VERIFIABLE DIFFERENTIAL PRIVACY

Ari Biswas[1] and Graham Cormode[2]

[1]University Of Warwick
[2]Meta AI

January 20, 2023

ABSTRACT

Differential Privacy (DP) is often presented as a strong privacy-enhancing technology with broad applicability and advocated as a de-facto standard for releasing aggregate statistics on sensitive data. However, in many embodiments, DP introduces a new attack surface: a malicious entity entrusted with releasing statistics could manipulate the results and use the randomness of DP as a convenient smokescreen to mask its nefariousness. Since revealing the random noise would obviate the purpose of introducing it, the miscreant may have a perfect alibi. To close this loophole, we introduce the idea of *Verifiable Differential Privacy*, which requires the publishing entity to output a zero-knowledge proof that convinces an efficient verifier that the output is both DP and reliable. Such a definition might seem unachievable, as a verifier must validate that DP randomness was generated faithfully without learning anything about the randomness itself. We resolve this paradox by carefully mixing private and public randomness to compute verifiable DP counting queries with theoretical guarantees and show that it is also practical for real-world deployment. We also demonstrate that computational assumptions are necessary by showing a separation between information-theoretic DP and computational DP under our definition of verifiability.

1 Introduction

We are living in an age of delegation, where the bulk of our digital data is held and processed by others in an opaque fashion. Our interactions are collated by digital applications that continually send our personal information to the "cloud". Servers in the cloud, typically owned by large monolithic organizations, such as Google, AWS or Microsoft, then perform computations on our private data to publish aggregate statistics for social utility. For example, we send our GPS coordinates to services like Strava and Google which, in exchange, use this information to recommend low-traffic cycling routes [RHML21]. Similarly, we let entertainment companies like Netflix, YouTube, TikTok and Hulu know our personal preferences so that they can better recommend content for us to consume [BK07]. National census bureaus collect personal information to publish aggregate statistics about the population, and consider doing so a moral duty to ensure transparency in the government's policies [BS22].

However, it is often the case that published aggregate statistics leak information about the activity of individuals. For example, Garfinkel *et al.* and Kasiviswanathan *et al.* describe practical reconstruction attacks that can be used to infer an individual's private data from aggregate population statistics [GAM19, KRS13]. Boyd *et al.* show that published census data has been used to discriminate against groups in society based on race [BS22]. Hence the information that is released, and how it is computed, requires careful scrutiny.

In response to these concerns, the privacy and security community have sought to apply various privacy enhancing technologies to protect the privacy of individuals contributing to data releases. Most relevant to this discussion is Differential Privacy (DP) and its generalizations, which require computations to be randomized, in order to offer the (informally stated) promise that users will not be adversely affected by allowing their data to be used. Typically, this is achieved by adding carefully calibrated random noise to the output, at the expense of reducing the accuracy of the computation. Differential privacy is most commonly studied in the *trusted curator* model, where a single entity receives all the sensitive data, and is entrusted to execute the algorithm to apply the random noise. Variations

that modify the trust and computational model include local privacy [War65], shuffle privacy [BBGN19, CSU19], computational differential privacy [MPRV09] and multi-party differential privacy [MMP+10].

A consistent theme across all existing work is to view DP simply as a privacy preserving mechanism. In this paper we shift the focus and view differential privacy through an adversarial lens: *what if an adversary seeks to abuse the protocol and pick noise chosen to distort the statistics, using differential privacy as an attack vector?* That is, a malicious entity may tamper with the computation in order to publish biased statistics, and claim this reflects the true outcome; any discrepancies may be dismissed as artifacts of random noise. Consider a counting query DP protocol to determine the winner of a plurality election, where the users vote for 1 out of M candidates (say, which topping people prefer on their pizza). A corrupted aggregator might not be interested in any particular user's vote but in biasing the aggregate output of the protocol instead. Thus, if that server has auxiliary information about the preferences of a subset of users, they might tamper with the protocol to exclude those honest voters from the election or tamper with the protocol to bias the results of the election (say, to pineapple) and blame any discrepancies in the result on random noise introduced by DP. Note that some loss in accuracy for privacy is unavoidable. By definition, DP requires the output be perturbed by private randomness. Often, outputting such random statistics creates tensions between publishing entity and the downstream consumer. In 2021, the State Of Alabama filed a lawsuit claiming that the use of DP on census data was illegal [Jus21], citing the inaccuracies introduced by DP. Thus, to ensure public trust in DP, it is critical to verify that any loss in utility can be attributed solely to unavoidable DP randomness.

To that effect, we formally introduce the idea of *publicly verifiable differential privacy* in both the trusted curator setting and the multi-party setting in presence of active adversaries[1]. Our contributions are as follows:

1. We formally introduce *publicly verifiable differential privacy* in both the trusted curator and client-server multiparty setting [BDO14]. Informally, the entity responsible for releasing DP statistics is required to also output a zero-knowledge proof to verify that the statistic was computed correctly and the private randomness generated faithfully. Such a proof reveals no additional information and still enforces that user privacy is protected via DP but ensures that the curator cannot use DP randomness maliciously.
2. We show concrete instantiations of verifiable DP by computing DP counting queries (histograms) in the trusted curator setting and the client-server multiparty setting. In the trusted curator setting, there is a single aggregating server that sees client data in plaintext and is responsible for outputting a DP histogram and a proof that the DP noise was generated faithfully. In the client-server MPC setting, clients secret share the inputs and send them to $K \geq 2$ servers, who then participate in an MPC protocol to output DP histograms. The protocol itself is secure in that not even the participating servers are able to learn any new information beyond the output nor are they able to tamper with the protocol.
3. We conduct experiments to show that our protocols with formal theoretical guarantees are also practical. Additionally, we describe how our protocol $\Pi_{\texttt{Bin}}$, for verifiable DP counting, can be combined with existing (non-verifiable) DP-MPC protocols, such as PRIO [CGB17] and Poplar [BBCG+22], to enforce verifiability.
4. We demonstrate that information-theoretic verifiable DP is impossible. Specifically, if both the prover and verifier are computationally unbounded, then both statistical zero knowledge and unconditional soundness cannot hold. Thus we could either prevent an all-powerful curator from manipulating DP protocols or an all-powerful verifier from being able to distinguish between neighbouring datasets from the output, but not both. This result is related to an open problem (Open Problem 10.6) of Vadhan [Vad17], which asks *"Is there a computational task solvable by a single curator with computational differential privacy but is impossible to achieve with information-theoretic differential privacy?"*. In Section 5 we relate our result to efforts at resolving this question.

2 Preliminaries

2.1 Notation

We write $x \xleftarrow{R} U$ to denote that x was uniformly sampled from a set U. We denote vectors with an arrow on top as in $\vec{x} \in \mathbb{Z}_q^M$, where M represents the number of coordinates in the vector and $\mathbb{Z}_q$ represents a prime order finite field of integers of size q. We write $\vec{a} + \vec{b}$ to mean coordinate-wise vector addition $a + b \mod q$, where a and b are arbitrary coordinates of $\vec{a}$ and $\vec{b}$. Similarly, when we write $\vec{a} \times \vec{b}$, we refer to the coordinate-wise Hadamard product between the two vectors.

[1] By active adversaries, we mean participants that may deviate from protocol specifications arbitrarily

2.2 Privacy and Security Background

Indistinguishability. We define a computational notion of indistinguishability.

Definition 1 (Computational Indistinguishability) *Fix security parameter $\kappa \in \mathbb{N}$. Let $\{X_\kappa\}_{\kappa\in N}$ and $\{Y_\kappa\}_{\kappa\in N}$ be probability distributions over $\{0,1\}^{poly(\kappa)}$. We say that $\{X_\kappa\}_{\kappa\in N}$ and $\{Y_\kappa\}_{\kappa\in N}$ are computationally indistinguishable $\{X_\kappa\}_{\kappa\in N} \stackrel{c}{\equiv} \{Y_\kappa\}_{\kappa\in N}$ if for all non-uniform PPT turing machines D ("distinguishers"), there exists a negligible function $\mu(\kappa)$ such for every $\kappa \in \mathbb{N}$*

$$\Big|\Pr[D(X_\kappa)=1] - \Pr[D(Y_\kappa)=1]\Big| \leq \mu(\kappa) \tag{1}$$

Commitments. Commitments are used in our schemes to ensure that participants cannot change their response during the protocol.

Definition 2 (Commitments) *Let $\kappa \in \mathbb{N}$ be the security parameter. A non-interactive commitment scheme consists of a pair of probabilistic polynomial time algorithms ($\texttt{Setup}$, $\texttt{Com}$). The setup algorithm $\texttt{pp} \leftarrow \texttt{Setup}(1^\kappa)$ generates public parameters $\texttt{pp}$. Given a message space $\texttt{M}_{\texttt{pp}}$ and randomness space $\texttt{R}_{\texttt{pp}}$, the commitment algorithm $\texttt{Com}_{\texttt{pp}}$ defines a function $\texttt{M}_{\texttt{pp}} \times \texttt{R}_{\texttt{pp}} \to \texttt{C}_{\texttt{pp}}$ that maps a message to the commitment space $\texttt{C}_{\texttt{pp}}$ using the random space. For a message $x \in \texttt{M}_{\texttt{pp}}$, the algorithm samples $r_x \stackrel{R}{\leftarrow} \texttt{R}_{\texttt{pp}}$ and computes $c_x = \texttt{Com}_{\texttt{pp}}(x, r_x)$. When the context is clear, will drop the subscript and write $\texttt{Com}_{\texttt{pp}}$ as $\texttt{Com}$.*

Definition 3 (Homomorphic Commitments) *A homomorphic commitment scheme is a non-interactive commitment scheme such that $\texttt{M}_{\texttt{pp}}$ and $\texttt{R}_{\texttt{pp}}$ are fields (with $(+,\times)$) and $\texttt{C}_{\texttt{pp}}$ are is an abelian groups with the $\otimes$ operator on which the discrete log problem (Definition 9) is hard, so that for all $x_1, x_2 \in \texttt{M}_{\texttt{pp}}$ and $r_1, r_2 \in \texttt{R}_{\texttt{pp}}$ we have*

$$\texttt{Com}(x_1, r_1) \otimes \texttt{Com}(x_2, r_2) = \texttt{Com}(x_1 + x_2, r_1 + r_2) \tag{2}$$

Throughout this paper, when we use a commitment scheme, we mean a non-interactive homomorphic commitment scheme with the following properties (stated informally here, but formalized in the Appendix A):

1. **Hiding:** A commitment c_x reveals no information about x and r_x to a computationally bounded adversary (Definition 10).
2. **Binding:** Given a commitment c_x to x using r_x, there is no efficient algorithm that can find x' and $r_{x'}$ such that $\mathrm{Com}(x', r_{x'}) = c_x = \mathrm{Com}(x, r_x)$ (Definition 11).
3. **Zero Knowledge OR Opening:** Given c_x, the commiting party is able to prove to a polynomial time verifier that c_x is a commitment to either 1 or 0 without revealing exactly which one it is. We denote such a proof as $\Pi_{\texttt{OR}}$ and say it securely computes the oracle $\mathcal{O}_{\texttt{OR}}$, which computes if $c_x \in L_{\texttt{Bit}}$

   $$L_{\texttt{Bit}} = \{c_x : x \in \{0,1\} \wedge c_x = \mathrm{Com}(x, r_x)\} \tag{3}$$

   where for some $r_x \in \mathbb{Z}_q$. See Appendix C for a concrete construction of the Σ-OR proof using Pedersen Commitment schemes proposed by [Dam00].

In all our experiments and security proofs, we use Pedersen Commitments (PC), though one could replace PC with [WYKW21, DIO20, BMRS21], and still satisfy all the above properties.

Differential Privacy (DP and IND-CDP). We consider two variants of the privacy definition.

Definition 4 (Information Theoretic DP [Vad17]) *Fix $n \in \mathbb{N}, \epsilon \geq 0$ and $\delta \leq n^{-\omega(1)}$. An algorithm $\mathcal{M} : \mathcal{X}^n \times Q \to \mathcal{Y}$ satisfies (ϵ, δ) differential privacy if for every two neighboring datasets $X \sim X'$ such that $||X \sim X'||_1 = 1$ and for every query $Q \in \mathcal{Q}$ we have for all $T \subseteq \mathcal{Y}$*

$$\Pr[M(X,Q) \in T] \leq e^\epsilon \Pr[M(X',Q) \in T] + \delta \tag{4}$$

A direct corollary of the above definition is that, given $M(X,Q)$ and $M(X',Q)$, with probability $1-\delta$ even an unbounded Turing Machine D is unable to distinguish between the outputs up to statistical distance ϵ. We will often write $M(X,Q) \stackrel{(\epsilon,\delta)}{\equiv} M(X',Q)$ as short hand to say that $\mathcal{M}$ is DP.

Algorithm 1 Π_{morra} A protocol for sampling a public coin

Input: $\lambda_1, \ldots, \lambda_K$
Output: $z \xleftarrow{R} \{0,1\}$

1. Each server $k \in [K]$ is asked to sample $m_k \xleftarrow{R} \mathbb{Z}_q$ uniformly at random.
2. *Commit*: Each server samples $r_{m_k} \xleftarrow{R} \mathbb{Z}_q$ and broadcasts $c_k = \text{Com}(m_k, r_k)$ to all other servers. Assume without loss of generality that the servers broadcast their commitments in natural lexicographical order $k \in [K]$.
3. *Reveal*: Once all servers have received c_k, they now broadcast m_k, r_{m_k} to all servers in the reverse order in which the commitments arrived. It is important that the reverse order is respected as it guarantees that each server's inputs are independent of the inputs of other servers. Once all commitments are revealed, each server verifies that $\text{Com}(m_k, r_k) = c_k$. If this test fails for any k or one of the servers does not respond, the protocol is aborted.
4. Each server computes $X = (m_1 + \cdots + m_k) \mod q$. We have $X \xleftarrow{R} \mathbb{Z}_q$. If $X \leq \lceil \frac{q}{2} \rceil$ then $c_i = 0$. Otherwise $c_i = 1$. Thus we can use this protocol to generate unbiased coins and uniformly random values.

