Shared-Noise Mechanisms for Verifiable Differentially Private Counting

Sehyeon Park[1,2] **Chenglin Fan**[1,2]*
[1]Department of Computer Science and Engineering
[2]Algorithmic Foundations of Data Science Lab
Seoul National University
Seoul 08826, South Korea
{shlion,fanchenglin}@snu.ac.kr

Abstract

Verifiable differential privacy enables public auditing of privacy-preserving aggregate computations, addressing the trust deficit in standard implementations. However, existing verifiable counting approaches that distribute trust across multiple provers often require each prover to contribute independent noise. This duplicates the noise added to the aggregate and substantially degrades utility, making accurate private counting difficult in distributed settings.

We introduce a shared-noise paradigm for verifiable differentially private counting. Instead of duplicating noise across provers, our approach generates a single noise sample in additive-shared form among mutually distrustful provers, while Pedersen commitments let a public verifier audit consistency without learning the noise. This yields shared-noise, verifier-auditable instantiations of two standard noise families: the Shared-Noise Verifiable Binomial Mechanism (SN-VBM) and the Shared-Noise Verifiable Laplace Mechanism (SN-VLM).

We prove that both mechanisms satisfy differential privacy and public verifiability, including completeness, soundness against malicious provers, and zero knowledge against a malicious verifier colluding with corrupted provers. Among the two instantiations, SN-VLM is the more scalable option: it avoids the quadratic growth in shared randomness required by the binomial construction under strict privacy budgets. Experiments on preprocessing load, randomness-generation time, and noise error show that SN-VLM substantially reduces cryptographic overhead while improving empirical accuracy in the evaluated privacy regimes.

1 Introduction

Differential privacy (DP) [14] is the de facto standard for privacy-preserving data release [15], with private counting serving as a basic primitive for downstream analytics [12]. However, standard DP mechanisms lack public verifiability: users and auditors cannot independently check whether the released output was computed with honest randomness [4]. This trust gap limits deployment in sensitive domains such as healthcare [18] and census statistics [1], where transparency is essential.

Verifiable DP addresses this gap by allowing an external verifier to audit whether the mechanism was consistently executed and the injected randomness was honestly generated [4]. Prior multi-prover verifiable counting protocols often duplicate noise across provers, increasing aggregate error and degrading utility [4, 27]. The main challenge is noise generation: local prover-generated noise can be biased, while transparent noise can be subtracted by the verifier to recover the exact count [4].

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To address this problem for counting queries, we introduce a shared-noise paradigm for verifiable differential privacy. Instead of assigning independent noise to each prover, we generate one noise sample in additively shared form among multiple provers [29, 26]. Our framework ensures that the protocol meets three essential requirements: completeness (honest executions are accepted), soundness (malicious provers cannot forge outputs), and zero-knowledge (the verifier learns nothing beyond the DP answer) [20, 4].

1.1 Our Contributions

- **Auditable Shared-Noise Framework:** We propose a shared-noise paradigm leveraging Pedersen-committed additive sharing. We construct **SharedCoin**, a protocol using zero-knowledge OR-opening proofs and committed Beaver-triple multiplication to generate additive-shared random bits that are hidden from the verifier but publicly auditable.
- **SN-VBM and SN-VLM Mechanisms:** We instantiate two mechanisms for counting. **SN-VBM** realizes centered binomial noise through direct shared bit sampling. **SN-VLM** realizes truncated discrete-Laplace-type noise using a verifier-auditable additive-shared Bernoulli sampler.
- **Complexity Analysis:** We prove that **SN-VLM** avoids the quadratic shared-coin growth of the binomial instantiation under strict privacy budgets. For fixed δ and number of provers m, the shared-coin complexity decreases from $\Theta(\epsilon^{-2})$ for **SN-VBM** to $\Theta(\log \epsilon^{-1} \log\log \epsilon^{-1})$ for **SN-VLM**; the committed Beaver-triple count follows the same asymptotic growth up to the constant factors induced by the subprotocols.
- **Empirical Evaluation:** We evaluate the two instantiations through Beaver triple consumption, runtime decomposition, and noise error. Our results show that, in the evaluated regimes, **SN-VLM** substantially reduces the preprocessing and randomness-generation cost compared with the binomial instantiation, while also yielding lower empirical noise error.

1.2 Technical Overview

The core of our framework is the transition from centralized or duplicated noise generation to a single distributed, auditable noise sample. We achieve this through three primary layers: a verifiable randomness primitive, specialized noise-sampling mechanisms, and a commitment-based verification architecture.

The Foundation: Verifiable Shared Coins. The fundamental building block of our system is the **SharedCoin** protocol. Unlike standard multi-party coin-flipping, which often reveals the bit to all parties, our protocol generates a random bit $a \in \{0, 1\}$ such that it is additively shared among m provers as $[[a]]$. To ensure this bit is unbiased and valid without revealing its value to the verifier, we employ two cryptographic safeguards:

- **Bit Validity:** Each prover commits to a prover-contributed bit in shared form, and a zero-knowledge OR-opening proof certifies that the reconstructed commitment opens to a bit.
- **Unbiasedness via XOR:** The provers compute the XOR sum of their bits using a committed variant of Beaver-triple multiplication [2]. This ensures that as long as at least one randomness-contributing party is honest, the resulting shared bit is computationally unbiased [4].

From Coins to Noise: SN-VBM and SN-VLM. We leverage these shared coins to construct privacy noise:

- **SN-VBM:** This mechanism directly aggregates n_b independent shared coins. The released value is $Q(X) + \sum a_i - n_b/2$, which is the exact count plus centered binomial noise. While intuitive, this mechanism requires n_b to scale as $1/\epsilon^2$ for fixed δ, leading to high overhead in high-privacy regimes.
- **SN-VLM:** To reduce the shared-coin cost, we implement a **SharedBern** sampler. This sampler quantizes a target probability p into an α-bit threshold and performs a bitwise comparison against a uniform random string generated by **SharedCoin**. By combining these Bernoulli samples, we construct a truncated and quantized discrete-Laplace-type noise. For fixed δ, this reduces the shared-coin growth from quadratic to near-logarithmic in $1/\epsilon$.

Verification and Security Analysis. The final layer checks that the noisy output is consistent with the committed inputs and noise shares. By the additive homomorphism of Pedersen commitments, the verifier can audit each opened noisy share via the homomorphic combination of the corresponding input-share and noise-share commitments, without seeing client inputs or individual noise shares. Soundness follows from the auditability of the sampling subprotocols and the binding property of Pedersen commitments; zero knowledge follows by simulating the public transcript from the released DP output and public parameters.

1.3 Related Work

Differentially Private Counting. Classic DP counting mechanisms rely on Laplace, Gaussian, or binomial noise injection to achieve (ϵ, δ)-DP [14, 15, 6]. While they guarantee output privacy, they are *not publicly auditable*; an external verifier cannot confirm from the transcript whether the prescribed noise was honestly sampled.

Distributed and Verifiable DP. Distributed paradigms (e.g., local DP, shuffle DP, secure aggregation) reduce centralized trust but leave hidden randomness hard to audit [21, 11, 7, 16, 5, 8]. Verifiable DP uses commitments and zero-knowledge proofs to audit inputs, noise generation, and aggregation [4]. Prior distributed-noise and MPC-sampling work showed how to generate discrete-Laplace-type noise from biased Bernoulli bits in the binary representation of a geometric variable [13, 28].

Commitments and MPC for Verifiable Sampling. Preprocessing primitives for MPC like daBits [24] and eDaBits [17] provide correlated randomness with consistency between arithmetic shares and binary representations. Publicly auditable MPC based on SPDZ similarly augments additive shares with Pedersen commitments [23] and commitment-opening shares so that Beaver multiplications [3] can be checked from a public transcript [2, 10]. We use this committed-share principle as an idealized interface for verifier-auditable DP noise generation, but our protocols do not require arithmetic-binary conversion or bit-decomposition consistency.

Gap and Our Contribution. The **VBM** and **VDDLM** baselines duplicate independent noise across provers, causing error growth that scales with the number of contributors [4, 27]. Sabater et al. [25] combine shared inputs with verifiably sampled shared noise to release a DP aggregate. Their sampling protocols are analyzed in an identifiable-abort model, but they do not provide an end-to-end Verifiable DP proof for the resulting DP protocol under the client–prover–verifier definition used here. We instantiate this shared-noise approach as **SN-VBM** and **SN-VLM**, using binomial and discrete-Laplace-type noise, respectively. Under ideal committed-triple preprocessing, we establish end-to-end Verifiable DP for both counting mechanisms by proving completeness, soundness, zero knowledge, and differential privacy. We also give explicit utility and complexity guarantees and empirically evaluate their computational efficiency and statistical error.

2 Preliminaries

Notation. We write $x \xleftarrow{\$} S$ to denote that x was uniformly sampled from the given set S. We denote the clients by $Cli_1, \ldots, Cli_n$, the provers by $Pv_1, \ldots, Pv_m$, and the verifier by Vfr. We use $i \in [n]$ to index clients and $j \in [m]$ to index provers. Unless stated otherwise, additions and multiplications of shared values and opening randomnesses are over $\mathbb{Z}_q$ for prime q.

2.1 Additive sharing and Pedersen commitments

Let $\mathbb{G}$ be a cyclic group of prime order q, and let $g, h \in \mathbb{G}$ be generators such that $\log_g h$ is unknown. We use Pedersen commitments $\mathsf{Com}(x; r) := g^x h^r \in \mathbb{G}$, where $x, r \in \mathbb{Z}_q$ [23]. Operations on commitments, such as products, inverses, and exponentiations, are group operations in $\mathbb{G}$.

Pedersen commitments are additively homomorphic [23]: $\mathsf{Com}(x_1; r_1) \cdot \mathsf{Com}(x_2; r_2) = \mathsf{Com}(x_1 + x_2; r_1 + r_2)$. They are perfectly hiding with uniformly random opening randomness and computationally binding under the discrete-logarithm assumption [23]. Informally, hiding means that a commitment reveals no information about the committed value, whereas binding prevents a prover from opening the same commitment to two different values.

Throughout the paper, we write $[[x]]$ for the vector of additive shares $(x_1, \ldots, x_m)$ of $x \in \mathbb{Z}_q$, where $x = \sum_{j=1}^{m} x_j \pmod q$. A Pedersen-committed additive sharing of x is denoted by $([[x]], \vec{s}_x, \vec{X})$, where $\vec{s}_x = (s_{x_1}, \ldots, s_{x_m}), \quad \vec{X} = (X_1, \ldots, X_m), \quad X_j = \mathsf{Com}(x_j; s_{x_j})$. Each prover Pv_j holds (x_j, s_{x_j}), while $\vec{X}$ is public. We also use a zero-knowledge OR-opening proof, denoted by O_{OR}, to prove that a commitment opens to a bit without revealing which bit it is. Formally, it proves membership in $L_{\mathsf{bit}} = \{C \in \mathbb{G} : \exists x \in \{0,1\},\ \exists r \in \mathbb{Z}_q \text{ such that } C = \mathsf{Com}(x; r)\}$. Formal commitment definitions and a concrete OR-opening proof are given in Appendix B.3.

2.2 Verifiable DP and threat model

We first recall the standard definition of differential privacy [15]. Two databases $X = (x_1, \ldots, x_n)$ and $X' = (x'_1, \ldots, x'_n)$ are neighboring, denoted $X \sim X'$, if they differ in the input of at most one client. A randomized mechanism M is (ϵ, δ)-differentially private if, for every pair of neighboring databases $X \sim X'$ and every measurable set S of outputs, $\Pr[M(X) \in S] \leq e^{\epsilon} \Pr[M(X') \in S] + \delta$. In this work, we focus on binary counting queries. Each client Cli_i holds an input bit $x_i \in \{0,1\}$, the database is $X = (x_1, \ldots, x_n)$, and the counting query is $Q(X) = \sum_{i=1}^{n} x_i$. Thus, for neighboring databases $X \sim X'$, we have $|Q(X) - Q(X')| \leq 1$. We denote the released noisy answer by $M_Q(X)$. In this one-bit-per-client model, record-level and user-level adjacency coincide.

We consider a client-prover-verifier protocol for releasing $M_Q(X)$ [4]. Clients secret-share their input bits among the provers and publish commitments to the shares. Randomness-contributing parties supply private bits, and the provers jointly compute shares of one global noise variable. The provers use the committed shares and verifier-auditable shared randomness to compute additive shares of a noisy count. The verifier checks a public transcript of commitments, proof transcripts, opened masked values, and final output openings. If verification accepts, the released value is the sum of the accepted output shares; otherwise, the protocol rejects.

Following the verifiable-DP framework of Biswas and Cormode [4], we require completeness, soundness, and zero knowledge. Informally, completeness means that an honest execution is accepted except with negligible probability; soundness means that malicious provers cannot make the verifier accept an output inconsistent with the prescribed DP mechanism except with negligible probability; and zero knowledge means that the verifier's view can be simulated from the released DP output and public parameters. A protocol is verifiably differentially private for the counting query if every accepted output is distributed according to the target (ϵ, δ)-DP mechanism, except with negligible soundness error.

We allow any subset of provers to be malicious. For soundness, the verifier follows the prescribed verification algorithm, since acceptance or rejection is the verifier's decision. For zero knowledge, the verifier may be malicious and may collude with corrupted provers. Input privacy against the provers requires at least one honest prover, since a coalition of all provers can reconstruct additive shares. For verifier-auditable randomness generation, we assume that at least one randomness-contributing party is honest; otherwise, unbiased randomness generation cannot be guaranteed in the plain model. Formal definitions of completeness, soundness, and zero knowledge are given in Appendix B.1.

2.3 Verifier-auditable committed Beaver triples

Our protocols use verifier-auditable committed Beaver triples as preprocessing. This interface follows the publicly auditable MPC approach of augmenting additive shares and Beaver triples with Pedersen commitments and commitment-opening shares, so that later multiplication steps can be checked from the public transcript [2]. We model this preprocessing by an ideal functionality $\mathcal{F}_{\mathsf{VCT}}$.

On each invocation, $\mathcal{F}_{\mathsf{VCT}}$ samples $u, v \xleftarrow{\$} \mathbb{Z}_q$, sets $w := uv$, and outputs committed additive sharings

$$([[u]], \vec{\mu}, \vec{U}), \quad ([[v]], \vec{\nu}, \vec{V}), \quad ([[w]], \vec{\omega}, \vec{W}),$$

where $U_j = \mathsf{Com}(u_j; \mu_j)$, $V_j = \mathsf{Com}(v_j; \nu_j)$, and $W_j = \mathsf{Com}(w_j; \omega_j)$. Prover Pv_j receives the local openings $(u_j, \mu_j), (v_j, \nu_j), (w_j, \omega_j)$, while $\vec{U}, \vec{V}, \vec{W}$ are public. Higher-level protocols open only masked differences against these triples and derive output commitments from the public transcript, allowing the verifier to audit consistency without learning the secret shares.

The full ideal-functionality description is given in Appendix B.4. We use $\mathcal{F}_{\mathsf{VCT}}$ as an ideal preprocessing primitive and count each invocation of $\mathcal{F}_{\mathsf{VCT}}.\mathsf{Triple}$ as one committed Beaver triple in the performance analysis.

3 Verifier-Auditable Additive Shared Coin Generation

3.1 Committed Multiplication

We use a committed multiplication protocol, denoted by

$$([[c]], \vec{s_c}, \vec{C}) = \textbf{ComMult}\big(([[a]], \vec{s_a}, \vec{A}), ([[b]], \vec{s_b}, \vec{B})\big),$$

which takes two Pedersen-committed additive sharings of a and b over $\mathbb{Z}_q$ and outputs a Pedersen-committed additive sharing of $c = ab$. The protocol is the standard Beaver multiplication identity applied to Pedersen-committed additive shares, following the publicly auditable MPC multiplication-and-audit approach [3, 2]. It uses one verifier-auditable committed Beaver triple from $\mathcal{F}_{\mathsf{VCT}}$, opens only the masked differences $d = a - u$ and $e = b - v$, and locally derives the output shares and their public commitments.

