. The adversary aims to differentiate D_0 and D_1. The challenger's goal is to build a mechanism M to prevent $M(D_0)$ and $M(D_1)$ from being distinguished by the adversary.
- Mechanism M is called ε-DP 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
	$\xleftarrow{D_0, D_1}$	
$b \leftarrow\$ \{0,1\}$		
$M(D_b), M(D_{1-b})$		
	$\xrightarrow{M(D_b), M(D_{1-b})}$	
		$b' \leftarrow 0$, **if** $M(D_{1-b}) \in Q$
		$b' \leftarrow 1$, **otherwise**
		if $b == b'$, A **wins**.

Protocol 2.1: A example of differentially privace mechanism.

Suppose the adversary A has chosen the following databases:

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

The test set Q is an interval $[T, 1]$ with a threshold T. After choosing the threshold T, the adversary output $b' \leftarrow 0$ when $T < M(D_{1-b}) < 1$, or $b' \leftarrow 1$ when $0 < M(D_{1-b}) \leq T$. The

[1] https://win-vector.com/2015/10/02/a-simpler-explanation-of-differential-privacy/

adversary's goal is to find a *good* threshold T that helps him output a b' that equals b with high probability and win the game.

The Deterministic Case. Suppose mechanism M compute the mean value of the given databases D_0 and D_1, where $M(D_0) = 0$ and $M(D_1) = 0.01$. The adversary can set the test set $Q = [0.005, 1]$ win every game. In Fig. 2.2, the blue line represents the value of $M(D_0)$, whereas the orange line represents the value of $M(D_1)$ (plotted upside down for clarity). The vertical dotted line is the threshold $T = 0.005$ that separates D_0 and D_1 correctly.

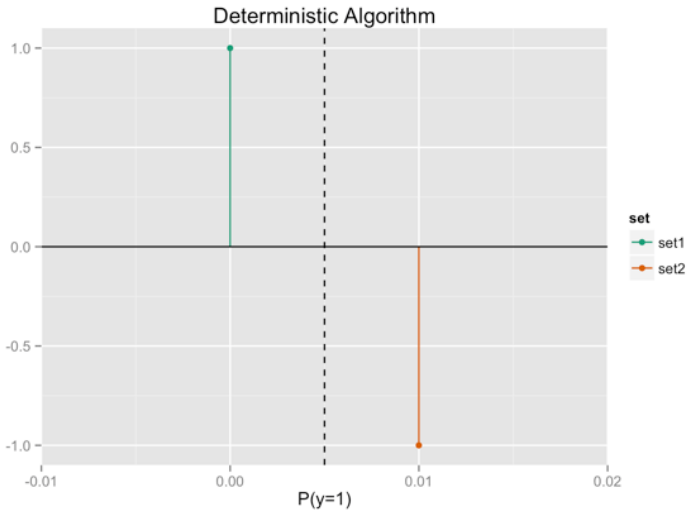

Figure 2.2: Deterministic algorithm.

The Indeterministic Case. The challenger takes some measures to *blur* the outputs of $M(D_0)$ and $M(D_1)$ and make them hard to differentiate. Suppose the challenger decides to add the 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 probability that $M(D_0)$ returns a value greater than the threshold T, and the adversary mistakes D_0 for D_1. In contrast, the shaded orange area is the probability that the adversary identify D_1 successfully. The challenger can decrease the adversary's success probability by adding stronger noise as Fig. 2.4 shows, where the shaded blue and orange areas are almost of the same size. In fact, we have $\varepsilon = \left|\ln\left(\frac{blue\ area}{orange\ area}\right)\right|$, where ε describes the degree of differential privacy. A smaller ε indicates a stronger privacy protection. Note that the challenger can add more noise to decrease the adversary's success probability, but the accuracy of the mean estimation decreases.