Definition 5 (Computational DP [MPRV09]) *Fix $\kappa \in \mathbb{N}$ and $n \in \mathbb{N}$. Let $\epsilon \geq 0$ and $\delta(\kappa) \leq \kappa^{-\omega(1)}$ be a negligible function, and let $\mathcal{M} = \{\mathcal{M}_\kappa : \mathcal{X}_\kappa^n \to \mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of randomised algorithms, where $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ can be represented by $\texttt{poly}(\kappa)$-bit strings. We say that $\mathcal{M}$ is computationally ϵ-differentially private if for every non-uniform PPT TM's ("distinguishers") D, for every query $Q \in \mathcal{Q}$, and for every neighbouring dataset $X \sim X'$, $\forall T \subseteq \mathcal{Y}_\kappa$ we have*

$$\Pr\left[D(\mathcal{M}(X, q) \in T) = 1\right] \leq e^\epsilon \cdot \Pr\left[D(\mathcal{M}(X', q) \in T) = 1\right] + \delta(\kappa) \tag{5}$$

We will often write $M(X, Q) \overset{(\epsilon,\delta)-CDP}{\equiv} M(X', Q)$ as short hand to say that $\mathcal{M}$ is IND-CDP.

Definition 6 *(DP-Error) Let $\mathcal{M} : \mathcal{X} \times \mathcal{Q} \to \mathcal{Y}$ be a (ϵ, δ)-DP mechanism over $\mathcal{Q}$. Assume that the L_1 norm is well-defined on $\mathcal{Y}$. For any $n \in \mathbb{N}$, $X \in \mathcal{X}^n$, we define the expected error of the mechanism $\mathcal{M}$ relative to Q as*

$$\textit{Err}_{\mathcal{M},Q} = \mathbb{E}[\|Q(X) - \mathcal{M}(X, Q)\|] \tag{6}$$

where the expectation is taken over internal randomness of $\mathcal{M}$.

When the context is abundantly clear, to simply notation we will drop the subscripts and refer to equation (6) as just $\texttt{Err}$. It is well known that for negligible δ, the counting query (i.e., DP histograms) has error $\texttt{Err} = O(\frac{1}{\epsilon})$ in the trusted curator model and MPC model [Vad17, CGB17].

Binomial Mechanism. We use Binomial noise to achieve privacy.

Lemma 2.1 (Binomial Mechanism) *Let $X = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ and define counting query $Q(X) = \sum_{i=1}^n x_i$. Fix $n_b > 30$, $0 < \delta \leq o(\frac{1}{n_b})$ and let $Z \sim \texttt{Binomial}(n_b, \frac{1}{2})$. Then $Z + Q(X)$ is an (ϵ, δ)-differentially private with $\epsilon = 10\sqrt{\frac{1}{n_b} \ln \frac{2}{\delta}}$.*

It is easy to see that the binomial mechanism incurs constant DP-error (i.e., it is independent of n, and depends only on ϵ, δ). The proof for Lemma 2.1 can be found in [GGK+20], which we re-derive in Appendix B for completeness.

Morra. We will prove zero knowledge (or security for MPC) assuming that the provers and verifiers (or all participants of the MPC, respectively) have access to an oracle that returns a polynomial sized stream of publicly random unbiased bits. In other words, we assume that all parties have access to an oracle functionality $\mathcal{O}_{\texttt{morra}}(1^\kappa, \lambda_1, \ldots, \lambda_K) = z$ where $z \xleftarrow{R} \{0,1\}$ where and λ_k refers to the empty string for all $k \in [K]$.

In practice, this oracle is replaced by a lightweight MPC protocol such as $\Pi_{\texttt{morra}}$ defined in Algorithm 1, which is a modification of an ancient game called Morra[2], that securely computes $\mathcal{O}_{\texttt{morra}}$ in the presence of a dishonest majority of active participants. It is easy to see that as long as one participant is honest and samples its value uniformly at

[2] https://en.wikipedia.org/wiki/Morra_(game)

random, the final protocol produces an unbiased coin. Since the commitment is hiding, a corrupt party cannot infer any information about the other parties choice m_k from the published c_{m_k} and by the binding property, a participant cannot change their decision after observing another party's opening. A formal simulator-styled proof can be found in Blum's seminal work for flipping coins over a telephone [Blu83] or any introductory textbook on MPC (under the title weak coin flipping). If we omit the final thresholding step, the above protocol can be used to sample $z \xleftarrow{R} Z_q$.

3 Security Models for Verifiable DP

This section introduces verifiable DP in both the single trusted curator and MPC model. In both settings, the input comes from n distinct clients. Informally, the main difference between the two models is that the former has plaintext access to the client data. In contrast, in MPC-DP, the clients secret share (or partition) their inputs and each server receives information theoretically hiding shares (or a partial view) of client inputs. Additionally, instead of a single trusted entity computing $\mathcal{M}$, the servers participate in an MPC protocol Π to securely compute $\mathcal{M}$ without revealing any information other than $\mathcal{M}(x_1, \ldots, x_n, Q)$.

For some queries $Q \in \mathcal{Q}$, the protocol requires that the client inputs come from a restricted subset $L \subseteq \mathcal{X}$. For such cases, if the servers are operating on information-theoretically hiding shares of the inputs, the clients must send a zero-knowledge proof so that the provers can verify that the inputs come from the specified language, without learning any other information about the inputs. Examples of such proofs can be found in the prior literature [BBCG$^+$22, BGI19, CGB17, BBB$^+$18]. In the definitions below and in what follows, we use the terms $\mathtt{Pv}$ (prover), server and curator interchangeably and the terms analyst and $\mathtt{Vfr}$ (verifier) to refer to the same entity.

3.1 Verifiable DP

In what follows, we describe the MPC model and then discuss how it can be specialized to the trusted curator model. Let $\mathcal{M}$ be a DP (or IND-CDP) mechanism as described in Definition 4 (or Definition 5 resepectively) for a query class $\mathcal{Q}$. Let $\kappa \in \mathbb{N}$ denote the security parameter. A verifiable DP mechanism for $\mathcal{M}$ for query class $\mathcal{Q}$ on a dataset X, consists of two interactive protocols $\mathtt{Setup}$ and Π. The first protocol denoted by $\mathtt{Setup}$ is used to generate public parameters. Let $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\kappa)$ denote such public parameters. The second protocol Π is a multiparty protocol between $K+1$ "next-message-computing-algorithms" algorithms $\mathtt{Vfr}$ and $(\mathtt{Pv}_1, \ldots, \mathtt{Pv}_K)$. The total number of rounds of message passing is upper bounded by some polynomial $\mathtt{poly}(\kappa)$. In message passing algorithms, $\mathtt{Vfr}$'s (respectively $\mathtt{Pv}_k$'s) message m_i at round i is determined by its input, messages it has received so far from $\mathtt{Pv}_k$ (respectively $\mathtt{Vfr}$) and internal randomness $\vec{r}_v$ (respectively $\vec{r}_{\mathtt{Pv}_k}$). Let $\vec{\mathtt{Pv}}$ denote a succinct representation for $(\mathtt{Pv}_1, \ldots, \mathtt{Pv}_K)$. Each prover $\mathtt{Pv}_k$ receives on its input tape n inputs $\left([\![x_1]\!]_k, \ldots, [\![x_n]\!]_k\right)$, succinctly denoted by $\vec{X}_k$. Let $\vec{r}_{\mathtt{Pv}_k} \in \{0,1\}^{\mathtt{poly}(\kappa)}$ denote the internal randomness for $\mathtt{Pv}_k$; let $z \in \{0,1\}^{\mathtt{poly}(\kappa)}$ denote auxiliary input available to the verifier; and let $\vec{r}_v \in \{0,1\}^{\mathtt{poly}(\kappa)}$ denote the internal randomness used by the verifying algorithm. At the end of the protocol, the provers send $\vec{y} \in \mathcal{Y}$ to the $\mathtt{Vfr}$, who then outputs either 0 or 1, with 1 indicating that the verifier accepts the provers' claim that, the real protocol output is indistinguishable from an ideal computation, i.e. $\vec{y} = \mathcal{M}(X, Q)$. Let $\mathtt{out}(\mathtt{Vfr}, \vec{y}, \vec{r}_v, \vec{r}_{\vec{\mathtt{Pv}}}, z, \vec{\mathtt{Pv}}, \mathtt{pp}) \in \{0,1\}$ denote the verifying algorithm's decision. In the definition below, we write $\mathtt{out}(\mathtt{Vfr}, \mathtt{Pv})$ as shorthand for $\mathtt{out}(\mathtt{Vfr}, \vec{y}, \vec{r}_v, \vec{r}_{\vec{\mathtt{Pv}}}, z, \vec{\mathtt{Pv}}, \mathtt{pp})$. The trusted curator can be understood as essentially this model with a single prover, i.e., we set $K = 1$. Thus the only functional difference between MPC-DP and trusted curator DP is that in the latter case, the curator sees all the data in plaintext. In MPC, the data may be secret, shared or partitioned across the provers. In both cases, the prover(s) must prove that they did not tamper with the protocol to generate an output that is distinguishable from the ideal computation of $\mathcal{M}$.

Definition 7 (Verifiable DP) *A constant round interactive verifiable DP protocol for $\mathcal{M}$ consists of two algorithms $\mathtt{Setup}$ and Π, such that for $n \in \mathbb{N}$ clients, $K \geq 1$ provers denoted by $\vec{\mathtt{Pv}}$ and a single verifier $\mathtt{Vfr}$, there exists negligible functions δ_c and δ_s such that*

1. ***Completeness:*** *Let $X = (x_1, \ldots, x_n) \in \mathcal{X}^n$ be the client inputs that have been split in K shares $(\vec{X}_1, \ldots, \vec{X}_K)$, where $\vec{X}_j$ denotes the input sent to the j'th prover, then as long as the $\vec{\mathtt{Pv}}$ and $\mathtt{Vfr}$ honestly execute Π, then we have*

$$\Pr\left[\mathit{out}(\mathtt{Vfr}, \vec{\mathtt{Pv}}) = 0 : \begin{array}{c} \mathit{pp} \leftarrow \mathit{Setup}(1^\kappa) \\ \mathtt{Pv}_j \leftarrow (\vec{X}_j, \vec{r}_{\mathtt{Pv}_j}, \mathit{pp}) \\ \mathtt{Vfr} \leftarrow (z, \vec{r}_v, \mathit{pp}) \\ \vec{y} \leftarrow \Pi(\vec{\mathtt{Pv}}, \mathtt{Vfr}, \mathit{pp}) \end{array}\right] \leq \delta_c.$$

2. ***Soundness:*** *For every $X \in \mathcal{X}^n$ and any subset $I \subseteq [K]$, let $\vec{\mathtt{Pv}}^*$ denote the collection of corrupted provers, indexed by I, that deviate from Π, such that the final output $\vec{y} \neq \mathcal{M}(X, Q)$, then we have*

$$\Pr\left[\mathit{out}(\mathtt{Vfr}, \vec{\mathtt{Pv}}) = 1 : \begin{array}{c} \mathit{pp} \leftarrow \mathit{Setup}(1^\kappa) \\ \mathtt{Pv}_j \leftarrow (\vec{X}_j, \vec{r}_{\mathtt{Pv}_j}, \mathit{pp}) \\ \mathtt{Vfr} \leftarrow (z, \vec{r}_v, \mathit{pp}) \\ \vec{y} \leftarrow \Pi(\vec{\mathtt{Pv}}^*, \mathtt{Vfr}, \mathit{pp}) \end{array}\right] \leq \delta_s.$$

Note that the correctness of the protocol is defined in terms of the actual inputs the clients sent to $\mathcal{M}$ and not the inputs a corrupted set of provers might have used. Thus if corrupted prover(s) tampered with the inputs on its input tape and then executed the protocol faithfully, by our definitions, it still counts as cheating.

3. ***Zero-Knowledge****: Let $\vec{\mathtt{Pv}}^*$ denote the collection of provers corrupted by any verifying algorithm $\mathtt{Vfr}^*$ and $I \subset [K]$ denote their indices. Let $\vec{\mathtt{Pv}}^*$ denote the set of honest provers not indexed by I. Let $\mathit{View}\Big[\Pi\Big((\vec{\mathtt{Pv}}, \vec{\mathtt{Pv}}^*), \mathtt{Vfr}^*\Big)\Big]$ be the joint distribution[3] of messages and output received during the execution of Π in the presence of corrupted parties. There exists a PPT Simulator $\mathsf{Sim}_{(\mathtt{Vfr}^*, I)}$ such that for all $\vec{y} = \mathcal{M}(X, Q)$*

$$\mathit{View}\Big[\Pi\Big((\vec{\mathtt{Pv}}, \vec{\mathtt{Pv}}^*), \mathtt{Vfr}^*\Big)\Big] \equiv \mathsf{Sim}_{(\mathtt{Vfr}^*, I)}(\vec{y}, \vec{r}_v, z, \mathit{pp})$$

where $z \in \{0,1\}^{\mathit{poly}(\kappa)}$ represents auxiliary input available to all the corrupted parties. Contrary to soundness, for zero-knowledge to hold, the simulated transcript should be indistinguishable from the actual protocol transcript, based on the inputs adversaries used and not the ones the clients sent to a set of corrupted provers.