Both the verifier and all provers check that the opened masked differences are consistent with the input commitments and the triple commitments. The output commitments $\vec{C}$ are then publicly derived from the triple commitments and the opened values d, e. Hence, by the hiding property of Pedersen commitments, the commitments reveal no additional information about the underlying shares. By the binding property, a corrupted prover cannot change its output share c_j while preserving consistency with the public commitment C_j. The complete line-by-line description, correctness proof, and security discussion of **ComMult** implementation are given in Appendix C.

3.2 Additive shared coin generation

We now describe our verifier-auditable additive shared coin-generation protocol, **SharedCoin**. The protocol outputs a Pedersen-committed additive sharing $([[a]], \vec{\rho}, \vec{A})$ of a random bit $a \in \{0, 1\}$, so that the bit remains hidden from the verifier while the public commitment vector $\vec{A}$ enables auditability.

The protocol combines prover-contributed private bits $b_1, \ldots, b_m$ with a verifier-contributed bit c and outputs $a = b_1 \oplus \cdots \oplus b_m \oplus c$. The prover-side XOR is evaluated over $\mathbb{Z}_q$ using

$$x \oplus y = x + y - 2xy \quad \text{for } x, y \in \{0, 1\},$$

where the multiplication term is computed by **ComMult**. The verifier's bit is opened only after the prover-side XOR chain has been fixed, ensuring that any honest randomness-contributing party makes the final bit unbiased.

Figure 1 describes the full **SharedCoin** protocol. Lines 1–4 generate committed additive sharings of the prover-contributed bits and use O_{OR} to prove that each reconstructed commitment $B_i = \prod_j C_{ij}$ opens to a bit. Lines 5–6 commit the verifier's bit c before the prover-side XOR chain is evaluated.

Lines 7–9 iteratively maintain a committed additive sharing $([[t^{(k)}]], \vec{s}_t^{(k)}, \vec{T}^{(k)})$ of $t^{(k)} = b_1 \oplus \cdots \oplus b_k$ using **ComMult**. Lines 10–12 open c and fold it into the final committed output: if $c = 0$, the protocol outputs the current XOR sharing, and if $c = 1$, it outputs a committed sharing of $1 - t^{(m)}$. Hence the output is a Pedersen-committed additive sharing $([[a]], \vec{\rho}, \vec{A})$ of $a = b_1 \oplus \cdots \oplus b_m \oplus c$.

The O_{OR} checks are performed by both the verifier and all provers; the protocol aborts if any party rejects. Therefore, if at least one checking party is honest, non-bit commitments cannot pass the public bitness checks except with negligible probability.

3.3 Security guarantees

We state the main guarantees of the shared coin-generation protocol below and defer the proofs to Appendix D.1. The guarantees cover honest-execution correctness, unbiasedness under one honest randomness-contributing party, auditability via commitments, and zero knowledge.

	Verifier (Vfr **)**		**Prover (** Pv_j **)**
1:			For each $i \in [m] \setminus \{j\}$:
	$(C_{ij})_{i,j\in[m],\, i\neq j}$	$\xleftarrow{(C_{ij})_{i,j\in[m],\, i\neq j}}$	$b_{ij}, r_{ij} \xleftarrow{\$} \mathbb{Z}_q, \quad C_{ij} := \mathsf{Com}(b_{ij}; r_{ij})$. Send (b_{ij}, r_{ij}) to Pv_i by secure channel.
2:			For each $i \in [m] \setminus \{j\}$: Check $C_{ji} \stackrel{?}{=} \mathsf{Com}(b_{ji}; r_{ji})$.
3:	$(C_{jj})_{j\in[m]}$	$\xleftarrow{(C_{jj})_{j\in[m]}}$	Sample $b_j \xleftarrow{\$} \{0,1\}, \quad r_{jj} \xleftarrow{\$} \mathbb{Z}_q$. Compute $b_{jj} := b_j - \sum_{i\neq j} b_{ji}$, $C_{jj} := \mathsf{Com}(b_{jj}; r_{jj})$.
4:	$\forall i \in [m]$ Send B_i to O_{OR}, $\forall i \in [m]$ Check $O_{\mathsf{OR}}(B_i) = 1$.	$B_j := \prod_{i=1}^m C_{ji}$ O_{OR}	Send $(b_j; \sum_{i=1}^m r_{ji})$ to O_{OR}. For each $i \in [m] \setminus \{j\}$: Check $O_{\mathsf{OR}}(B_i) = 1$.
5:	Sample $c \xleftarrow{\$} \{0,1\}, \quad r_c \xleftarrow{\$} \mathbb{Z}_q$. Compute $C_c := \mathsf{Com}(c; r_c)$.	$\xrightarrow{C_c}$	
6:	Send $(c; r_c)$ to O_{OR}.	O_{OR}	Send C_c to O_{OR} and check $O_{\mathsf{OR}}(C_c) = 1$.
7:	$\vec{T}^{(0)} = (1, \ldots, 1)$		$[[t^{(0)}]] = (0, \ldots, 0), \quad \vec{s}_t^{(0)} = (0, \ldots, 0)$.
	For $k \in [m]$, Perform 8-9:		
8:	Send $\vec{T}^{(k-1)}, \vec{C}_k$. Receive $\vec{L}^{(k)}$.	**ComMult**	Send $(t_j^{(k-1)}, s_{t_j}^{(k-1)}), (b_{kj}, r_{kj})$. Receive $(l_j^{(k)}, \lambda_j^{(k)})$.
9:	For each $j \in [m]$, compute $T_j^{(k)} = T_j^{(k-1)} \cdot C_{kj} \cdot (L_j^{(k)})^{-2}$.		Compute $t_j^{(k)} = t_j^{(k-1)} + b_{kj} - 2l_j^{(k)}$, $s_{t_j}^{(k)} = s_{t_j}^{(k-1)} + r_{kj} - 2\lambda_j^{(k)}$.
10:	Open (c, r_c)	$\xrightarrow{(c,r_c)}$	Check $C_c \stackrel{?}{=} \mathsf{Com}(c; r_c)$.
11:	For each $j \in [m]$, compute A_j. If $c = 0 : A_j := T_j^{(m)}$. If $c = 1 : A_j := g^{\delta_{j1}} (T_j^{(m)})^{-1}$.		Compute a_j, ρ_j. If $c = 0 : a_j := t_j^{(m)}, \quad \rho_j := s_{t_j}^{(m)}$. If $c = 1 : a_j := \delta_{j1} - t_j^{(m)}, \quad \rho_j := -s_{t_j}^{(m)}$.
12:	$\vec{A}$	Output	(a_j, ρ_j)

Figure 1: Verifier-auditable additive shared coin-generation protocol, **SharedCoin**.

Theorem 1 (Completeness and committed-output correctness)**.** *If all parties honestly execute **SharedCoin**, it does not abort and outputs a Pedersen-committed additive sharing $([[a]], \vec{\rho}, \vec{A})$ such that*

$$a = b_1 \oplus \cdots \oplus b_m \oplus c, \qquad \forall j \in [m],\ A_j = \mathsf{Com}(a_j; \rho_j).$$

Theorem 2 (Unbiasedness and auditability)**.** *Assume that at least one randomness-contributing party among $Pv_1, \ldots, Pv_m, Vfr$ is honest. Under the soundness of O_{OR}, the correctness of **ComMult**, and the hiding and computational binding properties of Pedersen commitments, upon successful completion of **SharedCoin**, except with negligible probability, either $([[a]], \vec{\rho}, \vec{A})$ is a Pedersen-committed additive sharing of a bit $a \in \{0,1\}$ that is computationally indistinguishable from uniform, or the provers' internal output state is inconsistent with $\vec{A}$.*

Theorem 3 (Zero knowledge)**.** *For any PPT malicious verifier possibly colluding with any subset of corrupted provers, the view of **SharedCoin** can be simulated without the honest provers' private randomness and output shares, assuming the hiding property of Pedersen commitments, the zero-knowledge property of O_{OR} and **ComMult**.*

The proofs of Theorems 1, 2, and 3 are deferred to Appendix D.1.

4 Shared-Noise Verifiable DP Mechanisms

4.1 Shared-Noise Verifiable Binomial Mechanism (SN-VBM)

We instantiate our shared-noise framework with centered binomial noise. Given privacy parameters ϵ, δ, the mechanism first sets the number of coins n_b, then invokes the additive shared coin-generation protocol n_b times. Let $([[a_i]], \vec{\rho}_i, \vec{A}_i)$ denote the i-th generated committed additive-shared coin, where

	Verifier (Vfr)		**Prover** (Pv_j)
1:	Agree on pp; set n_b.	Setup $n_b := \lceil 100 \ln(2/\delta)/\epsilon^2 \rceil$	Agree on pp; set n_b.
2:	Receive $(X_{ij})_{i\in[n],j\in[m]}$ from each client Cli_i.	$X_{ij} := \mathsf{Com}(x_{ij}; r_{ij})$	Receive (x_{ij}, r_{ij}) from each client Cli_i.
3:	For each $i \in [n]$, check $O_{\mathsf{OR}}(X_i) = 1$. Exclude Cli_i if the check fails.	$X_i := \prod_{j=1}^{m} X_{ij}$ O_{OR}	Participate in client-input verification.
4:	Receive $(A_{ij})_{i\in[n_b],j\in[m]}$.	**SharedCoin**	Receive $(a_{ij}, \rho_{ij})_{i\in[n_b]}$.
5:	Receive (y_j, z_j).	$\xleftarrow{(y_j,z_j)}$	Compute $y_j := \sum_{i=1}^{n} x_{ij} + \sum_{i=1}^{n_b} a_{ij}$, $z_j := \sum_{i=1}^{n} r_{ij} + \sum_{i=1}^{n_b} \rho_{ij}$.
6:	For each $j \in [m]$, check $\prod_{i=1}^{n} X_{ij} \cdot \prod_{i=1}^{n_b} A_{ij} \stackrel{?}{=} \mathsf{Com}(y_j; z_j)$.	Verification	
7:	$M_Q(X) := \sum_{j=1}^{m} y_j - \frac{n_b}{2}$.	Output	

Figure 2: Shared-Noise Verifiable Binomial Mechanism (**SN-VBM**).

$a_i \in \{0, 1\}$. The provers add these shared coins to their local shares of the client count, and the verifier checks the resulting aggregate against the public commitments. The final released value is $M_Q(X) = Q(X) + \sum_{i=1}^{n_b} a_i - \frac{n_b}{2}$, which is the exact count plus centered binomial noise.

Figure 2 describes the full **SN-VBM** protocol. Lines 1–3 set the public parameters pp and the mechanism parameter n_b, and verify client inputs. Each client secret-shares its bit x_i among the provers and publishes commitments to the shares. The verifier checks, through O_{OR}, that the reconstructed client commitment $X_i = \prod_j X_{ij}$ opens to a bit. If a malicious client submits shares whose reconstructed value is not a bit, X_i cannot pass the verification except with negligible probability under the soundness of O_{OR}. A client whose proof fails is excluded from the aggregation.

Line 4 generates n_b independent verifier-auditable additive-shared coins using the shared coin-generation protocol from Figure 1. Lines 5–6 aggregate the committed input shares and committed noise shares. For each prover Pv_j, the verifier checks that the reported pair (y_j, z_j) is consistent with the product of the corresponding input and noise commitments. Line 7 releases the centered noisy count.

By homomorphism of Pedersen commitments, the verification equation in Line 6 ensures that each accepted y_j is committed to the sum of prover Pv_j's input shares and noise shares. Therefore, if all checks pass,

$$M_Q(X) = \sum_{j=1}^{m} y_j - \frac{n_b}{2} = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n_b} a_i - \frac{n_b}{2} \sim Q(X) + \mathsf{Bin}(n_b, 1/2) - \frac{n_b}{2}.$$

Theorem 4 (Verifiability of SN-VBM)**.** *Assume the security of **SharedCoin**, the hiding and computational binding properties of Pedersen commitments, and the zero-knowledge property of O_{OR}. Then **SN-VBM** satisfies perfect completeness, negligible soundness, and computational zero knowledge.*

Lemma 1 (Parameter setting for SN-VBM [4, 19])**.** *For $0 < \epsilon \leq 1$ and $0 < \delta < 1$, let $n_b = \left\lceil \frac{100}{\epsilon^2} \ln \frac{2}{\delta} \right\rceil$. Then the output distribution of **SN-VBM** is (ϵ, δ)-differentially private.*

Combining Theorem 4 with Lemma 1, **SN-VBM** is a verifiably differentially private protocol for counting. The proofs are deferred to Appendix D.2.

4.2 Shared-Noise Verifiable Laplace Mechanism (SN-VLM)

We next instantiate our framework with discrete-Laplace-type noise. The distributional idea follows prior bitwise constructions of discrete-Laplace-type noise, where the bits of a geometric random variable are sampled independently using biased Bernoulli coins [13, 28]. Our contribution is to realize these Bernoulli samples in verifier-auditable additive-shared form. Given a public target probability $p \in (0, 1)$ and accuracy parameter α, the sampler first quantizes p into an α-bit threshold

$$\lfloor p \cdot 2^{\alpha} \rfloor = \sum_{k=0}^{\alpha-1} p_k 2^k, \qquad \forall k \in [0, \alpha - 1],\ p_k \in \{0, 1\}.$$

	Verifier (Vfr)		**Prover** (Pv_j)
1:	Compute $(p_0, \ldots, p_{\alpha-1}) \in \{0,1\}^\alpha$.	$\lfloor p \cdot 2^\alpha \rfloor = \sum_{k=0}^{\alpha-1} p_k 2^k$	Compute $(p_0, \ldots, p_{\alpha-1}) \in \{0,1\}^\alpha$.
2:	Receive $(A_{kj})_{k\in[0,\alpha-1], j\in[m]}$.	**SharedCoin**	Receive $(a_{kj}, \rho_{kj})_{k\in[0,\alpha-1]}$.
3:	$\vec{A}'^{(0)} = (1, \ldots, 1)$.		$[[a'^{(0)}]] = (0, \ldots, 0), \quad \vec{\rho}'^{(0)} = (0, \ldots, 0)$.
	For $k \in [0, \alpha-1]$, perform 4–6:		
4:	For each $j \in [m]$, compute $S_j^{(k)}$. $S_j^{(k)} := g^{\delta_{j1} p_k} \cdot A_{kj}^{1-2p_k}$.		$s_j^{(k)} := \delta_{j1} p_k + (1-2p_k) a_{kj}$. $r_{s_j}^{(k)} := (1-2p_k)\rho_{kj}$.
5:	Send $\vec{A}'^{(k)}, \vec{S}^{(k)}$. Receive $\vec{L}^{(k)}$.	**ComMult**	Send $(a_j'^{(k)}, \rho_j'^{(k)}), (s_j^{(k)}, r_{s_j}^{(k)})$. Receive $(l_j^{(k)}, \lambda_j^{(k)})$.
6:	For each $j \in [m]$, compute $A_j'^{(k+1)} := (S_j^{(k)})^{p_k} \cdot A_j'^{(k)} \cdot (L_j^{(k)})^{-1}$.		$a_j'^{(k+1)} := p_k s_j^{(k)} + a_j'^{(k)} - l_j^{(k)}$. $\rho_j'^{(k+1)} := p_k r_{s_j}^{(k)} + \rho_j'^{(k)} - \lambda_j^{(k)}$.
7:	$\vec{A}' := \vec{A}'^{(\alpha)}$.	Output	$(a_j', \rho_j') := (a_j'^{(\alpha)}, \rho_j'^{(\alpha)})$.

