VDDP: Verifiable Distributed Differential Privacy under the Client-Server-Verifier Setup

Haochen Sun
University of Waterloo

Xi He
University of Waterloo

Abstract

Despite differential privacy (DP) often being considered the de facto standard for data privacy, its realization is vulnerable to unfaithful execution of its mechanisms by servers, especially in distributed settings. Specifically, servers may sample noise from incorrect distributions or generate correlated noise while appearing to follow established protocols. This work analyzes these malicious behaviors in a general differential privacy framework within a distributed client-server-verifier setup. To address these adversarial problems, we propose a novel definition called *Verifiable Distributed Differential Privacy* (VDDP) by incorporating additional verification mechanisms. We also explore the relationship between zero-knowledge proofs (ZKP) and DP, demonstrating that while ZKPs are sufficient for achieving DP under verifiability requirements, they are not necessary. Furthermore, we develop two novel and efficient mechanisms that satisfy VDDP: (1) the Verifiable Distributed Discrete Laplacian Mechanism (VDDLM), which offers up to a 4×10^5x improvement in proof generation efficiency with only 0.1-0.2x error compared to the previous state-of-the-art verifiable differentially private mechanism; (2) an improved solution to Verifiable Randomized Response (VRR) under local DP, a special case of VDDP, achieving up a reduction of up to 5000x in communication costs and the verifier's overhead.

1 Introduction

In today's data-driven world, vast amounts of information are collected and analyzed to drive decision-making across various sectors, including healthcare, finance, and marketing. However, privacy and efficiency concerns often make it infeasible to collect and analyze such vast amounts of data in a centralized manner. Therefore, the concept of *distributed data analysis* [19, 30, 39] has emerged as a viable solution. In this approach, data is shared secretly and processed across multiple nodes, which can enhance privacy by reducing sensitive information exposure and improve efficiency by leveraging the computational resources of several nodes.

With differential privacy (DP) [37, 38] established as the de facto standard for data privacy, the distributed data analysis mechanisms have also benefited from an additional layer of privacy guarantee provided by DP, thus giving rise to *distributed differential privacy* [28, 35, 69, 92]. This approach includes various implementations such as multi-party computations (MPCs) of differential privacy schemes [17, 18, 34, 44, 51, 53] and differentially private federated learning (FL) schemes [43, 66, 78, 93]. These methods ensure that individual data points remain private while still allowing for accurate and efficient data analysis across distributed systems.

While enjoying the privacy guarantee provided by DP, most distributed DP mechanisms assume that the servers performing the randomized computations involved in the protocols are semi-honest. That is, all servers strictly adhere to the prescribed protocol and only passively attempt to learn additional information about the database. However, malicious servers can deviate from the DP mechanism during execution, potentially in a collusive manner, which can further compromise the privacy guarantees. This type of deviation is elusive, since attackers can attribute any anomalous results to this randomness. Under the assumption that a certain portion of servers are honest, malicious-secure MPC protocols have been designed to resist such attacks [4, 24, 36, 91]. However, to convince an external party (e.g., a data analyst) receiving the result of its authenticity, the assumption has to be lifted, and the honesty of the servers still needs to be verified. This type of verification is particularly difficult because the privacy requirements forbid a thorough examination of the execution.

Several previous studies have explored the verifiable execution of DP mechanisms, including the randomized response [61] and continuous (floating point) Gaussian mechanism [84]. Moreover, the verifiable distributed binomial mechanism (VDBM) [14] among multiple clients and servers has also been developed. These studies utilize zero-knowledge proofs (ZKP) [11, 42, 49, 50, 85], from which the verifiability and privacy protections are inherited. However, there remains a need for verifiable DP mechanisms with lower overhead, thereby enhancing scalability and practicality, while achiev-

1

ing better utility and privacy trade-offs. For example, the total execution time of the binomial mechanism exceeds 30 minutes to achieve $(10^{-3}, 10^{-10})$-DP on a single-dimensional input, while it is unclear whether the application of discrete cryptographic primitives may exacerbate the numerical issues of the continuous Gaussian mechanisms [54, 71].

Furthermore, there lacks a formal framework for verifiable differential privacy, especially under the distributed setting with multiple clients and servers. Directly transplanting the notions of ZKPs, which typically operate over deterministic arithmetic circuits, fails to capture the correctness over the DP mechanisms with inherent randomness. The ZKP protocols, typically between a single prover and a single verifier, do not capture the interactions among multiple clients holding the data and servers executing the computations. For example, multiple servers may correlate their randomness to bias the output distribution, while their marginal distributions remain correct. Though pioneering infrastructural cryptographic developments have extended ZKPs to the distributed setup (e.g., MPCs) [7, 59, 76], the notion of distributed DP with respect to each client's local database with the additional requirement of verifiability remains unclear.

To address these challenges, we present **Verifiable Distributed Differential Privacy (VDDP)**. Our contributions can be summarized as follows:

- We rigorously define verifiable differential privacy under the distributed client-server-verifier setup, capturing the potential for collusive deviations from the protocol by clients and servers, as well as all privacy leakages from one client to colluding clients, servers, and the verifier. **(Section 3)**
- We explore the relationship between ZKP and DP. While ZKPs are provably sufficient for achieving DP, we construct a concrete counterexample showing that they are not necessary for achieving verifiable DP. **(Section 3.4.2)**
- We develop the *Verifiable Distributed Discrete Laplacian Mechanism (VDDLM)*. Compared with VDBM, VDDLM is up to 4×10^5x faster in proof generation with only 0.1-0.2x error under the same privacy costs. **(Section 4)**
- We propose an improved solution to *Verifiable Randomized Response (VRR)* under local DP, with asymptotically and empirically improved overhead, achieving up to a 5000x improvement in communication cost and verifier's overhead as a special case of VDDP. **(Section 5)**

2 Preliminaries

In this study, we assume that all parties involved, including clients, servers, verifiers, as well as generic adversaries, are probabilistic polynomial time (PPT). We also assume that the number of clients and servers and the size of each local database are polynomial in the security parameter λ. We use $\mathsf{negl}(\lambda)$ to denote the set of functions $\{\mu(\lambda)\}$ such that $\lim_{\lambda \to +\infty} p(\lambda)\mu(\lambda) = 0$ for any polynomial p. We use $\mathbb{F}$ to represent a prime order finite field and $\mathbb{G}$ to represent a mul-

tiplicative group isomorphic to the addition in $\mathbb{F}$, where the Diffie-Hellman assumption holds and g, h are two generators of $\mathbb{G}$. A list of integer values from 0 to $n-1$ is denoted by $[n]$. We denote vectors with an arrow on top, such as $\overrightarrow{\mathsf{Ser}}$ for the set of all servers and $\overrightarrow{\mathsf{Cli}}$ for the set of clients. We summarize the rest of the notations used in this study in Table 6 of Appendix A and present the necessary notions and primitives next.

2.1 Differential Privacy (DP)

We first present the standard definition that focuses on the scenario of *central DP*, where a server collects the database from multiple clients and executes the DP mechanism, and transmits its output to a data analyst.

Definition 2.1 (Differential Privacy [38]). *A mechanism $\mathcal{M}$: $\mathcal{D} \to \mathcal{Y}$, where $\mathcal{D}$ is the space of possible input databases and $\mathcal{Y}$ is the space of possible outputs, is (ε, δ)-differentially private if for any pair of neighbouring (different in one record) databases D and D' and any measurable subset $S \subseteq \mathcal{Y}$,*

$$\Pr[\mathcal{M}(D) \in S] \le e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta. \tag{1}$$

$\mathcal{M}$ *achieves ε-pure differential privacy when $\delta = 0$.*

In *local DP* [37, 41] (see Appendix A.1 for the definition), each client runs a DP perturbation on their own record before directly submitting it to the data analyst.

Due to the application of cryptographic primitives in this study, we consider *computational differential privacy (CDP)*.

Definition 2.2 (Computational Differential Privacy [8, 72]). *A mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Y}$, where $\mathcal{D}$ is the space of possible input databases and $\mathcal{Y}$ is the space of possible outputs, is (ε, δ)-differentially private if for each PPT adversary $\mathcal{A}$, there exists a function $\mu(\lambda) \in \mathsf{negl}(\lambda)$ such that for any pair of neighbouring databases D and D',*

$$\Pr[1 \leftarrow \mathcal{A}(\mathcal{M}(D))] \le e^{\varepsilon} \Pr[1 \leftarrow \mathcal{A}(\mathcal{M}(D'))] + \delta + \mu(\lambda).$$

Note that the original definition of CDP [72] considers the case where $\varepsilon = \varepsilon(\lambda)$ is a function of the security parameter λ, and $\delta = 0$, which aligns with pure DP. However, later works have extended the definition to δ being a non-negligible function $\delta(\lambda) \notin \mathsf{negl}(\lambda)$ [9, 14, 40, 70]. In this study, it suffices to consider the special case with constant (ε, δ).

Discrete Laplacian Mechanism. Due to the numerical issues of continuous DP mechanisms (e.g., Laplacian mechanism and Gaussian mechanism) [54, 71], their discrete versions have been developed to prevent unintentional privacy leakage from floating-point arithmetic. Hence, we focus on the discrete DP mechanisms in our study due to the application of cryptographic primitives. The discrete Laplacian mechanism [6, 48] involves perturbing the result with additive discrete Laplacian noise $\mathsf{Lap}_{\mathbb{Z}}(t)$, where for any $r \in \mathbb{Z}$,

$$\Pr[r \leftarrow \mathsf{Lap}_{\mathbb{Z}}(t)] = \frac{e^{\frac{1}{t}} - 1}{e^{\frac{1}{t}} + 1} \cdot \exp\left(-\frac{|r|}{t}\right). \tag{2}$$

For any query $q : \mathcal{D} \to \mathbb{Z}$ with sensitivity Δ (i.e., $|q(D) - q(D')| \leq \Delta$ for any pair of neighbouring databases D and D'), perturbing the output $q(D)$ by the additive noise of $\mathsf{Lap}_{\mathbb{Z}}\left(\frac{\Delta}{\varepsilon}\right)$ achieves $(\varepsilon, 0)$-DP. For a multi-dimensional query, adding i.i.d. discrete Laplacian samples to each dimension of the output achieves the same DP guarantee when L1 distance in the output spaces are used to define the sensitivity [38].

Randomized Response (RR). K-ary randomized response mechanism [56, 58, 86–89] is an important local DP mechanism, where each client randomizes its input $x \in [K]$ to $k \in [K]$ with the following probability

$$\Pr\left[\mathcal{M}(x) = k\right] = p_{(k-x) \bmod K}, \tag{3}$$

where $\sum_{k \in [K]} p_k = 1$. This mechanism achieves ε-pure DP where $\varepsilon = \log \frac{p_0}{\min_{k=1}^{K-1} p_k}$. It is common to set $p_0 > p_1 = p_2 = \cdots = p_{K-1}$ for achieving DP and optimal utility.

2.2 Cryptographic Primitives

To facilitate the applications of the cryptographic primitives, all variables and their arithmetic relations are mapped to a finite field $\mathbb{F}$. Specifically, a d-dimensional $\mathbf{v} \in \mathbb{F}^d$ is encoded as a degree-$(d-1)$ polynomial F through the *number-theoretic transform (NTT)*, where for a multiplicative subgroup Ω of $\mathbb{F}$ with $|\Omega| = d$ and generator ω, $F(\omega^i) = \mathbf{v}_i$ for $i \in [d]$. We utilize the following cryptographic primitives over $\mathbb{F}$:

Pseudo-random Number Generator (PRNG). To verify large amounts of random values under high dimensionality, we use pseudo-random number generators (PRNGs) to facilitate the process and reduce communication overhead between the prover and verifier. For efficiency, we use the Legendre Pseudo-random Function (LPRF) [12, 13, 32, 68] with parallel PRNG construction [81] to generate fair coins, as applied in previous MPC and ZKP schemes for machine learning and data management [25, 52]. Given random seed $s \leftarrow\!\!\!\$\ \mathbb{F}$ and dimensionality d,

$$\mathsf{LPRF}(s; d) := (L_s(0), L_s(1), \ldots, L_s(d-1)) \tag{4}$$

where $L_s(k)$ is 1 if there exists x such that $x^2 = k + s$, and 0 otherwise. The correctness of $L_s(k)$ can be verified by precomputing a non-perfect-square t of $\mathbb{F}$, and then computing $x := \begin{cases} \sqrt{t(k+s)} & \text{if } b = 0 \\ \sqrt{k+s} & \text{if } b = 1 \end{cases}$ so it suffices to verify the correctness of the square relation and that b is either 0 or 1, i.e.,

$$x^2 = ((1-b)t + b)(k+s) \wedge b(1-b) = 0. \tag{5}$$

Zero-knowledge Proofs (ZKP). A *zero-knowledge proof (ZKP)* [11, 42, 49, 50, 85] is a potentially interactive protocol

$$\Pi := (\mathsf{P}(w, x, y) \leftrightarrow \mathsf{V}(x, y)) \tag{6}$$

where a possibly malicious prover P demonstrates to a semi-honest verifier V (both parties are PPT) that a statement in the form of $y = C(w\|x)$ is true, where C is a deterministic arithmetic circuit, the output y and part of the input x are known to both parties, while the other part of the input w is known only to the prover. Note that Equation (6) represents the exact protocol that an honest prover adheres to. Meanwhile, for a generic and possibly malicious prover P, we represent the execution of Π as $\mathsf{P} \leftrightarrow \mathsf{V}(x, y)$, $\mathsf{V}(\mathsf{P}, x, y)$, or $\Pi(\mathsf{P}, x, y)$. For a valid ZKP scheme with security parameter $\lambda \in \mathbb{N}$, it satisfies the following properties for any x, y:
- **(Completeness)** For any w such that $y = C(w\|x)$, V outputs 1 (i.e., acceptance) at the end of the execution of Π.
- **(Soundness)** If there does not exist w such that $y = C(w\|x)$, for any PPT prover P, $\Pr\left[\mathsf{V}(\mathsf{P}, x, y) = 1\right] \leq \mathsf{negl}(\lambda)$.
- **(Zero-knowledge)** There exists a PPT simulator Sim such that for any w such that $y = C(w\|x)$, $\mathsf{Sim}(x, y)$ is computationally indistinguishable from the view of V in Π.

However, in our context, merely preventing the non-existence of w is insufficient, as it does not stop the prover from biasing the output distribution. Therefore, we require knowledge soundness:
- **(Knowledge Soundness)** There exists a PPT extractor Ext with rewinding access to the provers that can extract a valid w from accepted provers with negligible soundness error, specifically,

$$\Pr\left[\begin{array}{c} y = C(w\|x) : \\ w \leftarrow \mathsf{Ext}(\mathsf{P}, x, y) \end{array}\right] \geq \Pr\left[\mathsf{V}(\mathsf{P}, x, y) = 1\right] - \mathsf{negl}(\lambda).$$

In this study, we utilize Pedersen commitment [77] for scalars and KZG commitment scheme [60] for multi-dimensional entities encoded as polynomials. These commitment schemes operate over $\mathbb{F}$ and $\mathbb{G}$, and achieves computational binding and hiding. Coupled with the *interactive oracle proofs (IOPs)* [5, 16, 26, 45, 65] that maintain completeness, soundness, and zero-knowledge, we construct the proofs for the arithmetic circuits C. The circuit C captures the correct execution of LPRF as in Equation (5), as well as the transformation from the uniformly random bits to the final output with desired distributions.

Secret Sharing. A secret sharing scheme [15, 83] is a method by which a secret x is divided into n shares

$$(\llbracket x \rrbracket_i)_{i \in [n]} \leftarrow\!\!\!\$\ \mathsf{SecretShare}(x) \tag{7}$$

such that the secret can be reconstructed only when a sufficient number of shares are combined. Specifically, a threshold t is set, and any t or more shares can be used to recover the secret via $\mathsf{RecSec}(\cdot)$, while fewer than t shares reveal no information about the secret. In this study, we utilize additive secret sharing (where $t = n$) to instantiate the new protocols.

3 Privacy Definition

In this section, we describe the problem setup as a distributed mechanism among the clients, servers, and a data analyst (verifier), and identify the challenges for making the mechanism

verifiable to the data analyst. Then, we describe our solution framework to tackle the challenges with formal security and privacy guarantees.

3.1 Problem Setup

Client-server-verifier Setup. The setup considered in this study involves $n_{\mathsf{Cli}} \geq 1$ **potentially malicious clients**, $n_{\mathsf{Ser}} \geq 1$ **potentially malicious servers**, and a **semi-honest** data analyst (who also acts as a **verifier**). We represent them as $\overrightarrow{\mathsf{Cli}} = (\mathsf{Cli}_j)_{j \in [n_{\mathsf{Cli}}]}$, $\overrightarrow{\mathsf{Ser}} = (\mathsf{Ser}_i)_{i \in [n_{\mathsf{Ser}}]}$, and V respectively. Each Cli_j possesses a local database D_j with the same schema, and makes secret-shares of D_j to a subset of the servers indexed by I_j. Correspondingly, each server Ser_i aggregates the secret shares from the clients indexed by $J_i := \{j : i \in I_j\}$, the set of clients who secret-share their data to Ser_i, executes the mechanism, and submits the result to V. V verifies the validity of the results received, which reduces to **1)** the validity of the D_js possessed by the clients and **2)** the correctness of the computations performed by the servers.

Note that central DP and local DP can also be treated as special cases of the client-server-verifier setup: for central DP, $n_{\mathsf{Ser}} = 1$ and $I_j = \{0\}$ for each j; for local DP, $n_{\mathsf{Ser}} = n_{\mathsf{Cli}}$ and $I_j = \{j\}$ for each j. The secret sharing schemes are trivial in both cases. Meanwhile, for generic secret-sharing based distributed DP mechanisms where each client secret-shares to all servers [17, 18, 51, 53], i.e., $I_j = [n_{\mathsf{Ser}}]$ for each j and $J_i = [n_{\mathsf{Cli}}]$ for each i. Therefore, to unify these three common cases, we assume that for any pair of clients Cli_j and $\mathsf{Cli}_{j'}$, either $I_j = I_{j'}$ or $I_j \cap I_{j'} = \emptyset$. By elementary combinatorics, it also holds that for any pair of servers Ser_i and $\mathsf{Ser}_{i'}$, either $J_i = J_{i'}$ or $J_i \cap J_{i'} = \emptyset$. Therefore, for any set of servers I, where all have received secret shares from the same set of clients, we use J_I to denote their shared set of clients.

We first **temporarily** assume that all clients and servers are semi-honest except that they may withdraw from the protocol prematurely. Under this temporary assumption, the verifiability requirement is not necessary. We then formalize the other components of such a distributed protocol:

- $\left([\![D_j]\!]_i \right)_{i \in I_j} \leftarrow\!\!\$\ \mathsf{SecretShare}(D_j)$: Each Cli_j makes secret shares and sends them to the servers;
- $[\![D]\!]_i \leftarrow \mathsf{AggrShare}\left([\![D_j]\!]_i : j \in J^* \cap J_i \right)$: Only a subset $J^* \subseteq [n_{\mathsf{Cli}}]$ of clients remain in the protocol, and each server aggregates the secret shares from them.
- $[\![y]\!]_i \leftarrow\!\!\$\ \mathcal{F}\left([\![D]\!]_i \right)$: Each Ser_i performs a prescribed secret-shared and randomized computation (the *server function*) $\mathcal{F}$ and computes the output $[\![y]\!]_i$.
- $(I, J) \leftarrow \mathsf{IdUsable}(I^*, J^*), y \leftarrow \mathsf{Aggr}\left([\![y]\!]_i : i \in I \right)$: Due to the withdrawal of the clients and servers, the final output can only be computed with respect to the input databases from a subset of clients $J \subseteq J^*$, using the secret-shared outputs from only a subset of clients $I \subseteq I^*$. V identifies I and J with $\mathsf{IdUsable}$ and aggregates the secret-shared outputs

Figure 1: Problem setup of DDP and VDDP ($n_{\mathsf{Cli}} = 3, n_{\mathsf{Ser}} = 4$) when targeting Cli_0's data. All parties except Cli_0 and H_0 form a collusion $\mathfrak{C}$ and collectively keep an internal state st. All transmissions leaving the green region (solid lines) are honestly computed but may cause information leakage and enable $\mathfrak{C}$ to update its internal state. The additional leakages under VDDP (commitments and proofs) are highlighted in blue. The transmissions from $\mathfrak{C}$ to the honest parties (dashed lines) may be forged. The transmissions within $\mathfrak{C}$ (dotted lines) may or may not occur due to the internal arrangements of $\mathfrak{C}$ and are invisible to the honest parties.

using Aggr. We require that, for $(I, J) \leftarrow \mathsf{IdUsable}(I^*, J^*)$, the equivalent input database to the protocol can be fully recovered directly as $D_{\mathsf{Cli}} \leftarrow \mathsf{AggrDB}(D_j : j \in J)$. AggrDB can be instantiated as concatenation for generic databases, or summation when D_j are the local histograms for counting queries. Meanwhile, we also require that the same database can be recovered from the servers' aggregated secret-shares as $D_{\mathsf{Ser}} \leftarrow \mathsf{RecDB}([\![D]\!]_i : i \in I)$, such that $D_{\mathsf{Cli}} = D_{\mathsf{Ser}}$.