An interesting point to note is that in verifiable differential privacy, the verifier plays a dual role. An honest verifier ensures that the output is faithfully generated and thus plays an active role in generating the DP noise without ever seeing it in plaintext. On the other hand, a dishonest verifier tries to tamper with the protocol to breach privacy. In non-verifiable DP, the analysts (verifier) only have access to DP the statistic. They have no agency over how the output is generated. Thus the verifier participating in verifiable DP has a greater attack surface than a classical adversary in traditional non-verifiable DP. We elaborate on this in Section 5, when trying to establish separations between statistical DP and computational DP. Additionally, just like in standard MPC, in the presence of a dishonest majority of corrupted participants, we do not treat early exiting by corrupted parties as a breach of security. This is easily detected by the honest parties, and the output is ignored. Verifiable DP, just like interactive zero-knowledge proofs [Gol07] comes in 24 different flavours based on the capabilities of the corrupted parties:

1. **Distinguishability:** Based on the distinguishability properties of the simulator algorithm, the protocol may be perfect, statistical or computationally zero-knowledge. The protocol described in Section 4 is computationally zero-knowledge.
2. **Verifier specifications:** Based on whether the verifier is expected to follow the rules of the protocol (semi-honest) or may deviate arbitrarily (active), we get honest-verifier zero knowledge or just zero knowledge. All our results are zero-knowledge.
3. **Soundness:** Based on the power of the corrupted provers, the protocol may be computationally sound (also known as arguments) or statistically sound (secure against unbounded provers). The verifiable DP protocol in Section 4 is computationally sound.
4. **Inputs:** Based on whether the verifier has access to the auxiliary input, the protocol could be plaintext zero-knowledge or auxiliary input zero knowledge. Our protocols allow for the verifier to have auxiliary input.

4 Verifiable Binomial Mechanism

This section describes how to compute counting queries verifiably with differential privacy in both the single curator and client-server MPC models. We consider the trusted curator model to be a special instantiation of the general MPC

[3] As $\mathcal{M}$ is a random function, the *joint distribution* of the view of the adversary and their output must be indistinguishable from the simulated transcript (and not just the view of the adversary). See [Lin17] for more details.

model where the number of provers $K = 1$. In Section 4.1 we describe intuitions for our protocol, and in Section 4.2 we explain what is needed for verifiability in the MPC setting, and tackle the additional challenges of verifying client inputs. We describe how prior efforts at verifying clients fall short of the security expectations of Definition 7. Finally, in Section 4.3 we describe a protocol that computes counting queries with DP verifiably.

Set $\mathcal{X} = \mathbb{Z}_q = \mathcal{Y}$, where $\mathbb{Z}_q$ is a prime order finite field of size q over the integers. Let $X = (x_1, \ldots, x_n)$ denote the client inputs and the Q be the counting query $Q(X) = \sum_{i=1}^{n} x_i$. Let $[\![x_i]\!]_k$ denote the k'th additive secret[4] share of a client input x_i. Each client splits their input into K secret shares and distributes them across the provers. We will assume that $n \ll q$ and $\kappa = \lfloor \log_2 q \rfloor$ can be viewed as the security parameter. For $K \geq 1$ provers and 1 verifier, define the oracle functionality $\mathcal{M}_{\texttt{Bin}}$ in the ideal world as follows:

1. $\mathcal{M}_{\texttt{Bin}}$ receives public privacy parameters ϵ and δ. It then computes n_b based on Lemma 2.1.
2. Let $\left([\![x_1]\!]_k, \ldots, [\![x_n]\!]_k\right)$ denote the inputs on the k'th prover's input tape. Each prover $\texttt{Pv}_k$ is expected to compute $X_k = \sum_{i=1}^{n} [\![x_i]\!]_k$ and sends to $\mathcal{M}_{\texttt{Bin}}$ as its input X_k. A corrupted prover might send an arbitrary input.
3. $\mathcal{M}_{\texttt{Bin}}$ samples $\Delta_k \sim \texttt{Binomial}(n_b, 1/2)$ independently for each input X_k it receives. It then computes
$$y = \sum_{k=1}^{K} (X_k + \Delta_k) \tag{7}$$
4. $\mathcal{M}_{\texttt{Bin}}$ sends the tuple (y, Δ_k) as output to each prover $\texttt{Pv}_k$. On receiving its output, the $\texttt{Pv}_k$ sends `CONTINUE` to $\mathcal{M}_{\texttt{Bin}}$. Once $\mathcal{M}_{\texttt{Bin}}$ receives the continue signal from prover $\texttt{Pv}_k$ it moves on to deliver output to $\texttt{Pv}_{k+1}$.
5. After all K provers have sent `CONTINUE`, $\mathcal{M}_{\texttt{Bin}}$ sends y as output to the verifier `Vfr`. If a single prover fails to send the continue message and thereby exits the protocol early, the verifier and the remaining provers do not receive any output.

When $K = 1$, i.e., the trusted curator setting, the single prover receives n client inputs in plaintext, so $[\![x_i]\!]_k = x_i$ for all $i \in [n]$. This is equivalent to an adversary corrupting all K provers. Thus in the MPC setting with $K \geq 2$ servers, it is safe to assume at least one of them will follow the protocol. Our goal is to be able to come up with an interactive protocol $\Pi_{\texttt{Bin}}$, which allows us to compute $\mathcal{M}_{\texttt{Bin}}$ verifiably as per Definition 7. Notice that in the ideal model definition above, the oracle adds K independent copies of DP noise to the output, whereas Lemma 2.1 only calls for a single copy. This is because, as we allow up to $K - 1$ provers to collude with a corrupted verifier, the corrupted provers could simply not add any noise to the output. Ben Or *et al.* 's completeness results [BOGW19] imply that K independent copies of noise are *necessary* to guarantee differential privacy unless the number of corruptions can be restricted to being strictly less than $\frac{K}{3}$, so each prover must independently generate enough noise to guarantee DP. Our protocols defined below are secure against computationally bounded provers and verifiers that may deviate arbitrarily from protocol specifications and have access to auxiliary inputs.

4.1 An Intuitive But Incomplete Protocol

Before describing the full protocol in Section 4.3 and Figure 2, we provide the reader with some intuition as to why the protocol works for a single curator and verifier. *In this section, we make the unrealistic assumption that prover and verifier behave faithfully*. Assume all parties have joint oracle access to $\mathcal{O}_{\texttt{Morra}}$ (as described in Section 2.2) to jointly sample unbiased bits $(b_1, \ldots, b_{n_b})$. It is easy to see that using $(\sum_{i=1}^{n_b} b_i)$ as DP randomness results in the desired distribution defined in $\mathcal{M}_{\texttt{Bin}}$. However, the oracle output is publicly known to both the verifier and prover; therefore, it cannot be directly used to guarantee differential privacy. As discussed earlier, this problem of proving that a prover faithfully sampled random bits without disclosing them lies at the heart of any verifiable DP protocol. Thus the protocol must combine public coins that satisfy verifiability requirements and private coins that ensure secrecy.

The protocol for verifiable DP counting proceeds in n_b identical and independent invocations (run in parallel). In copy i, the prover samples $v_i \in \{0, 1\}$, which it keeps private. Note that a prover could sample this bit using any arbitrary bias. As this is the provers' private coin, the verifier has no control over how the prover generates this information. After the prover has sampled their private bit, the prover and verifier make one call to $\mathcal{O}_{\texttt{Morra}}$ to get an unbiased coin denoted by b_i. Next, the prover locally computes $\hat{v}_i = b_i \oplus v_i$. Here $\oplus$ refers to the boolean XOR operation. It is easy to see that $\hat{v}_i$ has the same distribution as b_i, but its value is known only to the parties with access to v_i, i.e., the prover. After n_b rounds, the prover computes $Q(X)$ and $Z = \sum_{i=1}^{n_b} \hat{v}_i$ and outputs $Q(X) + Z$ where Z is used as DP randomness. By the assumption that the prover and verifier are faithful, Z is distributed according to the desired

[4] Although we describe our protocols with additive secret sharing, any linear secret sharing such as Shamir's secret sharing also applies to all our results.

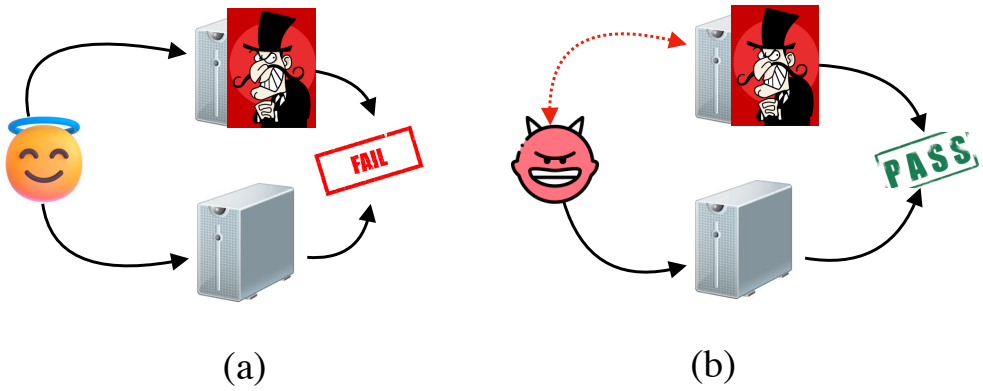

Figure 1: Two types of attacks that go undetected in Poplar. In (a) regardless of what the honest client sends, a corrupted server simply ignores the input and excludes the client from the protocol based on auxiliary information. In (b) a dishonest client colludes with the corrupted server by revealing secret values, so that an illegal input is included. In both cases, the honest server cannot distinguish between an honest run and a corrupted run of the protocol.

distribution stated in Theorem 2.1, and its value is only known to the prover. To make this protocol practical, we need to resolve a few issues.

1. Although the above description requires a bitwise XOR operation to ensure the right distribution is used, we operate with arithmetic circuits in the actual protocol. Thus, the provers could sample arbitrary values $v^* \in \mathbb{Z}_q$ such that $v^* \notin \{0, 1\}$, and we need to fix how to express the XOR operation via arithmetic circuits.
2. Even if we could verify that the prover sampled a private bit correctly, we still need to verify that they faithfully performed the local operations discussed above.

Thus, if we could guarantee that each server performed its computations correctly and sampled a private value from the correct set, we would get the desired outcome of verifiable and DP counting queries.

4.2 Extending To Client-Server MPC-DP

To compute DP histograms verifiably in the client-server MPC-DP setting, we use the same computational model used for PRIO [CGB17] and Poplar [BBCG+22], two real-world systems deployed at scale by Mozilla[5]. As discussed earlier, in this setting n clients secret share their inputs $x_i \in L$ amongst $K \geq 2$ provers, where $L \subseteq \mathcal{X}$ defines the language of legal inputs to the protocol. For computing M-bin histograms over n inputs, L is the set of all one-hot encoded vectors of size M. For the core problem of a single-dimensional counting query, $M = 1$ and $L = \{0, 1\}$. Since the inputs on the prover's tapes reveal no information about a client's input, for the protocol to be useful the provers must first verify in zero-knowledge that $x_i \in L$ before using such inputs to compute aggregate statistics. This additional step of verifying a client is not required in the trusted curator model, as the prover decides what inputs should be included in the computation and can see them in plaintext.

Verifying Clients in MPC-DP. Poplar and PRIO use efficient sketching techniques from [BGI16] to validate a client's input in zero knowledge *without* relying on any public key cryptography. Thus, as long as at least one out of K provers does not reveal the inputs it received, even an unbounded adversary corrupting the remaining provers cannot ascertain any information about an honest client's input. While such a system protects an honest client's privacy from an unbounded adversary, it is not verifiable as per Definition 7. Specifically, for the techniques used in PRIO and Poplar, a single corrupted prover could tamper with its inputs and exclude an honest client from the protocol by forcing them to fail the verification test. Alternatively, a corrupt client could collude with a prover to include arbitrary inputs, jeopardising the correctness of the output. Figure 1 summarises these attacks on Poplar and PRIO[6]. By our definitions of verifiability, the protocol's output *must* be a function of the inputs provided by honest clients only. Thus the protocol described in Section 4.3 provides the following additional guarantees:

[5] https://blog.mozilla.org/security/2019/06/06/next-steps-in-privacy-preserving-telemetry-with-prio/

[6] Concretely, in scenario (b), the dishonest client reveals the values κ and $[v]_0$ to the server. This allows the server to set $z_1 = -z_0, z_1^* = -z_0^*$ and $z_1^{**} = -z_0^{**}$, thereby admitting an illegal input into the protocol.