Figure 3: Verifier-auditable additive-shared Bernoulli sampler, **SharedBern**.

The threshold is computed by deterministic public preprocessing with downward rounding. It then generates α independent shared coins and compares the resulting α-bit uniform value with the threshold over committed arithmetic.

Figure 3 describes the full protocol. Lines 1–2 quantize the target probability and generate the α shared random bits used for comparison. Lines 3–6 maintain a committed additive sharing of the running comparison result. At bit position k, the value $s^{(k)} = a_k \oplus p_k$ detects whether the sampled bit and threshold bit differ. The update overwrites the running output with p_k whenever a difference is detected, so the final value is determined by the most significant differing bit. Thus the final output is a committed additive sharing of a Bernoulli random variable with probability $\lfloor p \cdot 2^\alpha \rfloor / 2^\alpha$.

We will denote the protocol as **SharedBern**(p, α) for convenience.

Lemma 2 (Bernoulli-sampler guarantee)**.** *Assume the security of **SharedCoin** and **ComMult**, and assume that at least one randomness-contributing party in each invocation of **SharedCoin** is honest. Under the soundness of O_{OR}, the correctness of **ComMult**, and the hiding and computational binding properties of Pedersen commitments, upon successful completion of **SharedBern**, except with negligible probability, either $([[a']], \vec{\rho}', \vec{A}')$ is a Pedersen-committed additive sharing of a bit $a' \in \{0,1\}$ whose distribution is computationally indistinguishable from* Bernoulli$(\lfloor p 2^\alpha \rfloor / 2^\alpha)$*, or the provers' internal output state is inconsistent with the public commitment vector $\vec{A}'$. Moreover, **SharedBern** satisfies perfect completeness and computational zero knowledge.*

We now use the Bernoulli sampler to construct quantized and truncated discrete-Laplace-type noise. Let $s = e^{-\epsilon}$. For $k = 0, \ldots, t-1$, the mechanism samples a Bernoulli bit a_k with probability $p_k = s^{2^k}/(1+s^{2^k})$, and accumulates the weighted sum $u := \sum_{k=0}^{t-1} a_k 2^k$, which gives a truncated geometric component. A final Bernoulli bit with probability $s/(1+s)$ determines the sign, producing noise supported on $[-2^t, 2^t - 1]$.

Figure 4 describes the full **SN-VLM** protocol. Lines 1–3 are identical to **SN-VBM**: clients secret-share their inputs and prove bit validity through O_{OR}. Lines 4–6 sample a committed additive-shared truncated geometric variable by accumulating independently sampled Bernoulli bits with weights 2^k. Lines 7–9 sample a sign bit and convert the geometric variable into a symmetric truncated discrete Laplace noise share. Lines 10–12 aggregate the committed input shares and noise shares, check commitment consistency, and release the noisy count. If all checks pass, then $M_Q(X) = Q(X) + n$, where n is the truncated and quantized discrete Laplace noise generated by the protocol.

Theorem 5 (Verifiability of SN-VLM)**.** *Assume the security of **SharedBern**, **ComMult**, the hiding and computational binding properties of Pedersen commitments, and the zero-knowledge property of O_{OR}. Then **SN-VLM** satisfies perfect completeness, negligible soundness, and computational zero knowledge.*

Lemma 3 (Parameter setting for SN-VLM)**.** *Let* $s = e^{-\epsilon}, \quad t = \left\lceil \log_2 \left(1 + \frac{\ln(2/\delta)}{\epsilon}\right) \right\rceil, \quad \alpha = 3 + \left\lceil \log_2 \frac{t+1}{\delta} \right\rceil$ *. Then the protocol in Figure 4 is (ϵ, δ)-differentially private for $\epsilon \le 1$ and $\delta \in (0,1)$.*

	Verifier (Vfr)		**Prover** (Pv_j)
1:	Agree on pp, s, t, α.	Setup	Agree on pp, s, t, α.
2:	Receive $(X_{ij})_{i\in[n],j\in[m]}$.	$\xleftarrow{(X_{ij})_{i\in[n],j\in[m]}}$	Receive (x_{ij}, r_{ij}) from each client Cli_i. $X_{ij} := \mathsf{Com}(x_{ij}; r_{ij})$.
3:	For each $i \in [n]$, check $O_{\mathsf{OR}}(X_i) = 1$. Exclude Cli_i if the check fails.	$X_i := \prod_{j=1}^{m} X_{ij}$ O_{OR}	Participate in client-input verification.
4:	$\vec{U}^{(0)} = (1, \ldots, 1)$.		$[[u^{(0)}]] = (0, \ldots, 0), \quad \vec{s_u}^{(0)} = (0, \ldots, 0)$.
	For $k \in [0, t-1]$, perform 5–6:		
5:	Receive $(A_{kj})_{j\in[m]}$.	$p_k := s^{2^k}/(1+s^{2^k})$. **SharedBern**$(p_k, \alpha)$	Receive (a_{kj}, ρ_{kj}).
6:	For each $j \in [m]$, compute $U_j^{(k+1)} := U_j^{(k)} \cdot (A_{kj})^{2^k}$.		Compute $u_j^{(k+1)} := u_j^{(k)} + 2^k a_{kj}$, $s_{u_j}^{(k+1)} := s_{u_j}^{(k)} + 2^k \rho_{kj}$.
7:	Receive $(B_j)_{j\in[m]}$.	**SharedBern**$(\frac{s}{1+s}, \alpha)$	Receive (b_j, ρ_{b_j}).
8:	Send $\vec{U}^{(t)}, \vec{B}$. Receive $\vec{L}$.	**ComMult**	Send $(u_j^{(t)}, s_{u_j}^{(t)}), (b_j, \rho_{b_j})$. Receive (l_j, λ_j).
9:	For each $j \in [m]$, compute $N_j := U_j^{(t)} \cdot (B_j)^{-1} \cdot (L_j)^{-2}$.		Compute $n_j := u_j^{(t)} - b_j - 2l_j$, $s_{n_j} := s_{u_j}^{(t)} - \rho_{b_j} - 2\lambda_j$.
10:	Receive (y_j, z_j).	$\xleftarrow{(y_j, z_j)}$	$y_j := \sum_{i=1}^{n} x_{ij} + n_j$. $z_j := \sum_{i=1}^{n} r_{ij} + s_{n_j}$.
11:	For each $j \in [m]$, check $\prod_{i=1}^{n} X_{ij} \cdot N_j \stackrel{?}{=} \mathsf{Com}(y_j; z_j)$.		
12:	$M_Q(X) := \sum_{j=1}^{m} y_j$.	Output	

Figure 4: Shared-Noise Verifiable Laplace Mechanism (**SN-VLM**).

The exact truncated mechanism is (ϵ, δ_t)-DP, where $\delta_t = (1-s)s^{2^t-1}/[(1+s)(1-s^{2^t})]$. Quantization changes its distribution by total variation at most $\eta = (t+1)2^{-\alpha}$, giving $(\epsilon, \delta_t + (e^\epsilon + 1)\eta)$-DP. The selected parameters bound both terms by $\delta/2$. We use signed output decoding in $(-q/2, q/2)$ and require $q > 2(n + 2^t)$ to prevent modular wraparound.

Combining Theorem 5 with Lemma 3, SN-VLM is a verifiably differentially private protocol for counting. The proofs are deferred to Appendix D.3.

5 Performance Overview

We defer the full performance evaluation to Appendix A. Our experiments compare SN-VBM and SN-VLM along three axes: preprocessing load, verifiable coin-generation time, and statistical error. For preprocessing, we report the number of ideal committed Beaver triples consumed by each mechanism, treating triple generation as an abstract preprocessing functionality.

Theorem 6 (Primitive complexity and online communication)**.** *Fix the security parameter and the number of provers m. The input stage publishes nm share commitments and verifies n reconstructed input bits for both **SN-VBM** and **SN-VLM**.*

*Since each invocation of **SharedCoin** uses m committed triples, **SN-VBM** requires n_b shared coins and mn_b triples. Under the standard binomial calibration, this scales as $\Theta(\epsilon^{-2})$ for fixed δ. In contrast, **SN-VLM** calls a verifier-auditable Bernoulli sampler $(t+1)$ times, each using α shared coins, so its shared-coin complexity is $(t+1)\alpha = \Theta(\log \epsilon^{-1} \log\log \epsilon^{-1})$ for fixed δ. Its committed-triple count is $(m+1)(t+1)\alpha + 1$.*

The noise-stage bitness-proof counts are $(m+1)n_b$ and $(m+1)(t+1)\alpha$, respectively. Excluding constant-size proof transcripts, online communication cost is $O(nm + m^2 n_b)$ and $O(nm + m^2 t\alpha)$, respectively.

Committed triple-generation cost is excluded from online complexity and measured runtime.

We also evaluate the online cost of verifiable randomness generation using an implementation of Pedersen commitments over the Ristretto curve, with $m = 2$ and $\delta = 10^{-10}$. The measurements separate client input verification, coin generation, multiplication, aggregation, and final verification. They show that coin generation is the dominant cost for the binomial-based mechanisms, because the number of required shared coins grows quadratically in $1/\epsilon$. By contrast, SN-VLM requires only a near-logarithmic number of shared coins, leading to substantially smaller randomness-generation time in high-privacy regimes.

Finally, we compare utility by simulating the noise distributions under the same privacy parameters. Since the released count has the form $M_Q(X) = Q(X) + Z$, the absolute error is exactly $|Z|$. We report both mean absolute error and the empirical 95th-percentile absolute error over 10^5 independent noise samples. For fixed $m = 2$ and $\delta = 10^{-10}$, all mechanisms show the expected ϵ^{-1} error scaling. However, in the evaluated regimes, **SN-VLM** has much smaller empirical error than the binomial mechanisms, while also reducing the cryptographic preprocessing and online randomness-generation costs.

Table 2 in Appendix A.1 compares **VBM**, **SN-VBM**, **VDDLM** and **SN-VLM** in the aspect of mechanism and their expected absolute error: shared noise removes the additional $\sqrt{m}$ expected-error factor at the cost of cross-prover interaction and preprocessing. **SN-VLM**'s computational improvement over **SN-VBM** comes from its noise generation mechanism. Appendix A.3 gives high-probability errors $O(\epsilon^{-1}\sqrt{\ln(1/\delta)\ln(1/\gamma)})$ for **SN-VBM** and $O(\epsilon^{-1}\ln(1/\gamma))$ for **SN-VLM**, each with probability at least $1 - \gamma$.

6 Conclusion and Limitations

We introduced a shared-noise framework for verifiable differentially private counting. Our main contribution is the construction and end-to-end analysis of verifiable counting mechanisms that avoid duplicating privacy noise across provers. Existing multi-prover approaches that add independent prover-side noise incur a $\sqrt{m}$ growth in aggregate error. In contrast, our protocols generate a single verifier-auditable noise sample in additive-shared form: the provers hold shares of the noise, while Pedersen commitments allow the verifier to audit consistency without learning the noise itself. Input privacy against the provers requires at least one honest prover, and unbiased randomness generation requires at least one honest randomness-contributing party.

We instantiated this idea as **SN-VBM** and **SN-VLM**. While **SN-VBM** directly replaces duplicated binomial noise with shared binomial noise, **SN-VLM** uses an auditable Bernoulli sampler to realize truncated and quantized discrete-Laplace-type noise. Our evaluation indicates that **SN-VLM** is more efficient in the mechanism-level metrics considered here: although one invocation of **SharedCoin** is slower than the baseline private-coin procedure, **SN-VLM** requires far fewer shared coins, reduces estimated coin-generation time, and achieves more than an order-of-magnitude smaller empirical error than the binomial-based mechanisms in the evaluated privacy regimes. Asymptotically, the client validation scales with nm and shared-coin communication with m^2. **SN-VBM**'s $n_b = \Theta(\epsilon^{-2}\log(1/\delta))$ makes **SN-VLM** more preferable under strict privacy.

The main limitation is that our protocols rely on committed multiplication and therefore require verifier-auditable committed Beaver triples as preprocessing. We model this through the ideal functionality $\mathcal{F}_{\mathsf{VCT}}$ and report triple consumption as a preprocessing metric; implementing and benchmarking a concrete committed triple-generation protocol remains for future work.

Our experiments also simulate all parties sequentially on a single machine. Accordingly, the reported measurements capture the aggregate prover- and verifier-side cryptographic computation required to generate the protocol transcript and help identify the dominant computational components of the mechanisms. They do not capture network latency, bandwidth limits, or synchronization overheads in real deployments.

Multiple binary submissions per physical client can reuse the validity proof for each logical input, but user-level privacy requires a public contribution bound and noise calibrated to the resulting sensitivity. Coordinate-wise histograms additionally require a cross-coordinate one-hot or bounded-frequency proof. End-to-end extension to these settings remains for future work.

Acknowledgements

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2026-25484051), the New Faculty Startup Fund of Seoul National University, and the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2025-02214654). Sehyeon Park was additionally supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (No. RS-2026-25570665).

References

[1] John M Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2867–2867, 2018.

[2] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In *International Conference on Security and Cryptography for Networks*, pages 175–196. Springer, 2014.

[3] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Annual international cryptology conference*, pages 420–432. Springer, 1991.

[4] Ari Biswas and Graham Cormode. Interactive proofs for differentially private counting. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1919–1933, 2023.

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[6] Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. In *Annual Conference on Neural Information Processing Systems*, 2020.

[7] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Advances in Cryptology – EUROCRYPT 2019*, volume 11476 of *Lecture Notes in Computer Science*, pages 375–403. Springer, 2019.

[8] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 259–282. USENIX Association, 2017.

[9] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

[10] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual cryptology conference*, pages 643–662. Springer, 2012.

[11] John Duchi, Martin J Wainwright, and Michael I Jordan. Local privacy and minimax bounds: Sharp rates for probability estimation. *Advances in neural information processing systems*, 26, 2013.

[12] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[13] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[15] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3-4):211–487, 2014.

[16] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[17] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. Improved primitives for mpc over mixed arithmetic-binary circuits. In *Annual international cryptology conference*, pages 823–852. Springer, 2020.

[18] Joseph Ficek, Wei Wang, Henian Chen, Getachew Dagne, and Ellen Daley. Differential privacy in health research: A scoping review. *Journal of the American Medical Informatics Association (JAMIA)*, 28(10):2269–2276, 2021.

[19] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 463–488. Springer, 2021.

[20] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, 1985.

[21] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[22] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 830–842, 2016.

[23] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[24] Dragos Rotaru and Tim Wood. Marbled circuits: Mixing arithmetic and boolean circuits with active security. In *International Conference on Cryptology in India*, pages 227–249. Springer, 2019.

[25] César Sabater, Florian Hahn, Andreas Peter, and Jan Ramon. Private sampling with identifiable cheaters. *Proceedings on privacy enhancing technologies*, 2023(2), 2023.

[26] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[27] Haochen Sun and Xi He. Vddp: Verifiable distributed differential privacy under the client-server-verifier setup. *arXiv preprint arXiv:2504.21752*, 2025.

[28] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. Securely sampling discrete gaussian noise for multi-party differential privacy. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2262–2276, 2023.