Defining Distributed DP. Due to the distributed nature, the notion of DP in such a protocol can only be defined on a per-client basis, by considering a pair of neighboring local databases D_j and D'_j for each client Cli_j. Moreover, if all servers that process D_j were *colluding*, achieving any degree of DP would be impossible, as D_j can be fully recovered due to the secret-sharing scheme. Therefore, DP can only be achieved for the clients when the number of servers that are not fully honest is bounded. Denoting all fully honest servers in I_j as H_j, we treat all other clients than Cli_j, all servers other than H_j, as well as V collectively as a collusion $\mathfrak{C}$.

Example 3.1. *Figure 1 illustrates the setup with 3 clients and 4 servers. Cli_0 sends its secret shares to three servers, Ser_0, Ser_1, and Ser_2. The largest colluding gang $\mathfrak{C}$ interested in learning Cli_0's data includes Cli_1, Cli_2, Ser_2, Ser_3, as well as V. Adding Ser_0 or Ser_1 to $\mathfrak{C}$ will enable them to recover the true data of Cli_0 as $\mathfrak{C}$ would receive all secret shares from*

Cli_0. In this case, $\mathfrak{C}$ receives $[\![D_0]\!]_2$ from Cli_0, and $[\![y]\!]_0$ and $[\![y]\!]_1$ from Ser_0 and Ser_1, respectively.

The outputs of the server function do not solely depend on D_j itself, but also the secret shares S_{out} that Cli_j transmits to $\mathfrak{C}$, the secret shares S_{in} transmitted back to H_j, and the subset of clients J^*. Therefore, instead of treating the outputs of $\mathcal{F}$ as a single DP mechanism, we define it as a DP mechanism that is post-processed by S_{out}, S_{in}, and J^*.

Definition 3.2. *A server function $\mathcal{F}$ satisfies $(\tau, \varepsilon, \delta)$-distributed differential privacy (DDP) if there exists an (ε, δ)-computational DP mechanism $\mathcal{M}$ and a PPT function $\mathfrak{S}$ such that for any Cli_j and any subset of honest servers $H_j \subset I_j$ such that $|H_j| \geq |I_j| - \tau$,*

$$\mathsf{View}_{\mathcal{F}}^{\mathfrak{C}}(D_j, S_{out}, S_{in}, J^*) \equiv \mathfrak{S}(\mathcal{M}(D_j), S_{out}, S_{in}, J^*), \quad (8)$$

where $S_{out} = ([\![D_j]\!]_i)_{i \in I_j \setminus H_j}$ are the secret shares from Cli_j to the servers not in H_j, $S_{in} = ([\![D_{j'}]\!]_i)_{i \in H_j, j' \in I_{H_j} \setminus \{j\}}$ are the secret shares from the other clients to H_j, and $J^* \subseteq J_{H_j}$ (where $j \in J^*$) is the subset of the clients that remain in the protocol. $\mathsf{View}_{\mathcal{F}}^{\mathfrak{C}}$ is the view of the adversaries $\mathfrak{C}$ (consisting of all clients except Cli_j, all servers except H_j, and V) on the output of $\mathcal{F}$. With $\mathsf{PartialShare}(D_j, S_{out})$ being the conditional distribution of $([\![D_j]\!]_i)_{i \in H_j}$ given S_{out}, the view is:

$$\mathsf{View}_{\mathcal{F}}^{\mathfrak{C}}(D_j, S_{out}, S_{in}, J^*)$$

1: $([\![D_j]\!]_i)_{i \in H_j} \leftarrow\!\!\$\ \mathsf{PartialShare}(D_j, S_{out})$
2: **foreach** $i \in H_j$ **do**
3: $\quad [\![y]\!]_i \leftarrow\!\!\$\ \mathcal{F}(\mathsf{AggrShare}([\![D_{j'}]\!]_i : j' \in J^* \cap J_{H_j}))$
4: **endfor**
5: **return** $Y := ([\![y]\!]_i)_{i \in H_j}$

Note that the additional privacy parameter τ is highly related to the secret sharing scheme's threshold t for recovering the secret. Specifically, $\tau \leq t - 1$, since more than $t - 1$ colluding servers can perfectly recover the secret-shared D_j. Therefore, in the concrete designs of the protocols, we aim to achieve $\tau = t - 1$ to maximize the tolerance against colluding servers. Also, we employ the notion of computational DP to accommodate the application of PRNGs.

3.2 Towards Verifiability: Challenges

We further **remove the semi-honest assumption** on the clients and servers, and therefore consider the additional requirement for V to verify the output correctness. We identify several challenges in fulfilling this requirement:

Challenge 1: The server function $\mathcal{F}$ is vulnerable to attacks by malicious servers. Specifically, any malicious server Ser_i may deviate from the correct distribution of $\mathcal{F}(\cdot)$ when computing the output, and multiple malicious servers may correlate their sampling processes to further bias the distribution of the final output. However, randomized computations are not directly supported by existing proof systems [85] that operate over deterministic arithmetic circuits.

Challenge 2: As argued by Biswas and Cormode in 2023 [14], the verifier needs to distinguish between honest and dishonest clients and servers in the client-server-verifier model. On the one hand, the databases and computation results from dishonest clients and servers need to be removed so as not to affect the quality of the result received by the verifier. On the other hand, to achieve better utility of the protocol, upon discovering any dishonest parties, the protocol may continue such that the verifier can still receive meaningful and correct results from the honest parties. Therefore, unlike MPC protocols for DP mechanisms [17, 18, 24, 36, 91], the ideal functionality is not well-defined due to the varying input to the protocol as a result of the execution. Instead, the protocol design must properly capture the interference among **1)** the honesty of each client and server, **2)** whether each of them is accepted, and **3)** the output of the protocol.

Challenge 3: Unlike conventional secure computations of DP mechanisms [17, 18, 24, 36, 91], despite the inherited privacy guarantees on the protocol output, the proof of the correct execution may cause additional information leakage to the verifier, which breaks the original DP guarantee. Previous studies utilize ZK proofs [14, 61, 84] to prevent further leakages at this stage. However, as ZK is a stronger notion than DP, it is worth exploring if the ZK property of the proof is necessary for achieving end-to-end DP. More importantly, due to the distributed nature of the setting, the guarantee must be established with respect to each client's local database.

3.3 Randomness Disentanglement

In response to **Challenge 1**, it is necessary to convert the randomized $\mathcal{F}$ into a deterministic function, with auxiliary inputs that represent the randomnesses in $\mathcal{F}$. We first describe two simple attempts to achieve this conversion:

- If the servers solely determine the randomness, it is still impossible to affirm the underlying distribution of the randomness through one sample.
- If the verifier solely determines the randomness, the knowledge of the randomness term may break the DP guarantee. The additive noises (e.g., discrete Laplacian) are especially vulnerable to this simple attack, as the unperturbed output is fully recovered by removing the noise value.

The aforementioned security and privacy issues necessitate the distribution of the tasks of randomness generation between the verifier and the servers. Therefore, we employ a public-coin-private-coin model to disentangle the randomnesses between the two parties as in Definition 3.3.

Definition 3.3 (Randomness disentanglement (RD)). *Given the randomized function $\mathcal{F} : X \to \mathcal{Y}$, we call a triplet $(f, \mathcal{P}_\sigma, \mathcal{P}_{pc})$ a randomness disentanglement (RD) for $\mathcal{F}$, where $\mathcal{P}_\sigma, \mathcal{P}_{pc}$ are two probability distributions and $f : X \times \mathrm{Supp}(\mathcal{P}_\sigma) \times \mathrm{Supp}(\mathcal{P}_{pc}) \to \mathcal{Y}$ is a deterministic function, if and only if for any $x \in X$:*

- *For any fixed $\sigma \in \mathrm{Supp}(\mathcal{P}_\sigma)$, the distribution of $f(x, \sigma, pc) : pc \leftarrow\!\!\$ \mathcal{P}_{pc}$ is equivalent to $\mathcal{F}(x)$.*
- *For any fixed $pc \in \mathrm{Supp}(\mathcal{P}_{pc})$, the distribution of $f(x, \sigma, pc) : \sigma \leftarrow\!\!\$ \mathcal{P}_\sigma$ is equivalent to $\mathcal{F}(x)$.*

In Definition 3.3, the server controls the obfuscation term (private coin) σ, while the verifier controls the generation of the public coin pc. Hence, regardless of the choice of the σ by the server, it is unable to deviate from the desired distribution of the output. Meanwhile, regardless of the choice of pc by the verifier, the distribution of the output remain the same, therefore offering the same privacy protection such that the adversaries cannot obtain additional information through the additional view of pc.

Example 3.4 (RD with LPRF). *With LPRF introduced in Section 2.2, the server function can be instantiated as*

$$\mathcal{F}(\llbracket D \rrbracket_i) := C(\llbracket D \rrbracket_i, \mathrm{LPRF}(s_i)) : s_i \leftarrow\!\!\$ \mathbb{F}, \qquad (9)$$

where C is a deterministic arithmetic circuit that transforms the input and the sampled fair coins (output of LPRF) to the server's output. Therefore, the RD of Equation (9) can be constructed as

$$f(\llbracket D \rrbracket_i, \sigma_i, pc_i) := C(\llbracket D \rrbracket_i, \mathrm{LPRF}(\sigma_i + pc_i)). \qquad (10)$$

with σ_i and pc_i both sampled from the uniform distribution over $\mathbb{F}$. Here, fixing one of σ_i and pc_i and having the other uniformly randomly sampled results in the distribution of $\sigma_i + pc_i$ being the uniform distribution over $\mathbb{F}$, thus giving the same output distribution as in Equation (9), which satisfies Definition 3.3. The server therefore needs to prove to the verifier the correctness of computation over LPRF as in Equation (5), as well as that over C.

Resistance against server collusion. In addition to the malicious deviations performed by single servers, multiple servers may collude to further bias the output distribution by correlating their sampling processes. For example, in the distributed discrete Gaussian mechanism [57], multiple servers may add the same copy of noise that is drawn from the prescribed Gaussian distribution, instead of sampling them independently. Thus, even though the distribution of each server's output is correct, that of the aggregated final output is not. However, we observe that RD preempts this type of collusion.

Lemma 3.5. *For any joint distribution over $(\sigma_0, \sigma_1, \ldots, \sigma_{n_{Ser}-1})$, if each pc_i where $i \in [n_{Ser}]$ is i.i.d. sampled from $\mathcal{P}_{pc}$, then the joint distribution of $(f(x_i, \sigma_i, pc_i))_{i \in [n_{Ser}]}$ is identical to the Cartesian product of the distribution over all $\mathcal{F}(x_i)$s, i.e., $\bigotimes_{i \in [n_{Ser}]} \mathcal{F}(x_i)$.*

By Lemma 3.5, even if malicious servers attempt to correlate the choice of the obfuscation factors, the outputs of the server functions are always mutually independent as desired.

3.4 I2DP: Interactive Distributed Proof of Differential Privacy

Given the RD described in Section 3.3 which resolves **Challenge 1**, in this section, we define an *Interactive Distributed Proof of Differential Privacy (I2DP)*, through which potentially malicious clients and servers prove to the semi-honest verifier the validity of their local databases and the correct computations over the RDs. Before the execution of I2DP, the following pre-requisites prepare its inputs:

- $pp \leftarrow\!\!\$ \mathsf{Setup}(1^\lambda)$ generates public parameter pp required for the applications of the cryptographic primitives.
- $(\llbracket D_j \rrbracket_i)_{i \in I_n} \leftarrow\!\!\$ \mathsf{SecretShare}(D_j)$, as defined in Section 3.1.
- $\llbracket com_j \rrbracket_i \leftarrow \mathsf{CommitShare}(\llbracket D_j \rrbracket_i, \llbracket r_j \rrbracket_i; pp)$ computes a binding and hiding commitment of Cli_j's share to Ser_i, i.e., $\llbracket D_j \rrbracket_i$, with randomness r_j, where $i \in I_j$. We also assume that the commitment of original databases D_js (known by the clients) and the aggregated shares $\llbracket D \rrbracket_i$s can be computed directly without seeing the committed values, using two deterministic functions $\mathsf{RecDataCom}$ and $\mathsf{AggrShareCom}$, respectively. Note that this assumption holds true for common secret-sharing schemes including additive secret sharing scheme and Shamir's secret sharing scheme [15, 83]. Furthermore, we assume that the clients and servers agree on the commitments of the shares (and therefore the underlying committed values), which can be achieved by digital signatures on the commitments [21, 47, 55, 62, 80].
- $\psi_i \leftarrow \mathsf{CommitOb}(\sigma_i, \rho_i; pp)$ where Ser_i chooses the obfuscation term σ_i and computes a binding and hiding commitment of it with randomness ρ_i.
- $pc_i \leftarrow\!\!\$ \mathcal{P}_{pc}$ where the verifier samples the public coins independently for each server. Note that step is completed after $\mathsf{CommitOb}$ to prevent adversarial choices of σ_is.

An I2DP Π is described in Figure 2. Specifically, it consists of the following components:

- $b_j^{\mathsf{Cli}} \leftarrow\!\!\$ \Pi_{\mathsf{data}}(\mathsf{Cli}_j, com_j; pp)$ is an interactive proof of the validity of data bound by com_j, i.e., $D_j \in \mathcal{D}$. The output $b_j^{\mathsf{Cli}} = 1$ if the proof is accepted, and 0 otherwise (collectively denoted as $\mathbf{b}^{\mathsf{Cli}}$ for all js). If b_j^{Cli} is 0, Cli_j and its data are excluded from all subsequent computations.
- $b_i^{\mathsf{Ser}} \leftarrow\!\!\$ \Pi_{\mathsf{comp}}(\mathsf{Ser}_i, \llbracket y \rrbracket_i, \llbracket com \rrbracket_i, \psi_i, pc_i; pp)$ is an interactive proof of the correctness of Ser_i's computation, i.e., Ser_i has honestly computed $\llbracket y \rrbracket_i = f(\llbracket D \rrbracket_i, \sigma_i, pc_i)$. The output $b_i^{\mathsf{Ser}} = 1$ if the proof is accepted, and 0 otherwise (collectively denoted as $\mathbf{b}^{\mathsf{Ser}}$ for all is).
- $(I, J) \leftarrow \mathsf{IdUsable}(I^*, J^*)$, $y \leftarrow \mathsf{Aggr}(\llbracket y \rrbracket_i : i \in I)$, as defined in Section 3.1.

$$\Pi\left(\overrightarrow{\mathsf{Cli}}, \overrightarrow{\mathsf{Ser}}, \left(\llbracket\mathsf{com}_j\rrbracket_i\right)_{i,j}, (\psi_i)_i, (\mathsf{pc}_i)_i; \mathsf{pp}\right)$$

1: **for** $j \in [n_{\mathsf{Cli}}]$ **do**

2: $\quad \mathsf{com}_j \leftarrow \mathsf{RecDataCom}\left(\left(\llbracket\mathsf{com}_j\rrbracket_i\right)_{i\in I_j}; \mathsf{pp}\right)$

3: $\quad b_j^{\mathsf{Cli}} \leftarrow_\$ \Pi_{\mathsf{data}}\left(\mathsf{Cli}_j, \mathsf{com}_j; \mathsf{pp}\right)$

4: **endfor** $\quad /\!/ J^* := \{j : b_j^{\mathsf{Cli}} = 1\}$, i.e., accepted clients remain.

5: **for** $i \in [n_{\mathsf{Ser}}]$ **do**

6: $\quad \llbracket\mathsf{com}\rrbracket_i \leftarrow \mathsf{AggrShareCom}\left(\left(\llbracket\mathsf{com}_j\rrbracket_i\right)_{j\in J_i \cap J^*}; \mathsf{pp}\right)$

7: $\quad \llbracket y\rrbracket_i \leftarrow_\$ \mathsf{Ser}_i\left((b_j)_{j\in J_i}, \mathsf{pc}_i; \mathsf{pp}\right)$

8: $\quad b_i^{\mathsf{Ser}} \leftarrow_\$ \Pi_{\mathsf{comp}}\left(\mathsf{Ser}_i, \llbracket y\rrbracket_i, \llbracket\mathsf{com}\rrbracket_i, \psi_i, \mathsf{pc}_i; \mathsf{pp}\right)$

9: **endfor** $\quad /\!/ I^* := \{i : b_i^{\mathsf{Ser}} = 1\}$, i.e., accepted servers remain.

10: $(I, J) \leftarrow \mathsf{IdUsable}(I^*, J^*)$

11: **return** $y \leftarrow \mathsf{Aggr}\left(\llbracket y\rrbracket_i : i \in I\right)$

Figure 2: Interactive distributed proof of differential privacy with potentially malicious clients and servers.

3.4.1 Security Guarantees

In response to **Challenge 2**, we quantify the interference among the honesty of the clients and servers, the values of $\mathbf{b}^{\mathsf{Cli}}$ and $\mathbf{b}^{\mathsf{Ser}}$, and the value of the protocol output. Ideally, a client's database should be included in the computation if and only if the client and the servers handling it are honest, and the output is correct with respect to the union of the databases from all such clients. These properties are captured by *completeness* and *(knowledge) soundness* under the single-prover setting, as introduced in Section 2.2, and have been adopted in previous studies on verifiable differential privacy [14, 61, 84]. However, further adaptations of these notions are required to extend them into the client-server-verifier setting, which we present in this section. As $\mathsf{Aggr}(\cdot)$ is executed by the semi-honest verifier, we focus on the subset of servers I and clients J identified by the server using $\mathsf{IdUsable}(I^*, J^*)$, and the correctness of $(\llbracket y\rrbracket_i)_{i\in I}$ with respect to $(D_j)_{j\in J}$.

Under the distributed setting, an honest client cannot guarantee that its local database will be included in the aggregated result if too many servers it submits data to are dishonest. This interference makes it impossible to directly borrow the notion of completeness under the single-prover setting. Instead, we capture this type of interference in Definition 3.6, which additionally requires a lower bound on the number of honest servers for the client's local database to be processed.

Definition 3.6 (θ-completeness of I2DP). *An I2DP Π is θ-complete iff for any honest client Cli_j with at least $|I_j| - \theta$ honest servers indexed by I_j, $j \in J$ with probability 1.*

Proposition 3.7. *An I2DP Π is θ-complete if Π_{data} and Π_{comp} are complete, and for each client Cli_j, $\mathsf{IdUsable}(I^*, J^*)$ outputs $j \in J$ when $j \in J^*$ and $|I_j \cap I^*| \geq |I_j| - \theta$.*

We argue that soundness is insufficient to achieve the goal

of VDDP. In particular, soundness merely requires the existence of the inputs and obfuscations that match the commitments and secret-shared outputs. Hence, it does not prevent the scenario where a malicious prover forges a result, as long as the output is reachable due to randomness. Therefore, we enforce the requirement of knowledge soundness, which enforces the knowledge of the local databases by the clients, and that of the obfuscation factors by the servers, all of which match the final aggregated results.

Definition 3.8 (Knowledge soundness of I2DP). *An I2DP Π is knowledge sound iff there exists a knowledge extractor that, except for negligible probability, can extract from the accepted set of servers and clients I^* and J^* the local databases $(D_j)_{j\in J^*}$, the aggregated secret shares $(\llbracket D\rrbracket_i)_{i\in I^*}$, and the obfuscation factors $(\sigma_i)_{i\in I^*}$, such that $\llbracket y\rrbracket_i = f(\llbracket D\rrbracket_i, \sigma_i, \mathsf{pc}_i)$ for each $i \in I^*$, all extracted values match their commitments, and $\mathsf{AggrDB}(D_j : j \in J) = \mathsf{RecDB}(\llbracket D\rrbracket_i : i \in I)$ for $(I, J) \leftarrow \mathsf{IdUsable}(I^*, J^*)$.*

Theorem 3.9. *An I2DP Π is knowledge sound if the sub-protocols Π_{data} and Π_{comp} are knowledge sound.*

3.4.2 Privacy Guarantees

In response to **Challenge 3**, we develop the end-to-end DP guarantee that incorporates the privacy leakages from both within the execution of I2DP and its prerequisites.

Definition 3.10 ($(\tau, \varepsilon, \delta)$-verifiable distributed differential privacy). *An I2DP Π is $(\tau, \varepsilon, \delta)$-verifiably distributed differentially private (VDDP) if for any public parameter $\mathsf{pp} \in \mathsf{Supp}(\mathsf{Setup}(1^\lambda))$, any honest client Cli_j, any set of honest servers $H_j \subset I_j$ such that $|H_j| \geq |I_j| - \tau$, and any $\mathfrak{C}$ (consisting of V and all clients and servers except Cli_j and H_j) that follows a certain potentially malicious strategy as a PPT algorithm, $\mathsf{View}_\Pi^{\mathfrak{C}}(\cdot; \mathsf{pp})$ is (ε, δ)-computational DP, where $\mathsf{View}_\Pi^{\mathfrak{C}}(D_j; \mathsf{pp})$ is the resulting view of $\mathfrak{C}$ in the execution of Π (including its prerequisites).*

Note that in Definitions 3.6, 3.8, and 3.10, we have described the security and privacy guarantees as the properties of the I2DP. However, for an end-to-end mechanism/protocol that encloses I2DP and its prerequisites, we also describe it to satisfy these properties if the underlying I2DP satisfies the same properties. Such a mechanism/protocol satisfying all three properties is also generically referred to as a *VDDP mechanism/protocol* in the absence of ambiguity.