	Public Verifier($\texttt{Vfr}$)		**Prover**($\texttt{Pv}_k$)
1 :	$\texttt{pp} \leftarrow \texttt{Setup}(1^\kappa)$	Generate public parameters	$\texttt{pp} \leftarrow \texttt{Setup}(1^\kappa)$
2 :	$\left\{\left\{c_{i,k}\right\}_{k\in[K]}\right\}_{i\in[n]}$	Client inputs	$\left\{[\![x_i]\!]_k, r_{i,k}\right\}_{i\in[n]}$
3 :	Verify using $\mathcal{O}_{\texttt{OR}}$ that$\forall i \in [n], x_i \in L$		Verify using $\mathcal{O}_{\texttt{OR}}$ that$\forall i \in [n], x_i \in L$
4 :	$(c'_{1,k}, \ldots, c'_{n_b,k})$	$c'_{j,k} = \mathrm{Com}\left(v_{j,k}, s_{v_{j,k}}\right) \longleftarrow$	$\forall j \in [n_b]$ Samples and commits $v_{j,k} \in \{0,1\}$
5 :	$\forall j \in [n_b]$ Send $c'_{j,k}$	$\mathcal{O}_{\texttt{OR}}$	$\forall j \in [n_b]$ Send openings$(v_{j,k}, s_{j,k})$
6 :	$\forall j \in [n_b]$ Check $\mathcal{O}_{\texttt{OR}}(c'_{j,k}) = 1$		
7 :	$\forall j \in [n_b]$ Send empty string λ_j	$\mathcal{O}_{\texttt{Morra}}$	$\forall j \in [n_b]$ Send empty string λ_j
8 :	Receive $(b_{1,k}, \ldots, b_{n_b,k})$	$\forall j \in [n_b]\ b_{j,k} = \mathcal{O}_{\texttt{Morra}}(\lambda_j)$	Receive $(b_{1,k}, \ldots, b_{n_b,k})$
9 :			$\forall j \in [n_b]$ Adjust $v_{j,k}$ based on $b_{j,k}$, to get $\hat{v}_{j,k}$
10 :		$y_k \longleftarrow$	$y_k = \sum_{i=1}^{n} [\![x_i]\!]_k + \sum_{j=1}^{n_b} \hat{v}_{j,k}$
11 :		$z_k \longleftarrow$	$z_k = \left(\sum_{i=1}^{n} r_{i,k} + \sum_{j=1}^{n_b} s_{j,k}\right)$
12 :	Compute $\hat{c}'_{j,k}$ using $b_{j,k}$ for all $j \in [b_{n_b}]$		
13 :	Check that $\left(\prod_{i=1}^{n} c_{i,k} \times \prod_{j=1}^{n_b} \hat{c}'_{j,k}\right) = \mathrm{Com}(y_k, z_k)$		

Figure 2: The figure above describes the interaction between a single prover and verifier in $\Pi_{\texttt{Bin}}$. In the single trusted curator model $K = 1$ we have $x_i = [\![x_i]\!]_k$ where the prover can see client inputs in plaintext. In the MPC setting, each prover $\texttt{Pv}_k$ follows the exact same protocol on their respective inputs specified in Line 2. Thus at the end of the protocol, each prover $\texttt{Pv}_k$ outputs the tuple y_k, z_k. A public verifier aggregates the output from each prover to publish verifiable DP statistics.

1. **Guaranteed Inclusion Of Honest Clients**: If a client submits shares of an input $x \in L$, then the final output of the protocol is guaranteed to use this input untampered. Thus an honest client is assured that, as long as a single prover follows the protocol specifications, no one learns any information about their private input and their input is correctly used to compute the final output.
2. **Guaranteed Exclusion Of Corrupt Clients**: A corrupted client, even one that has control over any proper subset of the K provers, cannot include an invalid input to the protocol. Thus if $x \notin L$, x is discarded by our protocol with overwhelming probability.

It is important to note that as we operate under stricter notions of privacy and correctness, our results require the use of public-key cryptography and security holds only against computationally bounded adversaries. Furthermore, we show in Section 5 that it is impossible to satisfy verifiable DP and provide information theoretic guarantees.

4.3 Main Protocol Description

The protocol $\Pi_{\texttt{Bin}}$ described in Figure 2 provides a compact standalone description of the interaction between K provers and a public verifier for computing $\mathcal{M}_{\texttt{Bin}}$. As the verifier is public, anyone (even non-participants to $\Pi_{\texttt{Bin}}$) can see the messages it receives from the clients. We assume that both the provers and the verifier have access to oracles $\mathcal{O}_{\texttt{morra}}$ and $\mathcal{O}_{\texttt{OR}}$ as defined in Section 2.2. In the real world, $\mathcal{O}_{\texttt{Morra}}$ is replaced with $\Pi_{\texttt{Morra}}$ (see Algorithm 1) and $\mathcal{O}_{\texttt{OR}}$ is replaced by Cramer *et al.* 's Σ-OR proof [CDS94] (see Appendix C for an example implementation) which securely compute the oracle functionalities in the presence of adversaries that may deviate from protocol specifications. Thus, we define our protocol in the hybrid world, and by the sequential composition theorem[7] [Gol07], the security properties of the protocol are preserved. Next, we describe the protocol in detail with line references to Figure 2:

[7]Though we use sequential composition, both protocols $\Pi_{\texttt{morra}}$ and $\Pi_{\texttt{or}}$ can be parallelly composed.

Line 1: In the first step, the prover(s) and verifier agree upon the public parameters for the protocol. The public parameters include a description of $\mathcal{C}_{\texttt{pp}} = \mathbb{G}_q, \mathcal{M}_{\texttt{pp}} = \mathcal{X} = \mathcal{Y} = \mathbb{Z}_q, \mathcal{R}_{\texttt{pp}} = \mathbb{Z}_q$ and a description of $\mathcal{M}_{\texttt{bin}}$ as defined in equation (7). The group $\mathbb{G}_q$ satisfies the requirements of the homomorphic commitment scheme defined in Section 3.

Line 2: For each client $i \in [n]$, let $[\![x_i]\!]_k$ denote the k'th share of their input $x_i \in L$. Define $c_{i,k} = \text{Com}\left([\![x_i]\!]_k, r_{i,k}\right)$ as the commitment to the k'th share of x_i. The client sends to each prover $\texttt{Pv}_k$ the tuple $([\![x_i]\!]_k, r_{i,k})$ and publicly broadcasts the commitments to their shares $\left(c_{i,1}, \ldots, c_{i,K}\right)$ to the public verifier observable to all parties.

Line 3: Similar to PRIO and Poplar, we use $L = \{0, 1\}$, and thus verifier and the client use the oracle $\mathcal{O}_{\texttt{OR}}$ to check if the client's input is indeed a commitment to a bit. For input x_i, the verifier sends to $\mathcal{O}_{\texttt{OR}}$ the derived commitment $c_i = \prod_{k=1}^{K} c_{i,k}$ and the client sends the openings $\left(x_i, \sum_{k=1}^{K} r_{i,k}\right)$. The oracle responds with $\mathcal{O}_{\texttt{OR}}(c_i) = 1$ if $x_i \in \{0, 1\}$ and c_i is a commitment to x_i. Note that all messages sent to the verifier are public, so the servers can independently validate the verifier's claims. As there is always one honest participant in the protocol (at least one prover or the verifier), this step provides a public record of honest and dishonest clients. Such a record resolves the issues presented in Figure 1. From here on, the protocol only uses inputs from validated clients.

Line 4: $\texttt{Pv}_k$ samples $(v_{1,k}, \ldots, v_{n_b,k})$ where $v_{j,k} \in \{0, 1\}$ (private random bit) and sends to the verifier commitments to $v_{j,k}$ for $j \in [n_b]$. Let $c'_{j,k} = \text{Com}(v_{j,k}, s_{j,k})$ denote the commitment to $v_{j,k}$ with randomness $s_{j,k}$. To enforce consistency in notation and improve readability, we always use c to denote commitments to client inputs and c' to denote commitments to the prover's private inputs. Similarly, we will always use r and s to denote the randomness used for client input and prover bit commitments, respectively.

Line 5-6: The verifier uses $\mathcal{O}_{\texttt{OR}}$ to check if the messages sent by the prover were indeed commitments to 0 or 1. This step is essential for the boolean to the arithmetic conversion, as the linearisation of the XOR operation is only valid for values $v \in \{0, 1\}$ (see completeness property of Theorem 4.1).

Line 7-8: If for any $i \in n_b$, $\mathcal{O}_{\texttt{OR}} = 0$, the verifier aborts the protocol and publicly declares that $\texttt{Pv}_k$ cheated. Otherwise, once all commitments are verified, the prover and verifier jointly invoke $\mathcal{O}_{\texttt{morra}}$ to get n_b *public* unbiased bits $(b_{1,k}, \ldots, b_{n_b,k})$.

Line 9: For all $i \in [n_b]$, based on the value of $b_{j,k}$, the prover sets $\hat{v}_{j,k}$ as follows

$$\hat{v}_{j,k} = \begin{cases} 1 - v_{j,k} & \text{if } b_{j,k} = 1 \\ v_{j,k} & \text{otherwise.} \end{cases}$$

As long as $v_{j,k} \in \{0, 1\}$, the above set of equations is equivalent to setting $\hat{v}_{j,k} = v_{j,k} \oplus b_{j,k}$. An important feature of this step is that, conditioned on $b_{j,k}$, the operations described above are linear. In Line 11, we describe why this is critical for correctness to hold.

Line 10-11: The prover sends (y_k, z_k) to the verifier:

$$y_k = \left(\sum_{i=1}^{n} [\![x_i]\!]_k + \sum_{j=1}^{n_b} \hat{v}_{j,k}\right) \tag{8}$$

$$z_k = \left(\sum_{i=1}^{n} r_{i,k} + \sum_{j=1}^{n_b} s_{j,k}\right) \tag{9}$$

where (y_k, z_k) is the output for prover $\texttt{Pv}_k$

Line 12: Using the common public randomness $\{b_{j,k}\}_{j \in [n_b]}$ generated by $\mathcal{O}_{\texttt{morra}}$, the verifier updates their view of received commitments as follows:

$$\hat{c}'_{j,k} = \begin{cases} \text{Com}(1, 0) \times {c'}_{j,k}^{-1} & \text{if } b_{j,k} = 1 \\ c'_{j,k} & \text{otherwise.} \end{cases}$$

Note that $\texttt{Pv}_k$ never opens ${c'}_{j,k}$, and thus $\texttt{Vfr}$ never sees $\hat{v'}_{j,k}$ in plaintext. By the hiding property of commitments, an efficient verifier learns nothing about the prover's private values from these messages. However, as the update conditioned on $b_{j,k}$ is linear and $b_{j,k}$ is public, $\texttt{Vfr}$ can still compute a commitment to $1 - v_{j,k}$ without ever knowing $v_{j,k}$. As a direct consequence, as discussed in the soundness claim, the prover cannot deviate from its prescribed linear operation, as the verifier is able to check it. As we will show later, this step guarantees correctness, soundness and security.

Line 13: Finally, the verifier checks

$$\prod_{i=1}^{n} c_{i,k} \times \prod_{j=1}^{n_b} \hat{c}'_{j,k} = \mathrm{Com}(y_k, z_k) \tag{10}$$

From these outputs, we can derive the desired result: we treat the y_k's as shares, and calculate $y = \sum_{k=1}^{K} y_k$ as the noisy sum. We next show that this protocol achieves our desired properties.

Theorem 4.1 *Let $X = (x_1, \ldots, x_n)$ be the client input. Let $\mathcal{M}_{Bin}$ and $\mathcal{O} = (\mathcal{O}_{morra}, \mathcal{O}_{OR})$ be as defined above. Assuming Π_{Bin} is run with $K \geq 1$ provers and a single verifier, then the following is true*

Completeness: *For every $X \in \mathcal{X}^n$*

$$\Pr\left[out(\mathtt{Vfr}, \vec{\mathtt{Pv}}) = 0 : \begin{array}{c} pp \leftarrow Setup^{\mathcal{O}}(1^\kappa) \\ \mathtt{Pv}_k^{\mathcal{O}} \leftarrow [\![X]\!]_k, \vec{r}_{\mathtt{Pv}_k}, pp \\ \mathtt{Vfr}^{\mathcal{O}} \leftarrow z, \vec{r}_v, pp \\ y \leftarrow \Pi_{Bin}^{\mathcal{O}}(\vec{\mathtt{Pv}}, \mathtt{Vfr}) \end{array} \right] = 0$$

where $[\![X]\!]_k = ([\![x_1]\!]_k, \ldots, [\![x_n]\!]_k)$ and $\vec{r}_{\mathtt{Pv}_k}$ denotes $\mathtt{Pv}_k$'s private randomness.

Computational Soundness: *For every $X \in \mathcal{X}^n$ and any subset $I \subseteq [K]$, let $\vec{\mathtt{Pv}}^*$ denote the collection of provers, indexed by I, that have been corrupted by an adversary $\mathcal{A}$, such that the final output $y \neq \mathcal{M}_{Bin}(X, Q)$. Let $\vec{\mathtt{Pv}}$ denote the collection of honest provers not indexed by I. Let z denote the auxiliary input available to $\mathcal{A}$ and μ be its advantage in the discrete log game (Definition 9)*

$$\Pr\left[out(\mathtt{Vfr}, \vec{\mathtt{Pv}}^*, \vec{\mathtt{Pv}}) = 1 : \begin{array}{c} pp \leftarrow Setup^{\mathcal{O}}(1^\kappa) \\ \mathtt{Pv}_k^{\mathcal{O}} \leftarrow [\![X]\!]_k, \vec{r}_{\mathtt{Pv}_k}, pp \\ \mathtt{Vfr}^{\mathcal{O}} \leftarrow z, \vec{r}_v, pp \\ y \leftarrow \Pi_{Bin}^{\mathcal{O}}\Big((\vec{\mathtt{Pv}}, \vec{\mathtt{Pv}}^*), \mathtt{Vfr}\Big) \end{array} \right] \leq \mu(\kappa)$$

Note that as $I \subseteq [K]$, soundness, as defined above covers both the MPC and the trusted curator setting.

Computational Zero-Knowledge*: Let $\vec{\mathtt{Pv}}^*$ denote the collection of provers, indexed by $I \subset [K]$, that have been corrupted by a corrupt verifier $\mathtt{Vfr}^*$. There exists a PPT Simulator $\mathsf{Sim}_{(\mathtt{Vfr}^*, I)}$ such that for all $y = \mathcal{M}(X, Q)$*

$$View\left[\Pi\Big((\vec{\mathtt{Pv}}, \vec{\mathtt{Pv}}^*), \mathtt{Vfr}^*, pp\Big)\right] \stackrel{c}{\equiv} \mathsf{Sim}_{(\mathtt{Vfr}^*, I)}(y, \vec{r}_v, z, pp)$$

where $z \in \{0,1\}^{poly(\kappa)}$ and $\vec{r}_v \in \{0,1\}^{poly(\kappa)}$ represents auxiliary input and randomness available to all the corrupted parties.