[29] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's scope as verifiable differentially private counting with shared-noise generation. The main claims are supported by the formal privacy/verifiability theorems and the performance evaluation.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the model assumptions, including the need for at least one honest randomness-contributing party, the use of ideal committed Beaver-triple preprocessing, and the fact that experiments simulate parties sequentially on one machine rather than modeling network latency.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper states the cryptographic and privacy assumptions, including Pedersen commitment binding, additive sharing, honest-party assumptions, and the ideal preprocessing functionality. Formal proofs of differential privacy, completeness, soundness, and zero knowledge are provided in the main text and appendix.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the mechanisms evaluated, privacy parameters, number of parties, commitment scheme, preprocessing metrics, runtime decomposition, and error metrics. The experiments do not rely on private datasets; the evaluated outputs are determined by protocol parameters and sampled noise.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will upload the code used for the experiments as part of the supplementary material, together with instructions for reproducing the main preprocessing, runtime, and error-comparison results. The experiments use synthetic/no-data settings, so no external dataset release is required.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper has no training or test splits. It specifies the relevant experimental parameters, including $m = 2$, $\delta = 10^{-10}$, the evaluated ϵ values, the Ristretto-based Pedersen commitment implementation, and the runtime/error metrics.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The Performance section reports aggregate empirical statistics over repeated executions, including mean absolute error and high-percentile absolute error, to summarize both typical and tail behavior. The runtime comparison is based on averaged measurements of the dominant randomness-generation component.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The Performance section provides the compute setting used for the experiments, including that all parties are simulated sequentially on a single machine, and reports the running time of the main protocol components. These details, together with the supplementary code, are sufficient to reproduce the reported preprocessing, runtime, and error-comparison experiments.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is theoretical and experimental research on privacy-preserving and publicly auditable aggregate computation. It does not involve human subjects, sensitive datasets, or deployment on real users.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the positive impact of improving trust and accountability in private aggregate analysis. It also notes limitations and possible risks, including reliance on model assumptions and the danger that verifiability may be misunderstood as guaranteeing correctness beyond the specified protocol and threat model.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models, scraped datasets, or generative systems. The proposed mechanisms are cryptographic/privacy-preserving protocols for aggregate counting.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the relevant prior work and baseline implementation, including the Verifiable Binomial Mechanism of Biswas and Cormode. No external datasets or pretrained models are used.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The submission does not introduce a new dataset, pretrained model, or benchmark asset. The main contributions are mechanisms, protocols, analyses, and experimental evaluations.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research, so IRB approval is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methods, proofs, protocols, and experiments do not use LLMs as an important, original, or non-standard component.

Table 1: Parameter settings and preprocessing load for $\delta = 10^{-10}$ and $m = 2$. The coin count denotes the number of invocations of **SharedCoin**, i.e., the number of verifier-auditable additive-shared coins.

	SN-VBM		**SN-VLM**			
ϵ	Coins (n_b)	Triples	t	α	Coins	Triples
0.001	2,371,899,812	4,743,799,624	15	41	656	1,969
0.002	592,974,953	1,185,949,906	14	41	615	1,846
0.005	94,875,993	189,751,986	13	41	574	1,723
0.01	23,718,999	47,437,998	12	40	520	1,561
0.02	5,929,750	11,859,500	11	40	480	1,441
0.05	948,760	1,897,520	9	40	400	1,201
0.1	237,190	474,380	8	40	360	1,081
0.2	59,298	118,596	7	40	320	961
0.5	9,488	18,976	6	40	280	841
1	2,372	4,744	5	39	234	703

A Performance

We evaluate our Shared-Noise Verifiable Binomial and Laplace mechanisms along three axes: preprocessing load, coin-generation time, and error. The preprocessing-load analysis counts the number of ideal committed Beaver triples consumed by each mechanism. The coin-generation experiment measures the dominant online cost of generating verifiable randomness. The error comparison evaluates the utility of the released count under the same privacy parameters.

In all our experiments, we instantiate the Pedersen commitment scheme over the Ristretto curve. Unless otherwise stated, we fix the number of computing parties to $m = 2$ and set $\delta = 10^{-10}$. We vary

$$\epsilon \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}.$$

A.1 Preprocessing Load

We first compare the preprocessing load of our two mechanisms by counting the number of committed Beaver triples consumed by each protocol. Since our construction abstracts committed triple generation through the ideal functionality $\mathcal{F}_{\mathsf{VCT}}$, our implementation uses a dummy triple generator that preserves the interface and commitment-consistency checks. Thus, we report triple consumption as a mechanism-level preprocessing metric rather than as a benchmark of a particular triple-generation protocol.

SharedCoin consumes m committed Beaver triples, since it computes the XOR of the m prover-sampled bits via committed multiplication. Therefore, **SN-VBM** uses

$$\#\mathsf{Coins}_{\mathsf{Bin}} = n_b, \qquad \#\mathsf{Triples}_{\mathsf{Bin}} = mn_b.$$

For **SN-VLM**, each Bernoulli sample uses α additive-shared coins and α additional committed multiplications, for a cost of $\alpha(m+1)$ triples. **SN-VLM** calls this sampler $t+1$ times and performs one final committed multiplication, giving

$$\#\mathsf{Coins}_{\mathsf{Lap}} = (t+1)\alpha, \qquad \#\mathsf{Triples}_{\mathsf{Lap}} = (m+1)(t+1)\alpha + 1.$$

Table 1 summarizes the resulting parameter settings and preprocessing costs. **SN-VBM** requires $\Theta(\epsilon^{-2})$ additive-shared coins and committed Beaver triples, whereas **SN-VLM** requires only $(t+1)\alpha = \Theta(\log \epsilon^{-1} \log\log \epsilon^{-1})$ additive-shared coins for fixed δ. Its committed-triple count differs by only the constant factor $m+1$, plus one final committed multiplication. Hence, for fixed m and δ, **SN-VLM** grows much more slowly as ϵ decreases.

In **VDDLM**, n_{Lap} varies with the dynamic parameter-selection strategy used for privacy calibration. **SN-VLM** sets t, α explicitly from (ϵ, δ) using truncation and total-variation bounds. **VDDLM** has $\Theta(n_{\mathrm{Lap}})$ sampling/proof cost per server and $\Theta(mn_{\mathrm{Lap}})$ total communication and verifier overhead. **SN-VBM** and **SN-VLM** require noise-stage communication $O(m^2 n_b)$ and $O(m^2 t\alpha)$, respectively.

Table 2: Comparison excluding client-input costs. n_{Lap} counts local fair coins per **VDDLM** noise sample. Its counterpart in **SN-VLM** is $(t+1)\alpha$ **SharedCoin** calls per shared noise sample.

Mechanism	Noise and sampling	Primitive cost	Expected absolute error
VBM [4]	Centered binomial; m local samples	mn_b local coins	$\Theta(\sqrt{mn_b})$
SN-VBM	Centered binomial; one shared sample	n_b SharedCoin calls; mn_b committed-triples	$\Theta(\sqrt{n_b})$
VDDLM [27]	Modified discrete Laplace; m local samples	mn_{Lap} local coins	$\Theta(\sqrt{m}/\epsilon)$
SN-VLM	Modified discrete Laplace; one shared sample	$(t+1)\alpha$ SharedCoin calls; $(m+1)(t+1)\alpha+1$ committed-triples	$\Theta(1/\epsilon)$

The local baselines allow parallel sampling without committed-multiplication preprocessing, avoiding the additional factor of m in noise-stage communication incurred by shared sampling. Shared sampling removes the additional $\sqrt{m}$ factor in expected absolute error instead.

Repeated releases. For R independent releases of the same query on the same committed inputs, the nm input commitments and n bitness proofs are needed once. With fixed per-release privacy parameters, SN-VBM uses Rn_b shared coins and Rmn_b triples; SN-VLM uses $R(t+1)\alpha$ shared coins and $R((m+1)(t+1)\alpha+1)$ triples. Each release needs fresh coins and triples and separate aggregation and final verification. Online communication is $O(nm+Rm^2n_b)$ and $O(nm+Rm^2t\alpha)$, respectively. For a fixed total privacy budget, a composition accountant selects (ϵ_r, δ_r); recomputing n_b, t, α from these values can make the total cost grow faster than a factor of R.

A.2 Running time analysis

We next evaluate the running time of verifiable randomness generation. To identify the bottleneck in the full protocol execution, we set the security parameters $(\epsilon, \delta) = (0.1, 10^{-10})$ and the number of clients to $n = 10^6$ and computing parties to $m = 2$. We then run complete examples of our shared-noise binomial and Laplace mechanisms and decompose the prover/verifier running time into client input verification, coin generation, multiplication, aggregation, and final verification.

We simulate all parties on a single machine. The resulting measurements capture the aggregate prover/verifier-side computation needed to generate the same protocol transcript, rather than the wall-clock time of a distributed deployment. Since party-local computations that could be parallelized across machines are executed sequentially, this model gives a conservative estimate of the local computation component. Network latency and synchronization costs are not modeled.

We exclude client input-sharing time because it is not a prover/verifier-side cost, and we also exclude committed Beaver triple generation because it is implemented by a dummy generator for $\mathcal{F}_{\mathsf{VCT}}$ and reported separately as preprocessing load. The resulting breakdown is shown in Table 3.

We also compare against the verifiable randomness-generation component of the Verifiable Binomial Mechanism (**VBM**) baseline in Biswas and Cormode's work [4]. For this baseline, we use the authors' Rust implementation and measure the combined time for bit commitment, commitment verification, and the Morra-based public coin-generation protocol. These steps are jointly necessary in **VBM** for the provers to obtain secret randomness and for the verifier to audit that the randomness was generated correctly.

For our mechanisms, the corresponding operation is one invocation of **SharedCoin**, which outputs a verifier-auditable additive-shared coin. Hence, the comparison is not between identical cryptographic objects: **VBM** generates private coins for one prover, whereas our protocol generates secret additive-shared coins, both with public auditability. We therefore interpret the experiment as a component-level comparison of verifiable randomness generation, rather than as an end-to-end runtime comparison.

Since the number of coins required by each mechanism is determined by (ϵ, δ), and each coin-generation invocation is independent, we estimate the total coin-generation time from a per-coin benchmark. For each coin-generation procedure, we generate 10^5 coins to obtain the average time per coin.

Table 3: Runtime breakdown of our **SN-VBM** and **SN-VLM** mechanisms with setting $n = 10^6$, $m = 2$, $(\epsilon, \delta) = (0.1, 10^{-10})$. All parties are simulated sequentially on a single machine.

mechanism	client verify	coin generation	multiplication	aggregation	final verification
SN-VBM	143,005ms	661,321ms	-	198ms	271ms
SN-VLM	143,136ms	1,005ms	167ms	257ms	289ms

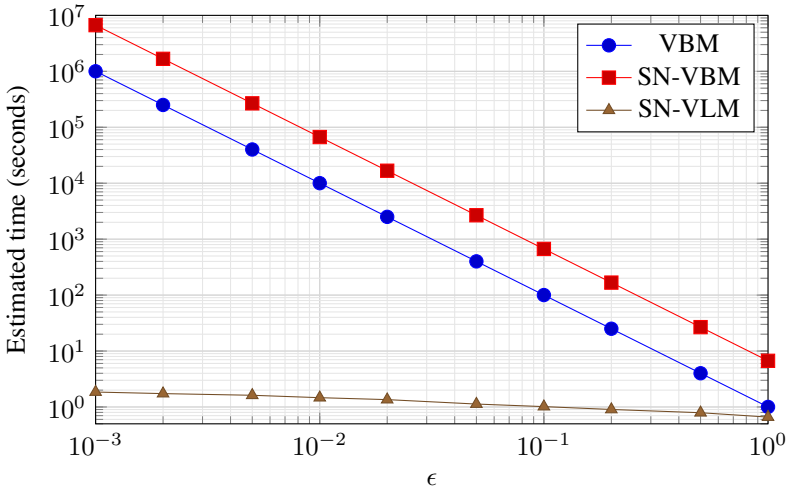


Figure 5: Estimated coin-generation time as a function of ϵ for $\delta = 10^{-10}$. The per-coin time is $422.27\,\mu s$ for the baseline private-coin procedure used by **VBM** and $2819.86\,\mu s$ for **SharedCoin** used by **SN-VBM / SN-VLM**.

We then multiply this per-coin time by the number of coins required by each mechanism. Figure 5 reports the estimated coin-generation time as a function of ϵ under $\delta = 10^{-10}$. This experiment shows how the dominant randomness-generation cost scales under the parameter settings required by each mechanism. Because the estimate is based on independent per-coin generation, it isolates the cost of randomness generation from other protocol components such as input sharing, aggregation, and final verification.

A.3 Error Comparison

We evaluate statistical utility by measuring the absolute error of the released count. For a counting query $Q(X)$ and released value $M_Q(X)$, the absolute error is

$$|M_Q(X) - Q(X)|\,.$$

Since all mechanisms considered here add data-independent noise to the exact count, this error is determined entirely by the noise distribution.

Noise distributions. **SN-VBM** adds centered binomial noise

$$Z_{\text{SN-VBM}} = \mathsf{Bin}(n_b, 1/2) - \frac{n_b}{2}.$$

VBM baseline generates independent noise for each of the m provers and aggregates the resulting noisy shares. Thus its total noise can be viewed as the sum of m independent centered binomial noises, which incurs an additional $\sqrt{m}$ factor in the standard deviation.

SN-VLM instead samples a single shared discrete-Laplace-type noise. In the ideal case, the symmetric discrete Laplace distribution with parameter $s = e^{-\epsilon}$ satisfies

$$\Pr[Z_{\mathsf{Lap}} = k] = \frac{1-s}{1+s} s^{|k|}.$$

Our implementation uses the truncated and quantized variant induced by the parameters t and α from Lemma 3. This noise is supported on

$$[-2^t, 2^t - 1],$$

and its distribution is within the privacy loss accounted for in Lemma 3.

High-probability error bounds. The above distributions yield different high-probability error behavior. For **SN-VBM**, the binomial tail bound gives the following error guarantee.

Theorem 7 (error bound for SN-VBM)**.** *For any failure probability $\gamma \in (0, 1)$, **SN-VBM** satisfies*

$$|M_Q(X) - Q(X)| = O\left(\frac{1}{\epsilon}\sqrt{\ln\frac{1}{\delta}\ln\frac{1}{\gamma}}\right)$$

with probability at least $1 - \gamma$.

For **SN-VLM**, the truncated geometric component is dominated by the tail of the ideal discrete Laplace distribution. Therefore, the high-probability error depends logarithmically on $1/\gamma$ and does not contain the $\sqrt{\ln(1/\delta)}$ factor of the binomial mechanism.

Theorem 8 (error bound for SN-VLM)**.** *For any failure probability $\gamma \in (0, 1)$, **SN-VLM** satisfies*

$$|M_Q(X) - Q(X)| = O\left(\frac{1}{\epsilon}\ln\frac{1}{\gamma}\right)$$

with probability at least $1 - \gamma$.

Theorem 9 (expected absolute error of SN-VLM)**.** *For $0 < \epsilon \leq 1$ and $0 < \delta < 1$, under Lemma 3, the implemented **SN-VLM** satisfies $\mathbb{E}[|M_Q(X) - Q(X)|] = \Theta(1/\epsilon)$, with constants independent of m.*

The proofs of Theorems 7, 8 and 9 are deferred to Appendix D.4.

Empirical noise simulation. We empirically compare **VBM**, **SN-VBM**, and **SN-VLM** under the same parameter settings used in the preprocessing analysis. Using the parameters in Table 1, we run noise-only simulations without executing MPC; for the binomial mechanisms, we sample directly from the induced binomial distributions rather than explicitly simulating individual coin flips.

