Interactive Proofs For Differential Privacy*

Ari Biswas
University Of Warwick
aribiswas3@gmail.com

Graham Cormode
Meta & University Of Warwick
gcormode@meta.com

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 tasked 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 *Interactive Proofs For 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 Motivation

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. It is most commonly studied in the *single curator* model, where a single entity receives all the sensitive data and is entrusted to output DP statistics. Variations that modify the trust and computational model include local privacy [War65], shuffle privacy [BBGN19, CSU19], computational differential privacy [MPRV09], multi-party (MPC) differential privacy [MMP+10] and client-server MPC-DP [CGB17]. Regardless of the computational and trust model, a consistent theme across all existing work is to view DP simply as a privacy-preserving mechanism. In this paper, we flip the script and ask the following question:

> What if the entity responsible for releasing aggregate DP statistics seeks to abuse the protocol, distort the output and use the randomness required by differential privacy as an alibi to hide its nefarious behaviour?

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 artefacts of random noise. Consider a counting query protocol to determine the DP 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 voter inputs; or they might manipulate the published output to bias the results of the election (say, to pineapple) and blame any discrepancies in the result on random noise introduced by DP. Given that differential privacy is used to compute approximations over sensitive information, it is likely that the output is also used to make socially important decisions and hence must be reliable. Note that in order to preserve the privacy of inputs, some loss in the accuracy of the output is unavoidable. By definition, DP requires the output to be perturbed by *private* randomness. This implies that the curator cannot just publish the randomness used to compute statistics as proof that it behaved honestly, as that would defeat the purpose of introducing randomness in the first place. Outputting approximate statistics already creates tensions between the 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, verifying that any utility loss can be attributed solely to unavoidable DP randomness is critical for the wider societal acceptance and adoption of DP.

*Some of the results discussed in this extended abstract will appear at The ACM Conference on Computer and Communications Security (CCS), 2023 under the title *Interactive Proofs For Differentially Private Counting.*

To this end, we formally introduce the idea of *Interactive Proofs For Differential Privacy* (IPDP) in both the single curator[2] setting and the client-server multi-party[3] setting [BDO14] in the presence of active adversaries[4]. Our contributions are as follows:

1. We formally introduce definitions for *Interactive Proofs For Differential Privacy* in both the trusted curator and client-server multiparty setting. Informally, the entity responsible for releasing DP statistics must also output a zero knowledge proof to verify that the output distribution was constructed correctly and the private randomness was generated faithfully. Such a proof reveals no additional information[5] but ensures that the curator cannot use DP randomness maliciously.
2. We show concrete instantiations of interactive proofs of DP by computing DP counting queries (histograms) in the trusted curator and client-server multiparty settings. 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 along with 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 report experiments that show that our protocols with formal theoretical guarantees are also practical. Additionally, we describe how our protocol Π, for verifiable DP counting, can be combined with existing (non-verifiable) DP-MPC protocols, such as PRIO [CGB17] and Poplar [BBCG+21], to enforce verifiability. The code for the experiments can be found at `https://github.com/abiswas3/Verifiable-Differential-Privacy`.
4. We demonstrate that interactive proofs for *information-theoretic* DP is impossible. Specifically, if the prover or verifier is computationally unbounded, either simulability[6] (DP guarantee) or unconditional soundness (the output distribution is not verifiable) fails to hold. This result provides a potential answer to Open Problem 10.6 from [Vad17, Chapter 10], 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?"*

2 Interactive Proofs For Differential Privacy (IPDP)

This section informally discusses the definitions for interactive proofs for DP. A formal description of all definitions can be found in Appendix A. Although we discuss the single curator model here for simplicity, the definitions apply to both the single curator and client-server MPC setting. In both settings, we will assume that the inputs come from $n \in \mathbb{N}$ distinct clients. Let $\kappa \in \mathbb{N}$ denote the security parameter. Given a description of (possibly uncountable) sets $\mathcal{X}$ and $\mathcal{Y}$, our goal is to compute some function $f : \mathcal{X}^n \to \mathcal{Y}$, which is computable efficiently by a PPT Turing Machine over inputs from n distinct input sources (clients), in a differentially private manner. In the single curator model, a single aggregating server receives client inputs $X \in \mathcal{X}^n$ in plaintext. It is responsible for constructing a distribution $\mathtt{M}(X, f)$, where $\mathtt{M}$ is a public[7] (ε, δ)-DP mechanism for computing f. The server never publishes the distribution $\mathtt{M}(X, f)$ in the clear but instead publishes at most $q(\kappa)$ samples from the distribution as the output of $\mathtt{M}$. We denote with $y \xleftarrow{\$} \mathtt{M}(X, f)$ the output of draws from $\mathtt{M}(X, f)$. But of course, the curator might be corrupted and could output samples from any distribution $\mathcal{D}$ arbitrarily far[8] from $\mathtt{M}(X, f)$. In what follows, we use the terms P (prover), server and curator interchangeably, and likewise use the terms analyst and V (verifier) to refer to the same entity. Unless otherwise specified, it is safe to assume that the prover P and verifier V are Turing Machines equipped with a *private* random and advice/auxiliary tape. In what follows, we use the terms interactive proofs and verifiable DP protocols interchangeably.