Proof 1

1. ***Completeness:*** *By the definition of $\mathcal{O}_{morra}$, $(b_{1,k}, \ldots, b_{n_b,k})$ are all unbiased bits. As per Π_{Bin}, when $b_{j,k} = 1$, $\hat{v}_{j,k} = 1 - v_{j,k}$ and when $b_{j,k} = 0$, $\hat{v}_{j,k} = v_{j,k}$. We know that an honest prover is guaranteed to have sampled a private value $v_{j,k} \in \{0,1\}$ for all $j \in [n_b]$. Thus the case-wise arithmetic operation described above is equivalent to setting $\hat{v}_{j,k} = v_{j,k} \oplus b_{j,k}$. This implies that for each server $\hat{v}_{j,k} \xleftarrow{R} \{0,1\}$ and $\sum_{j=1}^{n_b} \hat{v}_{j,k} \sim Binomial(n_b, 1/2)$. The output of each honest prover is thus $y_k = Binomial(n_b, 1/2) + \sum_{i=1}^{n} [\![x_i]\!]_k$. By linearity of secret-sharing, $\sum_{k \in [K]} y_k = \mathcal{M}_{Bin}(X, Q)$ where $\mathcal{M}_{Bin}$ is defined in equation (7).*

2. ***Soundness:*** *Beyond exiting the protocol early (which is trivially detected), an adversary $\mathcal{A}$ controlling a collection of dishonest provers could force a prover to cheat by doing at least one of the following:*

   (a) *(Cheat at Line 4): For any $j \in [n_b]$, $c'_{j,k}$ is not a commitment to a bit. As the verifier has access to oracle $\mathcal{O}_{OR}$, it would detect this immediately. Thus we can be guaranteed that $c'_{j,k}$ are commitments to 1 or 0.*

   (b) *(Cheat at Line 7): The prover could sample improper public randomness. However, this is impossible as the verifier and prover jointly use $\mathcal{O}_{Morra}$ to generate randomness.*

(c) *(Cheat at Line 10): Output messages* $(y'_k \neq y_k,\ z'_k \neq z_k)$. *If the verifier check from (Line 12) fails then the verifier knows* $\texttt{Pv}^*_k$ *cheated. If* $\mathrm{Com}(y_k, z_k) = \prod_{i=1}^{n} c_{i,k} \times \prod_{j=1}^{n_b} \hat{c}'_{j,k} = \mathrm{Com}(y'_k, z'_k)$, *then* $\mathcal{A}$ *has broken the binding property of the commitment scheme. As* $\mathcal{A}$ *has negligible success in winning the discrete log game, it has a negligible chance at breaking the commitment scheme.*

These are the only places where the $\texttt{Pv}^*$ *sends a message to the* $\texttt{Vfr}$ *and thus we have our result.*

3. ***Zero-Knowledge:*** *To prove zero knowledge we need to explicitly define the commitment scheme we are using. We use Pedersen Commitments which are defined as follows*

$$\mathrm{Com}(x, r) = g^x h^r \tag{11}$$

where $\mathcal{R}_{pp} = \mathcal{M}_{pp} = \mathbb{Z}_q$ *and* $\mathcal{C}_{pp} = \mathbb{G}_q$ *an abelian group where the discrete log problem is hard. To enhance readability, we will prove security for* $K = 2$ *provers and one verifier, but the result trivially generalises to* $K \geq 2$ *provers. To avoid confusion between the MPC and single curator setting, we defer the simpler security proof for single curators to Appendix D. Without loss of generality, assume that the verifier* $\texttt{Vfr}^*$ *and* $\texttt{Pv}_1$ *have been corrupted by a PPT adversary* $\mathcal{A}$ *and that* $\texttt{Pv}_2$ *is honest.* Sim *receives on its input tape the inputs for* $\texttt{Pv}_1$ *and* $\texttt{Vfr}^*$. *The ideal oracle functionality* $\mathcal{M}_{Bin}$ *is defined as before. Let* Sim *denote shorthand for* $\mathsf{Sim}_{\texttt{Vfr}^*,\texttt{Pv}_1}$. *We construct the simulator as follows:*

(a) Sim *receives the public messages* $\left\{ \left\{ c_{i,k} \right\}_{k\in[K]} \right\}_{i\in[n]}$ *and sets* $c_i = \prod_{k=1}^{K} c_{i,k}$.

(b) Sim *internally invokes* $\texttt{Pv}_1$ *to receive inputs* X_1. *If* $\texttt{Pv}_1$ *was honest then* $X_1 = \sum_{i=1}^{n} [\![x_i]\!]_1$. *Of course, we have no control over* $\mathcal{A}$, *and* X_1 *could be any arbitrary value. The definition of security requires that we prove security using the actual inputs used by the real-world adversary* $\mathcal{A}$ *and not the ones it was handed to at the start of the protocol.*

(c) Sim *invokes* $\mathcal{M}_{bin}$ *with input* X_1 *and receives* (y, Δ_1) *as defined in equation (7). Note* Sim *never has access to the honest party's input* X_2 *nor the randomness* Δ_2 *used by* $\texttt{Pv}_2$ *in the real protocol. It must simulate the messages and output of the real protocol from just its input and the output it receives from the ideal model.*

(d) Sim *sets* $y_1 = X_1 + \Delta_1$ *and computes* $y_2 = y - y_1$, *which by the definition of* $\mathcal{M}_{Bin}$, *is equal to* $(X_2 + \Delta_2)$.

(e) Sim *samples* $z_2 \xleftarrow{R} \mathcal{R}_{pp}$ *and sets* $c_2 = \mathrm{Com}(y_2, z_2)$.

(f) Sim *samples* $c'_{2,2}, \ldots, c'_{n_b,2}$ *such that* $c'_{j,2} = \mathrm{Com}(1, s_{j,2})$ *where* $s_{j,2} \xleftarrow{R} \mathcal{R}_{pp}$. *It sets* $c'_{1,2} = g^1 a_2$ *where* $a_2 = c_2 \times \left(\prod_{j=2}^{n_b} \hat{c}'_{j,2} \right)^{-1} \times \left(\prod_{i=1}^{n} c_{i,2} \right)^{-1} \times g^{-1}$. *Notice that* Sim *is actually unable to open* $c'_{1,2}$ *but is never required to do so, as opening a commitment to a private value violates DP. The only information* $\mathcal{A}$ *can check is if* $c'_{1,2}$ *is a commitment to a bit, which it is. Thus the simulator artificially constructs a set of commitments that align like the real-world protocol, without having the slightest idea what the randomness used by* $\texttt{Pv}_2$ *actually was. It is able to do so due to the hiding property of the commitment scheme.*

(g) Sim *sends over* $\{c_{j,2}\}_{j\in[n_b]}$ *to* $\mathcal{A}$ *pretending to be the honest prover (Line 4 of Figure 2).*

(h) Sim *pretends to be the prover and jointly invokes* $\mathcal{O}_{Morra}$ *with* $\mathcal{A}$ *to sample* n_b *unbiased public bits* $(b_{1,2}, \ldots, b_{n_b,2})$.

(i) *Finally* Sim *sends* y_2 *and* z_2 *to* $\mathcal{A}$ *and outputs whatever* $\mathcal{A}$ *outputs.*

5 Separation Under Verifiable DP

We show that information theoretic verifiable DP is impossible in the trusted curator model. To prove our result stated in Theorem 5.2, we rely on the impossibility of secure coin flipping by [HO14].

Theorem 5.1 (Impossibility Of Tossing A Fair Coin) *[HO14] Let* $(\texttt{Pv}, \texttt{Vfr})$ *be a coin tossing protocol and let* $B_\lambda = \mathbb{E}[\texttt{out}(\texttt{Pv}, \texttt{Vfr})(1^\kappa)]$ *be the bias of the output of such a protocol. Assuming that one-way-functions do not exist, then for any* $g \in \texttt{poly}(\kappa)$, *there exists a pair of efficient cheating strategies* $\texttt{Pv}^*$ *and* $\texttt{Vfr}^*$ *such that the following holds: for infinitely many* κ*'s, for each* $j \in \{0, 1\}$ *either* $\Pr[\texttt{out}(\texttt{Pv}^*, \texttt{Vfr})(1^\kappa) = j]$ *or* $\Pr[\texttt{out}(\texttt{Pv}, \texttt{Vfr}^*)(1^\kappa) = j]$ *is greater than* $\sqrt{B^j_\kappa} - \frac{1}{g(\kappa)}$, *where* $B^1_\kappa = B_\lambda$ *and* $B^0_\kappa = 1 - B_\lambda$. *In particular for* $B_\lambda = \frac{1}{2}$, *the corrupted party can bias the outcome by almost* $\frac{1}{\sqrt{2}} - \frac{1}{2}$.

The theorem above states that it is impossible for two unbounded parties to jointly sample an unbiased public coin. The result is stronger than the impossibility result by Cleve [Cle86], which states that its impossible to jointly flip an unbiased coin if we allow parties to exit early. The theorem above states that it's impossible even if we guarantee no party exists the protocol early.

Theorem 5.2 (Information Theoretic Verifiable DP is impossible) *Any constant round interactive protocols Π for an DP-mechanism $\mathcal{M}_{Bin}$ that satisfies Verifiable-DP (Definition 7) cannot have unconditional soundness and statistical zero-knowledge.*

Proof 2 *Verifiable DP requires that a verifier be able to guarantee that the randomness generated by a prover remains unbiased, without the verifier ever seeing the randomness. Theorem 5.1, states that it is impossible for two unbounded parties to even jointly sample a public unbiased coin without assuming one way functions. Thus commitment schemes are both necessary and sufficient to jointly sample an unbiased public coin.*

The task of jointly sampling unbiased private randomness is harder. If two parties could sample unbiased private randomness, then they could just use the same protocol to sample unbiased public randomness, by revealing the randomness. Thus, commitment schemes are a necessary condition for verifiable DP. Commitments cannot be both statistically binding and hiding, thus unbounded soundness and statistical zero-knowledge is impossible.

Connection With Open Problem.

Definition 8 (α-useful mechanism) *Fix $\alpha \in [0,1]$. Let $u : \mathcal{X}^n \times \mathcal{Y} \to \in \{0,1\}$ be an efficiently computable deterministic function. A mechanism $\mathcal{M}$ is α-useful for a utility function u if for some $Q \in \mathcal{Q}$ and for all $X \in \mathcal{X}^n$*

$$\Pr_{y \leftarrow \mathcal{M}(X,Q)}[u(X,y) = 1] \geq \alpha \tag{12}$$

In his survey on the complexity of DP, Vadhan [Vad17] asks the following question. Given $X \in \mathcal{X}^n$ and a differentially private mechanism $\mathcal{M} : \mathcal{X}^n \times \mathcal{Q} \to \mathcal{Y}$, is there an efficient utility function u that is α-useful when $\mathcal{M}$ is IND-CDP but not when $\mathcal{M}$ is information-theoretically DP. Groce *et al.* [GKY11] show that if the range of u is in $\mathcal{R}^n$ and the utility is measured in terms of the $\mathcal{L}_p$-norm, then statistical-DP and computational DP are equivalent. Thus for the separation to hold, the range of u must have a more complex structure, such as a graph, a circuit or a proof. Bun *et al.* corroborate this result by describing a utility function such that u is infeasible (not impossible) when $\mathcal{M}$ is statistical DP and efficient when $\mathcal{M}$ is computational DP [BCV16]. Similar to our definition of verifiability, their utility function u is cryptographic and unnatural from a data analysis point of view. Specifically, given $y = \mathcal{M}(X, Q)$, Bun *et al.* , define the utility as the answer to the question of whether y is a valid zap proof [DN00] of the statement "there exists a row in X that is a valid message signature pair". Meanwhile, we define our utility function as an interactive proof, that checks whether the real protocol output y, is indistinguishable from the output of an ideal run of $\mathcal{M}$. In Theorem 5.2, we show that verifiable DP is impossible in the presence of computationally unbounded adversaries. This provides a candidate for a separation between statistical DP and computational DP.

However, there are some key differences between our formulation of utility and how it was originally posed. For example, in Bun *et al.* , the utility function u is a deterministic non-interactive function that receives the output y and a dataset X of message-signature pairs. The task of evaluating utility is separate from the task of computing DP statistics. In verifiable DP, both the DP statistic and utility are computed simultaneously via a constant round interactive protocol. Furthermore, the number of rounds of the utility function is a function of the privacy parameter ϵ. Another point of difference is that, in verifiable DP, the verifier performs the dual role of evaluating the utility of the mechanism and generating randomness that prevents a curator from cheating (although it does not ever see this randomness). In Bun *et al.* , the verifier's task is just to verify the proof. They are not involved in generating the DP noise. Although we show that information theoretic verifiable DP is impossible, our definitions allow the adversary more agency. Thus the two settings are not directly comparable. We defer finding stronger connections between verifiable DP and finding a utility function that separates DP as per [Vad17] to future work.

6 Performance

This section quantifies the computational cost of $\Pi_{\texttt{Bin}}$, our protocol for computing verifiable DP counting queries. In the trusted curator model, the non-verifiable protocol simply involves summing over n inputs, sampling one draw of Binomial noise and aggregating the results. Meanwhile, verifiable DP requires computing commitments for n_b private coins, sending the verifier a non-interactive OR proof (as described in Appendix C) that the messages are commitments to either zero or one, playing n_b rounds of $\Pi_{\texttt{morra}}$ and performing exponentiation operations to check if

Table 1: The table above benchmarks the latency of each of the different stages of $\Pi_{\texttt{Bin}}$ for computing single dimension counting queries with parameters $n = 10^6, \epsilon = 1.25, \delta = 2^{-10}$. Setting $\epsilon = 0.88$ results in $n_b = 262144$ private coins for DP.