For each mechanism and each privacy setting, we run $R = 10^5$ independent noise samples and report two metrics in Figure 6. The first is the empirical mean absolute error,

$$\mathsf{MAE} = \frac{1}{R}\sum_{r=1}^{R}\left|Z^{(r)}\right|.$$

The second is the empirical 95th percentile absolute error,

$$\mathsf{Err}_{0.95} = \inf\left\{\tau : \Pr\left[|Z| \leq \tau\right] \geq 0.95\right\}.$$

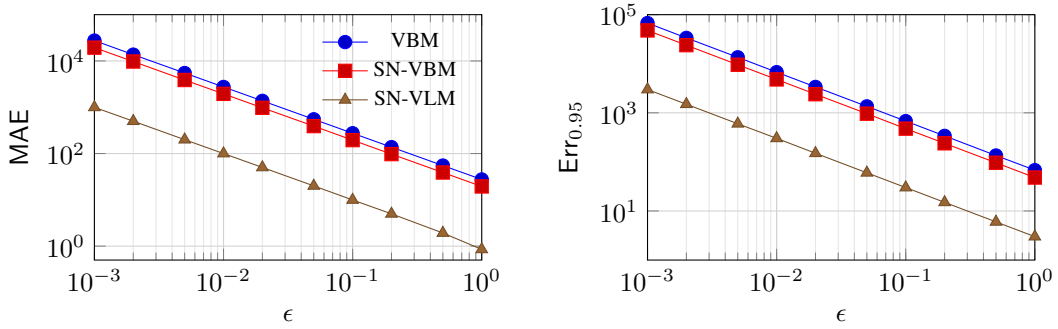
Here Z denotes the noise sampled by the corresponding mechanism. Since $M_Q(X) = Q(X) + Z$, these quantities are exactly the absolute errors of the released count. Figure 6 shows that for fixed $m = 2$ and $\delta = 10^{-10}$, the empirical errors of all three mechanisms follow the expected ϵ^{-1} scaling. In the evaluated parameter regime, **SN-VLM** consistently achieves more than an order-of-magnitude smaller error than the two binomial-based mechanisms.

B Formal definitions and cryptographic primitives

B.1 Verifiable DP definitions

This appendix formalizes the verifiable-DP security notions used in Section 2.2. We use the standard definition of differential privacy [15] and follow the standard view of verifiable differential privacy as a protocol-level guarantee composed of completeness, soundness, and zero knowledge [4].

Let Π be an interactive protocol executed by clients $Cli_1, \ldots, Cli_n$, provers $Pv_1, \ldots, Pv_m$, and verifier Vfr. Let $X = (x_1, \ldots, x_n)$ denote the client database, and let $M_Q(X)$ denote the intended



(a) Mean absolute error

(b) 95th percentile absolute error

Figure 6: Empirical absolute error of **VBM**, **SN-VBM**, and **SN-VLM** as a function of ϵ for $\delta = 10^{-10}$ and $m = 2$. For each privacy setting, we generate $R = 10^5$ independent noise samples and report (a) the mean absolute error and (b) the empirical 95th percentile absolute error.

differentially private output for query Q. The clients secret-share their inputs among the provers. We write $\vec{Pv} = (Pv_1, \ldots, Pv_m)$ for the collection of provers, and $out(Vfr, \vec{Pv}) \in \{0, 1\}$ for the verifier's decision, where 1 denotes acceptance and 0 denotes rejection. Public parameters are generated by

$$pp \leftarrow \mathsf{Setup}(1^\kappa),$$

where κ is the security parameter.

Definition 1 (Completeness)**.** *A protocol Π satisfies completeness if, for every legal client database X, an honest execution by the clients, provers, and verifier is accepted except with negligible probability. That is,*

$$\Pr\left[out(Vfr, \vec{Pv}) = 0 \ : \ pp \leftarrow \mathsf{Setup}(1^\kappa),\ y \leftarrow \Pi(\vec{Pv}, Vfr, pp)\right] \leq \mathsf{negl}(\kappa).$$

If the rejection probability is zero, we say that Π satisfies perfect completeness.

Definition 2 (Soundness)**.** *A protocol Π satisfies soundness if no probabilistic polynomial-time (PPT) malicious strategy of the provers can make the honest verifier accept an output inconsistent with the prescribed mechanism, except with negligible probability. More precisely, for any subset $I \subseteq [m]$ of corrupted provers, let $\vec{Pv}_I^{\,*}$ denote the corrupted provers indexed by I, and let $\vec{Pv}_{\bar{I}}$ denote the remaining honest provers. Then, for any output y inconsistent with the prescribed computation $M_Q(X)$,*

$$\Pr\left[out(Vfr, \vec{Pv}_{\bar{I}}, \vec{Pv}_I^{\,*}) = 1 \ : \ pp \leftarrow \mathsf{Setup}(1^\kappa),\ y \leftarrow \Pi(\vec{Pv}_{\bar{I}}, \vec{Pv}_I^{\,*}, Vfr, pp)\right] \leq \mathsf{negl}(\kappa).$$

Definition 3 (Zero knowledge)**.** *A protocol Π satisfies zero knowledge if the view of any probabilistic polynomial-time (PPT) malicious verifier, possibly colluding with any subset of corrupted provers, can be simulated from the released output and public information.*

Formally, for any subset $I \subseteq [m]$, let $\vec{Pv}_I^{\,*}$ denote the corrupted provers and let Vfr^* denote a malicious verifier. There exists a PPT simulator $\mathsf{Sim}_{Vfr^*,I}$ that interacts with the corrupted coalition and produces a simulated transcript such that, for every legal database X and every valid released output $y = M_Q(X)$,

$$\mathsf{View}\left[\Pi\big((\vec{Pv}_{\bar{I}}, \vec{Pv}_I^{\,*}), Vfr^*, pp\big)\right] \equiv_c \mathsf{Sim}_{Vfr^*,I}(y, pp),$$

where $\equiv_c$ denotes computational indistinguishability.

Definition 4 (Verifiable differential privacy)**.** *A protocol Π is a verifiably differentially private protocol for query Q if it satisfies completeness, soundness, and zero knowledge, and if the distribution of every accepted output is, except with negligible soundness error, distributed according to a target (ϵ, δ)-differentially private mechanism M_Q for Q.*

B.2 Commitment properties

We use Pedersen commitments [23] over a cyclic group $\mathbb{G}$ of prime order q. Let $g, h \in \mathbb{G}$ be generators such that $\log_g h$ is unknown. For $x, r \in \mathbb{Z}_q$, define

$$\mathsf{Com}(x; r) := g^x h^r.$$

Pedersen commitments are additively homomorphic [23]. For all $x_1, x_2, r_1, r_2 \in \mathbb{Z}_q$,

$$\mathsf{Com}(x_1; r_1) \cdot \mathsf{Com}(x_2; r_2) = \mathsf{Com}(x_1 + x_2; r_1 + r_2).$$

Moreover, for any $\alpha \in \mathbb{Z}_q$,

$$\mathsf{Com}(x; r)^\alpha = \mathsf{Com}(\alpha x; \alpha r).$$

Definition 5 (Hiding)**.** *A commitment scheme is hiding if no probabilistic polynomial-time adversary can distinguish commitments to two messages of its choice except with negligible advantage. Formally, for any PPT adversary $\mathcal{A}$, the quantity*

$$\left| \Pr\left[b' = b : \begin{array}{l} b \xleftarrow{\$} \{0,1\}, \\ (x_0, x_1) \leftarrow \mathcal{A}(pp), \\ r \xleftarrow{\$} \mathbb{Z}_q, \\ C := \mathsf{Com}(x_b; r), \\ b' \leftarrow \mathcal{A}(pp, C) \end{array} \right] - \frac{1}{2} \right|$$

is negligible in the security parameter.

Pedersen commitments with uniformly random opening randomness are perfectly hiding, which means the quantity above is 0 [23].

Definition 6 (Binding)**.** *A commitment scheme is binding if no probabilistic polynomial-time adversary can find two distinct openings of the same commitment except with negligible probability. Formally, for any PPT adversary $\mathcal{A}$,*

$$\Pr[(x, r, x', r') \leftarrow \mathcal{A}(pp),\ x \neq x',\ \mathsf{Com}(x; r) = \mathsf{Com}(x'; r')]$$

is negligible in the security parameter.

For Pedersen commitments, computational binding follows from the hardness of computing $\log_g h$ [23].

B.3 Zero-knowledge OR opening

We use a zero-knowledge OR-opening proof to prove that a Pedersen commitment opens to a bit. Define the language

$$L_{\mathsf{bit}} = \{C \in \mathbb{G} : \exists x \in \{0,1\},\ \exists r \in \mathbb{Z}_q \text{ such that } C = \mathsf{Com}(x; r)\}.$$

The oracle O_{OR} idealizes an interactive zero-knowledge proof for membership in L_{bit}. On common input C, the prover uses witness (x, r) and convinces the verifier that $C \in L_{\mathsf{bit}}$ without revealing whether $x = 0$ or $x = 1$.

In the real protocol, O_{OR} can be instantiated by the standard disjunctive Σ-protocol for Pedersen commitments [9]. We write

$$O_{\mathsf{OR}}(C) = 1$$

to denote successful verification that C commits to a bit, and $O_{\mathsf{OR}}(C) = 0$ otherwise.

B.4 Verifier-auditable committed Beaver triples

We define the ideal functionality $\mathcal{F}_{\mathsf{VCT}}$ for verifier-auditable Pedersen-committed Beaver triples in Figure 7. The functionality abstracts away the preprocessing phase and exposes only the committed triple interface used by higher-level protocols.

Functionality $\mathcal{F}_{\mathsf{VCT}}$
Parties: Provers $Pv_1, \ldots, Pv_m$, verifier Vfr

Triple: The functionality samples $u, v \xleftarrow{\$} \mathbb{Z}_q$, sets $w := uv$, and samples additive sharings

$$u = \sum_{j=1}^{m} u_j, \quad v = \sum_{j=1}^{m} v_j, \quad w = \sum_{j=1}^{m} w_j.$$

For each $j \in [m]$, it samples $\mu_j, \nu_j, \omega_j \xleftarrow{\$} \mathbb{Z}_q$ and defines

$$U_j := \mathsf{Com}(u_j; \mu_j), \quad V_j := \mathsf{Com}(v_j; \nu_j), \quad W_j := \mathsf{Com}(w_j; \omega_j).$$

1. Send $(u_j, \mu_j), (v_j, \nu_j), (w_j, \omega_j)$ to prover Pv_j.
2. Broadcast $\vec{U} = (U_1, \ldots, U_m)$, $\vec{V} = (V_1, \ldots, V_m)$, and $\vec{W} = (W_1, \ldots, W_m)$ to the public bulletin board.

Figure 7: Ideal functionality $\mathcal{F}_{\mathsf{VCT}}$ for verifier-auditable committed Beaver triples.

The committed-share interface of $\mathcal{F}_{\mathsf{VCT}}$ follows publicly auditable MPC/SPDZ techniques, which augment additive shares and Beaver triples with Pedersen commitments and commitment-opening shares so that multiplication gates can be checked from a public transcript [2]. In this work, we use $\mathcal{F}_{\mathsf{VCT}}$ as an ideal preprocessing primitive and count each invocation as one committed Beaver triple.

A concrete implementation of $\mathcal{F}_{\mathsf{VCT}}$ could be obtained by combining publicly auditable committed-share representations with malicious triple-generation techniques such as MASCOT [22], together with commitment-consistency checks. Implementing and benchmarking such a preprocessing phase is outside the scope of this work.

C Committed multiplication

This appendix gives the full line-by-line description, correctness proof, and security discussion for the committed multiplication protocol **ComMult**. The underlying multiplication identity and public commitment audit follow the publicly auditable MPC approach [2]; our presentation isolates the minimal interface needed for verifier-auditable DP noise generation. The protocol uses the verifier-auditable committed triple functionality $\mathcal{F}_{\mathsf{VCT}}$ defined in Appendix B.4.

The protocol **ComMult** takes as input two Pedersen-committed additive sharings $([[a]], \vec{s_a}, \vec{A})$, $([[b]], \vec{s_b}, \vec{B})$, and outputs a Pedersen-committed additive sharing $([[c]], \vec{s_c}, \vec{C})$ such that $c = ab$ holds.

We now explain each line of Figure 8.

Line 1. The parties invoke $\mathcal{F}_{\mathsf{VCT}}.\mathsf{Triple}$ to obtain a committed Beaver triple. Each prover Pv_j receives its local triple shares (u_j, μ_j), (v_j, ν_j), and (w_j, ω_j), while the verifier receives the public commitments $(U_j, V_j, W_j)_{j \in [m]}$. By definition of $\mathcal{F}_{\mathsf{VCT}}$,

$$\sum_{j=1}^{m} w_j = \left(\sum_{j=1}^{m} u_j \right) \left(\sum_{j=1}^{m} v_j \right).$$

Line 2. Each prover masks its input shares using its Beaver-triple shares:

$$d_j := a_j - u_j, \qquad e_j := b_j - v_j.$$

It also computes the corresponding opening randomness

$$s_{d_j} := s_{a_j} - \mu_j, \qquad s_{e_j} := s_{b_j} - \nu_j.$$

The pairs (d_j, s_{d_j}) and (e_j, s_{e_j}) are opened. These openings reveal only the masked differences $d = a - u$ and $e = b - v$, not the individual inputs a and b, because u and v are fresh random Beaver-triple values.

	Verifier (Vfr)		**Prover** (Pv_j)
1:	Receive $(U_j, V_j, W_j)_{j\in[m]}$	$\mathcal{F}_{\mathsf{VCT}}.\mathsf{Triple}$	Receive $(u_j, \mu_j),\ (v_j, \nu_j),\ (w_j, \omega_j)$
2:			Compute and open $d_j := a_j - u_j, \quad s_{d_j} := s_{a_j} - \mu_j,$ $e_j := b_j - v_j, \quad s_{e_j} := s_{b_j} - \nu_j.$
3:	For each $j \in [m]$: check $A_j U_j^{-1} \stackrel{?}{=} \mathsf{Com}(d_j; s_{d_j}),$ $B_j V_j^{-1} \stackrel{?}{=} \mathsf{Com}(e_j; s_{e_j}).$		
4:	For each $j \in [m]$: compute $C_j = W_j V_j^{d} U_j^{e} g^{\delta_{j1} de}.$	$d := \sum_{j=1}^{m} d_j,$ $e := \sum_{j=1}^{m} e_j.$	Compute $c_j = w_j + dv_j + eu_j + \delta_{j1} de,$ $s_{c_j} = \omega_j + d\nu_j + e\mu_j.$
5:	$\vec{C}$	Output	(c_j, s_{c_j})

Figure 8: Committed multiplication protocol **ComMult**.

Line 3. Verifier checks that each opened masked difference is consistent with the previously published commitments. By homomorphism of Pedersen commitments,

$$A_j U_j^{-1} = \mathsf{Com}(a_j; s_{a_j}) \cdot \mathsf{Com}(u_j; \mu_j)^{-1} = \mathsf{Com}(a_j - u_j; s_{a_j} - \mu_j),$$

and therefore the verifier accepts only if

$$A_j U_j^{-1} \stackrel{?}{=} \mathsf{Com}(d_j; s_{d_j}).$$

The same argument applies to $B_j V_j^{-1}$ and $\mathsf{Com}(e_j; s_{e_j})$. If any check fails, the protocol aborts.