Our aim is to define and construct a *verifiable* DP mechanism in which a prover convinces a verifier that the output it received was sampled from $\mathtt{M}(X, f)$ without revealing any information about the distribution itself.

[2]When we say single curator, we imply that there is a single server that can view client inputs in plaintext. However, this server could still be corrupted, so it must prove that the final released output was computed as prescribed by the DP protocol. Of course, we cannot protect client privacy in the single server setting. The focus is on ensuring the output is reliable

[3]Here, the clients first secret share their inputs before sending one share to each of the $K \geq 2$ curators. Thus as long as a single curator follows the protocol honestly, the client inputs are information theoretically secure.

[4]By active adversaries, we mean participants that may deviate from protocol specifications arbitrarily. In the MPC setting, we can guarantee both privacy of client inputs and that the output is reliable. In the single curator setting, active adversaries are equivalent to malicious verifier zero knowledge.

[5]Apart from what can be inferred by samples from the output DP distribution.

[6]See definition for Interactive Proofs For DP in Section A.3 for a definition of simulability.

[7]All involved parties know a description of $\mathtt{M}$.

[8]Throughout this document, we measure distances between distributions using total variation distance, denoted as $d_{\mathtt{TV}}(\cdot, \cdot)$.

A verifiable DP mechanism for $\mathtt{M}$ consists of a PPT algorithm $\mathtt{Setup}$ and an interactive proof system Π between two "next-message-computing-turing machines" P and V. A next-message computing-Turing machine computes, sends, and receives messages from the other party over a sequence of alternating rounds. Any message, for example, V's message m_i at round i is determined by its input, messages it has received so far from the other party and its internal random tape $r_{\mathsf{V}} \in \{0,1\}^*$.

$\mathtt{pp} \xleftarrow{\$} \mathtt{Setup}(1^\kappa)$ describes the PPT algorithm that receives a unary representation of the security parameter κ as input and generates public parameters $\mathtt{pp}$ randomly, that is broadcasted to all parties.

Π describes an interactive proof system for differential privacy between a prover (P) and a verifier (V). P receives as inputs (i) public parameters $\mathtt{pp}$, (ii) client inputs $X \in \mathcal{X}^n$ and (iii) random coins $r_{\mathsf{P}} \in \{0,1\}^*$ (on its random tape). The verifier receives inputs (i) public parameters $\mathtt{pp}$, (ii) auxiliary input $z \in \{0,1\}^{\mathtt{poly}(\kappa)}$ and (iii) random coins $r_{\mathsf{V}} \in \{0,1\}^*$. Our results still hold if the verifier also receives a neighbouring dataset X' (differs from X by at one location only) as auxiliary input, due to the DP definition of $\mathtt{M}$. The prover and verifier exchange messages for $t_\Pi = \lceil 2\mathtt{poly}(\kappa) \rceil$ rounds. As a convention, we denote with even indices rounds in which the verifier sends messages and denote with odd indices rounds when the prover sends messages (the last message is the verifier's decision).

In *at least* one round $j \in \{1, 3, \ldots, 2t_\Pi - 1\}$, the prover sends to the verifier V a message y_j with a special tag called output. Let $\vec{y}$ denote all the messages sent by the prover tagged as output. At the end of message exchanging rounds, the verifier uses a pre-specified deterministic polynomial time algorithm A to compute $y = A(\vec{y})$ before announcing its verdict. As for all $j \in \{1, 3, \ldots, 2t_\Pi - 1\}$, y_j is computed as a function of P_j's internal randomness, y is a random sample from some distribution $\mathcal{D}$ described by the prover and verifiers messages. We call this distribution $\mathcal{D}$, from which the verifier samples $y \xleftarrow{\$} \mathcal{D}$, the induced output distribution of $\Pi(\mathsf{P}, \mathsf{V})$. The verifiers' verdict (final output) is either 0 or 1, with 1 indicating that the verifier accepts the provers' claim that the induced distribution $\mathcal{D} = \mathtt{M}(X, f)$ and y is a sample from $\mathtt{M}(X, f)$, and 0 indicating otherwise. The proof system Π is an interactive proof system for DP if it satisfies the following conditions (stated informally, see Appendix A for formal definitions):

1. **Completeness**: If the prover and the verifier both behave honestly as prescribed by Π and the induced distribution $\mathcal{D} = \mathtt{M}(X, f)$, the verifier rejects with negligible probability.