Σ-proof	Σ-verification	Morra	Aggregation	Check
6609 ms	6708 ms	4987 ms	198 ms	263 ms

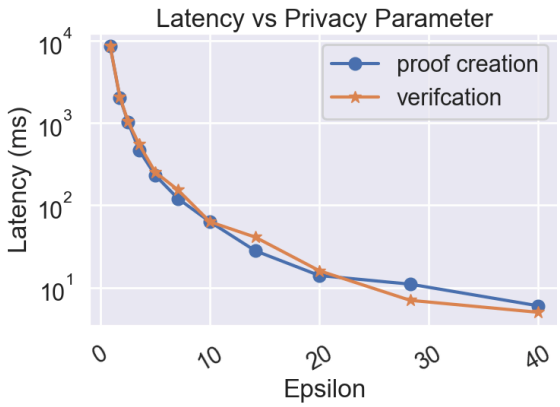

Figure 3: The figure above describes the latency of the two most expensive operations in $\Pi_{\texttt{Bin}}$ - the time it takes to prove and validate that the prover's private value is indeed a bit. The smaller the privacy parameter ϵ (high privacy), the more private coins n_b are needed to provide DP.

the prover messages align. In our experiments, we instantiate the commitment scheme using Pedersen Commitments (PC) [Ped91] and consider two choices for the prime order group $\mathbb{G}_q$ for PC. In what follows, we adopted $\mathbb{G}_q \subset \mathbb{Z}_p^*$ based on the finite field discrete log problem[8]. We also implemented Pedersen commitments over elliptic curves using the prime order Ristretto group[9], which gave slower results. A single exponentiation operation on an 8 core Apple M1 Mac took 35 μs for $\mathbb{G}_q \subset \mathbb{Z}_p^*$ and 328 μs over Curve25519. All results and plots are reproducible using code found at https://anonymous.4open.science/r/Verifiable-Differential-Privacy-0407/README.md.

Time cost for server verification. In Table 1 we describe the latency of the different phases of $\Pi_{\texttt{Bin}}$. The protocol's main computational bottleneck is verifying that the commitments to the prover's private values are indeed commitments to bits. Thus most of the time is spent creating and verifying non-interactive Σ-proofs. The aggregation column describes how long the prover takes to aggregate n elements of size κ bits, and the check column describes the time it takes the verifier to compute commitments to check that the prover's messages align with the commitments. The Morra column describes the time it takes to sample n_b public coins using $\Pi_{\texttt{morra}}$. We measure time spent doing local computations, and do not include time spent to communicate over the network. As working with the Σ-proof is our main bottleneck, Figure 3 describes how proof creation and verification latency scales with the privacy parameter ϵ. Note that for high privacy settings (small values of ϵ), the prover(s) need to generate more private coins to ensure indistinguishability. Specifically, the number of coins (n_b) is proportional to $1/\epsilon^2$ (Lemma 2.1), and the time cost is then linear in n_b.

Time cost for client verification. Clients submit secret shares of their inputs in the MPC setting. Thus the servers must verify that the client inputs are valid. For M-dimensional DP-histogram estimation, the client inputs are restricted to one-hot encoded vectors of size M. As discussed in Section 4.2, the sketching techniques used in PRIO and Poplar allow servers to verify clients with information-theoretic security but are vulnerable to attack by malicious servers. Our use of Σ-OR-protocols can defend against such attacks, but it comes at a higher computational cost due to its reliance on public key cryptography. Figure 4 benchmarks the increase in latency as a function of the number of input dimensions (M) of the secret shares. In both cases, the cost grows with the dimensionality of the input. The cost of making client verification robust to malicious servers is approximately an order of magnitude. The Σ protocol for verification can be run on each input dimension in parallel, and thus computation can be sped up using more cores. However, this increases the communication bandwidth of the protocol.

[8] Implemented using https://docs.rs/openssl/latest/openssl/bn/struct.BigNum.html
[9] https://doc.dalek.rs/curve25519_dalek/ristretto/struct.RistrettoPoint.html

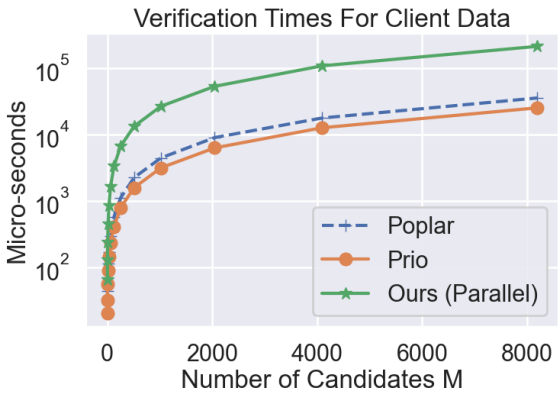

Figure 4: The figure above compares the times it takes to verify if a single client input is valid. PRIO and Poplar use lightweight sketching protocols and general-purpose MPC to check in zero knowledge whether a client's input is a one-hot vector. Instead, we use the Σ-OR proof to check if each coordinate of a client is legal.

Table 2: Summary of efforts MPC computation of aggregate DP statistics. The active security column describes if the protocols allowed participants to deviate arbitrarily. The Central DP column describes if the protocol output satisfies constant DP error independent of the number of clients participating in the protocol. The auditable property describes if the final output can be verified for correctness. Some interactive protocols leak additional information (such as prefix information about client input bits) beyond just the DP output. The leakage column describes if the prescribed protocols suffered from additional leakage.

Protocol	Active Security	Central DP	Auditable	Zero Leakage
Cryptographic RR [AJL04]	✓			✓
Verifiable Randomization Mechanism [KCY21]	✓		✓	✓
Securely Sampling Biased Coins [CSU19]		✓		✓
MPC-DP heavy hitters[BK21]		✓		✓
PRIO [CGB17]		✓		✓
Brave STAR [DSQ+21]				
Sparse Histograms [BBG+20]		✓		
Crypt-ϵ [RCWH+20]		✓		
Poplar [BBCG+22]	✓	✓		
Our work	✓	✓	✓	✓

7 Related Work

Dwork *et al.* introduced DP and described the Laplace mechanism for outputting histograms in the trusted curator model [DMNS06]. Soon after, McSherry *et al.* proposed the exponential mechanism [MT07] (equivalently, report noisy max [DKS+21]), which lets us compute the (approximately) most frequent bucket in a histogram, also under pure differential privacy. Although these mechanisms give us pure differential privacy and optimal error rates $O(\frac{1}{\epsilon})$, implementing such a "central" model requires trusting that the curator to follow the protocol and not exploit the client data that it sees in plaintext.

Therefore, researchers studied local privacy (LDP) [KLN+11] using randomised response [War65] to prevent any other party from seeing data in plaintext. Recently, Cheu, Smith and Ullman showed that the randomised response algorithm generalises all locally private protocols [CSU21]. This generalisation highlights two unavoidable disadvantages of local differential privacy. The first is that the accuracy of the protocol for even the binary histogram is $O(\sqrt{n})$ compared to $O(1)$ in the central model. The second is that randomised response systems offer a much weaker definition of privacy than the usual cryptography standards such as semantic security. For example, if the client flips their original answer with probability $p = 0.1$, the curator sees their sensitive information in plain text 90% of the time. Further increasing p reduces the accuracy of the protocol dramatically. Consider the example from [CGB17], where 1% of a million people answer "yes" to a survey about a sensitive topic. If we set $p = 0.49$, then one-third of the time the central analyser concludes that not a single member of the population answered "yes". Thus if we want

to preserve utility, this definition of security is considerably weaker than the indistinguishability guarantees provided by protocols such as secret sharing.

Shuffle privacy [Che21, BBGN19, EFM+20] analyses local mechanisms under the lens of central privacy and bridges the accuracy gap between local and central models. Recent results [GGK+19, BC20] prove that near central error guarantees are possible with distributed local transformations. Although this bypasses the accuracy issue of LDP, shuffle privacy assumes the existence of a secure shuffler, which is non-trivial to implement. In recent work, Bell *et al.* show that secure aggregation realises secure shuffling [BBG+20]. However, such protocols impose the impractical constraint of secure peer-to-peer communication between clients, and the curator is still a single source of failure. Despite the immense progress on differentially private histogram estimation, all known efficient implementations assume semi-honest participants and are a variant of either randomised response or the additive mechanism. It only takes a small fraction of clients to deviate from their prescribed protocol to destroy any utility of randomised response [CSU21]. Additive mechanisms involve adding carefully curated randomness to the statistic before being released as output.

To ensure central DP error without a trusted curator, Dwork *et al.* proposed using standard MPC for computing DP statistics [DKM+06]. They proposed that each of the K servers would own a fraction of the entire dataset used for computation. As long as not more than $\lfloor \frac{K}{3} \rfloor$ of the servers are dishonest, it is possible to compute DP-histograms with optimal accuracy. However, the protocol is not publicly auditable and breaks down in presence of a dishonest majority of adversarial corruptions. McGregor *et al.* show a separation between DP obtained using a trusted curator and that obtained using MPC [MMP+10]. Specifically, they show that there exist computations (such as inner product or hamming distance) where mechanisms with $(1, 0)$-DP incur $\Omega(\sqrt{n})$ reconstruction error compared to $O(1)$ in presence of a trusted curator. To bridge this gap, Mironov *et al.* defined computational differential privacy, a relaxation of traditional DP [MPRV09]. They show that as long as semi-honest OT exists, it is possible to compute any computationally DP function with the same error rates as information theoretic DP in a trusted curator model. Histograms, unlike inner product and hamming distance, can be computed using MPC with the same error rates as trusted curator DP, under infomation theoretic DP. Thus recent work has focused on computing histograms using MPC.

Bohler *et al.* use MPC to compute heavy hitters with semi-honest adversaries [BK21]. Researchers at Brave use oblivious pseudorandom functions (OPRF's) [JL09] and Shamir secret sharing [Sha79] to compute k-anonymous histograms in the two server setting [DSQ+21]. However, they do not include support for differential privacy. Researchers at Google use linear homomorphic encryption and OPRFs to compute differentially private sparse histograms in two-server models (2PC) [BGG+22], but require both the servers and clients to be semi-honest. Gibbs and Boneh propose PRIO, a protocol in which a small number of servers receive arithmetic shares of client input to compute differentially private histograms [CGB17]. PRIO uses shared non interactive proofs (SNIP's) to prevent clients from submitting illegal inputs but the protocol is only honest-verifier zero knowledge. Following the popularity of PRIO, Addanki *et al.* introduce PRIO+ to work over Boolean shares [AGJ+22]. Boneh *et al.* use distributed point functions (DPFs) [BGI19] to compute DP heavy-hitters in the two server model to propose a system called Poplar [BBCG+22] that is zero knowledge even in presence of active adversaries. Roy *et al.* introduce *Crypt-*ϵ, a generic system to compute differentially private statisitcs using garbled circuits and linear homomorphic encryption [RCWH+20]. The general purpose natue of *Crypt-*ϵ guarantees security only in the semi-honest threat model. Ambainis *et al.* proposed cryptographic randomised response [AJL04] but are able to only guarantee local differential privacy. Table 2 summarises the assumptions under which the latest MPC protocols that have been used to compute DP statistics. As described earlier, existing work either assumes semi-honest adversaries or is not auditable. In 2021, the State Of Alabama sued the US deparment of commerce with regard to the errors caused due to random noise [Jus21]. Differential Privacy by its defintion introduces a random noise blanket that tradesoff accuracy for privacy. This randomness is unavoidable if we wanted to protect individual privacy, but it also enables a corrupt aggregating server to disguise adversarial behaviour as randomness. In our paper, we first upgrade to security against active adversaries. Like existing literature we work in the dishonest majority model and further require the protocols to be publicly auditable. Our privacy constraints describe the most strict adversarial setting for practical deployment.

8 Concluding Remarks

We have introduced the notion of verifiable differential privacy to prevent malicious aggregators using random noise as an attack vector. We have demonstrated feasibility of this notion, and showed that computational DP is necessary to achieve verifiability. A natural open question is to provide protocols for more complex DP mechanisms. Our protocol deliberately uses simple randomness (a Binomial distribution constructed from Bernoulli random variables), as making verifiable Laplace or Gaussian noise is far from clear. Similarly, approaches based on sampling from an appropriate distribution (the exponential mechanism) may be challenging, since the distribution itself leaks information about the

private data. Another direction would be to support our approach within more expressive MPC frameworks (e.g., auditable-SPDZ [BDO14]).

References

[AGJ+22] Surya Addanki, Kevin Garbe, Eli Jaffe, Rafail Ostrovsky, and Antigoni Polychroniadou. Prio+: Privacy preserving aggregate statistics via boolean shares. In *International Conference on Security and Cryptography for Networks*, pages 516–539. Springer, 2022.

[AJL04] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic randomized response techniques. In *International Workshop on Public Key Cryptography*, pages 425–438. Springer, 2004.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[BBCG+22] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight Techniques for Private Heavy Hitters. *arXiv:2012.14884 [cs]*, January 2022. arXiv: 2012.14884.

[BBG+20] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[BBGN19] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, pages 638–667. Springer, 2019.

[BC20] Victor Balcer and Albert Cheu. Separating Local & Shuffled Differential Privacy via Histograms. *arXiv:1911.06879 [cs]*, April 2020. arXiv: 1911.06879.

[BCV16] Mark Bun, Yi-Hsiu Chen, and Salil Vadhan. Separating computational and statistical differential privacy in the client-server model. In *Theory of Cryptography Conference*, pages 607–634. Springer, 2016.

[BDO14] 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.

[BGG+22] James Bell, Adria Gascon, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. Distributed, private, sparse histograms in the two-server model. *Cryptology ePrint Archive*, 2022.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function Secret Sharing: Improvements and Extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1292–1303, Vienna Austria, October 2016. ACM.

[BGI19] Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2019.

[BK07] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, 9(2):75–79, 2007.

[BK21] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private heavy hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2361–2377, 2021.

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[BMRS21] Carsten Baum, Alex J Malozemoff, Marc B Rosen, and Peter Scholl. Mac'n'cheese : Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *Annual International Cryptology Conference*, pages 92–122. Springer, 2021.

[BOGW19] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 351–371. 2019.