Line 4. After all openings are accepted, the public values

$$d := \sum_{j=1}^{m} d_j, \qquad e := \sum_{j=1}^{m} e_j$$

are reconstructed. Each prover locally computes

$$c_j = w_j + dv_j + eu_j + \delta_{j1} de,$$

and the corresponding commitment randomness

$$s_{c_j} = \omega_j + d\nu_j + e\mu_j.$$

The verifier can publicly derive

$$C_j = W_j V_j^{d} U_j^{e} g^{\delta_{j1} de}.$$

By homomorphism, this is exactly a commitment to the prover's computed output share:

$$C_j = \mathsf{Com}(c_j; s_{c_j}).$$

Line 5. The public output is the commitment vector $\vec{C}$, while prover Pv_j retains its local output share c_j and the opening randomness s_{c_j}. Thus the output is a committed additive sharing

$$([[c]], \vec{s_c}, \vec{C}).$$

C.1 Correctness

Let

$$u := \sum_{j=1}^{m} u_j, \qquad v := \sum_{j=1}^{m} v_j, \qquad w := \sum_{j=1}^{m} w_j.$$

By the Beaver-triple property, $w = uv$. Since

$$d = \sum_{j=1}^{m} d_j = a - u, \qquad e = \sum_{j=1}^{m} e_j = b - v,$$

the reconstructed output satisfies

$$\begin{aligned}\sum_{j=1}^{m} c_j &= \sum_{j=1}^{m} (w_j + dv_j + eu_j + \delta_{j1}de) \\ &= w + dv + eu + de \\ &= uv + (a-u)v + (b-v)u + (a-u)(b-v) \\ &= ab.\end{aligned}$$

Therefore $([[c]], \vec{s_c})$ is an additive sharing of $c = ab$.

Moreover, for each $j \in [m]$,

$$\begin{aligned}C_j &= W_j V_j^d U_j^e g^{\delta_{j1}de} \\ &= \mathsf{Com}(w_j; \omega_j) \cdot \mathsf{Com}(v_j; \nu_j)^d \cdot \mathsf{Com}(u_j; \mu_j)^e \cdot \mathsf{Com}(\delta_{j1}de; 0) \\ &= \mathsf{Com}\left(w_j + dv_j + eu_j + \delta_{j1}de; \omega_j + d\nu_j + e\mu_j\right) \\ &= \mathsf{Com}(c_j; s_{c_j}).\end{aligned}$$

Thus each output share is consistent with its public commitment.

C.2 Security discussion

The protocol relies on two standard properties of Pedersen commitments: hiding and binding.

Hiding. The published values $A_j, B_j, U_j, V_j, W_j, C_j$ are Pedersen commitments. Assuming the commitment randomness is sampled independently and uniformly over $\mathbb{Z}_q$, these commitments are perfectly hiding. Therefore, the commitments themselves do not reveal the committed shares $a_j, b_j, u_j, v_j, w_j, c_j$.

The only values opened during the protocol are the masked differences $d_j = a_j - u_j$ and $e_j = b_j - v_j$, together with their opening randomness. Aggregating over all parties gives

$$d = a - u, \qquad e = b - v.$$

Since u and v are fresh random masks sampled independently of a and b, the public values d and e reveal no additional information about a and b beyond what is revealed by the final computation. This is the same privacy argument used in standard Beaver-triple multiplication.

Binding and auditability. The binding property prevents a corrupted prover from successfully opening a fixed commitment to two different values. Therefore, once A_j, B_j, U_j, V_j, W_j are fixed on the public bulletin board, the prover cannot adaptively change its input shares or triple shares while still passing the opening checks in Line 3.

Similarly, the output commitment C_j is not chosen freely by the prover. It is publicly derived from W_j, V_j, U_j, d, e as

$$C_j = W_j V_j^d U_j^e g^{\delta_{j1}de}.$$

Hence, after d and e are fixed, C_j is uniquely determined by the public transcript. A corrupted prover who later claims an output share $c'_j \neq c_j$ would need to produce randomness s'_{c_j} such that

$$C_j = \mathsf{Com}(c'_j; s'_{c_j}),$$

while $C_j = \mathsf{Com}(c_j; s_{c_j})$ already holds. This would violate the binding property of the commitment scheme.

Malicious behavior. A malicious prover can attempt to deviate in two places. First, it may open incorrect masked differences in Line 2. Such deviations are detected by the commitment checks in Line 3. Second, it may output a value inconsistent with the derived commitment in Line 5. Such deviations are detected whenever the output share is later opened or audited against C_j. Thus the protocol provides verifier-auditable consistency of the multiplication transcript.

D Formal Proofs

This appendix provides the deferred proofs for the protocol guarantees and error bounds stated in the main text.

D.1 Security of SharedCoin

We prove the correctness, unbiasedness, auditability, and zero-knowledge properties of **SharedCoin**.

D.1.1 Proof of Theorem 1

Proof. Assume that all parties honestly execute **SharedCoin**. Then all commitment openings and all O_{OR} checks in Lines 1–6 are valid, so the protocol does not abort. For each prover-contributed bit b_i, the shares satisfy

$$b_i = \sum_{j=1}^{m} b_{ij} \pmod q,$$

and the public commitments satisfy

$$C_{ij} = \mathsf{Com}(b_{ij}; r_{ij}).$$

Hence, by homomorphism of Pedersen commitments,

$$B_i = \prod_{j=1}^{m} C_{ij} = \mathsf{Com}\left(\sum_{j=1}^{m} b_{ij}; \sum_{j=1}^{m} r_{ij}\right) = \mathsf{Com}\left(b_i; \sum_{j=1}^{m} r_{ij}\right).$$

Since $b_i \in \{0,1\}$, the proof $O_{\mathsf{OR}}(B_i)$ accepts.

We next show by induction that after the k-th XOR update, the protocol maintains a Pedersen-committed additive sharing

$$([[t^{(k)}]], \vec{s}_t^{(k)}, \vec{T}^{(k)})$$

of

$$t^{(k)} = b_1 \oplus \cdots \oplus b_k.$$

For $k = 0$, the protocol initializes

$$[[t^{(0)}]] = (0, \ldots, 0), \qquad \vec{s}_t^{(0)} = (0, \ldots, 0), \qquad \vec{T}^{(0)} = (1, \ldots, 1),$$

which is a valid committed sharing of $t^{(0)} = 0$.

Assume the invariant holds for $k-1$. In Line 8, **ComMult** is invoked on the committed sharings of $t^{(k-1)}$ and b_k. By correctness of **ComMult**, it outputs a committed additive sharing

$$([[l^{(k)}]], \vec{\lambda}^{(k)}, \vec{L}^{(k)})$$

of

$$l^{(k)} = t^{(k-1)} b_k,$$

with

$$L_j^{(k)} = \mathsf{Com}(l_j^{(k)}; \lambda_j^{(k)}), \quad \forall j \in [m].$$

Line 9 defines

$$t_j^{(k)} = t_j^{(k-1)} + b_{kj} - 2l_j^{(k)}, \qquad s_{t_j}^{(k)} = s_{t_j}^{(k-1)} + r_{kj} - 2\lambda_j^{(k)}.$$

Therefore,

$$t^{(k)} = \sum_{j=1}^{m} t_j^{(k)} = t^{(k-1)} + b_k - 2t^{(k-1)} b_k = t^{(k-1)} \oplus b_k = b_1 \oplus \cdots \oplus b_k \in \{0,1\}$$

since $t^{(k-1)}, b_k \in \{0,1\}$.

It remains to check commitment consistency. By the induction hypothesis,

$$T_j^{(k-1)} = \mathsf{Com}(t_j^{(k-1)}; s_{t_j}^{(k-1)}),$$

and by construction,

$$C_{kj} = \mathsf{Com}(b_{kj}; r_{kj}).$$

Together with the output commitment of **ComMult**, we get

$$\begin{aligned} T_j^{(k)} &= T_j^{(k-1)} \cdot C_{kj} \cdot (L_j^{(k)})^{-2} \\ &= \mathsf{Com}(t_j^{(k-1)}; s_{t_j}^{(k-1)}) \cdot \mathsf{Com}(b_{kj}; r_{kj}) \cdot \mathsf{Com}(l_j^{(k)}; \lambda_j^{(k)})^{-2} \\ &= \mathsf{Com}\left(t_j^{(k-1)} + b_{kj} - 2l_j^{(k)}; s_{t_j}^{(k-1)} + r_{kj} - 2\lambda_j^{(k)}\right) \\ &= \mathsf{Com}(t_j^{(k)}; s_{t_j}^{(k)}). \end{aligned}$$

This proves the induction invariant.

After the loop, the protocol holds a committed additive sharing of

$$t^{(m)} = b_1 \oplus \cdots \oplus b_m.$$

The verifier then opens its committed bit c in Line 10. Since all parties are honest, the opening check

$$C_c \stackrel{?}{=} \mathsf{Com}(c; r_c)$$

accepts. If $c = 0$, the protocol sets

$$a_j = t_j^{(m)}, \qquad \rho_j = s_{t_j}^{(m)}, \qquad A_j = T_j^{(m)}.$$

Therefore

$$\sum_{j=1}^{m} a_j = t^{(m)}$$

and

$$A_j = \mathsf{Com}(a_j; \rho_j).$$

If $c = 1$, the protocol sets

$$a_j = \delta_{j1} - t_j^{(m)}, \qquad \rho_j = -s_{t_j}^{(m)}, \qquad A_j = g^{\delta_{j1}} (T_j^{(m)})^{-1}.$$

Then

$$\sum_{j=1}^{m} a_j = 1 - t^{(m)},$$

and

$$A_j = g^{\delta_{j1}} \cdot \mathsf{Com}(t_j^{(m)}; s_{t_j}^{(m)})^{-1} = \mathsf{Com}\left(\delta_{j1} - t_j^{(m)}; -s_{t_j}^{(m)}\right) = \mathsf{Com}(a_j; \rho_j).$$

Thus in both cases

$$A_j = \mathsf{Com}(a_j; \rho_j), \quad \forall j \in [m].$$

Moreover, the reconstructed output is

$$a = t^{(m)} \oplus c = b_1 \oplus \cdots \oplus b_m \oplus c.$$

Therefore **SharedCoin** outputs a Pedersen-committed additive sharing $([[a]], \vec{\rho}, \vec{A})$ satisfying the theorem statement. □

D.1.2 Proof of Theorem 2

Proof. Assume that **SharedCoin** completes successfully. If the provers' internal output state is inconsistent with the public commitment vector $\vec{A}$, then the theorem holds. Hence suppose that the output state is consistent with $\vec{A}$,

$$A_j = \mathsf{Com}(a_j; \rho_j), \quad \forall j \in [m].$$

Since the O_{OR} transcripts are checked by the verifier and all provers, successful completion implies that all honest checking parties accepted the public bitness proofs. Thus, if at least one checking

party is honest, every accepted aggregate commitment B_i and the verifier commitment C_c open to bits except with negligible probability by the soundness of O_{OR}. Therefore,

$$b_i \in \{0,1\} \quad \forall i \in [m], \qquad c \in \{0,1\}.$$

By the computational binding property of Pedersen commitments, once A_j is fixed, no PPT adversary can produce a different opening (a'_j, ρ'_j) with $a'_j \neq a_j$ such that

$$A_j = \mathsf{Com}(a'_j; \rho'_j),$$

except with negligible probability. Hence, except with negligible probability, the public commitment vector $\vec{A}$ fixes the output shares $a_1, \ldots, a_m$.

Since the protocol completes successfully, all committed multiplications in the XOR chain are accepted. By the correctness of **ComMult** and the inductive correctness argument from Theorem 1, the fixed output shares reconstruct

$$a = \sum_{j=1}^{m} a_j = b_1 \oplus \cdots \oplus b_m \oplus c.$$

It remains to show unbiasedness. Let η denote the bit contributed by an honest randomness-contributing party, and let

$$\beta := \bigoplus_{\xi \in \{b_1,\ldots,b_m,c\} \setminus \{\eta\}} \xi$$

be the XOR of all remaining contributions. Then

$$a = \eta \oplus \beta.$$

The honest contribution satisfies $\eta \xleftarrow{\$} \{0,1\}$. Moreover, η is independent of β: if $\eta = c$, then β is fixed before c is opened; if $\eta = b_h$ for an honest prover, then η is hidden from the adversary by Pedersen hiding and the zero-knowledge property of O_{OR}. Hence $\eta \oplus \beta$ is uniform, or computationally indistinguishable from uniform in the prover-honest case.

Therefore, under the stated assumptions, upon successful completion of **SharedCoin**, except with negligible probability, either the output is a Pedersen-committed additive sharing of a computationally uniform bit or the provers' internal output state is inconsistent with $\vec{A}$. □

D.1.3 Proof of Theorem 3

Proof. Fix an arbitrary PPT malicious verifier Vfr^* colluding with a subset $I \subseteq [m]$ of corrupted provers. We construct a simulator $\mathsf{Sim}_{Vfr^*,I}$ for the joint view of Vfr^* and $\vec{Pv}_I^*$ without using the honest provers' private bits, opening randomnesses, or output shares.

The simulator is given the public parameters and the internal inputs and randomnesses of the corrupted provers. It simulates all messages sent by honest provers as follows.

First, consider the share-distribution phase. For every honest prover Pv_h and corrupted receiver Pv_i with $i \in I$, the real protocol sends

$$(b_{hi}, r_{hi})$$

over a secure channel. In the real execution, these off-diagonal additive shares are sampled uniformly over $\mathbb{Z}_q$ and are independent of the underlying bit b_h. Therefore, $\mathsf{Sim}_{Vfr^*,I}$ samples

$$\widetilde{b}_{hi}, \widetilde{r}_{hi} \xleftarrow{\$} \mathbb{Z}_q$$

and sends $(\widetilde{b}_{hi}, \widetilde{r}_{hi})$ to the corrupted prover Pv_i. This has exactly the same distribution as the real message.

For public commitments generated by honest provers, $\mathsf{Sim}_{Vfr^*,I}$ samples commitments to dummy values, e.g.,

$$\widetilde{C}_{hj} := \mathsf{Com}(0; \widetilde{r}_{hj})$$

with fresh uniform randomness. By the hiding property of Pedersen commitments, these dummy commitments are identically distributed to commitments to the real honest shares. Similarly, the aggregate commitment

$$\widetilde{B}_h = \prod_{j=1}^{m} \widetilde{C}_{hj}$$

is computationally indistinguishable from the real aggregate commitment B_h, because both are Pedersen commitments with uniformly random opening randomness.

Whenever an honest prover must prove that an aggregate commitment opens to a bit, $\mathsf{Sim}_{Vfr^*,I}$ invokes the zero-knowledge simulator for O_{OR}. Thus, the simulated proof transcript for

$$\widetilde{B}_h \in L_{\mathsf{bit}}$$

is computationally indistinguishable from a real proof transcript. The same argument applies to the verifier-bit commitment C_c if the verifier is honest. Hence the public bitness-proof transcripts reveal nothing beyond the fact that the corresponding commitments pass the O_{OR} verification.

Next, consider the XOR computation in Lines 7–9. Each iteration invokes **ComMult** on Pedersen-committed additive sharings. $\mathsf{Sim}_{Vfr^*,I}$ invokes the zero-knowledge simulator for **ComMult** to generate the transcript of each committed multiplication, including the opened masked differences and the output commitments $\vec{L}^{(k)}$, which are computationally indistinguishable from the real transcripts. The commitment updates

$$T_j^{(k)} = T_j^{(k-1)} \cdot C_{kj} \cdot (L_j^{(k)})^{-2}$$

are deterministic functions of the simulated public transcript, so they are also computationally indistinguishable from the corresponding real values.

Finally, $\mathsf{Sim}_{Vfr^*,I}$ constructs the final public commitment vector $\vec{A}$ by following the same deterministic update rule as the real protocol. If $c = 0$, it sets

$$\widetilde{A}_j := \widetilde{T}_j^{(m)}.$$

If $c = 1$, it sets

$$\widetilde{A}_j := g^{\delta_{j1}} \left(\widetilde{T}_j^{(m)}\right)^{-1}.$$

The honest provers' final output shares and opening randomnesses are never revealed to the verifier, and the public commitments are hiding. Therefore, the simulated final commitment vector is computationally indistinguishable from the real one.