Comparing with the adversary's view in DDP, Π only gives the adversary the additional view of the commitments and proofs due to the additional requirement of verifiability (see Appendix B.1 for details). Therefore, if all commitments are hiding and all proofs are zero-knowledge, no additional information leakage about the local database D_j results, such that the same degree of privacy protection can be achieved. We formalize this as Theorem 3.11.

Theorem 3.11. *An I2DP Π is $(\tau, \varepsilon, \delta)$-VDDP if the underlying server function is $(\tau, \varepsilon, \delta)$-DDP, the commitment schemes are hiding, and the underlying sub-protocols Π_{data} and Π_{comp} are both zero-knowledge.*

Theorem 3.11 states that ZK is sufficient for achieving VDDP. However, we further argue that the converse statement is not necessarily true. In particular, ZK can be viewed as a stronger case of DP with zero privacy loss. Therefore, since our goal is achieving end-to-end DP, the ZK requirement on the proof can be relaxed to DP, such that the leakages from the server function and proofs can still be composed and upper bounded as DP. To formalize and prove both statements, we state an extended version of Theorem 3.11. Theorem 3.11 is a special case of Theorem 3.12 where $\varepsilon' = \delta' = 0$.

Theorem 3.12. *An I2DP Π is $(\tau, \varepsilon + \varepsilon', \delta + \delta')$-VDDP if the underlying server function is $(\tau, \varepsilon, \delta)$-DDP, the commitment schemes are hiding, the underlying sub-protocol Π_{comp} is zero-knowledge, and there exists a simulator $\mathsf{Sim}_{\mathsf{data}}$ such that for any D_j, com_j, and pp, $\mathsf{Sim}_{\mathsf{data}}(D_j, \mathsf{com}_j; \mathsf{pp})$ indistinguishably simulates the verifier's view in $\Pi_{\mathsf{data}}(\mathsf{Cli}_j(D_j, r_j), \mathsf{com}_j; \mathsf{pp})$, and there exists a (ε', δ')-differentially private mechanism $\mathcal{G}$ and PPT function h that $\mathsf{Sim}_{\mathsf{data}}(D_j, \mathsf{com}_j; \mathsf{pp}) \equiv h(\mathcal{G}(D_j), \mathsf{com}_j; \mathsf{pp})$.*

Given Theorem 3.12, we analyze and modify the verifiable distributed binomial mechanism (VDBM) introduced by Biswas and Cormode [14] in CCS '23 under our framework, as a concrete example that ZK is not necessary for DP.

Example 3.13 (VDBM [14])**.** *In VDBM [14], each client possesses a local count $x_j \in \{0, 1\}$, and shares it with all servers using the additive secret share mechanism as $[\![x_j]\!]_i$. Each server then aggregates the secret shares as $[\![y]\!]_i \leftarrow \sum_{j \in J^*} [\![x_j]\!]_i + Binom(n_b, \frac{1}{2})$. The RD f is constructed by having the server and verifier each sample n_b fair coins, denoted $\sigma_i = (\sigma_{i,k})_{k \in [n_b]}$ and $\mathsf{pc}_i = (\mathsf{pc}_{i,k})_{k \in [n_b]}$, correspondingly, where $f([\![x]\!]_i) = \sum_{j \in J^*} [\![x_j]\!]_i + \sum_{k \in [n_b]} (\sigma_{i,k} \oplus \mathsf{pc}_{i,k})$. Compatible with the additive secret-sharing scheme, $\mathsf{IdUsable}(I^*, J^*) = \begin{cases} (I^*, J^*) & \text{if } I^* = [n_{\mathsf{Ser}}] \\ (\emptyset, \emptyset) & \text{otherwise} \end{cases}$, and Aggr can be instantiated as the simple summation.*

The server function can achieve $(n_{\mathsf{Ser}} - 1, \varepsilon, \delta)$-DDP for the same (ε, δ), i.e., the same DP guarantee as the central DP setting can be achieved when at least one server is fully honest. Π_{comp} can be constructed via the homomorphism of the commitments. Meanwhile, Π_{data}, where the clients act as the provers, is instantiated with Π_{Bin} which proves the knowledge of the committed value is either 0 or 1 [31, 33, 85]. Therefore, by Proposition 3.7 and Theorems 3.9 and 3.11, VDBM satisfies 0-completeness, knowledge soundness, and $(n_{\mathsf{Ser}} - 1, \varepsilon, \delta)$-VDDP.

In Example 3.13, it is possible to modify Π_{Bin} such that the modified version is no longer zero-knowledge but still

satisfies the condition in Theorem 3.12. The modification involves biasing the distribution of the transcript according to input $x_j \in \{0, 1\}$, as detailed in Appendix B.2. By Theorem 3.14, the modified version of VDBM still satisfies VDDP.

Theorem 3.14. *For any $\varepsilon' > 0$, there exists a modified and non-ZK version of Π_{Bin} such that VDBM introduced in Example 3.13 satisfies 0-completeness, knowledge soundness, and $(n_{\mathsf{Ser}} - 1, \varepsilon + \varepsilon', \delta)$-VDDP with the modified Π_{Bin}.*

4 VDDLM: Verifiable Distributed Discrete Laplacian Mechanism

In this section, we develop the *verifiable distributed discrete Laplacian mechanism (VDDLM)* for counting queries, which achieves a better privacy-utility trade-off and lower overhead compared with VDBM. Similar to VDBM, the clients' local counts are secret-shared to all servers using additive secret-sharing. Each server aggregates all secret shares from the clients and adds a copy of the discrete Laplacian noise elementwise and independently to each dimension as the output. The output from all servers is further aggregated by the data analyst (verifier) as the final output. We first present the sampling circuit and the interactive proof for the VDDLM and then rigorously analyze its privacy, utility, and overhead.

4.1 Construction of RD for VDDLM

We utilize the RD described in Example 3.4 for VDDLM. In particular, since the noise is additive, given aggregated d-dimensional secret-shared input counts $[\![\mathbf{x}]\!]_i \in \mathbb{F}^d$ for each Ser_i, the RD can be constructed as $f([\![\mathbf{x}]\!]_i, \sigma_i, \mathsf{pc}_i) := [\![\mathbf{x}]\!]_i + C_{\mathsf{Lap}}(\mathsf{LPRF}(\sigma_i + \mathsf{pc}_i))$. We focus on the construction of C_{Lap} for the rest of this section.

We utilize a decomposition of $\mathsf{Lap}_{\mathbb{Z}}(t)$ [36, 91] from elementary probability theory. As introduced in Section 2.1, for $r \leftarrow_\$ \mathsf{Lap}_{\mathbb{Z}}(t)$, the probability that $r = 0$ is $p_z^* = \frac{e^{\frac{1}{t}} - 1}{e^{\frac{1}{t}} + 1}$. Meanwhile, conditioning on $r \neq 0$, the distribution of $|r| - 1$ is the geometric distribution with success probability $p_g^* = 1 - e^{-\frac{1}{t}}$, i.e., $\mathsf{Geom}(p_g^*)$. Furthermore, a geometric distribution $r_g \leftarrow_\$ \mathsf{Geom}(p_g^*)$ can be represented as the sum of binary variables, i.e., $r_g = \sum_{i \geq 0} 2^i r_i$. Each r_i is independently sampled from the Bernoulli distribution $\mathsf{Ber}(p_i^*)$ where $p_i^* = \frac{1}{1 + p_g^{*-2^i}}$.

Proposition 4.1. *With mutually independently sampled $b_z \leftarrow_\$ \mathsf{Ber}(p_z^*)$, $s \leftarrow_\$ \{-1, 1\}$, and $r_i \leftarrow_\$ \mathsf{Ber}(p_i^*)$ for $i \geq 0$, $(1 - b_z) \cdot s \cdot (\sum_{i \geq 0} 2^i r_i + 1)$ follows the distribution of $\mathsf{Lap}_{\mathbb{Z}}(t)$.*

We construct C_{Lap} as Algorithm 1, directly based on Proposition 4.1. Since for large is, the corresponding p_i^*s are negligible, we only keep the least significant γ (range parameter) bits. Also, we utilize a subroutine $C_{\mathsf{Ber}}(\cdot; p^*, \nu)$ described in Appendix C.1 to sample from $\mathsf{Ber}(p^*)$ for any $p^* \in (0, 1)$

$\mathsf{Ser}_i\left(\left([\![\mathbf{x}_j]\!]_i\right)_j, \left([\![\mathbf{r}_j]\!]_i\right)_j, \sigma_i,\right.$ $\qquad$ $\mathsf{V}\left(\left([\![\mathsf{com}_j]\!]_i\right)_{i,j},\right.$

$\left.\rho_i, \left([\![\mathsf{com}_j]\!]_i\right)_j, \psi_i, \mathsf{pc}_i; \mathsf{pp}\right)$ $\qquad$ $\left.(\psi_i)_i, (\mathsf{pc}_i)_i; \mathsf{pp}\right)$

1:	$\sigma_i' \leftarrow \sigma_i + \mathsf{pc}_i$		$J^* \leftarrow \Pi_{\mathsf{data}}$
2:	$\psi_i' \leftarrow \psi_i \cdot g^{\mathsf{pc}_i}$		$\psi_i' \leftarrow \psi_i \cdot g^{\mathsf{pc}_i}$
3:	$\mathbf{z}_i \leftarrow \mathsf{LPRF}(\sigma_i')$		
4:	$\mathbf{t}_i \leftarrow\!\!\$ \; \mathbb{F}^d$		
5:	$\zeta_i \leftarrow \mathsf{Commit}(\mathbf{z}_i, \mathbf{t}_i; \mathsf{pp})$	$\xrightarrow{\zeta_i}$	ζ_i
6:	$\mathsf{P}_{\mathsf{LPRF}}\left(\sigma_i', \rho_i, \mathbf{z}_i, \psi_i', \zeta_i; \mathsf{pp}\right)$	$\xleftrightarrow{\pi_{\mathsf{LPRF}}}$	$\mathsf{V}_{\mathsf{LPRF}}(\psi_i', \zeta_i; \mathsf{pp})$
7:	$[\![\mathbf{x}]\!]_i \leftarrow \sum_{j \in J^*} [\![\mathbf{x}_j]\!]_i$		
8:	$[\![\mathbf{r}]\!]_i \leftarrow \sum_{j \in J^*} [\![\mathbf{r}_j]\!]_i$		
9:	$[\![\mathsf{com}]\!]_i \leftarrow \sum_{j \in J^*} [\![\mathsf{com}_j]\!]_i$		$[\![\mathsf{com}]\!]_i \leftarrow \sum_{j \in J^*} [\![\mathsf{com}_j]\!]_i$
10:	$[\![\mathbf{y}]\!]_i \leftarrow [\![\mathbf{x}]\!]_i + C_{\mathsf{Lap}}(\mathbf{z}_i)$	$\xrightarrow{[\![\mathbf{y}]\!]_i}$	$[\![\mathbf{y}]\!]_i$
11:	$\mathsf{P}_{\mathsf{Lap}}\left([\![\mathbf{x}]\!]_i, \mathbf{z}_i, [\![\mathbf{y}]\!]_i, [\![\mathsf{com}]\!]_i, \zeta_i; \mathsf{pp}\right)$	$\xleftrightarrow{\pi_{\mathsf{Lap}}}$	$\mathsf{V}_{\mathsf{Lap}}([\![\mathsf{com}]\!]_i, \zeta_i; \mathsf{pp})$

Figure 3: The instantiation of I2DP in VDDLM.

and precision parameter ν, such that given $\mathbf{b} \leftarrow\!\!\$ \{0,1\}^\nu$, $C_{\mathsf{Ber}}(\mathbf{b}; p^*, \nu) \sim \mathsf{Ber}\left(\frac{\lfloor 2^\nu p^* \rfloor}{2^\nu}\right)$, where the parameter is the closest multiple of $\frac{1}{2^\nu}$ to p^*. For p_z^* and each p_i^*, we denote p_z and p_i as the such realized parameters using C_{Ber}.

Algorithm 1 Sampling from $\mathsf{Lap}_{\mathbb{Z}}(t)$

Require: range parameter γ; precision parameters ν_z and $(\nu_i)_{i \in [\gamma]}$; precomputed p_z^* and $(p_i^*)_{i \in [\gamma]}$; random bits $\mathbf{b}_z \in \{0,1\}^{\nu_z}$, $b_s \in \{0,1\}$, and $\mathbf{b}_i \in \{0,1\}^{\nu_i}$ for each $i \in [\gamma]$.

1: **function** $C_{\mathsf{Lap}}(\mathbf{b}_z, b_s, (\mathbf{b}_i)_{i \in [\gamma]})$

2: $\quad b_z \leftarrow C_{\mathsf{Ber}}(\mathbf{b}_z; p_z^*, \nu_z)$ $\qquad \triangleright p_z^* := \frac{e^{\frac{1}{t}} - 1}{e^{\frac{1}{t}} + 1}$

3: $\quad s \leftarrow 2 \cdot b_s - 1$

4: $\quad a \leftarrow \sum_{i \in [\gamma]} 2^i \cdot C_{\mathsf{Ber}}(\mathbf{b}_i; p_i^*, \nu_i) + 1$

5: $\quad$ **return** $r \leftarrow (1 - b_z) \cdot s \cdot a$

6: **end function**

When there is no ambiguity on the hyperparameters required in Algorithm 1, we denote C_{Lap} as the output distribution of C_{Lap} with the input of uniformly random bits. Also, note that Algorithm 1 is unidimensional but can be trivially extended to the multidimensional case by having multiple instances of C_{Lap} assembled in parallel.

4.2 Design of the VDDLM Protocol

We utilize the cryptographic primitives introduced in Section 2.2 to instantiate the prerequisites, and describe the I2DP of VDDLM in Figure 3. The protocol terminates with no output if any server fails the proofs for either LPRF or C_{Lap}, which collectively act as Π_{data}. The protocol proceeds as follows:

- In Line 1, the verifier executes Π_{data} with each client and identifies the subset of clients J^* remaining in the protocol. Meanwhile, each server fuses the random seed σ_i' of LPRF using σ_i and the public coin pc_i decided by the verifier.
- In Line 2, both parties compute ψ_i', a valid commitment of the aggregated random seed σ_i'.
- In Line 3, the random bits $\mathbf{z}_i$ are generated from σ_i', which are committed in Lines 4 and 5, and proved in Line 6.
- In Lines 7 and 8, each server aggregates the secret shares $[\![\mathbf{x}_j]\!]_i$ and the commitment randomnesses $[\![\mathbf{r}_j]\!]_i$ from the clients that have passed Π_{data} (i.e., $j \in J^*$), as $[\![\mathbf{x}]\!]_i$ and $[\![\mathbf{r}]\!]_i$, respectively. Also, both parties aggregate the commitments of these values in Line 9.
- In Line 10, the secret share of each client is perturbed using the Laplacian mechanism realized by $C_{\mathsf{Lap}}(\cdot)$. Each server must add its own copy of the noise due to the potential risk of collusion between the verifier and some other clients.
- In Line 11, the correctness $[\![\mathbf{y}]\!]_i$ is established using the sub-protocol of the proof over the arithmetic circuit of $C_{\mathsf{Lap}}(\cdot)$, before V runs IdUsable and Aggr.

4.3 Analysis of VDDLM

Security and Privacy. The truncations and approximations in Algorithm 1 may cause additional privacy leakages, and need to be considered in the privacy analysis. However, unlike the previous analysis on a similar circuit for MPC [91], our analysis does not take the detour via the statistical distance from the original discrete Laplacian distribution, which enabled us to provide a tight bound on the privacy cost.

Theorem 4.2. *Given query* $q : \mathcal{D} \to \mathbb{Z}$ *with sensitivity* 1, *the modified discrete Laplacian mechanism as in Algorithm 1,* $\mathcal{M}(D) := q(D) + C_{\mathsf{Lap}}$ *satisfies* (ε, δ)-*DP such that*

$$\varepsilon = \log\left(\max\{a_z, a_0, \ldots, a_{\gamma-1}\}\right), \delta = \frac{2p_z}{(1 - p_z)\prod_{i=0}^{\gamma-1}(1 - p_i)},$$
(11)

where $a_z = \frac{2p_z}{(1-p_z)\prod_{i=0}^{\gamma-1}(1-p_i)}$ *and* $a_i = \frac{1-p_z}{2}\prod_{i=0}^{\gamma-1} p_i$ *for each* $i \in [\gamma]$. *Moreover, the values of* (ε, δ) *are tight.*

Proof Sketch. The privacy parameter ε corresponds to the ratio between the two probabilities that Algorithm 1 outputs two consecutive integers. As the support of this algorithm is no longer the entire $\mathbb{Z}$, δ accounts for the marginal case where the output of $\mathcal{M}(D)$ is out of the support of $\mathcal{M}(D')$ for neighboring databases D and D'. $\qquad\square$

Table 1: Utility (expected **L1** error) and overhead (each server's total running time, total communication, and verifier's total running time) comparison between VDBM [14] and VDDLM during the execution of Π_{comp}.

	L1	Server	Comm. & Verifier
VDBM	$\Theta\left(\frac{\sqrt{n_{\mathsf{Ser}}}d}{\varepsilon}\sqrt{\log\frac{1}{\delta}}\right)$	$\Theta\left(\frac{d}{\varepsilon^2}\log\frac{1}{\delta}\right)$	$\Theta\left(\frac{n_{\mathsf{Ser}}d}{\varepsilon^2}\log\frac{1}{\delta}\right)$
VDDLM	$\Theta\left(\frac{\sqrt{n_{\mathsf{Ser}}}d}{\varepsilon}\right)$	$\Theta(dn_{\mathsf{Lap}})$	$\Theta(n_{\mathsf{Ser}}n_{\mathsf{Lap}})$

The full proof and the generalized version of Theorem 4.2 for queries with generic sensitivities are in Appendix C.3 and C.4, respectively. The DP guarantee extends to the distributed settings when at least one server is fully honest:

Lemma 4.3. *The server function of VDDLM is $(n_{\mathsf{Ser}}-1,\varepsilon,\delta)$-DDP for ε and δ defined in Theorem 4.2.*

By utilizing the same IdUsable and Aggr as VDBM introduced in Example 3.13 and complete, knowledge sound, and ZK Π_{data} and Π_{comp}, the desired security and privacy guarantees can be achieved as stated in Proposition 4.4.

Proposition 4.4. *VDDLM is 0-complete, knowledge sound, and $(n_{\mathsf{Ser}}-1,\varepsilon,\delta)$-VDDP for ε and δ defined in Theorem 4.2.*

Utility and Overhead. The utilities of the modified discrete Laplacian mechanism can be measured by the L1 error, as stated in Theorem 4.5. Given that the modified discrete Laplacian mechanism is designed to closely approximate its original version, the L1 error remains $\Theta\left(\frac{1}{\varepsilon}\right)$ [6,48], and for n_{Ser} servers and d dimensions, the total expected L1 error is $\Theta\left(\frac{\sqrt{n_{\mathsf{Ser}}}d}{\varepsilon}\right)$. On the other hand, each server's overhead is proportional to the product of the total number of fair coins generated n_{Lap} and the dimensionality d. However, $n_{\mathsf{Lap}} = \nu_z + 1 + \sum_{i=0}^{\gamma-1}\nu_i$ results from the range and precision parameters in Algorithm 1, and does not have direct dependence on the privacy parameters (ε,δ).

Theorem 4.5. *In the modified discrete Laplacian mechanism as in Algorithm 1, the expected L1 error is given by*
$$\mathbb{E}\left|\mathcal{M}(D) - q(D)\right| = (1 - p_z)\left(1 + \sum_{i=0}^{\gamma-1} 2^i p_i\right).$$

Comaprison with VDBM. Given (ε,δ)-DP achieved in the central DP setting, both VDBM and VDDLM achieve 0-completeness, knowledge soundness, and $(n_{\mathsf{Ser}}-1,\varepsilon,\delta)$-VDDP. Note that for a smaller number of colluding servers $n' \le n_{\mathsf{Ser}}-1$, smaller ε and δ can also be achieved. However, as semi-honest servers cannot be detected, for the most robust privacy guarantee, we focus on the case when at most $n_{\mathsf{Ser}}-1$ servers may collude. Therefore, their utility and overhead can be compared fairly under the same privacy parameters,

Figure 4: Utility and overhead comparison between VDBM (a fixed $\delta = 10^{-10}$) and VDDLM ($\delta < 10^{-10}$).

as illustrated in Table 1[1]. Moreover, we plot the L1 error and number of coins under different εs under the central-DP and uni-dimensional case. It can be observed that VDDLM achieves a 5-10x reduction of error compared with the binomial mechanism. Moreover, the server's overhead grows significantly slower under VDDLM, making the scenarios with smaller εs (e.g., $\varepsilon = 10^{-3}$) much more feasible.