2. **Soundness**: Let $\alpha(\kappa)$ be a noticeable function[9]. If the prover is dishonest, and its messages result in a final induced distribution $\mathcal{D}$, such that $d_{\mathtt{TV}}(\mathcal{D}, \mathtt{M}(X, f)) \geq \alpha(\kappa)$, then the verifier accepts with negligible probability. This implies that the curator responsible for outputting DP statistics cannot tamper with the protocol noticeably without getting caught by the verifier. Thus a malicious curator can no longer use DP randomness as a smokescreen to hide its nefariousness.

3. **Zero Knowledge**: So far, we have not ruled out protocols where the prover sends the verifier its private randomness along with y. However, this would defeat the purpose of DP. Let $X' = X \setminus \{x\}$ denote any neighbouring dataset of X. We need to prove that a verifier that has access to X' still does not learn anything more about x than what it would learn from a sample from $\mathtt{M}(X, f)$. To do this, we borrow from the definition of zero knowledge and construct a PPT algorithm called the simulator that receives the same inputs as a corrupted verifier $\hat{\mathsf{V}}$, and has oracle access[10] to $\hat{\mathsf{V}}$ and sample access[11] to $\mathtt{M}(X, f)$. Then we show the view of the simulator is indistinguishable from the view of a corrupted verifier $\hat{\mathsf{V}}$ that tries to deviate from protocol. The view of a party is computed as a function of all the messages it receives, its internal randomness and the joint output of the protocol.

Remark 1 An interesting point to note is that here verifier plays a dual role. An honest verifier ensures that the output is faithfully generated and thus plays an active role in noise generation (as the final output y is a function of its and the provers' messages) without ever knowing the values on the provers' random tape. On the other hand, a dishonest verifier tries to tamper with the protocol to breach privacy (by trying to learn more than just the output). In non-verifiable DP, the analyst (verifier) only has sample access to the induced distribution. They have no agency over how this distribution is constructed. Thus the verifier participating in verifiable DP has a greater attack surface than a classical adversary in traditional non-verifiable DP. In the full version of this work [BC22], we elaborate on this when trying to establish separations between statistical DP and computational DP.

Remark 2 When extending the above definitions to the MPC setting, just as 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.

[9] A noticeable function is a function which is *not* negligible in the security parameter, where a negligible is defined in Definition 3 in Appendix A.

[10] By oracle access we mean black box access as defined [Gol07, Chapter 4].

[11] We say an algorithm has sample access to a distribution $\mathcal{D}$ if it can query an oracle $\mathcal{O}_{\mathcal{D}}$ to retrieve samples $y \xleftarrow{\$} \mathcal{D}$.

3 Main Results

In this section, we state our theorems informally and briefly discuss our techniques. First, we describe efficient constructions for interactive proof systems to verifiably compute Computationally Differentially Private (CDP, formalized in Definition 7) counting queries using the Binomial Mechanism (Lemma 1). In these constructions, n clients publicly commit to their inputs before sending them to the curator. The honest verifier uses these commitments to ensure the curator computes the query f correctly. The hiding property of commitments ensures that the verifier learns nothing about the inputs from these public commitments. To ensure the prover samples *provable but private* unbiased randomness for the binomial mechanism, we describe a novel algorithm using homomorphic commitments by which the verifier can audit the prover's randomness without seeing the prover's random coins. This allows the prover to commit to using the correct randomness required for DP without actually revealing the randomness. Once we have established commitments for client inputs and prover randomness, we use cryptographic verification techniques from [BSCG+13, BCV16] to verify the final computation. The essence of the protocol is to use interaction between the prover and the verifier to create a new set of committed binary coins that are truly random (provided that the prover and verifier do not collude). The sum of the new coin values, interpreted as bits, provides the right Binomial noise distribution that yields differential privacy. The homomorphic nature of the commitments ensures that the verifier can construct a commitment to the sum of the inputs and all the coins, which is the output that the prover can decommit to. As our results depend heavily on homomorphic commitment schemes (hence one-way functions), we restrict both the prover and the verifier to be PPT and thus can only guarantee results for CDP.

Theorem 1 (Informal) There exists an efficient construction for an interactive proof of CDP for computing counting queries using the binomial mechanism.

The techniques to output counting queries verifiably can be further generalised for computing CDP mechanisms represented by any polynomial-sized arithmetic circuit. Addition gates are handled naturally by homomorphism. For multiplication gates, we use Beaver's triples and interaction [DKL+13] to verify multiplications of committed inputs and randomness.

Theorem 2 (Informal) There exists an efficient construction for an interactive proof of CDP for computing any mechanism described by a polynomial-sized arithmetic or boolean circuit.