Thus, for every PPT malicious verifier colluding with any subset of corrupted provers, the simulated view constructed by $\mathsf{Sim}_{Vfr^*,I}$ is computationally indistinguishable from the real view. Therefore, **SharedCoin** is computational zero knowledge. □

D.2 Security and privacy of SN-VBM

We prove the verifiability and differential privacy guarantees of **SN-VBM**.

Proof of Theorem 4. We prove perfect completeness, negligible soundness, and computational zero knowledge.

Perfect completeness. Assume that all parties honestly execute **SN-VBM**. Each client Cli_i publishes commitments

$$X_{ij} = \mathsf{Com}(x_{ij}; r_{ij})$$

to an additive sharing of its input bit x_i. Hence

$$X_i = \prod_{j=1}^{m} X_{ij} = \mathsf{Com}\left(\sum_{j=1}^{m} x_{ij}; \sum_{j=1}^{m} r_{ij}\right) = \mathsf{Com}\left(x_i; \sum_{j=1}^{m} r_{ij}\right).$$

Since $x_i \in \{0, 1\}$, the $O_{\mathsf{OR}}(X_i)$ proof accepts. Moreover, each invocation of **SharedCoin** honestly outputs a committed additive sharing $([[a_i]], \vec{\rho}_i, \vec{A}_i)$ of a bit a_i. For each prover Pv_j, the reported values are

$$y_j = \sum_{i=1}^{n} x_{ij} + \sum_{i=1}^{n_b} a_{ij}$$

and

$$z_j = \sum_{i=1}^{n} r_{ij} + \sum_{i=1}^{n_b} \rho_{ij}.$$

By the homomorphism of Pedersen commitments,

$$\prod_{i=1}^{n} X_{ij} \cdot \prod_{i=1}^{n_b} A_{ij} = \mathsf{Com}(y_j; z_j).$$

Thus all verification checks pass, and the protocol outputs

$$M_Q(X) = \sum_{j=1}^{m} y_j - \frac{n_b}{2}.$$

Therefore, **SN-VBM** satisfies perfect completeness.

Soundness. Assume that a malicious set of provers makes the verifier accept an output not consistent with the prescribed **SN-VBM** computation. We show that this implies either a violation of a subprotocol soundness guarantee or a violation of Pedersen binding.

First, if a corrupted client causes a non-bit client input to be accepted, then the corresponding aggregate commitment

$$X_i = \prod_{j=1}^{m} X_{ij}$$

must pass the $O_{\mathsf{OR}}(X_i)$ check even though it does not commit to a bit. This violates the soundness of O_{OR}, except with negligible probability.

Second, if any generated noise coin is not a valid accepted output of **SharedCoin**, then the adversary violates the soundness/auditability guarantee of **SharedCoin**, except with negligible probability. Hence, except with negligible probability, every accepted noise commitment vector $\vec{A}_i$ defines a committed additive sharing of a bit a_i, unless the provers' internal state is inconsistent with the public commitments.

Finally, suppose all client bitness checks and all **SharedCoin** invocations are valid, but the accepted released value is still inconsistent with the prescribed sum. Then, for some prover Pv_j, the reported pair (y_j, z_j) must satisfy the verifier's check

$$\prod_{i=1}^{n} X_{ij} \cdot \prod_{i=1}^{n_b} A_{ij} = \mathsf{Com}(y_j; z_j)$$

while y_j is not the sum of the committed input and noise shares for that prover. However, the left-hand side is already fixed by the public commitment transcript and, by homomorphism, is a Pedersen commitment to the committed sum of those shares. Producing a different opening (y_j, z_j) for the same commitment would violate the computational binding property of Pedersen commitments, except with negligible probability.

Therefore, except with negligible probability, every accepted transcript defines the output

$$M_Q(X) = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n_b} a_i - \frac{n_b}{2},$$

where each a_i is the bit generated by an accepted invocation of **SharedCoin**. Thus malicious provers cannot make the verifier accept an output inconsistent with the prescribed **SN-VBM** mechanism except with negligible probability.

Zero knowledge. Fix any PPT malicious verifier Vfr^*, possibly colluding with a subset $I \subseteq [m]$ of corrupted provers. We construct a PPT simulator $\mathsf{Sim}_{Vfr^*,I}$ for its view. The simulator is given the corrupted parties' inputs and randomnesses, the public parameters, and the released output value $M_Q(X)$.

For honest clients and honest provers, $\mathsf{Sim}_{Vfr^*,I}$ replaces Pedersen commitments to private input shares and noise shares with commitments to dummy values using fresh randomness. By the hiding

property of Pedersen commitments, these dummy commitments are computationally indistinguishable from the real commitments. For every client-input bitness proof, $\mathsf{Sim}_{Vfr^*,I}$ invokes the zero-knowledge simulator for O_{OR} to generate a simulated proof transcript indistinguishable from the real one.

For the n_b invocations of **SharedCoin**, $\mathsf{Sim}_{Vfr^*,I}$ invokes the simulator guaranteed by Theorem 3. Therefore, the simulated coin-generation transcripts are computationally indistinguishable from the real transcripts and reveal no honest noise shares.

It remains to simulate the final aggregation messages. Let the shares of the corrupted provers $\vec{Pv}_I^*$ be fixed by the adversary's internal state. The simulator $\mathsf{Sim}_{Vfr^*,I}$ chooses dummy values for the honest provers' reported shares so that their sum is consistent with the released output $M_Q(X)$ after subtracting the public value $n_b/2$. It then publishes Pedersen commitments and opening randomnesses consistent with these dummy aggregate shares. Since these commitments are hiding, and since only the final released sum is exposed, this aggregation transcript is computationally indistinguishable from the real one. Hence the simulated view is computationally indistinguishable from the real view.

Therefore, **SN-VBM** satisfies perfect completeness, negligible soundness, and computational zero knowledge. □

Lemma 4 (Differential privacy of the binomial mechanism [4, 19])**.** *Let $X = (x_1, \ldots, x_n) \in \{0,1\}^n$ and let $Q(X) = \sum_{i=1}^n x_i$. Fix $0 < \delta < 1$ and an integer $n_b \geq 100 \ln(2/\delta)$. Let $Z \sim \mathsf{Bin}(n_b, 1/2)$. Then*

$$Q(X) + Z$$

is (ϵ, δ)-differentially private with

$$\epsilon = 10\sqrt{\frac{1}{n_b} \ln \frac{2}{\delta}}.$$

Proof. Let $p_k = \Pr[Z = k]$ and $I = \{k : |k - n_b/2| \leq \epsilon n_b/10\}$. The stated parameters give $\epsilon \leq 1$ and $n_b > 5/\epsilon$. For $k \in I$, the adjacent ratios

$$\frac{p_k}{p_{k-1}} = \frac{n_b - k + 1}{k}, \qquad \frac{p_k}{p_{k+1}} = \frac{k+1}{n_b - k}$$

are at most

$$1 + \frac{\epsilon n_b/5 + 1}{2n_b/5} \leq 1 + \epsilon \leq e^{\epsilon}.$$

Hoeffding's inequality gives

$$\Pr[Z \notin I] \leq 2e^{-\epsilon^2 n_b/50} = \delta^2/2 \leq \delta.$$

For neighboring counts, the shift is in $\{-1, +1\}$, summing these ratio bounds over any output event and charging at most δ to $Z \notin I$ proves differential privacy. □

This is the standard binomial-mechanism privacy bound used in prior verifiable counting protocols.

Proof of Lemma 1. By Lemma 4, it suffices to choose n_b such that

$$10\sqrt{\frac{1}{n_b} \ln \frac{2}{\delta}} \leq \epsilon.$$

The choice

$$n_b = \left\lceil \frac{100}{\epsilon^2} \ln \frac{2}{\delta} \right\rceil$$

satisfies this inequality and also ensures $n_b > 30$. Therefore,

$$Q(X) + \mathsf{Bin}(n_b, 1/2)$$

is (ϵ, δ)-differentially private. The **SN-VBM** output is

$$Q(X) + \mathsf{Bin}(n_b, 1/2) - \frac{n_b}{2}.$$

Subtracting the public constant $n_b/2$ is post-processing and therefore does not affect differential privacy. Hence the output distribution of **SN-VBM** is (ϵ, δ)-differentially private. □

D.3 Security and privacy of SharedBern and SN-VLM

We prove the Bernoulli-sampler guarantee and the verifiability and privacy guarantees of **SN-VLM**.

D.3.1 Proof of Lemma 2

Proof. By Theorem 2, each successful invocation of **SharedCoin** outputs, except with negligible probability, either a Pedersen-committed additive sharing of a computationally uniform bit or an internal state inconsistent with its public commitment vector. If any such inconsistency occurs, then the lemma follows. Hence suppose all generated shared coins are commitment-consistent.

Let

$$R = \sum_{k=0}^{\alpha-1} a_k 2^k$$

be the integer represented by the α shared coins, and let

$$P = \lfloor p \cdot 2^\alpha \rfloor = \sum_{k=0}^{\alpha-1} p_k 2^k.$$

Since the a_k's are computationally indistinguishable from independent uniform bits, R is computationally indistinguishable from the uniform distribution on $\{0, \ldots, 2^\alpha - 1\}$.

The update rule in **SharedBern** implements the comparison

$$a' = \mathbf{1}[R < P].$$

Indeed, at each bit position k, the value

$$s^{(k)} = a_k \oplus p_k$$

indicates whether the sampled bit and the threshold bit differ. The update overwrites the running output exactly when a difference occurs, so after all bits have been processed, the output is determined by the most significant bit on which R and P differ. Therefore $a' = 1$ exactly when $R < P$.

Consequently,

$$\Pr[a' = 1] = \Pr[R < P] = \frac{P}{2^\alpha} = \frac{\lfloor p \cdot 2^\alpha \rfloor}{2^\alpha},$$

up to computational indistinguishability inherited from the shared coins. Commitment consistency of the output follows from the correctness of **ComMult** and Pedersen homomorphism, exactly as in the proof of Theorem 1. Binding implies that any different internal output state opening to the same public vector $\vec{A}'$ can occur only with negligible probability.

Perfect completeness follows because honest executions of **SharedCoin** and **ComMult** complete successfully and all commitment identities are satisfied. Computational zero knowledge follows by composing the simulators for **SharedCoin** and **ComMult**, together with the hiding property of Pedersen commitments. □

D.3.2 Proof of Theorem 5

We prove perfect completeness, negligible soundness, and computational zero knowledge.

Perfect completeness. Assume that all parties honestly execute **SN-VLM**. As in **SN-VBM**, each client Cli_i publishes commitments

$$X_{ij} = \mathsf{Com}(x_{ij}; r_{ij})$$

to an additive sharing of its input bit x_i. Hence the aggregate commitment

$$X_i = \prod_{j=1}^{m} X_{ij}$$

opens to $x_i \in \{0, 1\}$, and the $O_{\mathsf{OR}}(X_i)$ proof accepts.

We next consider the noise-generation part. For each $k \in \{0, \dots, t-1\}$, **SharedBern** honestly outputs a Pedersen-committed additive sharing

$$([[a_k]], \vec{\rho}_k, \vec{A}_k)$$

of a Bernoulli bit sampled with probability

$$p_k = \frac{s^{2^k}}{1 + s^{2^k}}.$$

The protocol accumulates the truncated geometric component by setting

$$u_j^{(k+1)} = u_j^{(k)} + 2^k a_{kj}$$

and

$$s_{u_j}^{(k+1)} = s_{u_j}^{(k)} + 2^k \rho_{kj}.$$

By induction and Pedersen homomorphism, after the loop we have

$$U_j^{(t)} = \mathsf{Com}(u_j^{(t)}; s_{u_j}^{(t)})$$

and

$$\sum_{j=1}^{m} u_j^{(t)} = \sum_{k=0}^{t-1} 2^k a_k.$$

The sign bit is then generated by another invocation of **SharedBern** with probability $s/(1+s)$, producing a committed additive sharing

$$([[b]], \vec{\rho}_b, \vec{B})$$

with

$$B_j = \mathsf{Com}(b_j; \rho_{b_j}).$$

The protocol invokes **ComMult** on the committed sharings of $u^{(t)}$ and b. By correctness of **ComMult**, the output $([[l]], \vec{\lambda}, \vec{L})$ satisfies

$$l = u^{(t)} b$$

and

$$L_j = \mathsf{Com}(l_j; \lambda_j).$$

Line 9 defines

$$n_j = u_j^{(t)} - b_j - 2l_j$$

and

$$s_{n_j} = s_{u_j}^{(t)} - \rho_{b_j} - 2\lambda_j.$$

Therefore the reconstructed noise is

$$n = \sum_{j=1}^{m} n_j = u^{(t)} - b - 2u^{(t)} b.$$

Equivalently, if $b = 0$ then $n = u^{(t)}$, and if $b = 1$ then $n = -u^{(t)} - 1$, giving the truncated discrete-Laplace-type noise used by **SN-VLM**. Moreover, by Pedersen homomorphism,

$$N_j = U_j^{(t)} \cdot B_j^{-1} \cdot L_j^{-2} = \mathsf{Com}(n_j; s_{n_j}).$$

Finally, each prover reports

$$y_j = \sum_{i=1}^{n} x_{ij} + n_j$$

and

$$z_j = \sum_{i=1}^{n} r_{ij} + s_{n_j}.$$

Thus

$$\prod_{i=1}^{n} X_{ij} \cdot N_j = \mathsf{Com}(y_j; z_j)$$

for every $j \in [m]$. Hence all verification checks pass, and the protocol outputs

$$M_Q(X) = \sum_{j=1}^{m} y_j = Q(X) + n.$$

Therefore, **SN-VLM** satisfies perfect completeness.

Soundness. Assume that malicious provers make the verifier accept an output inconsistent with the prescribed **SN-VLM** computation. If a non-bit client input is accepted, then the adversary violates the soundness of O_{OR}. If a Bernoulli sample used in the geometric or sign component is not a valid accepted output of **SharedBern**, then the adversary violates the guarantee of Lemma 2. If the final multiplication between $u^{(t)}$ and b is inconsistent, then the adversary violates the correctness/auditability of **ComMult**.

It remains only to consider the final aggregation check. If all subprotocol outputs are valid but the released value is inconsistent with the committed input and noise shares, then for some prover Pv_j the adversary must produce (y_j, z_j) satisfying

$$\prod_{i=1}^{n} X_{ij} \cdot N_j = \mathsf{Com}(y_j; z_j)$$

while y_j is not the committed sum of that prover's input shares and noise share. Since the left-hand side is fixed by the public commitment transcript, this gives a different opening of a fixed Pedersen commitment, contradicting computational binding except with negligible probability. Therefore **SN-VLM** has negligible soundness error.

Zero knowledge. Computational zero knowledge follows by composition. A simulator for a malicious verifier colluding with corrupted provers replaces honest-party Pedersen commitments with commitments to dummy values, using the hiding property of Pedersen commitments. It simulates all client-input O_{OR} proofs using the zero-knowledge simulator for O_{OR}. It simulates each **SharedBern** invocation using the simulator guaranteed by Lemma 2, and simulates the final **ComMult** transcript using the zero-knowledge simulator for **ComMult**. The remaining messages, including $\vec{U}^{(k)}$, $\vec{N}$, and the final verification commitments, are deterministic homomorphic functions of already simulated commitments or are Pedersen commitments to dummy shares chosen consistently with the released output $M_Q(X)$. By Pedersen hiding and the zero-knowledge properties of the subprotocols, the simulated transcript is computationally indistinguishable from the real transcript.