[BS22] Danah Boyd and Jayshree Sarathy. Differential perspectives: Epistemic disconnects surrounding the us census bureaus use of differential privacy. *Harvard Data Science Review (Forthcoming)*, 2022.

[CDS94] 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.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, Robust, and Scalable Computation of Aggregate Statistics. *arXiv:1703.06255 [cs]*, March 2017. arXiv: 1703.06255.

[Che21] Albert Cheu. Differential privacy in the shuffle model: A survey of separations. *arXiv preprint arXiv:2107.11839*, 2021.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369, 1986.

[CSU19] Jeffrey Champion, Abhi Shelat, and Jonathan Ullman. Securely sampling biased coins with applications to differential privacy. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–614, 2019.

[CSU21] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 883–900. IEEE, 2021.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 418–430. Springer, 2000.

[DIO20] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. *Cryptology ePrint Archive*, 2020.

[DKM+06] 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.

[DKS+21] Zeyu Ding, Daniel Kifer, Thomas Steinke, Yuxin Wang, Yingtai Xiao, Danfeng Zhang, et al. The permute-and-flip mechanism is identical to report-noisy-max with exponential noise. *arXiv preprint arXiv:2105.07260*, 2021.

[DMNS06] 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.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 283–293. IEEE, 2000.

[DSQ+21] Alex Davidson, Peter Snyder, EB Quirk, Joseph Genereux, and Benjamin Livshits. Star: Distributed secret sharing for private threshold aggregation reporting. *arXiv preprint arXiv:2109.10074*, 2021.

[EFM+20] lfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. *arXiv:1811.12469 [cs, stat]*, July 2020. arXiv: 1811.12469.

[GAM19] Simson Garfinkel, John M Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Communications of the ACM*, 62(3):46–53, 2019.

[GGK+19] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages. *arXiv preprint arXiv:1908.11358*, 2019.

[GGK+20] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the Power of Multiple Anonymous Messages. *arXiv:1908.11358 [cs, stat]*, May 2020. arXiv: 1908.11358.

[GKY11] Adam Groce, Jonathan Katz, and Arkady Yerukhimovich. Limits of computational differential privacy in the client/server setting. In *Theory of Cryptography Conference*, pages 417–431. Springer, 2011.

[Gol07] Oded Goldreich. *Foundations of cryptography. Vol. 1: Basic tools*, volume 1. Cambridge Univ. Press, Cambridge, digitally print. 1. paperback version edition, 2007.

[HO14] Iftach Haitner and Eran Omri. Coin flipping with constant bias implies one-way functions. *SIAM Journal on Computing*, 43(2):389–409, 2014.

[JL09] Stanisław Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography Conference*, pages 577–594. Springer, 2009.

[Jus21] Brennan Center For Justice. Alabama v. u.s. dept of commerce, 2021.

[KCY21] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 43–60. Springer, 2021.

[KLN+11] 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.

[KRS13] Shiva Prasad Kasiviswanathan, Mark Rudelson, and Adam Smith. The power of linear reconstruction attacks. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1415–1433. SIAM, 2013.

[Lin17] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography*, pages 277–346, 2017.

[Mau09] Ueli Maurer. Unifying zero-knowledge proofs of knowledge. In *International Conference on Cryptology in Africa*, pages 272–286. Springer, 2009.

[MMP+10] Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. The limits of two-party differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 81–90. IEEE, 2010.

[MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Annual International Cryptology Conference*, pages 126–142. Springer, 2009.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[RCWH+20] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypt: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 603–619, 2020.

[RHML21] Varun Raturi, Jinhyun Hong, David Philip McArthur, and Mark Livingston. The impact of privacy protection measures on the utility of crowdsourced cycling data. *Journal of Transport Geography*, 92:103020, 2021.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Tha20] Justin Thaler. Proofs, arguments, and zero-knowledge, 2020.

[Vad17] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1074–1091. IEEE, 2021.

A Formal Security Definitions

Definition 9 (Discrete Log Assumption) *For all PPT adversaries $\mathcal{A}$, there exists a negligible function μ such that*

$$\Pr\left[x = x' : \begin{array}{c}(\mathbb{G}_q, g) \leftarrow \mathtt{Setup}(1^\kappa) \\ x \xleftarrow{R} \mathbb{Z}_q, h = g^x \\ x' \leftarrow \mathcal{A}(\mathtt{pp}, h)\end{array}\right] \leq \mu(\kappa)$$

Definition 10 *(Hiding Commitments) Let κ be the security parameter. A commitment scheme is said to be hiding for all PPT adversaries $\mathcal{A}$ the following quantity is negligible. The commitment is perfectly hiding if $\mu(\kappa) = 0$.*

$$\Pr\left[b = b' : \begin{array}{c}\mathtt{pp} \leftarrow \mathtt{Setup}(1^\kappa) \\ b \xleftarrow{R} \{0,1\}, r_{x_b} \xleftarrow{R} R_{\mathtt{pp}} \\ (x_0, x_1) \in \mathcal{M}^2_{\mathtt{pp}} \leftarrow \mathcal{A}(\mathtt{pp}) \\ c = \mathrm{Com}(x_b, r_{x_b}), b' = \mathcal{A}(\mathtt{pp}, c)\end{array}\right] \leq \mu(\kappa)$$

Definition 11 *(Binding Commitments) Let κ be the security parameter. A commitment scheme is said to be binding if, for all PPT adversaries $\mathcal{A}$, there exists a negligible function μ such that*

$$\Pr\left[(c_{x_0} = c_{x_1}) \wedge (x_0 \neq x_1) : \begin{array}{c}\mathtt{pp} \leftarrow \mathtt{Setup}(1^\kappa) \\ x_0, r_{x_0}, x_1, r_{x_1} \leftarrow \mathcal{A}(\mathtt{pp})\end{array}\right] \leq \mu(\kappa)$$

The commitment is perfectly binding if $\mu(\kappa) = 0$.

B The Binomial mechanism

In this Appendix, we spell out the details of the differential privacy properties of Binomial noise addition (the Binomial mechanism). The results here were originally shown by Ghazi *et al.* [GGK+20], and we include them here for completeness (we do not claim any novelty in this section).

Definition 12 *[GGK+20] A function $q : \mathcal{X}^n \to \mathbb{Z}^M$ is said to be k-incremental if for all neighbouring datasets $X \sim X'$, $||f(X) - f(X')||_\infty \leq k$.*

It is easy to see that counting queries for histogram estimation are 1-incremental. The following definition describes valid noise distributions to ensure differential privacy.

Definition 13 *[GGK+20] A distribution D over $\mathbb{Z}$ is (ϵ, δ, k)-smooth if for all $k' \in [-k, k]$ we have*

$$\Pr_{Y \sim D}\left[\frac{\Pr_{Y' \sim D}[Y' = Y]}{\Pr_{Y' \sim D}[Y' = Y + k']} \geq e^{|k'|\epsilon}\right] \leq \delta \tag{13}$$

The result for the Binomial mechanism follows by showing that adding noise drawn from a smooth distribution ensures differential privacy and then showing that the Binomial distribution meets the smoothness definition.

Lemma B.1 (Lemma 4.11 in Appendix C of [GGK+20]) *Suppose $q : \mathcal{X}^n \to \mathbb{Z}^M$ is k-incremental i.e., for all neighboring datasets $X \sim X'$ we have $||q(X) - q(X')||_\infty = k$ and $\Delta(q) = ||q(X) - q(X')||_1 = \Delta$. Let $\mathcal{D}$ be a (ϵ, δ, k)-smooth distribution. Then the mechanism M*

$$M_{(Y_1,\ldots,Y_M)}(X, q) = q(X) + (Y_1, \ldots, Y_M) \tag{14}$$

is $(\epsilon\Delta, \delta\Delta)$ differentially private, where $(Y_1, \ldots, Y_M) \overset{i.i.d}{\sim} D$.

Proof 3 *Let $X = (x_1, \ldots, x_n)$ where $x_i \in \mathcal{X}$ and $X' = (x_1, \ldots, x'_n)$. Let $\vec{y} = (y_1, \ldots, y_M)$ and $\vec{Y} = (Y_1, \ldots, Y_M)$ be i.i.d draws from $\mathcal{D}$. Assume that Equation (15) holds:*

$$\Pr_{\vec{y} \sim \mathcal{D}}[g(\vec{y}) \geq e^{\epsilon'}] \leq \delta' \tag{15}$$

where $g(\vec{y}) = \frac{\Pr_{(\vec{Y} \sim \mathcal{D})}[M_{\vec{Y}}(X,q) = q(X) + \vec{y}]}{\Pr_{(\vec{Y} \sim \mathcal{D})}[M_{\vec{Y}}(X',q) = q(X') + \vec{y}]}$.

Let $S \subseteq \mathbb{Z}^M$ be an arbitrary subset in the range of M. Let $T = \{M_{\vec{Y}}(X,q) | g(\vec{y}) < e^{\epsilon'}\}$ represent a set of outputs of M over draws of $\vec{Y}$ such that $g(\vec{y}) < e^{\epsilon'}$. Then from equation (15) we can show that M is (ϵ', δ') differentially private

$$\Pr_{\vec{y}\sim\mathcal{D}}[M_{\vec{y}}(X,q) \in S] \leq \delta' + \Pr_{\vec{y}\sim\mathcal{D}}[M_{\vec{y}}(X,q) \in S \cap T] \tag{16}$$

$$= \delta' + \sum_{w \in S\cap T} \Pr_{\vec{y}\sim\mathcal{D}}[M_{\vec{y}}(X,q) = w] \tag{17}$$

$$\leq \delta' + \sum_{w \in S\cap T} e^{\epsilon'} \Pr_{\vec{y}\sim\mathcal{D}}[M_{\vec{y}}(X',q) = w] \tag{18}$$

$$\leq \delta' + \sum_{w \in S} e^{\epsilon'} \Pr_{\vec{y}\sim\mathcal{D}}[M_{\vec{y}}(X',q) = w] \tag{19}$$

$$= \delta' + e^{\epsilon'} \Pr_{\vec{y}\sim\mathcal{D}}[M_{\vec{y}}(X',q) \in S] \tag{20}$$

Equation (16) is from the law of total probability, equation (18) comes from equation (15) assumption and equation (19) is true as $T \cap S \subseteq S$. Therefore all that remains is to show that equation (15) is true if $\mathcal{D}$ is as defined and $\epsilon' = \epsilon\Delta, \delta' = \delta\Delta$ to complete the proof.

Define $k_j = q(X)_j - q(X')_j$. As each coordinate of $q(X)$ is independently perturbed and q is a deterministic function, equation (15) is equivalent to equation (21).

$$\Pr_{\vec{y}\sim\mathcal{D}}\left[\prod_{j=1}^{M} \frac{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j]}{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j + k_j]} \geq e^{\epsilon'}\right] \leq \delta' \tag{21}$$

Thus, in order to prove equation (15) is true, it suffices to show that equation (21) holds. We know that $\mathcal{D}$ is a smooth distribution i.e for each $j \in [M]$

$$\Pr_{\vec{y}\sim\mathcal{D}}\left[\frac{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j]}{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j + k_j]} \geq e^{|k_j|\epsilon}\right] \leq \delta \tag{22}$$

We can apply the union bound to get the probability of the joint distribution over all indices.

$$\Pr_{\vec{y}\sim\mathcal{D}}\left[\prod_{j=1}^{M} \frac{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j]}{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j + k_j]} \geq e^{\sum_{j=1}^{m}|k_j|\epsilon}\right] \tag{23}$$

$$\leq \delta \sum_{j=1}^{M} \mathbb{I}(k_j \neq 0) \tag{24}$$

Given the sensitivity of q is Δ, at most Δ indices for k_j can be non-zero and $(\sum_{j=1}^{M}|k_j|) \leq \Delta$. Finally we get the result we seek

$$\Pr_{\vec{y}\sim\mathcal{D}}\left[\prod_{j=1}^{M} \frac{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j]}{\Pr_{(Y_j\sim\mathcal{D})}[Y_j = y_j + k_j]} \geq e^{\Delta|k_j|\epsilon}\right] \leq \delta\Delta \tag{25}$$

Lemma B.2 (Based on Lemma 4.12 of Appendix C in [GGK+20]) *Let $n \in \mathbb{N}$, $p \in [0, 1/2]$, $\alpha \in [0, 1)$ and $k \leq \frac{n\alpha p}{2}$. Then the binomial distribution $\mathrm{Bin}(n,p)$ is a (ϵ, δ, k)-smooth distribution.*

Proof 4 *Let $Y \sim \mathrm{Bin}(n,p)$, then $\Pr[Y = y] = \binom{n}{y}p^y(1-p)^{n-y}$. For any $-k \leq k' \leq k$, define an interval $\epsilon := [(1-\alpha)np + k', (1+\alpha)np - k']$. This an interval of size k around the mean of the distribution. Note that as long as $k \leq \frac{np}{2}\alpha$, then the interval $\epsilon' := [(1-\alpha/2)np, (1+\alpha/2)np]$ is contained inside of ϵ. Thus if $y \sim \mathrm{Bin}(np)$ is not in ϵ, it is also not inside ϵ'. We know how to bound the probability that $y \notin \epsilon'$ by using the multiplicative Chernoff bound. Invoking it, we get*

$$\Pr_{y\sim\mathrm{Bin}(n,p)}[y \notin \epsilon] \leq \Pr_{y\sim\mathrm{Bin}(n,p)}[y \notin \epsilon']$$

$$\leq e^{-\frac{-\alpha^2 np}{8}} + e^{-\frac{-\alpha^2 np}{8+2\alpha}}$$

$$= \delta$$

Now for all $y \in \epsilon$, we have for $0 \leq k' \leq k$

$$\frac{\Pr[Y=y]}{\Pr[Y=y+k']} = \Big(\frac{1-p}{p}\Big)^{k'} \prod_{i=1}^{k'} \frac{y+i}{n-y-i+1} \tag{26}$$

$$\leq \Big(\frac{1-p}{p}\Big)^{k'} \Big(\frac{y+k'}{n-y-k'}\Big)^{k'} \tag{27}$$

$$\leq \Big(\frac{1-p}{p}\Big)^{k'} \Big(\frac{(1+\alpha)np}{n-(1+\alpha)np}\Big)^{k'} \tag{28}$$

$$= (1+\alpha)^{k'} \Big(\frac{1-p}{1-p-p\alpha}\Big)^{k'} \tag{29}$$

(28) *comes from our assumption $y \in \epsilon$ and so when $y = (1+\alpha)np - k'$ the ratio above is maximal.*