5 VRR: Verifiable Randomized Response

In this section, we construct the verifiable randomized response (VRR) scheme. As a special case of distributed DP, in this local DP mechanism with a pure DP guarantee, a client also takes the role of the only server that executes the randomized computation on its output. Therefore, we aim at achieving 0-completeness as well as $(0,\varepsilon,0)$-VDDP (the client is accepted and its privacy is preserved as long as it and its own server perform the computation correctly), in addition to knowledge soundness. Due to the low interference among different clients, we omit their indices in this section.

5.1 Construction of RD for VRR

We begin by constructing an RD that is compatible with the cryptographic primitives applied while adhering to the original protocol of RR as introduced in Section 2.1. First, we map the input space $[K]$ to a multiplicative cyclic subgroup X of a prime order finite field $\mathbb{F}$, where X has order K and has a generator χ, such that $i \mapsto \chi^i$ forms an isomorphism between $[K]$ and X. Therefore, by Equation (3), given any input $x \in X$ held by a client, it is supposed to submit $y = x \cdot \chi^k$ with probability p_k for any $k \in [K]$.

Moreover, to represent the probability distribution defined by p_ks, we utilize another cyclic multiplicative subgroup Ω with generator ω, and construct a cyclic subgroup such that $p_k = \frac{A_k}{|\Omega|}$ for each $0 \le k \le K-1$, where all A_ks are integers. Therefore, there exists a degree-$(|\Omega|-1)$ polynomial F such that the multiset $\left\{F(\omega^i) : i \in [|\Omega|]\right\}$ has A_k copies of χ^k for

---

[1]Utility and overhead of VDBM is analyzed in details in Appendix C.2. The instantiation of Π_{data} depends on the exact scenario and is interchangeable between the two mechanisms.

Cli $(x, i_\sigma, r_x, r_\sigma,$ com$, \psi, i_{pc}; pp)$		V $($com$, \psi, i_{pc}; pp)$
1 : $\quad y \leftarrow xF\left(\omega^{i_\sigma + i_{pc}}\right)$	$\xrightarrow{\quad y \quad}$	$y^K \overset{?}{=} 1$
2 : $\quad i_t \leftarrow i_\sigma + i_{pc}$		
3 : $\quad r_t \leftarrow r_\sigma \omega^{i_{pc}}$		
4 : $\quad$com$_t \leftarrow \psi^{\omega^{i_{pc}}}$		com$_t \leftarrow \psi^{\omega^{i_{pc}}}$
5 : $\quad z = yx^{-1}, r_z \leftarrow\!\!{\scriptstyle\$}\, \mathbb{F}$		
6 : $\quad$com$_z \leftarrow g^z h^{r_z}$	$\xrightarrow{\quad \text{com}_z \quad}$	com$_z$
7 : $\quad \alpha$	$\xleftarrow{\quad \alpha \quad}$	$\alpha \leftarrow\!\!{\scriptstyle\$}\, \mathbb{F}$
8 : $\quad F_\alpha \leftarrow F + \alpha F_\Omega$		$g_\alpha \leftarrow g^{F(\tau)} \cdot \left(g^{F_\Omega(\tau)}\right)^\alpha$
9 : $\quad$P$_{\text{EvSc}}\left(z, r_z, \omega^{i_t}, r_t, F_\alpha,\right.$ $\left.\text{com}_z, \text{com}_t; pp\right)$	$\overset{\pi_{\text{EvSc}}}{\longleftrightarrow}$	V$_{\text{EvSc}}\,($com$_z,$ com$_t,$ $g_\alpha; pp)$
10 : $\quad$P$_{\text{Prod}}\left(y, z, x, 0, r_z, r_x,\right.$ $\left.g^y, \text{com}_z, \text{com}; pp\right)$	$\overset{\pi_{\text{Prod}}}{\longleftrightarrow}$	V$_{\text{Prod}}\,(g^y, \text{com}_z, \text{com}; pp)$

Figure 5: The instantiation of I2DP in VRR.

each $k \in [K]$. That is, the probability space is quantized with a precision of $\frac{1}{|\Omega|}$. Moreover, with $i \leftarrow\!\!{\scriptstyle\$}\, [|\Omega|]$, the probability that $F\left(\omega^i\right) = \chi^k$ is p_k for each $k \in [K]$. Therefore, $xF\left(\omega^i\right)$ has exactly the same distribution as the client's output y.

We further note that the sampling of $xF\left(\omega^i\right)$ where $i \leftarrow\!\!{\scriptstyle\$}\, [|\Omega|]$ can be decomposed between the client and verifier, such that by having $\mathcal{P}_\sigma$ (under the control of the client) and $\mathcal{P}_{pc}$ (under the control of the verifier) as the uniform distribution over Ω, a valid RD can be constructed as

$$f(x, \sigma, pc) := xF\left(\sigma \cdot pc\right). \qquad (12)$$

For simplicity, we also use the indices i_σ and i_{pc} to identify σ and pc, such that $\sigma = \omega^{i_\sigma}$ and pc $= \omega^{i_{pc}}$, where i_σ and i_{pc} can be uniformly sampled from $[|\Omega|]$.

Therefore, in Section 5.2, we focus on establishing the proof protocol of the validity of data (i.e., $x \in \mathcal{X}$) and the correctness of the computation over f, as the components of Π_{data} and Π_{comp} described in Figure 2 of Section 3.4.

5.2 Design of the VRR Protocol

To instantiate Π_{data} and Π_{comp} as in Figure 2, both executed between the client (which acts as the server that handles its own data) and the verifier, we first identify the arithmetic relations to be proved.

Note that for Π_{comp}, in addition to the correctness over Equation (12), it is also necessary to verify that $\sigma \in \Omega$, which is equivalent to $\sigma \cdot pc \in \Omega$ since pc $\in \Omega$. On the other hand, for Π_{data}, the client needs to prove $x \in \mathcal{X}$. Since $F(\sigma \cdot pc) \in \mathcal{X}$ for any $\sigma \cdot pc \in \Omega$, $x \in \mathcal{X}$ iff $y = xF(\sigma \cdot pc) \in \mathcal{X}$. Therefore, it

Table 2: Overhead (each client's total running time, total communication, and verifier's total running time) comparison between KCY21 [61] and our solution.

	Client	Comm.& Verifier
KCY21	$\Theta(|\Omega|K)$	$\Theta(n_{\text{Cli}}|\Omega|K)$
Ours	$\Theta(|\Omega|)$	$\Theta(n_{\text{Cli}})$

suffices for the verifier to directly check that $y \in \mathcal{X}$. By elementary algebra, the membership in the cyclic multiplicative subgroups $\mathcal{X}$ and Ω are equivalent to having the vanishing polynomials, $F_{\mathcal{X}}(X) = X^K - 1$ and $F_\Omega(X) = X^{|\Omega|} - 1$, evaluating to 0. Therefore, the I2DP between a client Cli and a verifier V, as described in Figure 5, proves that

$$F_{\mathcal{X}}(y) = 0 \land y = f(x, \sigma, pc) \land F_\Omega(\sigma \cdot pc) = 0. \qquad (13)$$

The interactive protocol begins after the client commits to its data x and obfuscation i_σ, as com $= g^x h^{r_x}$ and $\psi = g^{\omega^{i_\sigma}} h^{r_\sigma}$ with randomness $r_x, r_\sigma \leftarrow\!\!{\scriptstyle\$}\, \mathbb{F}$, and the verifier decides on the public coin i_{pc}. The protocol proceeds as follows:

- In Line 1, the client computes the output y as in Equation (12), and sends y to the verifier. The verifier immediately checks if $y^K = 1$, i.e., $F_{\mathcal{X}}(y) = 0$.
- In Lines 2 to 4, the client fuses the obfuscation i_t and public coin i_{pc} as $i_t \leftarrow i_\sigma + i_{pc}$, or equivalently, $t \leftarrow \omega^{i_\sigma} \omega^{i_{pc}} = \sigma \cdot pc$. Both parties compute com$_t$, a valid commitment of t using the homomorphic property of the commitment scheme.
- In Lines 5 and 6, the client computes and commits to the intermediate value $z = yx^{-1} = F(\sigma \cdot pc)$, and sends the commitment com$_z$ to the verifier.
- In Lines 7 to 9, the verifier chooses $\alpha \leftarrow\!\!{\scriptstyle\$}\, \mathbb{F}$ and transmits it to the client, and asks the client to prove that $F(t) + \alpha F_\Omega(t) = z$. By the Schwartz-Zippel Lemma [82, 94], this is equivalent to $F(t) = z \land F_\Omega(t) = 0$ with overwhelming probability over the randomness of α. Here, P$_{\text{EvSc}} \leftrightarrow$ V$_{\text{EvSc}}$ is a complete, knowledge-sound, and zero-knowledge interactive proof of the evaluation of a public polynomial with secret input and output (see Appendix D.1).
- In Line 10, the client proves to the verifier that $y = zx$ (i.e., $z = yx^{-1}$) as elements of $\mathbb{F}$ bound by their commitments, via a folklore complete, knowledge-sound, and zero-knowledge interactive proof P$_{\text{Prod}} \leftrightarrow$ V$_{\text{Prod}}$ [62, 67].

5.3 Analysis of VRR

Security and Privacy. The security and privacy guarantees of VRR are formalized in Theorem 5.1. The completeness and knowledge soundness are inherited from the subroutines in Figure 5. Meanwhile, without the verifications, the only information that a prover gets from a client (and its own server) is the output y, which is $(\varepsilon, 0)$-DP and therefore, $(0, \varepsilon, 0)$-DDP. Therefore, the ZK subroutines make VRR $(0, \varepsilon, 0)$-VDDP.

Table 3: Comparison between VDDLM and VDBM. The hyperparameters (e.g., n_b) of VDBM are computed from the desired $(\varepsilon^*, \delta^* = 10^{-10})$. The exact (ε, δ) are computed directly from the configuration of each experiment using Theorem 4.2. d: dimension of input; T_{Ser}: average computing and proving time of servers; C: communication cost between each server and client; T_V: the verifier's running time to validate each server's output; L1: the L1 error of the final output. The figures marked by $\sim$ are estimated, and therefore only kept to 1 significant figure.

d	ε^*	VDBM [14]				VDDLM (Ours)					
		T_{Ser} (s)	C (MB)	T_V (s)	L1	ε	δ	T_{Ser} (s)	C (MB)	T_V (s)	L1
16	10^{-3}	$\sim 7 \times 10^6$	1.6×10^6	$\sim 5 \times 10^6$	1.7×10^5	9.64×10^{-4}	8.26×10^{-11}	17	2.2	8.0	2.8×10^4
	10^{-2}	6.9×10^4	1.6×10^4	5.0×10^4	1.4×10^4	9.71×10^{-3}	1.68×10^{-11}	13	1.7	6.1	2.1×10^3
	10^{-1}	7.1×10^2	1.6×10^2	5.3×10^2	1.4×10^3	9.89×10^{-2}	1.30×10^{-12}	10	1.3	4.8	2.7×10^2
	10^0	8.8	1.6	6.6	1.3×10^2	9.68×10^{-1}	2.72×10^{-14}	6.9	0.95	3.5	24
64	10^{-3}	$\sim 3 \times 10^7$	6.5×10^6	$\sim 2 \times 10^7$	5.2×10^5	9.64×10^{-4}	8.26×10^{-11}	63	2.2	7.9	1.0×10^5
	10^{-2}	$\sim 3 \times 10^5$	6.5×10^4	$\sim 2 \times 10^5$	5.9×10^4	9.71×10^{-3}	1.68×10^{-11}	48	1.7	6.0	1.1×10^4
	10^{-1}	2.8×10^3	6.5×10^2	2.1×10^3	5.1×10^3	9.89×10^{-2}	1.30×10^{-12}	37	1.3	4.7	1.0×10^3
	10^0	33	6.5	26	5.4×10^2	9.68×10^{-1}	2.72×10^{-14}	26	0.95	3.4	97
256	10^{-3}	$\sim 1 \times 10^8$	2.6×10^7	$\sim 8 \times 10^7$	2.2×10^6	9.64×10^{-4}	8.26×10^{-11}	2.5×10^2	2.2	8.1	3.2×10^5
	10^{-2}	$\sim 1 \times 10^6$	2.6×10^5	$\sim 8 \times 10^5$	2.1×10^5	9.71×10^{-3}	1.68×10^{-11}	1.9×10^2	1.7	6.2	3.4×10^4
	10^{-1}	1.1×10^4	2.6×10^3	8.0×10^3	2.1×10^4	9.89×10^{-2}	1.30×10^{-12}	1.5×10^2	1.3	4.8	3.6×10^3
	10^0	1.2×10^2	26	84	2.3×10^3	9.68×10^{-1}	2.72×10^{-14}	1.2×10^2	0.97	3.4	3.0×10^2
1024	10^{-3}	$\sim 4 \times 10^8$	1.0×10^8	$\sim 3 \times 10^8$	8.7×10^6	9.64×10^{-4}	8.26×10^{-11}	1.0×10^3	2.2	8.9	1.4×10^6
	10^{-2}	$\sim 4 \times 10^6$	1.0×10^6	$\sim 3 \times 10^6$	8.9×10^5	9.71×10^{-3}	1.68×10^{-11}	7.7×10^2	1.7	6.8	1.4×10^5
	10^{-1}	4.2×10^4	1.0×10^4	3.2×10^4	9.2×10^4	9.89×10^{-2}	1.30×10^{-12}	7.0×10^2	0.4	5.3	1.4×10^4
	10^0	4.3×10^2	1.0×10^2	3.2×10^2	8.7×10^3	9.68×10^{-1}	2.72×10^{-14}	4.3×10^2	0.98	3.6	1.3×10^3

Theorem 5.1. *The VRR protocol is 0-complete, knowledge-sound, and $(0, \varepsilon, 0)$-VDDP.*

Utility and Overhead. As explained in Section 5.1, the output distribution of the server function of VRR is exactly the same as the original version of RR without the additional verifications, given the same set of parameters A_ks and therefore p_ks. Therefore, VRR can achieve the same utility. Moreover, due to the crafted quantization of the probability space of RR, and the compatible protocol design using highly efficient cryptographic primitives, our solution to VRR enjoys a significant reduction of overhead compared with the previous solution (KCY21, [61]), as shown in Table 2. In particular, the total running time of each client is linear in $|\Omega|$, and does not depend on the number of classes K or the privacy parameter ε. Moreover, the communication and verifier's overhead is only constant per client, significantly improving the scalability.

6 Experiments

We implement VDDLM and VRR in C++ using the MCL cryptography library [73] and report their overheads and utilities in this section. We use the BLS12-381 curve, one of the most prevalent elliptic curves in modern ZKP systems, which provides 128-bit security [21]. The experiments were run with 32 cores allocated from an Intel Xeon Platinum 8358 CPU @ 2.60GHz and 231 GB of memory.

Experiments on VDDLM. In complement to the advantages analyzed from a theoretical and numerical perspective in Section 4.3, we present the experimental comparison between VDDLM and VDBM. The comparisons focus on Π_{comp}, the major difference between VDDLM and VDBM, as the instantiation of Π_{data} in VDDLM (Line 1 of Figure 3) is specific to the exact scenario and a valid Π_{data} for VDDLM would also be valid for VDBM under the same scenario. As the original implementation of VDBM focuses on a one-dimensional setting, we execute it element-wise to produce the experimental figures under the multi-dimensional setting.

As shown in Table 3[2], VDDLM achieves a remarkable reduction of overhead from almost all perspectives. The improvement is particularly significant when the dimension of the problem is larger (e.g., 1024), and only a small (e.g., $\varepsilon = 10^{-3}$) privacy budget is available: the server's and verifier's running times are boosted by 4×10^5 and 3×10^7 times, respectively, and the communication cost is compressed by a factor of 4×10^7. Moreover, aligned with Figure 4, VDDLM only suffers 0.1x to 0.2x numerical error compared with VDBM. Therefore, VDDLM greatly improves the practicality and scalability of the VDDP mechanisms.

Experiments on VRR. We compare our new VRR mechanism in Section 5 with the previous solution [61]. The decisive factor of the overhead is the quantization precision of the probability space, i.e., $|\Omega|$. When using the same quantization of the probability space, the same privacy cost and utility. However, as introduced in Section 5.3, our solution shows an asymptotic improvement in both the client and verifier's

---

[2] T_{Ser}, C, T_V has been amortized and theoretically irrelevant to n_{Ser}, and hence does not significantly vary with n_{Ser}. Therefore, we fix $n_{\mathsf{Ser}} = 2$ following VDBM [14].

Table 4: Per-client overhead of verifiable K-class randomized response using the previous (KCY21, [61]) and our solutions. $|\Omega|$: probability space quantizaiton accuracy; T_{Cli}: running time of each client; C: communication between each client and verifier; T_V: the verifier's running time for each client.

$\lvert\Omega\rvert$	K	Solution	T_{Cli} (s)	C (kB)	T_V (s)
64	8	KCY21	0.29	41	0.17
		Ours	0.049	2.0	8.1×10^{-3}
	32	KCY21	1.0	1.6×10^2	0.62
		Ours	0.051	2.0	8.0×10^{-3}
256	8	KCY21	1.0	1.6×10^2	0.63
		Ours	0.18	2.0	8.1×10^{-3}
	32	KCY21	4.1	6.6×10^2	2.6
		Ours	0.18	2.0	8.1×10^{-3}
	128	KCY21	17	2.6×10^3	11
		Ours	0.18	2.0	8.2×10^{-3}
1024	8	KCY21	4.3	6.6×10^2	2.5
		Ours	0.72	2.0	8.1×10^{-3}
	32	KCY21	17	2.6×10^3	11
		Ours	0.77	2.0	8.1×10^{-3}
	128	KCY21	72	1.0×10^4	45
		Ours	0.74	2.0	8.1×10^{-3}

overhead and the communication cost.

Table 4 demonstrates the improvement achieved by our novel solution to VRR under different configurations of quantizing the probability space. It can be observed that using our solution, the communication and running time are constant and fixed at approximately 2.0kB and 8ms per client, which significantly reduces the overhead for the verifier, enabling it to organize data collection from a significantly larger population of clients. Furthermore, thanks to the improvement of each client's overhead from $O(|\Omega|K)$ to $O(|\Omega|)$, the clients also enjoy a 5x to 100x speedup under different configurations. We also conducted a runtime evaluation for each component of VRR (shown in Appendix E, Figure 9), which shows that the generation of the proofs executed by the clients is the most time-consuming component.

7 Related Work

Pioneering steps in verifiable executions of differentially private mechanisms involve cryptographic proofs on the correctness of deterministic fundamental computation steps in differentially private database systems like **VFuzz** [74] and **DPrio** [63]. More recent advancements have shifted their focus to the correct sampling from noise distributions, including randomized response (**KCY21**, [61]), floating-point Gaussian mechanisms (**STC+24**, [84]), and binomial mechanisms (**VDBM**, [14]). More broadly, other studies on secure computation for randomness generation [3, 20] and differential privacy [10, 29], with multi-party computation

Table 5: Comparison of security and privacy models with previous work on MPCs of DP mechanisms (**MPC-DP**, [17, 18, 24, 36, 91]) and verifiable executions of DP mechanisms [14, 61, 63, 74, 84]. *VD*, *VC*, *VR*: authenticity of data, correct deterministic computation, or correct sampling from the prescribed random distributions is verifiable (to an external data analyst); *N*: resilience against numerical issues of DP due to compatibility with discrete cryptographic primitives; *CSV*: client-server-verifier model; *E2EDP*: end-to-end DP guarantee, incorporating additional leakages from the proof.

	VD	VC	VR	N	CSV	E2EDP
MPC-DP	✗	✗	✗	✔	✗	N/A
VFuzz [74]	✔	✔	✗	✗	✗	✗
DPrio [63]	✔	✗	✗	✔	✔	✗
KCY21 [61]	✔	✔	✔	✔	✗	✗
STC+24 [84]	✔	✔	✔	✗	✗	✗
VDBM [14]	✔	✔	✔	✔	✔	✗
Ours	✔	✔	✔	✔	✔	✔

(MPC) [17, 18, 24, 34, 36, 44, 91], have laid the foundation for the secure computation of DP mechanisms, especially in distributed settings. However, despite the similarities in multiple aspects, they do not cover the scenario when an external data analyst needs to verify the authenticity of the data and correctness of computation, especially the randomness involved. We compare this study's security and privacy models with the aforementioned studies in Table 5. We discuss additional related work in Appendix F.

8 Conclusion

In this study, we have rigorously defined verifiable distributed differential privacy and systematically explored its relationship with zero-knowledge. We further proved the feasibility of VDDP by providing two concrete instantiations, VDDLM and VRR, which have significantly outperformed the previous state-of-the-art solutions in both utility and efficiency. Besides these two protocols, we also discuss the feasibility and challenges of extending VDDLM to the discrete Gaussian mechanism [23] in Appendix C.5. Based on this study, we raise three open questions: **1)** constructions of VDDP with non-zero-knowledge proofs while satisfying Theorem 3.12, such that the overhead can be reduced asymptotically on either the client, server, or verifier's running time, or the communication cost; **2)** alternative sampling methods for the discrete Gaussian mechanism that are compatible with existing or novel proof protocols, enabling efficient constructions of verifiable distributed discrete Gaussian mechanisms; **3)** extensions of VDDP to more complex settings, e.g., where extensive communications among the servers are involved and need to be audited by the verifier.