TODO: need reproduce following figures

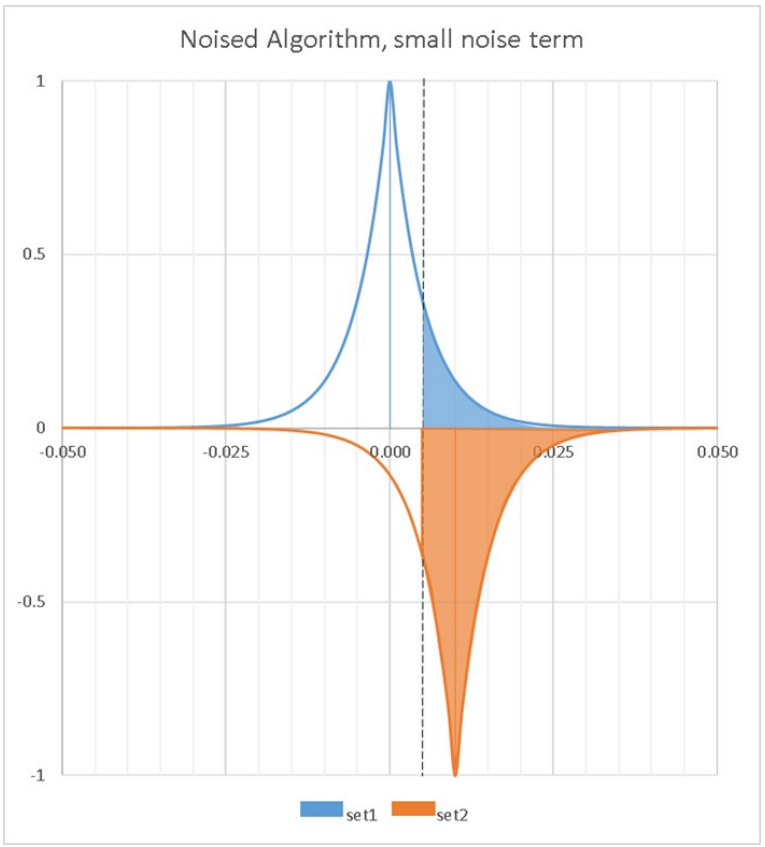

Figure 2.3: Indeterministic algorithm with small noise ($b = 0.005$).

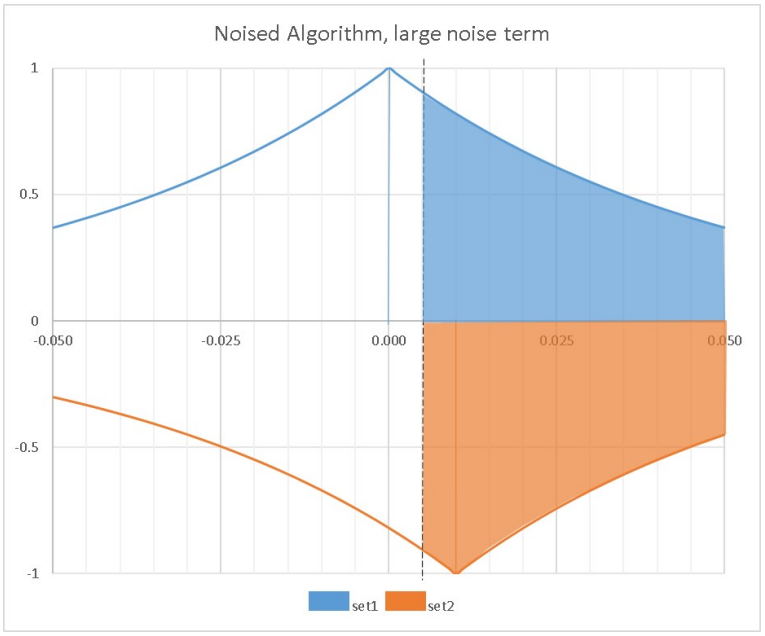

Figure 2.4: Indeterministic algorithm with large noise ($b = 0.05$).

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.

Definition 2.3.16 (Laplace Mechanism [DR$^+$14]). *Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The Laplace mechanism is defined as $M_{Lap}(X) = f(X) + (Y_1, \ldots, Y_k)$, 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 2. *The Laplace Mechanism satisfies ε-DP [DR$^+$14].*

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

Theorem 3. *The Gaussian mechanism satisfies (ε, δ)-DP [DR$^+$14].*

Properties of Differential Privacy

We introduce three fundamental properties of DP [Dwo06; DMNS06] based on work [DR+14].

Proposition 1 (**Post-Processing**). *Let $M : \mathcal{X}^n \to \mathcal{Y}$ be a (ε, δ)-DP mechanism, and let $F : \mathcal{Y} \to \mathcal{Z}$ be an arbitrary randomized mapping. Then $F \circ M$ is (ε, δ)-DP.*

The post-processing property implies that once a database is privatized, it is also differentially private after further processing.

Proposition 2 (**Group Privacy**). *Let $M : \mathcal{X}^n \to \mathcal{Y}$ be a (ε, δ)-DP mechanism. For all $T \subseteq \mathcal{Y}$,*

$$\Pr[M(D_0) + T] \leq e^{k\varepsilon} \cdot \Pr[M(D_1) \in T] + \delta,$$

where D_0, $D_1 \in \mathcal{X}^n$ are databases that differ in exactly k entries.

The group privacy property indicates that DP can also be extended to the case when two databases have more than one entry difference. However, as k increases, the privacy decay rate $e^{k\varepsilon}$ also increases, which implies a weaker level of privacy protection.