Similarly for $-k \leq k' \leq 0$ we have

$$\frac{\Pr[Y=y]}{\Pr[Y=y+|k'|]} = \Big(\frac{p}{1-p}\Big)^{|k'|} \prod_{i=1}^{|k'|} \frac{n-y+i}{y-i+1} \tag{30}$$

$$\leq \Big(\frac{p}{1-p}\Big)^{|k'|} \Big(\frac{n-y+|k'|}{y-|k'|}\Big)^{|k'|} \tag{31}$$

$$\leq \Big(\frac{1+p\alpha-p}{(1-\alpha)(1-p)}\Big)^{|k'|} \tag{32}$$

$$\leq \Big(\frac{1+\alpha}{1-\alpha}\Big)^{|k'|} \tag{33}$$

$$= e^{|k|\epsilon} \tag{34}$$

(32) *comes from plugging in the smallest value for y and* (33) *comes from the fact that $p \leq 1/2$. It is easy to see that when $p \leq \frac{1}{2}$, that we have*

$$\Big(\frac{1-p}{1-p-p\alpha}\Big)^{k'} \leq \Big(\frac{1}{1-\alpha}\Big)^{|k'|} \tag{35}$$

Therefore we can upper bound equation (29) *by setting*

$$(1+\alpha)^{k'} \Big(\frac{1-p}{1-p-p\alpha}\Big)^{k'} \leq e^{|k|\epsilon} \tag{36}$$

Finally, we can prove smoothness using Bayes' rule. Let $g(y) = \frac{\Pr_{Y\sim \mathrm{Bin}(n,p)}[Y=y]}{\Pr_{Y\sim \mathrm{Bin}(n,p)}[Y=y+k']}$ and $\mathcal{D} = \mathrm{Bin}(n,p)$. Then:

$$\begin{aligned} \Pr_{y\sim\mathcal{D}}\Big[g(y) \geq e^{|k'|\epsilon}\Big] &\leq \Pr_{y\sim\mathcal{D}}\Big[g(y) \geq e^{|k'|\epsilon} \Big| y \notin \epsilon\Big] + \delta \\ &\leq e^{-\frac{-\alpha^2 np}{8}} + e^{-\frac{-\alpha^2 np}{8+2\alpha}} \\ &\leq \delta \end{aligned} \tag{37}$$

(37) *comes from the fact that when $y \in \epsilon$ we have $\frac{\Pr[Y=y]}{\Pr[Y=y+k']} \leq e^{\epsilon|k'|}$ by how we defined ϵ in equation* (33)*. So we have*

$$\Pr_{y\sim\mathrm{Bin}(n,p)}\left[\frac{\Pr_{Y\sim\mathrm{Bin}(n,p)}[Y=y]}{\Pr_{Y\sim\mathrm{Bin}(n,p)}[Y=y+k']} \geq e^{|k'|\epsilon} \Big| y \notin \epsilon\right] = 0$$

Putting all this together, we can now prove Lemma 2.1:

Proof 5 (Proof of Lemma 2.1) *From the definition of k-incremental functions in Definition 12 of Appendix B, it is easy to see that the counting query is 1-incremental and has global sensitivity $\Delta = 1$. So, by Lemma B.1 in Appendix B*

if we add noise $Z \sim \mathcal{D}$ where $\mathcal{D}$ is $(\epsilon, \delta, 1)$-smooth we get our result. Thus if we could show that $\mathrm{Bin}(n_b, \frac{1}{2})$ *is* $(\epsilon, \delta, 1)$*-smooth we would be done.*

From Lemma B.2 in Appendix B we have that for $\alpha = \frac{e^\epsilon - 1}{e^\epsilon + 1}$ *and* $k \leq \frac{n_b \alpha p}{2}$*, where* $p \leq \frac{1}{2}$*, the binomial distribution* $\mathrm{Bin}(n_b, p)$ *is* (ϵ, δ, k)*-smooth, where* $\delta \geq e^{-\frac{-\alpha^2 n_b p}{8}} + e^{-\frac{-\alpha^2 n_b p}{8+2\alpha}}$*. Observe that by the definition of* α*, for all* $\epsilon \in [0, 1]$*,* $\alpha \geq \frac{\epsilon}{\sqrt{5}}$*. Setting* $p = \frac{1}{2}$*, we could write*

$$n_b p \alpha \geq n_b \frac{\epsilon}{2\sqrt{5}} \tag{38}$$

All we need is an adequate value for $\epsilon \in [0, 1]$ *by which we get* $n_b p \alpha \geq 2$*, which would prove that* $\mathrm{Bin}(n_b, \frac{1}{2})$ *is 1-smooth. Notice that*

$$e^{-\frac{-\alpha^2 n_b p}{8}} + e^{-\frac{-\alpha^2 n_b p}{8+2\alpha}} \leq 2e^{-\frac{-\alpha^2 n_b p}{10}} \leq 2e^{-\frac{-\epsilon^2 n_b p}{50}} = \delta \tag{39}$$

Re-arranging the terms and setting $p = \frac{1}{2}$ *we get* $\epsilon = 10\sqrt{\frac{1}{n_b} \ln \frac{2}{\delta}}$*. Plugging it back into equation (38) we get*

$$\begin{aligned} n_b p \frac{\epsilon}{\sqrt{5}} = \frac{\epsilon}{2\sqrt{5}} = n_b \sqrt{\frac{1}{5n_b} \ln \frac{2}{\delta}} &\geq \sqrt{\frac{n_b}{5} \ln 2} \\ &\geq 2 \textit{ for } n_b > 30 \end{aligned}$$

Therefore we have shown that $\mathrm{Bin}(n_b, \frac{1}{2})$ *is* $(\epsilon, \delta, 1)$*-smooth*

	Verifier		**Prover**
1 :		Common Input $g, h, \mathbb{G}_q, q, c$	
2 :			$v_1, e_1 \xleftarrow{R} \mathbb{Z}_q^2$
3 :			Set d_1 such that $d_1 \left(\frac{c}{g}\right)^{e_1} = h^{v_1}$
4 :			$b \xleftarrow{R} \mathbb{Z}_q$ and set $d_0 = h^b$
5 :	(d_0, d_1)	$\xleftarrow{(d_0, d_1)}$	(d_0, d_1)
6 :	$e \xleftarrow{R} \mathbb{Z}_q$	$\xrightarrow{e}$	$e_0 = e - e_1 \mod q$
7 :			$v_0 = b + e_0 r$
8 :	Check $e_1 + e_0 = e$	$\xleftarrow{(v_0, e_0, v_1, e_1)}$	
9 :	Check $d_0 c^{e_0} = h^{v_0}$ and $d_1 c^{e_1} = g^{e_1} h^{v_1}$		

Figure 5: Proof for convincing `Vfr` that $c = gh^r$ is in $L_{\texttt{Bit}}$ without revealing that $x = 1$.

C OR Protocol

Define as public parameters a cyclic prime order group $\mathbb{G}_q$ and generators g and h for $\mathbb{G}_q$. Let $\mathcal{M}_{\texttt{pp}} = \mathcal{R}_{\texttt{pp}} = \mathbb{Z}_q$. Pedersen Commitments defined below satisfy all the properties described in Section 2.2.

$$\mathrm{Com}(x, r_x) = g^x h^{r_x} \tag{40}$$

For the sake of completeness, we describe the interactive disjunctive OR proof using Σ-protocols from [CDS94]. In all implementations in this paper, we use the Fiat-Shamir transform, which makes protocols non-interactive and can be shown to be secure in the random oracle model as described in [Tha20]. Note we choose the non-interactive version for efficiency reasons but the Σ protocols are zero-knowledge even without a random oracle. Maurer [Mau09] shows that if the verifier's challenge space is polynomial sized then the protocol can be shown to be zero-knowledge. This does affect the soundness of the protocol but it can be made negligible by a constant number of repetitions. Damgard *et al.* show that by using Trapdoor commitments [Dam00], one can preserve soundness and get zero-knowledge but

	Verifier		Prover
1 :		Common Input $g, h, \mathbb{G}_q, q, c$	
2 :			$v_0, e_0 \xleftarrow{R} \mathbb{Z}_q^2$
3 :			Set d_0 such that $d_0 c^{e_0} = h^{v_0}$
4 :			$b \xleftarrow{R} \mathbb{Z}_q$ and set $d_1 = h^b$
5 :	(d_0, d_1)	$\xleftarrow{(d_0, d_1)}$	(d_0, d_1)
6 :	$e \xleftarrow{R} \mathbb{Z}_q$	$\xrightarrow{e}$	$e_1 = e - e_0 \mod q$
7 :			$v_1 = b + e_1 r$
8 :	Check $e_1 + e_0 = e$	$\xleftarrow{(v_0, e_0, v_1, e_1)}$	
9 :	Check $d_0 c^{e_0} = h^{v_0}$ and $d_1 c^{e_1} = g^{e_1} h^{v_1}$		

Figure 6: Proof for convincing $\texttt{Vfr}$ that $c_x = h^{r_x}$ is in $L_{\texttt{Bit}}$ without revealing that $x = 0$.

the protocol now has 4 messaging rounds instead of 3. Next we describe the Σ-protocol that can used to verify the OR condition.

Let $x \in \{0, 1\}$ and $c_x = \text{Com}(x, r_x)$ for $r_x \xleftarrow{R} \mathbb{Z}_q$ be the commitment to x. Given c_x, $\Pi_{\texttt{OR}}$ is an interactive zero-knowledge proof between a prover $\texttt{Pv}$ and a verifier $\texttt{Vfr}$ to show that $c_x \in L_{\texttt{Bit}}$. The security properties can be found in [Tha20, Dam00, CDS94].

$$L_{\texttt{Bit}} = \{c_x : x \in \{0, 1\} \wedge c_x = \text{Com}(x, r_x)\} \tag{41}$$

where for some $r_x \in \mathbb{Z}_q$. Based on the value for x, the prover uses the protocol described in Figure 5 or Figure 6. The verifier cannot distinguish between the protocol's two runs as the messages are indistinguishable. In case the the inputs $\vec{x}$ are bit strings of size M (like in PRIO and Poplar) and only one coordinate can be non-negative, the prover sends to the verifier $r = \sum_{j=1}^{M} r_{x_j}$ along with the Σ-proofs, where r_{x_j} is the randomness used to create commitments for the coordinate $x_j \in \{0, 1\}$. As $\vec{x} \in L$, implies $\vec{x} \in \{0, 1\}^M$ and $||\vec{x}||_1 = 1$, the OR proofs verify the first criterion and the second criterion is easily verified by checking $c_{||\vec{x}||} = \prod_{j=1}^{M} c_{x_m}$ is a a commitment to one i.e., check if $g^1 h^r = c$.

D Deferred Security Proofs

Single Curator Simulator Proof.

Theorem D.1 *Let $\texttt{Vfr}^*$ denote the corrupted verifier. There exists a PPT Simulator $\mathsf{Sim}_{(\texttt{Vfr}^*)}$ such that for all $y = \mathcal{M}_{Bin}(X, Q)$*

$$\mathit{View}[\Pi(\texttt{Pv}, \texttt{Vfr})] \overset{c}{\equiv} \mathsf{Sim}_{(\texttt{Vfr}^*}(y, \vec{r}_v, z, \mathit{pp})$$

where $z \in \{0, 1\}^{poly(\kappa)}$ and $\vec{r}_v \in \{0, 1\}^{poly(\kappa)}$ represents auxiliary input and randomness available to all the corrupted parties.

Proof 6 *Denote the corrupted verifier as $\texttt{Vfr}^*$. Sim receives on its input tape the inputs for $\texttt{Vfr}^*$. The ideal oracle functionality $\mathcal{M}_{Bin}$ is defined as before. Let Sim denote shorthand for $\mathsf{Sim}_{\texttt{Vfr}^*}$. We construct the simulator as follows:*

1. Sim *receives the public messages $\{c_i\}_{i \in [n]}$.*
2. Sim *invokes $\mathcal{M}_{bin}$ with the empty string λ and receives y as defined in equation (7).*
3. Sim *samples $z \xleftarrow{R} \mathcal{R}_{pp}$ and sets $c = \text{Com}(y, z)$.*
4. Sim *samples $c'_2, \ldots, c'_{n_b}$ such that $c'_j = \text{Com}(1, s_j)$ where $s_j \xleftarrow{R} \mathcal{R}_{pp}$. It sets $c'_1 = g^1 a$ where $a = c \times \left(\prod_{j=2}^{n_b} \hat{c}'_j\right)^{-1} \times \left(\prod_{i=1}^{n} c_i\right)^{-1} \times g^{-1}$.*

5. Sim *sends over* $\{c_j\}_{j\in[n_b]}$ *to* $\mathcal{A}$ *pretending to be the honest prover (Line 4 of Figure 2).*
6. Sim *pretends to be the prover and jointly invokes* $\mathcal{O}_{\texttt{Morra}}$ *with* $\mathcal{A}$ *to sample* n_b *unbiased public bits* $(b_1, \ldots, b_{n_b})$.
7. Sim *sends* y *and* z *to* $\mathcal{A}$ *and outputs whatever* $\mathcal{A}$ *outputs.*