Open Science

Upon publication, the implementations of VDDLM and VRR will be available at https://github.com/jvhs0706/vddp, where the scripts for reproducing the experimental results in Section 6 will also be included.

Ethics Considerations

This study inherently enhances data privacy. We identify the clients, servers, and verifier (data analyst) as the stakeholders involved in this study. By the nature of this study, the unethical behaviors of malicious clients and servers submitting invalid data and deviating from the prescribed mechanisms are exposed to the verifier, due to the requirement of soundness (Definition 3.8). Moreover, this study also ensures that the honest clients and servers can pass the verification conducted by the verifier. We are aware that the interference among the clients and servers may cause honest clients' data not to be included in the computation. However, this study has striven to limit the occurrence of this undesired exclusion via the requirement of θ-completeness (Definition 3.6).

This study has allowed for verifiable DP with additional privacy leakages and raised the open question of trading privacy protection for better overhead. We acknowledge that this suggested future direction may result in further privacy leakage than the executed DP mechanisms. However, in response to this, we have preemptively bounded the end-to-end privacy leakage of the entire protocol with Theorem 3.12.

References

[1] Naman Agarwal, Peter Kairouz, and Ziyu Liu. The skellam mechanism for differentially private federated learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 5052–5064, 2021.

[2] Naman Agarwal, Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed SGD. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7575–7586, 2018.

[3] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic randomized response techniques. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 425–438. Springer, 2004.

[4] Balamurugan Anandan and Chris Clifton. Laplace noise generation for two-party computational differential privacy. In Ali A. Ghorbani, Vicenç Torra, Hüseyin Hisil, Ali Miri, Ahmet Koltuksuz, Jie Zhang, Murat Sensoy, Joaquín García-Alfaro, and Ibrahim Zincir, editors, *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pages 54–61. IEEE Computer Society, 2015.

[5] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[6] Victor Balcer and Salil P. Vadhan. Differential privacy on finite computers. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 43:1–43:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[7] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014.*

Proceedings, volume 8642 of *Lecture Notes in Computer Science*, pages 175–196. Springer, 2014.

[8] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 451–468. Springer, 2008.

[9] James Bell, Adrià Gascón, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. Distributed, private, sparse histograms in the two-server model. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 307–321. ACM, 2022.

[10] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1253–1269. ACM, 2020.

[11] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[12] Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. Cryptanalysis of the legendre PRF and generalizations. *IACR Trans. Symmetric Cryptol.*, 2020(1):313–330, 2020.

[13] Ward Beullens and Cyprien Delpech de Saint Guilhem. Legroast: Efficient post-quantum signatures from the legendre PRF. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 130–150. Springer, 2020.

[14] Ari Biswas and Graham Cormode. Interactive proofs for differentially private counting. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security,*

CCS 2023, Copenhagen, Denmark, November 26-30, 2023, pages 1919–1933. ACM, 2023.

[15] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge, MARK 1979, New York, NY, USA, June 4-7, 1979*, pages 313–318. IEEE, 1979.

[16] Andrew J. Blumberg, Justin Thaler, Victor Vu, and Michael Walfish. Verifiable computation using multiple provers. *IACR Cryptol. ePrint Arch.*, page 846, 2014.

[17] Jonas Böhler and Florian Kerschbaum. Secure multiparty computation of differentially private median. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 2147–2164. USENIX Association, 2020.

[18] Jonas Böhler and Florian Kerschbaum. Secure multiparty computation of differentially private heavy hitters. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2361–2377. ACM, 2021.

[19] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *CoRR*, abs/1611.04482, 2016.

[20] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 762–776. IEEE, 2021.

[21] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[22] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 635–658, 2016.

[23] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[24] Jeffrey Champion, Abhi Shelat, and Jonathan R. Ullman. Securely sampling biased coins with applications to differential privacy. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 603–614. ACM, 2019.

[25] Ian Chang, Katerina Sotiraki, Weikeng Chen, Murat Kantarcioglu, and Raluca A. Popa. HOLMES: efficient distribution testing for secure collaborative learning. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 4823–4840. USENIX Association, 2023.

[26] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 499–530. Springer, 2023.

[27] Wei-Ning Chen, Ayfer Özgür, and Peter Kairouz. The poisson binomial mechanism for unbiased federated learning with secure aggregation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 3490–3506. PMLR, 2022.

[28] Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 375–403. Springer, 2019.

[29] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Cryptε: Crypto-assisted differential privacy on untrusted servers. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 603–619. ACM, 2020.

[30] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 259–282. USENIX Association, 2017.

[31] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[32] Ivan Damgård. On the randomness of legendre and jacobi sequences. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 163–172. Springer, 1988.

[33] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430. Springer, 2000.

[34] Alex Davidson, Peter Snyder, E. B. Quirk, Joseph Genereux, and Benjamin Livshits. STAR: distributed secret sharing for private threshold aggregation reporting. *CoRR*, abs/2109.10074, 2021.

[35] Cynthia Dwork. Differential privacy in distributed environments: An overview and open questions. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, page 5. ACM, 2021.

[36] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[37] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[38] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[39] Tariq Elahi, George Danezis, and Ian Goldberg. Privex: Private collection of traffic statistics for anonymous communication networks. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 1068–1079. ACM, 2014.

[40] Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida. Efficient noise generation to achieve differential privacy with applications to secure multiparty computation. In Nikita Borisov and Claudia Díaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2021.

[41] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 211–222. ACM, 2003.

[42] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[43] Jie Fu, Yuan Hong, Xinpeng Ling, Leixia Wang, Xun Ran, Zhiyu Sun, Wendy Hui Wang, Zhili Chen, and Yang Cao. Differentially private federated learning: A systematic review. *CoRR*, abs/2405.08299, 2024.

[44] Yucheng Fu and Tianhao Wang. Benchmarking secure sampling protocols for differential privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 318–332. ACM, 2024.

[45] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, page 953, 2019.

[46] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008.

[47] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[48] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM J. Comput.*, 41(6):1673–1693, 2012.

[49] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer, 1986.

[50] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985.

[51] Slawomir Goryczka and Li Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE Trans. Dependable Secur. Comput.*, 14(5):463–477, 2017.

[52] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. Mpc-friendly symmetric key primitives. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 430–443. ACM, 2016.

[53] Mikko A. Heikkilä, Eemil Lagerspetz, Samuel Kaski, Kana Shimizu, Sasu Tarkoma, and Antti Honkela. Differentially private bayesian learning on distributed data. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3226–3235, 2017.

[54] Jiankai Jin, Eleanor McMurtry, Benjamin I. P. Rubinstein, and Olga Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 473–488. IEEE, 2022.

[55] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (eddsa). *RFC*, 8032:1–60, 2017.

[56] Peter Kairouz, Kallista A. Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2436–2444. JMLR.org, 2016.

[57] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5201–5212. PMLR, 2021.

[58] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2879–2887, 2014.

[59] Sanket Kanjalkar, Ye Zhang, Shreyas Gandlur, and Andrew Miller. Publicly auditable mpc-as-a-service with succinct verification and universal setup. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, pages 386–411. IEEE, 2021.

[60] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.

[61] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. In Ken Barker and Kambiz Ghazinour, editors, *Data and Applications Security and Privacy XXXV - 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19-20, 2021, Proceedings*, volume 12840 of *Lecture Notes in Computer Science*, pages 43–60. Springer, 2021.

[62] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[63] Dana Keeler, Chelsea Komlo, Emily Lepert, Shannon Veitch, and Xi He. Dprio: Efficient differential privacy with high utility for prio. *Proc. Priv. Enhancing Technol.*, 2023(3):375–390, 2023.

[64] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36. Springer, 2005.

[65] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 1–34. Springer, 2021.

[66] Junxu Liu, Jian Lou, Li Xiong, Jinfei Liu, and Xiaofeng Meng. Cross-silo federated learning with record-level personalized differential privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 303–317. ACM, 2024.

[67] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2009.

[68] Alexander May and Floyd Zweydinger. Legendre PRF (multiple) key attacks and the power of preprocessing. In *35th IEEE Computer Security Foundations Symposium, CSF 2022, Haifa, Israel, August 7-10, 2022*, pages 428–438. IEEE, 2022.

[69] Fredrik Meisingseth and Christian Rechberger. Sok: Computational and distributed differential privacy for MPC. *Proc. Priv. Enhancing Technol.*, 2025(1):420–439, 2025.

[70] Fredrik Meisingseth, Christian Rechberger, and Fabian Schmid. Practical two-party computational differential privacy with active security. *Proc. Priv. Enhancing Technol.*, 2025(1):341–360, 2025.

[71] Ilya Mironov. On significance of the least significant bits for differential privacy. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 650–661. ACM, 2012.

[72] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil P. Vadhan. Computational differential privacy. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2009.

[73] Shigeo Mitsunari. A portable and fast pairing-based cryptography library. https://github.com/herumi/mcl, 2024. Commit hash: 52366af5f5e06d02c9cd2f64ae32541786aebe28.

[74] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. Verifiable differential privacy. In Laurent Réveillère, Tim Harris, and Maurice Herlihy, editors, *Proceedings of the Tenth European Conference on Computer Systems, EuroSys 2015, Bordeaux, France, April 21-24, 2015*, pages 28:1–28:14. ACM, 2015.

[75] Joseph P. Near, David Darais, Chike Abuah, Tim Stevens, Pranav Gaddamadugu, Lun Wang, Neel Somani, Mu Zhang, Nikhil Sharma, Alex Shan, and Dawn Song. Duet: An expressive higher-order language and linear type system for statically enforcing differential privacy. *CoRR*, abs/1909.02481, 2019.

[76] Alex Ozdemir and Dan Boneh. Experimenting with collaborative zk-snarks: Zero-knowledge proofs for distributed secrets. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX*

Security 2022, Boston, MA, USA, August 10-12, 2022, pages 4291–4308. USENIX Association, 2022.

[77] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[78] Thorsten Peinemann, Moritz Kirschte, Joshua Stock, Carlos Cotrini, and Esfandiar Mohammadi. S-BDT: distributed differentially private boosted decision trees. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 288–302. ACM, 2024.

[79] Roie Reshef, Anan Kabaha, Olga Seleznova, and Dana Drachsler-Cohen. Verification of neural networks' local differential classification privacy. In Rayna Dimitrova, Ori Lahav, and Sebastian Wolff, editors, *Verification, Model Checking, and Abstract Interpretation - 25th International Conference, VMCAI 2024, London, United Kingdom, January 15-16, 2024, Proceedings, Part II*, volume 14500 of *Lecture Notes in Computer Science*, pages 98–123. Springer, 2024.

[80] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[81] John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. Parallel random numbers: as easy as 1, 2, 3. In Scott A. Lathrop, Jim Costa, and William Kramer, editors, *Conference on High Performance Computing Networking, Storage and Analysis, SC 2011, Seattle, WA, USA, November 12-18, 2011*, pages 16:1–16:12. ACM, 2011.

[82] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[83] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[84] Ali Shahin Shamsabadi, Gefei Tan, Tudor Cebere, Aurélien Bellet, Hamed Haddadi, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-dpproof: Confidential proof of differentially private training. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[85] Justin Thaler. Proofs, arguments, and zero-knowledge. *Found. Trends Priv. Secur.*, 4(2-4):117–660, 2022.

[86] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 729–745. USENIX Association, 2017.

[87] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. Answering multi-dimensional analytical queries under local differential privacy. In Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 159–176. ACM, 2019.

[88] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 127–143. IEEE Computer Society, 2018.

[89] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. Locally differentially private frequency estimation with consistency. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[90] Yuxin Wang, Zeyu Ding, Daniel Kifer, and Danfeng Zhang. Checkdp: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 919–938. ACM, 2020.

[91] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. Securely sampling discrete gaussian noise for multi-party differential privacy. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 2262–2276. ACM, 2023.

[92] Yu Wei, Jingyu Jia, Yuduo Wu, Changhui Hu, Changyu Dong, Zheli Liu, Xiaofeng Chen, Yun Peng, and Shaowei Wang. Distributed differential privacy via shuffling versus aggregation: A curious study. *IEEE Trans. Inf. Forensics Secur.*, 19:2501–2516, 2024.

Notation	Definition
$x \leftarrow_\$ \mathcal{P}$	x is independently sampled from a distribution $\mathcal{P}$,
$x \leftarrow_\$ S$	the uniform distribution over a set S, or the output
$x \leftarrow_\$ \mathcal{F}(\cdot)$	distribution of a randomized function $\mathcal{F}(\cdot)$
$X \sim \mathcal{P}$	The random variable X has distribution $\mathcal{P}$
$n_{\mathsf{Cli}}, n_{\mathsf{Ser}}$	the total number of clients/servers
$\overrightarrow{\mathsf{Cli}}, \overrightarrow{\mathsf{Ser}}$	the set of potentially malicious clients/servers
Cli_j, D_j	the jth client with sensitive local database D_j
Ser_i	the ith server who computes over the sensitive data
$[\![D_j]\!]_i$	the secret share of D_j transmitted to Ser_i
I_j	the set of servers receiving secret-shares from Cli_j
J_i	the set of clients secret-sharing its data to Ser_i
V	the semi-honest verifier (e.g., data analyst) that is interested in the computation output and therefore verifies its authenticity
$\mathcal{P}_1 \equiv \mathcal{P}_2$	two distributions $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent
$a \overset{?}{=} b$	verifier checks if $a = b$

Table 6: Notations

[93] Shuangqing Xu, Yifeng Zheng, and Zhongyun Hua. Camel: Communication-efficient and maliciously secure federated learning in the shuffle model of differential privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 243–257. ACM, 2024.

[94] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

A Additional Background

We summarize the notations used in this study in Table 6.

A.1 Local Differential Privacy

Definition A.1 (Local Differential Privacy). *A mechanism $\mathcal{M} : X \to \mathcal{Y}$ is (ε, δ)-locally differentially private if for any $x, x' \in X$ and any measurable subset $S \subseteq \mathcal{Y}$,*

$$\Pr[\mathcal{M}(x) \in S] \le e^\varepsilon \Pr[\mathcal{M}(x') \in S] + \delta. \quad (14)$$

A.2 Discrete Gaussian Mechanism

Similar to the discrete Laplacian mechanism, the discrete Gaussian mechanism [23] involves perturbing the result with additive discrete Gaussian noise $\mathcal{N}_{\mathbb{Z}}(\sigma^2)$, where

$$\Pr\left[z \leftarrow_\$ \mathcal{N}_{\mathbb{Z}}(\sigma^2)\right] \propto \exp\left(-\frac{z^2}{2\sigma^2}\right). \quad (15)$$

For any query $q : \mathcal{D} \to \mathbb{Z}$ with sensitivity Δ (i.e., $|q(D) - q(D')| \le \Delta$ for any pair of neighboring databases D and D'), and any $\varepsilon > 0, \delta > 0$, perturbing the output $q(D)$ by the additive noise of $\mathcal{N}_{\mathbb{Z}}\left(\frac{\Delta^2}{\varepsilon^2}\right)$ achieves $\frac{1}{2}\varepsilon^2$-concentrated DP [22] and therefore (ε', δ)-DP, where $\varepsilon' = \frac{1}{2}\varepsilon^2 + \varepsilon \cdot \sqrt{2\log\frac{1}{\delta}}$.

Moreover, for any multi-dimensional query $q : \mathcal{D} \to \mathbb{Z}^d$ where d is the dimensionality of the output, adding i.i.d. discrete Gaussian samples to each dimension of the output, $\mathcal{N}_{\mathbb{Z}^d}\left(\frac{\Delta^2}{\varepsilon^2}\right)$, achieves the same DP guarantee when L2 distances in the output space $\mathbb{Z}^d$ are used to define the sensitivity Δ [22, 23].

A.3 Additional Details of RR

As stated in Theorem A.2, an unbiased estimator of the unperturbed histogram can be efficiently constructed.

Theorem A.2. *Given the histogram that the data analyst receives, $\mathbf{n}' = (n'_0, n'_1, \ldots, n'_{k-1})^\top$, an unbiased estimation of the unperturbed histogram $\hat{\mathbf{n}}$ can be formulated as*

$$\hat{\mathbf{n}} \leftarrow \begin{pmatrix} p_0 & p_{K-1} & \cdots & p_1 \\ p_1 & p_0 & \cdots & p_2 \\ \vdots & \vdots & \vdots & \vdots \\ p_{K-1} & p_{K-2} & \cdots & p_0 \end{pmatrix}^{-1} \mathbf{n}'. \quad (16)$$

Proof of Theorem A.2. Given the unperturbed counts

$$\mathbf{n} = (n_0, n_1, \ldots, n_{K-1}),$$

by Equation (3), for each $k \in [K]$,

$$\mathbb{E}\left[n'_k\right] = \sum_{k'=0}^{K-1} p_{(k-k') \bmod K} \cdot n_{k'}, \quad (17)$$

such that

$$\mathbb{E}\mathbf{n}' = \begin{pmatrix} p_0 & p_{K-1} & \cdots & p_1 \\ p_1 & p_0 & \cdots & p_2 \\ \vdots & \vdots & \vdots & \vdots \\ p_{K-1} & p_{K-2} & \cdots & p_0 \end{pmatrix} \mathbf{n}. \quad (18)$$

Therefore, Equation (16) is an unbiased estimator of $\mathbf{n}$. $\square$

A.4 Additional Details of PRNG

A PRNG is considered cryptographically secure if its output is computationally indistinguishable from true randomness [62], as formalized in Definition A.3:

Definition A.3. *A PRNG $G : S \to R$, where S is the space of random seeds, and R is the space of random outputs, is cryptographically secure if for any PPT adversary $\mathcal{A}$, there exists $\mu(\lambda) \in \mathsf{negl}(\lambda)$, such that*

$$\left| \Pr\left[\begin{array}{c} \mathcal{A}(G(s)) = 1 : \\ s \leftarrow\!\!\$\, S \end{array} \right] - \Pr\left[\begin{array}{c} \mathcal{A}(r) = 1 : \\ r \leftarrow\!\!\$\, R \end{array} \right] \right| \leq \mu(\lambda). \quad (19)$$

A.5 Additional Details of ZKP

The KZG polynomial commitment [60] utilizes pairing, formalized as Definition A.4.

Definition A.4 (Pairing [46, 64]). *Assume cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with prime order q where the discrete log assumptions hold. A **pairing** is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that*

- *(**Bilinear**) $e\left(g_1^a, g_2^b\right) = e\left(g_1, g_2\right)^{ab}$;*
- *(**Non-degenerate**) $e \neq 1$;*
- *(**Computability**) e can be efficiently computed.*

The scheme comprises the following components:
- $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{KZG}}\left(\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, d-1\right)$ sets up the public parameters used for committing to the polynomials. Specifically, for $|\mathbb{F}| = |\mathbb{G}_1| = |\mathbb{G}_2| = q$, it samples $g, h \leftarrow\!\!\$\, \mathbb{G}_1$, $g_2 \leftarrow\!\!\$\, \mathbb{G}_2$ and $\tau \leftarrow\!\!\$\, \mathbb{F}$, and computes

$$\mathsf{pp} := \left(g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^{d-1}}, h, h^\tau, h^{\tau^2}, \ldots, h^{\tau^{d-1}}, g_2, g_2^\tau\right).$$

Note that this operation is of $O(d)$ complexity.
- $\mathsf{com}_F \leftarrow \mathsf{Commit}_{\mathsf{KZG}}\left(F, R; \mathsf{pp}\right)$ computes the commitment of $F_{\leq d-1}[X]$ using randomly sampled $R \leftarrow\!\!\$\, F_{\leq d-1}[X]$, as

$$\mathsf{com}_F := g^{F(\tau)} h^{R(\tau)} = \prod_{j=0}^{d-1} \left(g^{\tau^j}\right)^{F_j} \left(h^{\tau^j}\right)^{R_j}.$$

Note that this operation is of $O(d)$ complexity, while the size of the commitment is $O(1)$.
- $\pi \leftarrow \mathsf{P}_{\mathsf{KZG}}\left(y, x, F, R; \mathsf{pp}\right)$ proves that $y = F(x)$ by showing $(X - x)$ divides $(F(X) - y)$. Specifically,

$$\pi := \left(\rho := R(x), \gamma := g^{\frac{F(\tau)-y}{\tau-x}} h^{\frac{R(\tau)-r}{\tau-x}}\right).$$

Note that this operation is of $O(d)$ complexity, while the size of the proof is $O(1)$.
- $\mathsf{accept}/\mathsf{reject} \leftarrow \mathsf{V}_{\mathsf{KZG}}\left(y, x, \pi, \mathsf{com}; \mathsf{pp}\right)$. Specifically, the verifier checks whether

$$e\left(\mathsf{com}_{F'}, g_2^\tau \cdot g_2^{-x}\right) = e\left(\mathsf{com}_F \cdot g^{-y} \cdot h^{-r}, g_2\right).$$

Note that this operation is of $O(1)$ complexity.

In the absence of the requirement of hiding (e.g., committing to the publicly known polynomial F), the randomness R is set as 0 and omitted in the notations. Moreover, for single-dimensional entities, the commitment scheme (i.e.,the Pedersen commitment scheme [77]) corresponds to the case that $d - 1$, such that $x \in \mathbb{F}$ is committed as $g^x h^r$ where $r \leftarrow\!\!\$\, \mathbb{F}$.