Proposition 3 (**Basic Composition**). *Suppose $M = (M_1 \dots M_k)$ is a sequence of $(\varepsilon_i, \delta_i)$-DP 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.*

The basic composition property provides a way to evaluate the overall privacy guarantee of the released result when k DP mechanisms are applied to the same dataset.

Challenges of Differential Privacy

Differential privacy provides a method to guarantee and quantify individual privacy at the theoretical level. However, it faces a series of challenges in practical application scenarios.

Sensitivity Calculation. As discussed in [**subsubsec:DPMechanisms**], we first need to compute or estimate the sensitivity of the query function before applying mechanism. Take a database with human ages as an example, the ages should be bounded in the interval $[0, 150]$ (the 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 a *reasonable* value range and limit the data within that range. If the value range is chosen too wide (i.e., a larger sensitivity estimation), the mechanism applies a too strong noise to perturb the data, and the utility of the result might be destroyed. Nevertheless, a narrow value range estimation may also decrease the utility as too much data beyond the value range is discarded.

Implementation of DP Mechanisms. Generally, the security analysis of differentially private mechanisms is based on two implicit assumptions: (i) Computations are performed on real numbers and require machines to have infinite precision, (ii) The noise is sampled from a probability distribution that is very close to the theoretically correct probability distribution. However, the practical implementation of differentially private mechanisms is typically based on floating-point or fixed-point arithmetic that only provides finite order of accuracy. Mironov [Mir12] demonstrated that the Laplace random noise sampled with textbook algorithm under floating-point implementation could lead to violation of differential privacy. Jin et al. [JMRO22] presented a series of floating-point attacks against the Gaussian noise samplers, and timing attacks against the discrete distribution samplers. 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.

List of Figures

List of Tables

List of Protocols

List of Abbreviations

SMPC Secure Multi-Party Computation

DP 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

SIMD Single Instruction Multiple Data

Bibliography

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

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

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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).

[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.

[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.

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

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[War65] S. L. WARNER. **"Randomized response: A survey technique for eliminating evasive answer bias"**. In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69.

[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:

$$\left(\left\langle c_{a,b} \right\rangle, \left\langle \boldsymbol{y_{a,b}} \right\rangle^B\right) = \begin{cases} \left(\langle c_a \rangle, \langle \boldsymbol{y_a} \rangle^B\right), & \text{if } \langle c_a \rangle == 1 \\ \left(\langle c_b \rangle, \langle \boldsymbol{y_b} \rangle^B\right), & \text{if } \langle c_a \rangle == 0 \wedge \langle c_b \rangle == 1 \\ (0,0) & \text{if } \langle c_a \rangle == 0 \wedge \langle c_b \rangle == 0, \end{cases} \tag{A.18}$$

which is equivalent to

$$\left\langle c_{a,b} \right\rangle = \langle c_a \rangle \oplus \langle c_b \rangle \oplus (\langle c_a \rangle \wedge \langle c_b \rangle) \tag{A.19}$$

$$\begin{aligned} \left\langle \boldsymbol{y_{a,b}} \right\rangle = \quad & (\langle c_a \rangle \oplus \langle c_b \rangle) \cdot \left(\langle \boldsymbol{y_a} \rangle^{B,UINT} \cdot \langle c_a \rangle \oplus \langle \boldsymbol{y_b} \rangle^{B,UINT} \cdot \langle c_b \rangle\right) \\ & \oplus (\langle c_a \rangle \wedge \langle c_b \rangle) \cdot \langle \boldsymbol{y_a} \rangle^{B,UINT} \end{aligned} \tag{A.20}$$

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

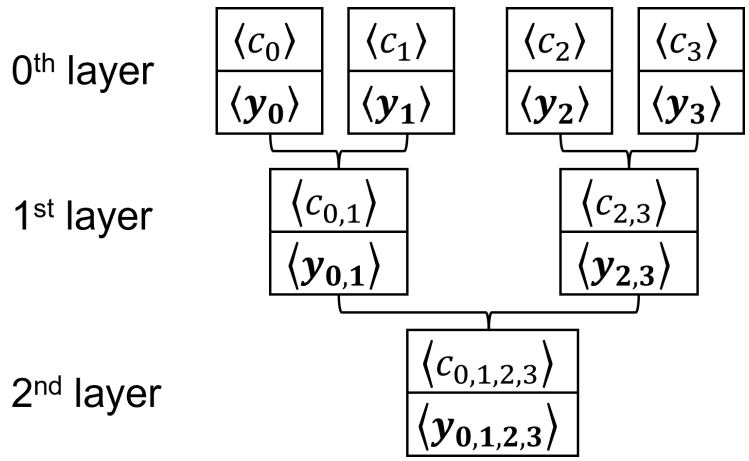

Figure A.1: Example inverted binary tree for $\Pi^{ObliviousSelection}$.