Therefore, **SN-VLM** satisfies perfect completeness, negligible soundness, and computational zero knowledge. □

Lemma 5 (Differential privacy of **SN-VLM**). *Let*

$$\epsilon = -\ln s,$$

$$\delta_t = \frac{(1-s)s^{2^t-1}}{(1+s)(1-s^{2^t})},$$

and

$$\delta = \delta_t + \left(\frac{1}{s} + 1\right)(t+1)2^{-\alpha}.$$

*Then the output distribution of **SN-VLM** is (ϵ, δ)-differentially private.*

D.3.3 Proof of Lemma 5

Proof. We compare the finite-precision mechanism implemented by **SN-VLM** with an ideal exact-truncated mechanism. We use tildes for the ideal mechanism.

First, consider the ideal version obtained by replacing every invocation of **SharedBern** with an exact Bernoulli sampler. For $k = 0, \ldots, t-1$, let

$$\widetilde{a}_k \sim \mathsf{Bernoulli}\left(\frac{s^{2^k}}{1+s^{2^k}}\right),$$

and define

$$\widetilde{u} := \sum_{k=0}^{t-1} 2^k \widetilde{a}_k.$$

For every $z \in \{0, \dots, 2^t - 1\}$, writing $z = \sum_{k=0}^{t-1} 2^k z_k$, we have

$$\Pr[\widetilde{u} = z] = \left(\prod_{k:z_k=1} \frac{s^{2^k}}{1+s^{2^k}} \right) \cdot \left(\prod_{k:z_k=0} \frac{1}{1+s^{2^k}} \right) = \frac{(1-s)s^z}{1-s^{2^t}}.$$

Let the ideal sign bit be

$$\widetilde{b} \sim \mathsf{Bernoulli}\left(\frac{s}{1+s}\right),$$

and define the ideal noise

$$\widetilde{n} = \widetilde{u} - \widetilde{b} - 2\widetilde{u}\widetilde{b}.$$

Thus, if $\widetilde{b} = 0$, then $\widetilde{n} = \widetilde{u}$, and if $\widetilde{b} = 1$, then $\widetilde{n} = -\widetilde{u} - 1$. Hence $\widetilde{n}$ is supported on $[-2^t, 2^t - 1]$ and for every z in this support,

$$\Pr[\widetilde{n} = z] = \frac{1-s}{(1+s)(1-s^{2^t})} s^{|z|}.$$

Let

$$\widetilde{M}(X) = Q(X) + \widetilde{n}$$

be the ideal exact-truncated mechanism.

We next show that $\widetilde{M}$ is (ϵ, δ_t)-DP. Let X, X' be neighboring databases, so that

$$|Q(X) - Q(X')| \le 1.$$

It suffices to consider the case $Q(X') = Q(X) + 1$; the other direction is symmetric. For all noise values whose shifted counterpart remains in the support, the discrete-Laplace ratio satisfies

$$\Pr[\widetilde{n} = z] \le e^{\epsilon} \Pr[\widetilde{n} = z - 1],$$

for $\epsilon = -\ln s$. The only possible violation comes from the truncation boundary, where the shifted noise value leaves the support. The boundary mass is at most

$$\delta_t = \frac{(1-s)s^{2^t-1}}{(1+s)(1-s^{2^t})}.$$

Therefore, for every event S in the output space, summing the pointwise ratio bound over S and charging the boundary event to δ_t gives

$$\Pr[\widetilde{M}(X) \in S] \le e^{\epsilon} \Pr[\widetilde{M}(X') \in S] + \delta_t.$$

Thus the ideal exact-truncated mechanism is (ϵ, δ_t)-DP.

We now account for the finite precision α used by **SharedBern**. The actual SN-VLM noise is

$$u := \sum_{k=0}^{t-1} 2^k a_k, \qquad n = u - b - 2ub,$$

and the actual mechanism is

$$M(X) = Q(X) + n.$$

For each Bernoulli parameter, quantization changes the parameter by at most $2^{-\alpha}$, which means the total variation distance between the exact Bernoulli distribution and its quantized version is at most $2^{-\alpha}$. Therefore, by applying a union bound, total variation distance between n and $\widetilde{n}$ is at most

$$\eta := (t+1)2^{-\alpha},$$

which is also the total variation distance between $M(X)$ and $\widetilde{M}(X)$.

We have, for every event S,

$$\Pr[M(X) \in S] \le \Pr[\widetilde{M}(X) \in S] + \eta$$

and

$$\Pr[\widetilde{M}(X') \in S] \le \Pr[M(X') \in S] + \eta.$$

Combining the total-variation bound with the (ϵ, δ_t)-DP guarantee of the ideal exact-truncated mechanism gives

$$\begin{aligned}\Pr[M(X) \in S] &\le \Pr[\widetilde{M}(X) \in S] + \eta \\ &\le e^{\epsilon} \Pr[\widetilde{M}(X') \in S] + \delta_t + \eta \\ &\le e^{\epsilon} \Pr[M(X') \in S] + \delta_t + (e^{\epsilon} + 1)\eta.\end{aligned}$$

Since $e^{\epsilon} = 1/s$ and $\eta = (t+1)2^{-\alpha}$, the actual SN-VLM mechanism is

$$\left(\epsilon, \delta_t + \left(\frac{1}{s} + 1\right)(t+1)2^{-\alpha}\right)$$

differentially private. By the definition of δ in the lemma, this is (ϵ, δ)-differential privacy. □

D.3.4 Proof of Lemma 3

Proof. By Lemma 5, it suffices to show that the chosen parameters satisfy

$$\delta_t + \left(\frac{1}{s} + 1\right)(t+1)2^{-\alpha} \le \delta.$$

We bound the two terms separately.

First, since

$$t = \left\lceil \log_2 \left(1 + \frac{\ln(2/\delta)}{\epsilon}\right) \right\rceil,$$

we have

$$2^t \ge 1 + \frac{\ln(2/\delta)}{\epsilon}.$$

Therefore,

$$2^t - 1 \ge \frac{\ln(2/\delta)}{\epsilon}.$$

Using $s = e^{-\epsilon}$, this gives

$$s^{2^t - 1} = e^{-\epsilon(2^t - 1)} \le e^{-\ln(2/\delta)} = \frac{\delta}{2}.$$

Moreover,

$$\frac{1 - s}{(1+s)(1 - s^{2^t})} \le 1,$$

and hence

$$\delta_t = \frac{(1-s)s^{2^t-1}}{(1+s)(1-s^{2^t})} \le s^{2^t-1} \le \frac{\delta}{2}.$$

Second, by the choice of α,

$$\alpha = 3 + \left\lceil \log_2 \frac{t+1}{\delta} \right\rceil,$$

we have

$$2^{-\alpha} \le \frac{\delta}{8(t+1)}.$$

Since $\epsilon \le 1$ and $s = e^{-\epsilon}$, we have

$$\frac{1}{s} + 1 = e^{\epsilon} + 1 \le e + 1 < 4.$$

Therefore,

$$\left(\frac{1}{s}+1\right)(t+1)2^{-\alpha} \le 4(t+1)\cdot\frac{\delta}{8(t+1)} = \frac{\delta}{2}.$$

Combining the two bounds,

$$\delta_t + \left(\frac{1}{s}+1\right)(t+1)2^{-\alpha} \le \frac{\delta}{2}+\frac{\delta}{2} = \delta.$$

Thus, by Lemma 5, the output distribution of **SN-VLM** is (ϵ,δ)-differentially private. □

Extension to approximate additive mechanisms. The shared-sampling and linear aggregation layers extend to additive mechanisms $M(X) = f(X) + Z$ once an auditable shared sampler is available. If M is (ϵ,δ_0)-DP and the implemented noise $\widehat{Z}$ has total variation distance at most η from Z, the same argument gives

$$f(X)+\widehat{Z} \quad \text{is} \quad (\epsilon, \delta_0 + (e^\epsilon+1)\eta)\text{-DP}.$$

A discrete-Gaussian mechanism can be an instance: clipped vector means can be aggregated coordinate-wise, with separate proofs of the clients' norm bounds. The primary technical challenge can be constructing an efficient, auditable finite-precision sampler for the target distribution and establishing its approximation and privacy guarantees.

D.4 Error bounds

We prove the high-probability error bounds used in Section A.3.

D.4.1 Proof of Theorem 7

Proof. The error of **SN-VBM** is exactly the centered binomial noise

$$Z_{\mathsf{Bin}} = \mathsf{Bin}(n_b, 1/2) - \frac{n_b}{2}.$$

By Hoeffding's inequality for a sum of n_b independent Bernoulli variables, for every $\tau > 0$,

$$\Pr\left[|Z_{\mathsf{Bin}}| \ge \tau\right] \le 2\exp\left(-\frac{2\tau^2}{n_b}\right).$$

Setting the right-hand side to γ gives

$$\tau = \sqrt{\frac{n_b}{2}\ln\frac{2}{\gamma}}.$$

Therefore, with probability at least $1-\gamma$,

$$|M_Q(X) - Q(X)| = |Z_{\mathsf{Bin}}| \le \sqrt{\frac{n_b}{2}\ln\frac{2}{\gamma}}.$$

By the parameter setting of **SN-VBM**,

$$n_b = \Theta\left(\frac{1}{\epsilon^2}\ln\frac{1}{\delta}\right).$$

Substituting this into the above bound yields

$$|M_Q(X) - Q(X)| = O\left(\frac{1}{\epsilon}\sqrt{\ln\frac{1}{\delta}\ln\frac{1}{\gamma}}\right)$$

with probability at least $1-\gamma$. □

D.4.2 Proof of Theorem 8

Proof. The error of **SN-VLM** n is generated by

$$u := \sum_{k=0}^{t-1} 2^k a_k, \quad n := u - b - 2ub$$

where

$$a_k \sim \mathsf{Bernoulli}\left(\frac{\lfloor s^{2^k}/(1+s^{2^k})\cdot 2^\alpha \rfloor}{2^\alpha}\right), \quad b \sim \mathsf{Bernoulli}\left(\frac{\lfloor s/(1+s)\cdot 2^\alpha \rfloor}{2^\alpha}\right).$$

We compare u with the ideal exact-truncated geometric component

$$\widetilde{u} := \sum_{k=0}^{t-1} 2^k \widetilde{a}_k,$$

where

$$\widetilde{a}_k \sim \mathsf{Bernoulli}\left(\frac{s^{2^k}}{1+s^{2^k}}\right).$$

As shown in the proof of Lemma 5, for every $z \in \{0,\dots,2^t-1\}$,

$$\Pr[\widetilde{u}=z] = \frac{(1-s)s^z}{1-s^{2^t}}.$$

In the actual finite-precision protocol, each Bernoulli parameter is rounded down by **SharedBern**. Hence each actual Bernoulli bit a_k is stochastically dominated by the corresponding ideal bit $\widetilde{a}_k$. Therefore, u is stochastically dominated by $\widetilde{u}$, and for every nonnegative integer τ,

$$\Pr[u \ge \tau] \le \Pr[\widetilde{u} \ge \tau].$$

Using the distribution of $\widetilde{u}$, we have

$$\Pr[\widetilde{u} \ge \tau] = \sum_{z=\tau}^{2^t-1} \frac{(1-s)s^z}{1-s^{2^t}} = \frac{s^\tau - s^{2^t}}{1-s^{2^t}} \le \frac{s^\tau}{1-s^{2^t}}.$$

Under the parameter setting of Lemma 3, we have

$$s^{2^t} \le s^{2^t-1} \le \frac{\delta}{2} \le \frac{1}{2},$$

as shown in the proof of Lemma 3. Thus,

$$\Pr[u \ge \tau] \le 2s^\tau.$$

If $b=0$, then $n=u$, and if $b=1$, then $n=-u-1$. Hence

$$|n| \le u+1.$$

Therefore, for every positive integer τ,

$$\Pr[|n| \ge \tau] \le \Pr[u \ge \tau - 1] \le 2s^{\tau-1}.$$

Setting the right-hand side to γ gives

$$\tau = 1 + \frac{\ln(2/\gamma)}{-\ln s} = 1 + \frac{\ln(2/\gamma)}{\epsilon}$$

since $s = e^{-\epsilon}$. Therefore,

$$|M_Q(X) - Q(X)| = |n| = O\left(\frac{1}{\epsilon}\ln\frac{1}{\gamma}\right)$$

holds with probability at least $1-\gamma$. □

D.4.3 Proof of Theorem 9

Proof. The error of **SN-VLM** $n = M_Q(X) - Q(X)$ is generated by

$$u := \sum_{k=0}^{t-1} 2^k a_k, \quad n := u - b - 2ub,$$

so $|n| = u + b$.

Downward rounding and comparison with the untruncated geometric variable give

$$\mathbb{E}[|n|] \le \frac{s}{1-s} + 1 \le \frac{2}{\epsilon}.$$

For a lower bound, recall that $0 < \epsilon \le 1$, $0 < \delta < 1$, and

$$t = \left\lceil \log_2 \left(1 + \frac{\ln(2/\delta)}{\epsilon}\right) \right\rceil, \qquad \alpha = 3 + \left\lceil \log_2 \frac{t+1}{\delta} \right\rceil.$$

Let

$$\ell = \lfloor \log_2(1/\epsilon) \rfloor, \qquad k = \min\{t-1, \ell\}, \qquad w = 2^k.$$

Since $t \ge 1$ and $\ell \ge 0$, the index k belongs to $\{0, \dots, t-1\}$ and therefore corresponds to a sampled bit.

We first bound its weight w. By the definition of ℓ,

$$2^\ell \le \frac{1}{\epsilon} < 2^{\ell+1}, \qquad \text{so} \qquad \frac{1}{2\epsilon} < 2^\ell \le \frac{1}{\epsilon}.$$

The definition of t also gives

$$2^t \ge 1 + \frac{\ln(2/\delta)}{\epsilon} > \frac{\ln 2}{\epsilon}.$$

Since $\ln 2 < 1$, both 2^ℓ and 2^{t-1} are at least $\ln 2/(2\epsilon)$. Thus

$$\frac{\ln 2}{2\epsilon} \le w = \min\{2^{t-1}, 2^\ell\} \le \frac{1}{\epsilon}.$$

The target success probability of a_k is

$$p_k = \frac{s^{2^k}}{1 + s^{2^k}} = \frac{1}{1 + e^{\epsilon w}} \ge \frac{1}{1+e},$$

where the inequality follows from $\epsilon w \le 1$. The implemented sampler rounds it down, so

$$\Pr[a_k = 1] = \frac{\lfloor p_k \cdot 2^\alpha \rfloor}{2^\alpha} \ge p_k - 2^{-\alpha}.$$

Moreover, the choice of α, together with $t \ge 1$ and $\delta < 1$, ensures

$$2^{-\alpha} \le \frac{\delta}{8(t+1)} \le \frac{\delta}{16} < \frac{1}{16}.$$

Consequently, the actual success probability remains bounded below by a positive absolute constant:

$$\Pr[a_k = 1] \ge \frac{1}{1+e} - \frac{1}{16} > 0.$$

Finally, $u = \sum_{j=0}^{t-1} 2^j a_j$ has only nonnegative terms, and $|n| = u + b \ge u$ because $b \in \{0, 1\}$. Keeping only the contribution of a_k therefore gives

$$|n| \ge 2^k a_k = w a_k.$$

Taking expectations and applying the preceding bounds yields

$$\mathbb{E}|n| \ge w\mathbb{E}[a_k] = w \Pr[a_k = 1] \ge \frac{\ln 2}{2\epsilon} \left(\frac{1}{1+e} - \frac{1}{16}\right).$$

The coefficient of $1/\epsilon$ is a positive absolute constant, establishing $\mathbb{E}|n| = \Omega(1/\epsilon)$ independently of m. □