Moreover, the commitment scheme satisfies the following properties:
- (**Binding**) For any pp, any PPT adversary $\mathcal{A}$, and any $\mathsf{com} \in \mathbb{G}$

$$\Pr\left[\begin{array}{c} F_1 \neq F_2 \\ \wedge \mathsf{Commit}_{\mathsf{KZG}}\left(F_1, R_1; \mathsf{pp}\right) = \mathsf{com} \\ \wedge \mathsf{Commit}_{\mathsf{KZG}}\left(F_2, R_2; \mathsf{pp}\right) = \mathsf{com} : \\ F_1, F_2, R_1, R_2 \leftarrow \mathcal{A}\left(\mathsf{com}, F; \mathsf{pp}\right) \end{array} \right] \leq \mathsf{negl}(\lambda),$$

i.e., it is hard for $\mathcal{A}$ to compute two valid polynomials with the same commitment.
- (**Hiding**) There exists a simulator Sim such that for any pp, $\mathsf{Sim}(\mathsf{pp})$ is computationally indistinguishable from

$$\mathsf{Commit}_{\mathsf{KZG}}\left(F, R; \mathsf{pp}\right) : R \leftarrow\!\!\$\, F_{\leq d-1}[X]$$

for any pp and F.

Note that the verifier only has access to com_F, while F and R are kept secret by the prover. The verifier learns no information about F due to the randomness of R.

A.6 Additional Details of Secret Sharing

In additive secret sharing scheme, the secret $x \in \mathbb{F}$ is split into n shares such that the sum of all shares equals the secret. Formally, for n shares, $(\llbracket x \rrbracket_i)_{i \in [n-1]}$ are uniformly randomly chosen from $\mathbb{F}$, and the final share is computed as:

$$\llbracket x \rrbracket_{n-1} = x - \sum_{i=0}^{n-2} \llbracket x \rrbracket_i. \quad (20)$$

This ensures that the sum of all shares equals the original secret x. The threshold t for recovering the secret in this scheme is n, meaning that any n shares can be used to reconstruct the secret, while fewer than n shares are uniformly random and provide no information about x.

The multi-dimensional version of this scheme is straightforwardly Equation (20) applied element-wise, with each element of the secret vector shared independently.

B Details for VDDP (Section 3)

B.1 Adversary's View in I2DP

In this appendix, we formalize the collective view of the adversaries $\mathfrak{C}$ in a I2DP Π as in Figure 6. Note that the adversaries $\mathfrak{C}$ update their collective internal state st upon receiving any additional information from the honest parties. In Lines 1 to 6, the honest parties compute the secret shares and their commitments, such that $\mathfrak{C}$ is transmitted the secret shares from j to all servers in $I_j \backslash H_j, S$, and the corresponding randomness terms R used in the commitment schemes, in addition to the commitments C of the secret share from Cli_j to each server in I_j, which have all been transmitted to the verifier. With additional information of S, R, C, in Lines 7 to 11, $\mathfrak{C}$ determines

$\underline{\text{View}_{\Pi}^{\mathfrak{C}}(D_j; \text{pp})}$

1 : $\left(\llbracket D_j \rrbracket_i\right)_{i \in I_n} \leftarrow\!\!\$ \ \text{SecretShare}(D_j)$

2 : $\llbracket r_j \rrbracket_i \leftarrow\!\!\$ \ \mathcal{P}_r, \forall i \in I_j$

3 : $\llbracket \text{com}_j \rrbracket_i \leftarrow \text{CommitShare}\left(\llbracket D_j \rrbracket_i, \llbracket r_j \rrbracket_i; \text{pp}\right), \forall i \in I_j$

4 : $S := \left(\llbracket D_j \rrbracket_i\right)_{i \in I_j \setminus H_j}$

5 : $R := \left(\llbracket r_j \rrbracket_i\right)_{i \in I_j \setminus H_j}$

6 : $C := \left(\llbracket \text{com}_j \rrbracket_i\right)_{i \in H_j}$

7 : $\left(\llbracket D_{j'} \rrbracket_i, \llbracket r_{j'} \rrbracket_i\right)_{i \in H_j, j' \in J_{H_j} \setminus \{j\}}, \text{st} \leftarrow\!\!\$ \ \mathfrak{C}_1\left(S, R, C; \text{pp}\right)$

8 : **foreach** $i \in H_j, j' \in J_{H_j} \setminus \{j\}$ **do**

9 : $\quad \llbracket \text{com}_{j'} \rrbracket_i \leftarrow \text{CommitShare}\left(\llbracket D_{j'} \rrbracket_i, \llbracket r_{j'} \rrbracket_i; \text{pp}\right)$

10 : **endfor**

11 : $\left(\llbracket \text{com}_{j'} \rrbracket_{i'}\right)_{i \notin H_j, j' \notin J_{H_j}}, \text{st} \leftarrow\!\!\$ \ \mathfrak{C}_2\left(\text{st}; \text{pp}\right)$

12 : **foreach** $i \in H_j$ **do**

13 : $\quad \sigma_i \leftarrow\!\!\$ \ \mathcal{P}_\sigma, \rho_i \leftarrow\!\!\$ \ \mathcal{P}_\rho$

14 : $\quad \psi_i \leftarrow \text{CommitOb}\left(\sigma_i, \rho_i; \text{pp}\right)$

15 : **endfor**

16 : $\Psi := (\psi_i)_{i \in H_j}$

17 : $(\psi_i)_{i \notin H_j}, \text{st} \leftarrow\!\!\$ \ \mathfrak{C}_3\left(\text{st}, \Psi; \text{pp}\right)$

18 : $\overrightarrow{\text{pc}} \leftarrow\!\!\$ \ \mathcal{P}_{\text{pc}}^{\otimes n_{\text{Ser}}}$

19 : $\pi_{\text{data}} \leftarrow\!\!\$ \ \text{tr}\left[\Pi_{\text{data}}\left(\text{Cli}_j, \text{com}_j; \text{pp}\right)\right]$

20 : **foreach** $j' \neq j$ **do**

21 : $\quad b_{j'}^{\text{Cli}}, \text{st} \leftarrow\!\!\$ \ \Pi_{\text{data}}\left(\mathfrak{C}_4(\text{st}, \overrightarrow{\text{pc}}, \pi_{\text{data}}), \text{com}_{j'}; \text{pp}\right)$

22 : **endfor**

23 : $J^* \leftarrow \left\{j' : b_{j'}^{\text{Cli}} = 1\right\}$ $\mathbf{\textit{l}} \ j \in J^*$

24 : **foreach** $i \in H_j$ **do**

25 : $\quad \llbracket D \rrbracket_i \leftarrow \text{AggrShare}\left(\llbracket D_j \rrbracket_i, J^* \cap J_i\right)$

26 : $\quad \llbracket y \rrbracket_i \leftarrow f\left(\llbracket D \rrbracket_i, \sigma_i, \text{pc}_i\right)$

27 : $\quad \pi_{\text{comp},i} \leftarrow\!\!\$ \ \text{tr}\left[\Pi_{\text{comp}}\left(\text{Ser}_i, \llbracket y \rrbracket_i, \llbracket \text{com} \rrbracket_i, \psi_i, \text{pc}_i; \text{pp}\right)\right]$

28 : **endfor**

29 : $Y := (\llbracket y \rrbracket_i)_{i \in H_j}$

30 : $\overrightarrow{\pi}_{\text{comp}} := (\pi_{\text{comp},i})_{i \in H_j}$

31 : **return** $\text{st}, Y, \overrightarrow{\pi}_{\text{comp}}$

Figure 6: Adversaries' view in an I2DP Π.

the secret shares transmitted back to the honest parties, and the commitments of the secret shares from all clients other than Cli_j. Then, in Lines 12 to 16, the servers in H_j compute and commit to the obfuscation factors σ_i, such that $\mathfrak{C}$ has the knowledge of their commitments Ψ. In Line 17, $\mathfrak{C}$ computes the commitments of the obfuscation factors of all servers not in H_j. In Lines 19 and 20 to 22, Cli_j and the other clients conduct Π_{data}, respectively. Since the verifier is semi-honest, Cli_j's proof π_{data} is accepted, and the accepting bits $b_{j'}^{\text{Cli}}$ for

other clients are also correctly determined by the verifier. In Lines 24 to 28, each server in H_j computes f honestly, accompanied with the accepted proof, on the data aggregated over the clients Cli_j and all other accepted clients in J^*. The final view of $\mathfrak{C}$ contains the output of the servers in H_j (denoted Y as in Line 29), the proofs (denoted $\overrightarrow{\pi}_{\text{comp}}$ as in Line 30), as well as the previous internal state st which may carry all the information transmitted from the honest parties.

B.2   Details of VDBM

We first formalize the translation from the central DP guarantee of the binomial mechanism to that of the distributed DP guarantee of VDBM as Lemma B.1. The proof of Lemma B.1 is deferred and merged with that of Lemma 4.3, as in Appendix C.3.

Lemma B.1. *The server function of VDBM is $(n_{\text{Ser}} - 1, \varepsilon, \delta)$-DDP where $\varepsilon = 10\sqrt{\frac{1}{n_b} \ln \frac{2}{\delta}}$ and $\delta = o\left(\frac{1}{n_b}\right)$, when $n_b > 30$ is a constant.*

We then present the modification of Π_{Bin} which violates the zero-knowledge property yet preserves the DP property as described in Theorem 3.12. First, observe that, in the original zero-knowledge version, Π_{Bin} has the same distribution of proof transcripts for $x = 0$ and $x = 1$. By considering 0 and 1 neighboring inputs, this is equivalent to 0-DP. Therefore, we aim at generalizing the original version with privacy parameters $\varepsilon' > 0$.

We present the modified protocol $\Pi_{\text{Bin}}^{K,\xi}$ in Figure 7, where $\mathcal{P}_{K,\xi}$ is defined for any integer $1 \leq K \leq \frac{|\mathbb{F}| - 1}{2}$ and real number $\xi \geq 1$ such that

$$\Pr\left[k \leftarrow\!\!\$ \ \mathcal{P}_{K,\xi}\right] = \begin{cases} \frac{2}{\xi+1} \frac{1}{|\mathbb{F}|} & \text{if } k = 0, 2, 4, \ldots, 2K-2 \\ \frac{2\xi}{\xi+1} \frac{1}{|\mathbb{F}|} & \text{if } k = 1, 3, 5, \ldots, 2K-1 \\ \frac{1}{|\mathbb{F}|} & \text{otherwise} \end{cases}.$$

(21)

Note that $\xi = 1$ corresponds to the original zero-knowledge version, and setting $\xi > 1$ is the only modification made to the original Π_{Bin}.

Lemma B.2. $\Pi_{\text{Bin}}^{K,\xi}$ *satisfies the conditions in Theorem 3.12 with $\varepsilon' = \log \xi$ and $\delta' = 0$.*

Proof of Lemma B.2. The view can be simulated by

$$
\begin{array}{ll}
\mathsf{P}_{\mathsf{Bin}}\left(x=0, r_x, \mathsf{com}_x; g, h\right) & \mathsf{V}_{\mathsf{Bin}}\left(\mathsf{com}_x; g, h\right)
\end{array}
$$

	$\mathsf{P}_{\mathsf{Bin}}\left(x=0, r_x, \mathsf{com}_x; g, h\right)$		$\mathsf{V}_{\mathsf{Bin}}\left(\mathsf{com}_x; g, h\right)$
$1:$	$b, e_1 \leftarrow\!\!\$\ \mathbb{F}, v_1 \leftarrow\!\!\$\ \mathcal{P}_{K,\xi}$		
$2:$	$d_0 \leftarrow h^b, d_1 \leftarrow h^{v_1}\left(\dfrac{c}{g}\right)^{-e_1}$	$\xrightarrow{\ (d_0,d_1)\ }$	(d_0, d_1)
$3:$	$e_0 \leftarrow e - e_1, v_0 = b + e_0 r_x$	$\xleftarrow{\ e\ }$	$e \leftarrow\!\!\$\ \mathbb{F}$
$4:$		$\xrightarrow{\ (v_0,e_0,v_1,e_1)\ }$	$e_0 + e_1 \overset{?}{=} e \wedge \mathsf{com}_x^{e_0} \overset{?}{=} h^{v_0} \wedge \mathsf{com}_x^{e_1} \overset{?}{=} g^{e_1}h^{v_1}$

	$\mathsf{P}_{\mathsf{Bin}}\left(x=1, r_x, \mathsf{com}_x; g, h\right)$		$\mathsf{V}_{\mathsf{Bin}}\left(\mathsf{com}_x; g, h\right)$
$1:$	$b, e_0 \leftarrow\!\!\$\ \mathbb{F}, v_0 \leftarrow\!\!\$\ \mathcal{P}_{K,\xi}$		
$2:$	$d_0 \leftarrow h^{v_0} c^{-e_0}, d_1 \leftarrow h^b$	$\xrightarrow{\ (d_0,d_1)\ }$	(d_0, d_1)
$3:$	$e_1 \leftarrow e - e_0, v_1 = b + e_1 r_x$	$\xleftarrow{\ e\ }$	$e \leftarrow\!\!\$\ \mathbb{F}$
$4:$		$\xrightarrow{\ (v_0,e_0,v_1,e_1)\ }$	$e_0 + e_1 \overset{?}{=} e \wedge \mathsf{com}_x^{e_0} \overset{?}{=} h^{v_0} \wedge \mathsf{com}_x^{e_1} \overset{?}{=} g^{e_1}h^{v_1}$

Figure 7: $\Pi_{\mathsf{Bin}}^{K,\xi}$, protocol for proving the knowledge of $x \in \{0,1\}$ and $r \in \mathbb{F}$ such that $\mathsf{com}_x = g^x h^r$. $\xi = 0$ corresponds to the original zero-knowledge version [31, 33, 85].

$$
\begin{array}{ll}
\multicolumn{2}{l}{\underline{\mathsf{Sim}_{\mathsf{Bin}}^{K,\xi}\left(x, \mathsf{com}_x; g, h\right)}} \\
1: & \textbf{if } x = 1 \textbf{ do} \\
2: & \quad (v_0, e_0, v_1, e_1) \leftarrow\!\!\$\ \mathcal{P}_{K,\xi} \times \mathbb{F} \times \mathbb{F} \times \mathbb{F} \\
3: & \textbf{elseif } x = 0 \textbf{ do} \\
4: & \quad (v_0, e_0, v_1, e_1) \leftarrow\!\!\$\ \mathbb{F} \times \mathbb{F} \times \mathcal{P}_{K,\xi} \times \mathbb{F} \\
5: & \textbf{endif} \\
6: & e \leftarrow e_0 + e_1 \\
7: & d_0 \leftarrow \mathsf{com}_x^{-e_0} \cdot h^{v_0} \\
8: & d_1 \leftarrow \mathsf{com}_x^{-e_1} \cdot g^{e_1} h^{v_1} \\
9: & \textbf{return } v_0, e_0, v_1, e_1, e, d_0, d_1,
\end{array}
$$

which satisfies the condition in Theorem 3.12, with the sampling of (v_0, v_1) as $\mathcal{G}(\cdot)$ and the remaining part as h.

In particular, note that the outputs of $\mathcal{G}(0)$ and $\mathcal{G}(1)$ are the uniform distributions over $\mathbb{F} \times \mathcal{P}_{K,\xi}$ and $\mathcal{P}_{K,\xi} \times \mathbb{F}$ respectively. For any $(a,b) \in \mathbb{F}^2$, the

$$
\frac{\Pr[\mathcal{G}(0) = (a,b)]}{\Pr[\mathcal{G}(1) = (a,b)]} \leq \frac{\frac{1}{|\mathbb{F}|} \cdot \frac{2\xi}{\xi+1} \frac{1}{|\mathbb{F}|}}{\frac{2}{\xi+1} \frac{1}{|\mathbb{F}|} \cdot \frac{1}{|\mathbb{F}|}} = \xi. \tag{22}
$$

Symmetrically, it also holds that $\frac{\Pr[\mathcal{G}(1)=(a,b)]}{\Pr[\mathcal{G}(0)=(a,b)]} \leq \xi$. Therefore, $\mathcal{G}$ is $(\log \xi, 0)$-DP. □

Therefore, the modified protocol still satisfies $(n_{\mathsf{Ser}} - 1, \varepsilon + \log \xi, \delta)$-VDDP. It is also worth noting that the completeness and knowledge soundness are preserved, as the modification only involves the shifting of the sampling distribution of either v_0 or v_1.

B.3 Deferred Proofs

In this appendix, we present the missing proofs in Section 3.

Proof of Lemma 3.5. Condition on any fixed tuple of

$$
(\sigma_0, \sigma_1, \ldots, \sigma_{n_{\mathsf{Ser}}-1}).
$$

By the Definition 3.3, the conditional distribution of $f(x_i, \sigma_i, \mathsf{pc}_i)$ is identical to $\mathcal{F}(x_i)$. Moreover, since all pc_is are i.i.d. sampled, the conditional distributional distributions of the $f(x_i, \sigma_i, \mathsf{pc}_i)$s is identical to $\bigotimes_{i \in [n_{\mathsf{Ser}}]} \mathcal{F}(x_i)$. Therefore, for any joint distribution of σ_is, the joint distribution of $f(x_i, \sigma_i, \mathsf{pc}_i)$ are identically $\bigotimes_{i \in [n_{\mathsf{Ser}}]} \mathcal{F}(x_i)$. □

Proof of Theorem 3.9. We construct the knowledge extractor Ext by assembling an independent instance of $\mathsf{Ext}_{\mathsf{data}}$ for each client, and an independent instance of $\mathsf{Ext}_{\mathsf{comp}}$ for each server. Denote $\mu_{\mathsf{data}}(\lambda), \mu_{\mathsf{comp}}(\lambda)$ as the soundness error of $\mathsf{Ext}_{\mathsf{data}}$ and $\mathsf{Ext}_{\mathsf{comp}}$, respectively.

For any deterministic set of prover strategies $\overrightarrow{\mathsf{Cli}^*}, \overrightarrow{\mathsf{Ser}^*}$, by running the multiple instances of Π_{data} and Π_{comp}, as well as $\mathsf{Ext}_{\mathsf{data}}$ and $\mathsf{Ext}_{\mathsf{comp}}$ independently, conditioned on any fixed $\mathbf{b}^{\mathsf{Cli}}$ (therefore fixed $[\![\mathsf{com}]\!]_i$s) and $Y := ([\![y]\!]_i)_{i \in [n_{\mathsf{Ser}}]}$, for any subset $I^* \subseteq [n_{\mathsf{Ser}}]$ and $J^* \subseteq [n_{\mathsf{Cli}}]$, there exists $\mu_{\mathsf{Ser}}(\lambda) \in$

negl(λ) such that the probability that

$$\prod_{i\in I^*}\Pr\left[\begin{array}{c}[\![y]\!]_i = f\left([\![D]\!]_i,\sigma_i,\mathsf{pc}_i\right)\\ \wedge[\![\mathsf{com}]\!]_i = \mathsf{Commit}\left([\![D]\!]_i,[\![r]\!]_i;\mathsf{pp}\right)\\ \wedge\psi_i = \mathsf{CommitOb}\left(\sigma_i,\rho_i;\mathsf{pp}\right):\\ \left([\![D]\!]_i,[\![r]\!]_i,\sigma_i,\rho_i\right)\leftarrow\\ \mathsf{Ext}_{\mathsf{comp}}\left(\mathsf{Ser}_i^*,[\![y]\!]_i,[\![\mathsf{com}]\!]_i,\psi_i,\mathsf{pc}_i;\mathsf{pp}\right)\end{array}\middle|\mathbf{b}^{\mathsf{Cli}},Y\right]\tag{23}$$

$$\geq\prod_{i\in I^*}\left(\Pr\left[b_i^{\mathsf{Ser}}=1\middle|\mathbf{b}^{\mathsf{Cli}},Y\right]-\mu_{\mathsf{Ser}}(\lambda)\right)\tag{24}$$

$$\geq\prod_{i\in I^*}\Pr\left[b_i^{\mathsf{Ser}}=1\middle|\mathbf{b}^{\mathsf{Cli}},Y\right]-n_{\mathsf{Ser}}\mu_{\mathsf{Ser}}(\lambda)\tag{25}$$

Therefore, averaging over all possible values of $\left(\mathbf{b}^{\mathsf{Cli}},Y\right)$,

$$\Pr\left[\begin{array}{c}[\![y]\!]_i = f\left([\![D]\!]_i,\sigma_i,\mathsf{pc}_i\right)\\ \wedge[\![\mathsf{com}]\!]_i = \mathsf{Commit}\left([\![D]\!]_i,[\![r]\!]_i;\mathsf{pp}\right)\\ \wedge\psi_i = \mathsf{CommitOb}\left(\sigma_i,\rho_i;\mathsf{pp}\right)\quad:\forall i\in I^*\\ \left([\![D]\!]_i,[\![r]\!]_i,\sigma_i,\rho_i\right)\leftarrow\\ \mathsf{Ext}_{\mathsf{comp}}\left(\mathsf{Ser}_i^*,[\![y]\!]_i,[\![\mathsf{com}]\!]_i,\psi_i,\mathsf{pc}_i;\mathsf{pp}\right)\end{array}\right]$$
$$\geq\Pr\left[b_i^{\mathsf{Ser}}=1:i\in I^*\right]-n_{\mathsf{Ser}}\mu_{\mathsf{Ser}}(\lambda)\quad(26)$$

Similarly, there exists $\mu_{\mathsf{Cli}}(\lambda)\in\mathsf{negl}(\lambda)$, such that

$$\Pr\left[\begin{array}{c}D_j\in\mathcal{D}\wedge\mathsf{com}_j=\mathsf{Commit}\left(D_j,r_j;\mathsf{pp}\right):\\ D_j,r_j\leftarrow\mathsf{Ext}_{\mathsf{Cli}}\left(\mathsf{Cli}_j^*,\mathsf{com}_j;\mathsf{pp}\right)\end{array}:\forall j\in J^*\right]$$
$$\geq\Pr\left[b_j^{\mathsf{Cli}}=1:j\in J^*\right]-n_{\mathsf{Cli}}\mu_{\mathsf{Cli}}(\lambda)\quad(27)$$

Furthermore, conditioned on any $\mathbf{b}^{\mathsf{Ser}}$ and $\mathbf{b}^{\mathsf{Cli}}$, for $(I,J)\leftarrow$ IdUsable(I^*,J^*), by the homomorphic commitments,

$$\mathsf{AggrDBCom}\left(\mathsf{com}_j:j\in J\right)=$$
$$\mathsf{RecDBCom}\left([\![\mathsf{com}]\!]_i:i\in I\right),\quad(28)$$

such that the LHS of (28) is a valid commitment of $D_{\mathsf{Cli}}\leftarrow$ AggrDB$\left(D_j:j\in J\right)$, and the RHS of (28) is a valid commitment of $D_{\mathsf{Ser}}\leftarrow$ RecDB$\left([\![D]\!]_i:i\in I\right)$. Therefore, by the binding properties of the commitment schemes, $\mu_{\mathsf{com}}(\lambda)\in\mathsf{negl}(\lambda)$ such that $\Pr\left[D_{\mathsf{Cli}}\neq D_{\mathsf{Ser}}\right]\leq\mu_{\mathsf{com}}(\lambda)$.

Summarizing all the above, the total soundness error of the extraction is $n_{\mathsf{Ser}}\mu_{\mathsf{Ser}}(\lambda)+n_{\mathsf{Cli}}\mu_{\mathsf{Cli}}(\lambda)+\mu_{\mathsf{com}}(\lambda)\in\mathsf{negl}(\lambda)$, since $n_{\mathsf{Ser}},n_{\mathsf{Cli}}\in\mathsf{negl}(\lambda)$.

$\square$

Proof of Theorem 3.12. Given that commitment schemes are hiding, Π_{comp} is zero-knowledge, and the existence of $\mathsf{Sim}_{\mathsf{data}}$, for any instantiation of $\mathfrak{C}$, there exist PPT algorithms $\mathfrak{C}_1$ and $\mathfrak{C}_2$ such that $\mathsf{View}_\Pi^{\mathfrak{C}}(D_j;\mathsf{pp})$ can be simulated by $\mathsf{Sim}_\Pi^{\mathfrak{C}}(D_j)$ as defined below:

$\underline{\mathsf{Sim}_\Pi^{\mathfrak{C}}(D_j;\mathsf{pp})}$

1: $\left([\![D_j]\!]_i\right)_{i\in I_n}\leftarrow\!\!\$\,\mathsf{SecretShare}(D_j)$

2: $S:=\left([\![D_j]\!]_i\right)_{i\in I_j\backslash H_j}$

3: $\left([\![D_{j'}]\!]_i\right)_{i\in H_j,j'\in J_{H_j}\backslash\{j\}},\mathsf{st}_1\leftarrow\!\!\$\,\mathfrak{C}_1\left(S;\mathsf{pp}\right)$

4: $G\leftarrow\!\!\$\,\mathcal{G}(D_j)$

5: $J^*,\mathsf{st}_2\leftarrow\!\!\$\,\mathfrak{C}_2\left(\mathsf{st}_1,G;\mathsf{pp}\right)\quad/\!\!/\,j\in J^*$

6: $\vec{\mathsf{pc}}\leftarrow\!\!\$\,\mathcal{P}_{\mathsf{pc}}^{\otimes n_{\mathsf{Ser}}}$

7: **foreach** $i\in H_j$ **do**

8: $\quad\sigma_i\leftarrow\!\!\$\,\mathcal{P}_\sigma$

9: $\quad[\![D]\!]_i\leftarrow\mathsf{AggrShare}\left([\![D_j]\!]_i,J^*\cap J_i\right)$

10: $\quad[\![y]\!]_i\leftarrow f\left([\![D]\!]_i,\sigma_i,\mathsf{pc}_i\right)$

11: **endfor**

12: $Y:=\left([\![y]\!]_i\right)_{i\in H_j}$

13: **return** $\mathsf{st}_2,Y,\vec{\mathsf{pc}}$

By the design of RD, pc_i causes no additional information leakage. Moreover, since the secret shares S leak no information, the distribution of S is the same given any D_j, which is denoted as $\mathcal{P}_S$. Therefore, the simulator is equivalent to:

$\underline{\mathsf{Sim}_\Pi^{\mathfrak{C}}(D_j;\mathsf{pp})}$

1: $S\leftarrow\!\!\$\,\mathcal{P}_S$

2: $\left([\![D_{j'}]\!]_i\right)_{i\in H_j,j'\in J_{H_j}\backslash\{j\}},\mathsf{st}_1\leftarrow\!\!\$\,\mathfrak{C}_1\left(S;\mathsf{pp}\right)$

3: $G\leftarrow\!\!\$\,\mathcal{G}(D_j)$

4: $J^*,\mathsf{st}_2\leftarrow\!\!\$\,\mathfrak{C}_2\left(\mathsf{st}_1,G;\mathsf{pp}\right)\quad/\!\!/\,j\in J^*$

5: $\left([\![D_j]\!]_i\right)_{i\in H_j}\leftarrow\!\!\$\,\mathsf{PartialShare}\left(D_j,S\right)$

6: **foreach** $i\in H_j$ **do**

7: $\quad[\![D]\!]_i\leftarrow\mathsf{AggrShare}\left([\![D_j]\!]_i,J^*\cap J_i\right)$

8: $\quad[\![y]\!]_i\leftarrow\!\!\$\,\mathcal{F}\left([\![D]\!]_i\right)$

9: **endfor**

10: $Y:=\left([\![y]\!]_i\right)_{i\in H_j}$

11: **return** st_2,Y

By Definition 3.2, Lines 5 to 10 are equivalent to

$$\mathsf{View}_{\mathsf{DDP}}^{\mathfrak{C}}\left(D_j,S,\left([\![D_{j'}]\!]_i\right)_{i\in H_j,j'\in J_{H_j}\backslash\{j\}},J^*\right)$$
$$=\mathfrak{S}\left(\mathcal{M}(D_j),S,\left([\![D_{j'}]\!]_i\right)_{i\in H_j,j'\in J_{H_j}\backslash\{j\}},J^*\right),\quad(29)$$

where $\mathcal{M}$ is (ε,δ)-DP and $\mathfrak{S}$ is a PPT function. Therefore, the simulator is equivalent to the following:

$$\underline{\mathsf{Sim}^{\mathfrak{C}}_{\Pi}(D_j;\mathsf{pp})}$$

1: $\quad S \leftarrow_\$ \mathcal{P}_S$

2: $\quad \left(\left[\!\left[D_{j'}\right]\!\right]_i\right)_{i \in H_j, j' \in J_{H_j} \setminus \{j\}}, \mathsf{st}_1 \leftarrow_\$ \mathfrak{C}_1(S;\mathsf{pp})$

3: $\quad G \leftarrow_\$ \mathcal{G}(D_j)$

4: $\quad J^*, \mathsf{st}_2 \leftarrow_\$ \mathfrak{C}_2(\mathsf{st}_1, G;\mathsf{pp})$

5: $\quad Y \leftarrow \mathfrak{S}\left(\mathcal{M}(D_j), S, \left(\left[\!\left[D_{j'}\right]\!\right]_i\right)_{i \in H_j, j' \in J_{H_j} \setminus \{j\}}, J^*\right)$

6: $\quad$ **return** st_2, Y

Moreover, observe the outputs of Lines 1 and 2 do not depend on D_j. Meanwhile, in Line 3, G is (ε', δ')-DP. Therefore, by sequential composition and post-processing,

$$S, \left(\left[\!\left[D_{j'}\right]\!\right]_i\right)_{i \in H_j, j' \in J_{H_j}}, J^*, \mathsf{st}_2$$

is (ε', δ')-DP. Moreover, since $\mathcal{M}(\cdot)$ is (ε, δ)-DP, by sequential composition and post-processing again, $\mathsf{Sim}^{\mathfrak{C}}_{\Pi}(D_j;\mathsf{pp})$ is $(\varepsilon + \varepsilon', \delta + \delta')$-DP. $\qquad\square$

C   Details for VDDLM (Section 4)

C.1   Verifiable Sampling from Bernoulli Distributions

To realize the verifiable sampling of the Bernoulli distribution with any parameter $0 < p^* < 1$, we utilize Algorithm 2 to convert the generated fair coins to unfair ones. We precompute an approximation of the real number p^* as a binary representation $p = 0.\overline{\beta_0 \beta_1 \ldots \beta_{\nu-1}}$. WLOG, we assume $\beta_{\nu-1} = 1$, or the trailing zero can be removed. The correctness of Algorithm 2 is stated in Lemma C.1.

Algorithm 2 Approximate sampling of $\mathsf{Ber}(p^*)$

Require: $p^* \in (0, 1)$; precision parameter ν; precomputed $p = 0.\overline{\beta_0 \beta_1 \ldots \beta_{\nu-1}}$; $\beta_{\nu-1} = 1$; input $\mathbf{b} \in \{0, 1\}^\nu$

1: **function** $C_{\mathsf{Ber}}(\mathbf{b}; p^*, \nu)$

2: $\quad r \leftarrow \mathbf{b}_{\nu-1}$

3: $\quad$ **for** $i \leftarrow \nu - 2, \nu - 3, \ldots, 0$ **do**

4: $\quad\quad$ **if** $\beta_i = 1$ **then** $r \leftarrow r \lor \mathbf{b}_i$ **else** $r \leftarrow r \land \mathbf{b}_i$

5: $\quad$ **end for**

6: $\quad$ **return** r

7: **end function**

Lemma C.1. *With* $\mathbf{b} \leftarrow_\$ \{0,1\}^\nu$, $C_{\mathsf{Ber}}(\mathbf{b}; p^*, \nu)$ *follows the distribution of* $\mathsf{Ber}\left(0.\overline{\beta_0 \beta_1 \ldots \beta_{\nu-1}}\right)$.

Proof of Lemma C.1. Clearly, when $\nu = 1$, $\mathbf{b}_0$ is a fair coin. Inductively, if Lemma C.1 holds for $\nu = N$, then when $\nu = N + 1$, $r \sim \mathsf{Ber}(p)$ where $p = 0.\overline{\beta_1 \beta_2 \ldots \beta_N}$ at the end of iteration $i = 1$. Then if $\beta_0 = 0$, by the $\land$ operation and uniform randomness of $\mathbf{b}_0$, after iteration $i = 0$, $r \sim \mathsf{Ber}\left(\frac{p}{2}\right)$

such that $\frac{p}{2} = 0.0\overline{\beta_1 \beta_2 \ldots \beta_N}$. Similarly, if $\beta_0 = 1$, by the $\lor$ operation and uniform randomness of $\mathbf{b}_0$, after iteration $i = 0$, $r \sim \mathsf{Ber}\left(\frac{1+p}{2}\right)$ such that $\frac{1+p}{2} = 0.\overline{1\beta_1 \beta_2 \ldots \beta_N}$. $\qquad\square$

C.2   Utility and Overhead Analysis of VDBM

In VDBM, the additive noise is drawn from $\mathsf{Binom}\left(n_b, \frac{1}{2}\right)$, such that (ε, δ)-DP is satisfied with $\varepsilon = 10\sqrt{\frac{1}{n_b} \log \frac{2}{\delta}}$. Therefore, $n_b = \Theta\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right)$, which, unlike VDDLM, admits a direct dependence on the privacy costs.

Therefore, with n_{Ser} servers, the expected L1 error of the binomial mechanism in each dimension after removing the bias terms becomes

$$\mathbb{E}_{x \leftarrow_\$ \mathsf{Binom}\left(n_{\mathsf{Ser}} n_b, \frac{1}{2}\right)} \left| x - \frac{n_{\mathsf{Ser}} n_b}{2} \right| \tag{30}$$

$$\leq 2^{-n_{\mathsf{Ser}} n_b} n_{\mathsf{Ser}} n_b \binom{n_{\mathsf{Ser}} n_b - 1}{\left\lfloor \frac{n_{\mathsf{Ser}} n_b}{2} \right\rfloor} \tag{31}$$

$$\leq \frac{\Gamma\left(\left\lceil \frac{n_{\mathsf{Ser}} n_b}{2} \right\rceil + \frac{1}{2}\right)}{\sqrt{\pi}\Gamma\left(\left\lceil \frac{n_{\mathsf{Ser}} n_b}{2} \right\rceil\right)} \tag{32}$$

By Stirling's formula, Equation (32) is $\Theta\left(\sqrt{n_{\mathsf{Ser}} n_b}\right)$. As $n_b = \Theta\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right)$, the overall expected L1 error is $\Theta\left(\frac{\sqrt{n_{\mathsf{Ser}}}}{\varepsilon}\sqrt{\log \frac{1}{\delta}}\right)$, and therefore $\Theta\left(\frac{\sqrt{n_{\mathsf{Ser}}} d}{\varepsilon}\sqrt{\log \frac{1}{\delta}}\right)$ with d dimensions. Moreover, the prover's overhead is proportional to the product of n_b and d, and is therefore $\Theta\left(\frac{d}{\varepsilon^2} \log \frac{1}{\delta}\right)$.

Since the communication and verifier's overhead highly depend on the exact implementation [85], we report these complexities of actual implementation in Table 1. VDDLM achieves a more significant improvement in these aspects than in the prover's overhead, thanks to the use of the data representation and KZG commitment scheme in Section 2.2.

C.3   Deferred Proofs

In this appendix, we present the missing the proofs in Section 4.

Proof of Theorem 4.2. For any subset $S \subset \mathbb{Z}$ and neighboring databases D, D', we consider

$$S_+ := S \cap \left(\mathsf{Supp}(\mathcal{M}(D)) \cap \mathsf{Supp}(\mathcal{M}(D'))\right) \tag{33}$$

$$S_- := S \cap \left(\mathsf{Supp}(\mathcal{M}(D)) \setminus \mathsf{Supp}(\mathcal{M}(D'))\right), \tag{34}$$

such that

$$\Pr[\mathcal{M}(D) \in S] = \Pr[\mathcal{M}(D) \in S_+] + \Pr[\mathcal{M}(D) \in S_-] \tag{35}$$

WLOG, assuming $q(D') - q(D) = 1$, then

$$\mathsf{Supp}(\mathcal{M}(D)) \setminus \mathsf{Supp}(\mathcal{M}(D')) = \{q(D) - 2^\gamma\}.$$

Therefore,

$$\Pr[\mathcal{M}(D) \in S_-] \le \Pr[\mathcal{M}(D) = q(D) - 2^\gamma] \qquad (36)$$

$$= \Pr[-2^\gamma \leftarrow \mathcal{C}_{\mathsf{Lap}}] \qquad (37)$$

$$= \Pr[b_z = 1, s = -1, a = 2^\gamma] \qquad (38)$$

$$= (1 - p_z) \cdot \frac{1}{2} \cdot \prod_{i=0}^{\gamma-1} p_i \qquad (39)$$

$$= \delta. \qquad (40)$$

Note that the equality holds iff $q(D) - 2^\gamma \in S$.

Furthermore, given any $r \in \{-2^\gamma + 1, \ldots, 2^\gamma\}$, the probability

$$\Pr[\mathcal{M}(D) = q(D) + r] = \Pr[r \leftarrow \mathcal{C}_{\mathsf{Lap}}] \qquad (41)$$

$$\Pr[\mathcal{M}(D') = q(D) + r] = \Pr[r - 1 \leftarrow \mathcal{C}_{\mathsf{Lap}}] \qquad (42)$$

As we aim at upper bounding $\Pr[\mathcal{M}(D) = q(D) + r]$ by $\Pr[\mathcal{M}(D') = q(D) + r]$, we only need to consider the case that $r \le 0$.

If $r = 0$, the ratio of the two probabilities

$$\frac{\Pr[\mathcal{M}(D) = q(D) + r]}{\Pr[\mathcal{M}(D') = q(D) + r]} = \frac{p_z}{\frac{1 - p_z}{2} \prod_{i=0}^{\gamma-1}(1 - p_i)} = a_z. \qquad (43)$$

Otherwise, by representing

$$r = -\mathfrak{b}1\underbrace{00\ldots0}_{i\times} \qquad (44)$$

$$r + 1 = -\mathfrak{b}0\underbrace{11\ldots1}_{i\times} \qquad (45)$$

for any bit string $\mathfrak{b}$, the ratio between the two probabilities becomes

$$\frac{\Pr[\mathcal{M}(D) = q(D) + r]}{\Pr[\mathcal{M}(D') = q(D) + r]} = \frac{p_i \prod_{j=0}^{i-1}(1 - p_j)}{(1 - p_i) \prod_{j=0}^{i-1} p_j} = a_i \qquad (46)$$

Therefore,

$$\Pr[\mathcal{M}(D) \in S_+] \le e^\varepsilon \Pr[\mathcal{M}(D') \in S], \qquad (47)$$

where the equality holds when the S_+ is a subset of the maximizer in Equation (11).

Summarizing all above,

$$\Pr[\mathcal{M}(D) \in S] \le e^\varepsilon \Pr[\mathcal{M}(D') \in S] + \delta, \qquad (48)$$

the equality holds for $S = S_+ \cup S_-$ where S_+ and S_- satisfy the aforementioned equality conditions. $\square$

Proof of Lemmas B.1 and 4.3. WLOG, consider Cli_0 with $H_0 = [\tau']$ where $\tau' \le n_{\mathsf{Ser}} - 1$. Consider any single-dimensional counting query $\mathcal{M}(x) := x + r$ (where $x \in \mathbb{F}$ is the unperturbed count, and r is an additive noise such that $\mathrm{Supp}(r) \subset \mathbb{F}$) that achieves (ε, δ)-DP, which is the case for

both the binomial mechanism and the modified Laplacian mechanism. The multidimensional version extends $\mathcal{M}$ to d-dimensional for any $d \ge 1$ such that $\mathcal{M}$ is applied element-wise, i.e.,

$$\mathcal{M}(x_0, x_1, \ldots, x_{d-1}) := (\mathcal{M}(x_0), \mathcal{M}(x_1), \ldots, \mathcal{M}(x_{d-1})).$$

By the independence among all d dimensions, for any $\mathbf{x}, \mathbf{x}' \in \mathbb{F}^d$ such that $\|\mathbf{x} - \mathbf{x}'\|_1 \le 1$,

$$\Pr[\mathcal{M}(\mathbf{x}) \in S] \le e^\varepsilon \Pr[\mathcal{M}(\mathbf{x}') \in S] + \delta.$$

Therefore, $\mathcal{M}$ achieves (ε, δ)-DP. For VDDLM where the additive randomness is sampled using LPRF, the actual realization of $\mathcal{M}$ achieves (ε, δ)-CDP.

Consider

$$\underline{\mathsf{Sim}(\mathbf{x}_0, S_{\mathsf{out}}, S_{\mathsf{in}}, J^*)}$$

1: $\mathbf{y}_0 \leftarrow_\$ \underbrace{\mathcal{M} \circ \cdots \circ \mathcal{M}}_{\tau' \times}(\mathbf{x}_0)$

2: **for** $i \leftarrow 1, 2, \ldots, \tau' - 1$ **do**

3: $\quad [\![\mathbf{y}_0]\!]_i \leftarrow_\$ \mathbb{F}^d$

4: $\quad [\![\mathbf{y}]\!]_i \leftarrow [\![\mathbf{y}_0]\!]_i + \sum_{j \in J^* \setminus \{0\}} [\![\mathbf{x}_j]\!]_i$

5: **endfor**

6: $[\![\mathbf{y}_0]\!]_0 \leftarrow \mathbf{y}_0 - \sum_{i=1}^{\tau'-1} [\![\mathbf{y}_0]\!]_i - \sum_{i=\tau'}^{n_{\mathsf{Ser}}-1} [\![\mathbf{x}_0]\!]_i$

7: $[\![\mathbf{y}]\!]_0 \leftarrow [\![\mathbf{y}_0]\!]_0 + \sum_{j \in J^* \setminus \{0\}} [\![\mathbf{x}_j]\!]_0$

8: **return** $[\![\mathbf{y}]\!]_0$

which perfectly simulates $\mathsf{View}_{\mathcal{F}}^{\mathfrak{C}}(D_j, S_{\mathsf{out}}, S_{\mathsf{in}}, J^*)$ for any

$$S_{\mathsf{out}} := ([\![\mathbf{x}_0]\!]_i)_{\tau' \le i \le n_{\mathsf{Ser}}-1} \qquad (49)$$

$$S_{\mathsf{in}} := ([\![\mathbf{x}_j]\!]_i, [\![\mathbf{x}_j]\!]_i, \ldots, [\![\mathbf{x}_j]\!]_i)_{\substack{i \in [\tau'] \\ 1 \le j \le n_{\mathsf{Cli}}-1}} \qquad (50)$$

and $J^* \subseteq [n_{\mathsf{Cli}}] \setminus \{0\}$. By post-processing, $\mathsf{Sim}(\mathbf{x}_0, S_{\mathsf{out}}, S_{\mathsf{in}}, J^*)$ satisfies the conditions in Definition 3.2. Therefore, for both VDBM and VDDLM, the underlying $\mathcal{F}$ are $(n_{\mathsf{Ser}} - 1, \varepsilon, \delta)$-DDP. $\square$

Proof of Theorem 4.5. We follow the notation in Section 4.1. Conditioning on b_z,

$$\mathbb{E}|\mathcal{M}(D) - q(D)| \qquad (51)$$

$$= p_z \cdot 0 + (1 - p_z)\mathbb{E}a \qquad (52)$$

$$= (1 - p_z)\left(1 + \sum_{i=0}^{\gamma-1} 2^i \Pr[r_i = 1]\right) \qquad (53)$$

$$= (1 - p_z)\left(1 + \sum_{i=0}^{\gamma-1} 2^i p_i\right) \qquad (54)$$

$\square$

C.4 Generalization of Theorem 4.2

Theorem C.2 (Theorem 4.2, generalized)**.** *Given query $q : \mathcal{D} \to \mathbb{Z}$ with sensitivity $\Delta \geq 1$, the modified discrete Laplacian mechanism defined in Theorem 4.2 satisfies (ε, δ)-differential privacy, where*

$$\varepsilon = \Delta \cdot \log \left(\max \left\{ a_z, a_0, \ldots, a_{\gamma-1} \right\} \right), \qquad (55)$$

$$\delta = \sum_{i \in [\Delta]} \Pr[-2^\gamma + i \leftarrow \mathcal{C}_{\mathsf{Lap}}], \qquad (56)$$

where a_z and a_is are defined in Theorem C.2.

Proof of Theorem C.2. For any subset $S \subset \mathbb{Z}$ and neighbouring databases D, D', we consider

$$S_+ := S \cap \left(\mathrm{Supp}(\mathcal{M}(D)) \cap \mathrm{Supp}(\mathcal{M}(D')) \right) \qquad (57)$$

$$S_- := S \cap \left(\mathrm{Supp}(\mathcal{M}(D)) \setminus \mathrm{Supp}(\mathcal{M}(D')) \right), \qquad (58)$$

such that

$$\Pr[\mathcal{M}(D) \in S] = \Pr[\mathcal{M}(D) \in S_+] + \Pr[\mathcal{M}(D) \in S_-]. \quad (59)$$

WLOG, assuming $q(D') - q(D) = \Delta$, then

$$\mathrm{Supp}(\mathcal{M}(D)) \setminus \mathrm{Supp}(\mathcal{M}(D')) = \{q(D) - 2^\gamma + i : i \in [\Delta]\}.$$

Therefore,

$$\Pr[\mathcal{M}(D) \in S_-] \leq \sum_{i=0}^{\Delta-1} \Pr[\mathcal{M}(D) = q(D) - 2^\gamma + i] = \delta. \quad (60)$$

Furthermore, given any $r \in \{-2^\gamma + \Delta, \ldots, 2^\gamma\}$, the probability

$$\Pr[\mathcal{M}(D) = q(D) + r] = \Pr[r \leftarrow \mathcal{C}_{\mathsf{Lap}}], \qquad (61)$$

$$\Pr[\mathcal{M}(D') = q(D) + r] = \Pr[r - \Delta \leftarrow \mathcal{C}_{\mathsf{Lap}}]. \qquad (62)$$

As we aim at upper bounding $\Pr[\mathcal{M}(D) = q(D) + r]$ by $\Pr[\mathcal{M}(D') = q(D) + r]$, by repeatedly applying the argument in the proof of Theorem 4.2, the ratio of the probability is upper bounded by

$$\left(\max \left\{ a_z, a_0, \ldots, a_{\gamma-1} \right\} \right)^\Delta,$$

such that

$$\Pr[\mathcal{M}(D) \in S_+] \leq e^\varepsilon \Pr[\mathcal{M}(D') \in S]. \qquad (63)$$

Summarizing all the above,

$$\Pr[\mathcal{M}(D) \in S] \leq e^\varepsilon \Pr[\mathcal{M}(D') \in S] + \delta. \qquad (64)$$

$\square$

C.5 Discussions on extending to discrete Gaussian mechanism

We argue that the extension of VDDLM to the distributed discrete Gaussian mechanism is likely to result in efficiency issues due to the underlying sampling methods of the discrete Gaussian mechanism. The current sampling algorithms of the discrete Gaussian distribution utilize rejection sampling [23], which is equivalent to a circuit with indefinite depth. To preserve completeness and ensure that all dimensions are accepted, all parties need to prescribe that the rejection sampling loop be executed for a sufficiently large number of iterations, especially for higher dimensions. Although the existing MPC mechanisms [91] propose dynamically terminating the rejection sampling loop by revealing the acceptance bits in each iteration, transplanting this workaround to VDDP will significantly increase the verifier's load by multiplying the total number of iterations. Therefore, the verifiable execution of the discrete Gaussian mechanism would benefit from the development of alternative sampling methods.

D Details for VRR (Section 5)

D.1 Proof of polynomial evaluation

As the the proposed scheme involves evaluations of publicly-known polynomials at private values, we describe in Figure 8 the corresponding protocol where the prover $\mathsf{P}_{\mathsf{EvSc}}$ to the verifier $\mathsf{V}_{\mathsf{EvSc}}$ that $y = F(x)$ for a publicly-known polynomial y included in the public parameters as $g^{F(\tau)}$ and private values $x, y \in \mathbb{F}$ committed as com_x and com_y, respectively.

Lemma D.1. *Given $2(K+2)$ accepting transcripts, namely*

$$\pi^{(k,i)} = \left(\mathsf{com}_{F'}, u^{(k)}, \mathsf{com}_{z'}^{(k)}, \pi_{\mathsf{Prod}}^{(k,i)}, \pi_{\mathsf{KZG}}^{(k)}, \pi_{\mathsf{KZG}}'^{(k)} \right)$$

for $1 \leq k \leq K+2$ and $i \in \{1, 2\}$, it can be extracted from $x, r_x, y, r_y \in \mathbb{F}$ such that

$$\mathsf{com}_x = g^x h^{r_x} \wedge \mathsf{com}_y = g^y h^{r_y} \wedge y = F(x).$$

Proof of Lemma D.1. For each $1 \leq k \leq K+2$, by the knowledge soundness of $\mathsf{P}_{\mathsf{Prod}} \leftrightarrow \mathsf{V}_{\mathsf{Prod}}$, since $u^{(k)}$ and $z^{(k)} = F\left(u^{(k)}\right)$ are both public, it can be extracted from $\pi^{(k,1)}$ and $\pi^{(k,2)}$ the values $x, r_x, y, r_y, z'^{(k)}, r_z'^{(k)}$ such that

$$y - F\left(u^{(k)}\right) = \left(x - u^{(k)}\right) z' \wedge$$

$$\mathsf{com}_x = g^x h^{r_x} \wedge \mathsf{com}_y = g^y h^{r_y} \wedge \mathsf{com}_{z'} = g^{z'} h^{r_z'}.$$

By the Diffie-Hellman assumption, the extracted x, r_x, y, r_y are the same.

Furthermore, in Line 6, by the knowledge of exponent assumption, it can be extracted $G^{(k)}, R_G^{(k)} \in \mathbb{F}_{\leq K}[X]$ such that

$$G^{(k)}(u) = 0 \wedge \mathsf{com}_{F'} \cdot \left(\mathsf{com}_{z'}^{(k)} \right)^{-1} = g^{G^{(k)}(\tau)} h^{R_G^{(k)}(\tau)},$$

$$\underline{\mathsf{P_{EvSc}}\left(y,r_y,x,r_x,F,\mathsf{com}_y,\mathsf{com}_x;\mathsf{pp}\right) \qquad\qquad\qquad\qquad\qquad \mathsf{V_{EvSc}}\left(\mathsf{com}_y,\mathsf{com}_x,g^{F(\tau)};\mathsf{pp}\right)}$$

$1:\quad F'(X) \leftarrow \dfrac{y-F(X)}{(x-X)}, R'_F(X) \leftarrow\!\!{\scriptstyle\$}\; \mathbb{F}_{\leq K}[X]$

$2:\quad \mathsf{com}_{F'} \leftarrow \mathsf{Commit_{KZG}}\left(F',R'_F;\mathsf{pp}\right) \qquad\qquad \xrightarrow{\;\mathsf{com}_{F'}\;} \qquad \mathsf{com}_{F'}$

$3:\quad z \leftarrow F(u), z' \leftarrow F'(u), r'_z \leftarrow\!\!{\scriptstyle\$}\;\mathbb{F} \qquad\qquad \xleftarrow{\quad u \quad} \qquad u \leftarrow\!\!{\scriptstyle\$}\;\mathbb{F}$

$4:\quad \mathsf{com}_{z'} \leftarrow g^{z'}h^{r'_z} \qquad\qquad\qquad\qquad\qquad \xrightarrow{\;\mathsf{com}_{z'}\;} \qquad \mathsf{com}_{z'}$

$5:\quad \mathsf{P_{Prod}}\left(y-z,r_y,x-u,r_x,z',r'_z,\mathsf{com}_y\cdot g^{-z},\mathsf{com}_x\cdot g^{-u},\mathsf{com}_{z'};\mathsf{pp}\right) \xleftarrow{\;\pi_{\mathsf{Prod}}\;} \mathsf{V_{Prod}}\left(\mathsf{com}_y\cdot g^{-z},\mathsf{com}_x\cdot g^{-u},\mathsf{com}_{z'};\mathsf{pp}\right)$

$6:\quad \pi_{\mathsf{KZG}} \leftarrow \mathsf{P_{KZG}}\left(z,u,F;\mathsf{pp}\right) \qquad\qquad \xrightarrow{\;\pi_{\mathsf{KZG}}\;} \qquad \mathsf{V_{KZG}}\left(z,u,\pi_{\mathsf{KZG}},\mathsf{com}_F;\mathsf{pp}\right)$

$7:\quad \pi'_{\mathsf{KZG}} \leftarrow \mathsf{P_{KZG}}\left(0,u,F'-z',R'_F-r'_z;\mathsf{pp}\right) \xrightarrow{\;\pi'_{\mathsf{KZG}}\;} \mathsf{V_{KZG}}\left(0,u,\pi'_{\mathsf{KZG}},\mathsf{com}_{F'}\cdot \mathsf{com}_{z'}^{-1};\mathsf{pp}\right)$

Figure 8: Protocol for proving $y = F(x)$ for public $F \in \mathbb{F}[X]$ can secret $x,y \in \mathbb{F}$.

therefore, for $F'(X) \leftarrow G^{(k)}(X)+z', R'_F \leftarrow R_G^{(k)}(X)+r'_z$,

$$F'\left(u^{(k)}\right) = z' \wedge \mathsf{com}_{F'} = g^{F'(\tau)}h^{R'_F(\tau)},$$

where the uniqueness of F' and R'_F are also by the Diffie-Hellman assumption.

Therefore,

$$y - F\left(u^{(k)}\right) = \left(x - u^{(k)}\right)F'\left(u^{(k)}\right),$$

for all $K+2$ different $u^{(k)}$s. Since LHS and RHS are both at most of degree $K+1$, it must hold that $y-F(X) = (x-X)F'(X)$, such that $y = F(x)$. $\qquad\square$

Lemma D.2. *For any* $\mathsf{pp} \in \mathsf{Supp}\left(\mathsf{Setup}\left(1^\lambda\right)\right)$, $y,r_y,x,r_x \in \mathbb{F}$, $F \in \mathbb{F}_{\leq K}[X]$ *and* $\mathsf{com}_x \in \mathbb{G}$ *such that*

$$y = F(x) \wedge \mathsf{com}_x = g^x h^{r_x} \wedge \mathsf{com}_y = g^y h^{r_y},$$

there exists a simulator

$\underline{\mathsf{Sim_{EvSc}}\left(\mathsf{com}_y,\mathsf{com}_x,F;\mathsf{pp}\right)}$

$\mathsf{com}'_z \leftarrow\!\!{\scriptstyle\$}\;\mathbb{G}$

$u \leftarrow\!\!{\scriptstyle\$}\;\mathbb{F}$

$\pi_{\mathsf{Prod}} \leftarrow\!\!{\scriptstyle\$}\; \mathsf{Sim_{Prod}}\left(\mathsf{com}_z,\mathsf{com}_x\cdot g^{-u},\mathsf{com}_{z'};\mathsf{pp}\right)$

$\left(c',\pi'_{\mathsf{KZG}}\right) \leftarrow\!\!{\scriptstyle\$}\; \mathsf{Sim_{KZG}}\left(0,u;\mathsf{pp}\right)$

$\pi_{\mathsf{KZG}} \leftarrow \mathsf{P_{KZG}}\left(F(u),u,F;\mathsf{pp}\right)$

$\mathsf{com}_{F'} \leftarrow\!\!{\scriptstyle\$}\; c'\cdot \mathsf{com}_{z'}$

return $\mathsf{com}_{F'},u,\mathsf{com}_{z'},\pi_{\mathsf{Prod}},\pi_{\mathsf{KZG}},\pi'_{\mathsf{KZG}}$

that perfectly simulates the the view of $\mathsf{V_{EvSc}}$, *that is*

$$\mathsf{View}\begin{bmatrix} \mathsf{P_{EvSc}}\left(y,r_y,x,r_x,F,\mathsf{com}_y,\mathsf{com}_x;\mathsf{pp}\right) \\ \leftrightarrow \mathsf{V_{EvSc}}\left(\mathsf{com}_y,\mathsf{com}_x,g^{F(\tau)};\mathsf{pp}\right) \end{bmatrix}$$
$$= \mathsf{Sim_{EvSc}}\left(\mathsf{com}_y,\mathsf{com}_x,F;\mathsf{pp}\right), \quad (65)$$

Proof of Lemma D.2. We first consider the joint distribution of transcript π in $\mathsf{P_{EvSc}} \leftrightarrow \mathsf{V_{EvSc}}$ described in Figure 8. Clearly, the marginal distribution of $(u,\mathsf{com}_{z'})$ is the uniform distribution in $\mathbb{F}\times\mathbb{G}$. Moreover, due to the uniform randomness of R'_F, the conditional distribution of π'_{KZG} on $(u,\mathsf{com}_{z'})$, where

$$\pi'_{\mathsf{KZG}} = \left(\rho' := R'_F(u)-r'_z, \gamma := g^{\frac{F'(\tau)-F'(u)}{\tau-u}}h^{\frac{R'_F(u)-R'_F(\tau)}{\tau-u}}\right),$$
$$(66)$$

is the uniform distribution over $\mathbb{F}\times\mathbb{G}$. Therefore, the aforementioned joint distribution is the uniform distribution over

$$\mathbb{G}\times\mathbb{F}\times(\mathbb{F}\times\mathbb{G}),$$

which is perfectly simulated by $\mathsf{Sim_{EvSc}}$.

Moreover, $\mathsf{Sim_{EvSc}}$ correctly captures that

- π_{Prod} can be perfectly simulated using

$$\mathsf{Sim_{Prod}}\left(\mathsf{com}_z,\mathsf{com}_x\cdot g^{-u},\mathsf{com}_{z'};\mathsf{pp}\right),$$

- $\mathsf{com}_{F'}$ and π_{KZG} can be deterministically computed from

$$\left(u,\mathsf{com}'_z,\pi'_{\mathsf{KZG}}\right)$$

and the only inputs to $\mathsf{Sim_{EvSc}}$, i.e., $\mathsf{com}_y,\mathsf{com}_x,F,\mathsf{pp}$.

This concludes the proof of the equality of distributions in (65). $\qquad\square$

D.2 Proof of Theorem 5.1

By Theorem 3.11, to prove Theorem 5.1, it suffices to prove the completeness, soundness, and zero-knowledge of the protocol described in Figure 5.

Proof Sketch of Theorem 5.1 (completeness). The completeness of Figure 5 directly follows from that of the subroutines involved. $\square$

Proof of Theorem 5.1 (soundness). In Figure 5, consider $4(|\Omega|+2)$ accepting transcripts

$$\left(\mathsf{com}_z, \alpha^{(j)}, \pi_{\mathsf{EvSc}}^{(j,k)}, \pi_{\mathsf{Prod}}^{(j)}\right)$$

where $j \in \{1,2\}$ and $1 \leq k \leq 2(|\Omega|+2)$. By Lemma D.1, for each $j \in \{1,2\}$, it can be extracted z, r_z, t, r_t from $\pi_{\mathsf{Prod}}^{(j,k)}$s such that

$$z = F(t) + \alpha^{(j)}F_\Omega(t) \wedge g^z h^{r_z} = \mathsf{com}_z \wedge g^t h^{r_t} = \mathsf{com}_t.$$

With $a^{(1)} \neq a^{(2)}$, it must hold that

$$F(t) = z \wedge F_\Omega(t) = 0.$$

Therefore, for $i_t \leftarrow \log_\omega t$, and $i_\sigma \leftarrow i_t - i_{\mathsf{pc}}$, we have $z = F(\omega^{i_\sigma + i_{\mathsf{pc}}})$. Moreover, by

$$g^{\omega^{i_\sigma + i_{\mathsf{pc}}}} h^{r_t} = \mathsf{com}_t = \psi^{\omega^{i_{\mathsf{pc}}}},$$

it holds that for $r_\sigma \leftarrow r_t \omega^{-i_{\mathsf{pc}}}, g^{\omega^{i_\sigma}} h^{r_\sigma} = \psi$. Finally, from $\pi_{\mathsf{Prod}}^{(1)}$ and $\pi_{\mathsf{Prod}}^{(2)}$ it can be extracted x, r_x such that $\mathsf{com} = g^x h^{r_x}$ and $y = xz = xF(\omega^{i_\sigma + i_r})$. Moreover, since $y \in \mathcal{X}$ and $F(\omega^{i_\sigma + i_r})$, $x = yF(\omega^{i_\sigma + i_r})^{-1} \in \mathcal{X}$. $\square$

Proof Sketch of Theorem 5.1 (zero-knowledge). Given output y, hiding commitments com, ψ and the public coin i_{pc}, the view of verifier in Figure 5 can be simulated by

$$
\begin{array}{l}
\underline{\mathsf{Sim}_{\mathsf{RR}}(\mathsf{com}, \psi, y, i_{\mathsf{pc}}; \mathsf{pp})} \\[4pt]
\mathsf{com}_z \leftarrow_\$ \mathbb{G} \\[2pt]
\alpha \leftarrow_\$ \mathbb{F} \\[2pt]
\pi_{\mathsf{EvSc}} \leftarrow \mathsf{Sim}_{\mathsf{EvSc}}\left(\mathsf{com}_z, \psi^{\omega^{i_{\mathsf{pc}}}}, F + \alpha F_\Omega; \mathsf{pp}\right) \\[2pt]
\pi_{\mathsf{Prod}} \leftarrow \mathsf{Sim}_{\mathsf{Prod}}(g^y, \mathsf{com}_z, \mathsf{com}; \mathsf{pp}) \\[2pt]
\textbf{return } \mathsf{com}_z, \alpha, \pi_{\mathsf{EvSc}}, \pi_{\mathsf{Prod}}
\end{array}
$$

$\square$

E Additional experiments on VRR

We conduct further experiments to measure the running time of the different components in VRR and record the results as shown in Figure 9. It can be observed that the proving time is the most time-consuming component and linearly

Figure 9: Running time of each component of VRR.

increases with respect to $|\Omega|$ (the quantization accuracy of the probability space), as projected in Section 5.3. Meanwhile, the setup of the public parameters takes a similar amount of time but is only executed once in the entire system. The computing times admit a similar increase as the proving time but are negligible compared to the time needed to conduct the proofs by the same clients. Since only a constant amount of values need to be committed, the committing time by each client is mostly constant. Additionally, the verifier only takes constant time to authenticate the output by each client, as described in Section 5.3, therefore significantly increasing the scalability of VRR.

F Additional Related Work

Due to concerns related to compatibility with cryptographic primitives, the discrete DP mechanisms [6,23,48] and the distributed DP mechanisms [1,2,27,57] constructed over them via SecAgg [19] have been considered for constructing the secure computations, including MPCs and verifiable executions, of DP mechanisms.

While deviations from DP mechanisms may cause adversarial behaviors, the correctness of the mechanisms may also lead to unintentional privacy leakages. Therefore, verification tools in programming languages have been applied to ensure that the mechanisms, as programs, are bug-free and can achieve the claimed DP guarantees [75,79,90]. Note that these studies focus on a different aspect of security in DP mechanisms, which may serve as complementary but do not overlap with the verifiable executions of the protocols.