Remark 3 In all our constructions, the verifier is always a public coin protocol and contributes no private input to the proof system.

Next, we show that, without assumptions such as a random oracle (ROM) or common random string (CRS), it is impossible to have non-interactive proofs (Merlin-Arthur proofs) for DP. This result follows immediately from folklore results about verifiable coin flipping. If the prover's message is not a function of the verifier's messages, then the verifier has no control over what the prover sends. Given that the verifier can receive only a *single* draw from the output distribution, it cannot construct confidence intervals about the prover's distribution either. Thus interaction is necessary for proof of DP, and the verifier and the prover must construct randomness for DP jointly. Our results can be interpreted as the DP analogue of the honest verifier to malicious verifier transformation for statistical zero knowledge [Vad99], where the prover and verifier must jointly compute the random selection protocol [DGW94].

Theorem 3 (Informal) Non-interactive proofs of DP are impossible in the plain model.

The definition of information-theoretic DP states that even a computationally unbounded adversary that has access to a neighbouring input dataset X' cannot distinguish samples drawn from $\mathtt{M}(X, f)$ and $\mathtt{M}(X', f)$ where $\mathtt{M}$ is an (ε, δ)-DP mechanism for computing f. However, we show that if the prover tried to convince such an adversary that they used randomness correctly, it would violate differential privacy in the process (or the prover would be able to lie about their randomness). Our proofs use the seminal results of [HO14], which state that fair coin flipping implies the existence of one-way functions and otherwise incurs noticeable bias. Interactive proofs for the binomial mechanism in the presence of a computationally unbounded verifier and prover would not only allow us to flip fair coins but also to do so privately (which is a strictly harder task).

Theorem 4 (Informal) If either the prover or the verifier is computationally unbounded, then interactive proofs for *Information Theoretic* DP is impossible.

Finally, we provide performance benchmarks that show that constructions given in Theorems 1 and Theorem 2 are efficient and can be deployed in the real world for medium-sized datasets (about $n = 10^6$) clients: each individual check takes at most a couple of hundred milliseconds, while the entire interactive proof protocol can be performed in a matter of minutes, comparable to the running time of non-verified systems for working with private data.

References

[BBCG+21] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight Techniques for Private Heavy Hitters. *arXiv:2012.14884 [cs]*, 2021.

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

[BC22] Ari Biswas and Graham Cormode. Verifiable differential privacy for when the curious become dishonest. *arXiv preprint arXiv:2208.09011*, 2022.

[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, 2016.

[BDO14] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In *Security and Cryptography for Networks*, pages 175–196, 2014.

[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 *International Cryptology Conference*, pages 92–122, 2021.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108. Springer, 2013.

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

[CSU19] Jeffrey Champion, Abhi Shelat, and Jonathan Ullman. Securely sampling biased coins with applications to differential privacy. In *ACM CCS*, pages 603–614, 2019.

[DGW94] Ivan Damgård, Oded Goldreich, and Avi Wigderson. *Hashing functions can simplify zero-knowledge protocol design (too)*. Citeseer, 1994.

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

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits. In *European Symposium on Research in Computer Security*, pages 1–18, 2013.

[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. *SICOMP*, 43(2):389–409, 2014.

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

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

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

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

[Vad99] Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.

[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 *IEEE S&P*, pages 1074–1091, 2021.

A Formal Definitions

A.1 Notation

For any $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$ and $[a, b]$ to denote closed interval of of real numbers between $a < b$. All logarithms are base two unless otherwise specified. We use the convention that lowercase letters are the logarithm (base 2) of the corresponding capital letter (e.g., for any $n \in \mathbb{N}$, $N = 2^n$). We use the abbreviation PPT to denote probabilistic polynomial time. For any set U, we write $x \xleftarrow{\$} U$ to denote that x was uniformly sampled from the set U. For $n \in \mathbb{N}$, we denote by $\Delta(\Omega_{\mathsf{n}})$ the set of all probability distributions over the set Ω_{n}. For any $\mathcal{D} \in \Delta(\Omega_{\mathsf{n}})$ we write $x \xleftarrow{\$} \mathcal{D}$ to denote x was sampled according to $\mathcal{D}$. We identify a distribution $\mathcal{D}$ with its probability mass function: for every $x \in \Omega_{\mathsf{n}}$, $\mathcal{D}(x)$ denotes the probability $\Pr_{X \xleftarrow{\$} \mathcal{D}}[X = x]$ and for every set $S \subseteq \Omega_{\mathsf{n}}$, $\mathcal{D}(S)$ denotes the $\Pr_{X \xleftarrow{\$} \mathcal{D}}[X \in S]$. We use $S_{\mathcal{D}}$ to denote the support of the distribution $\mathcal{D}$.

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.

A.2 Preliminaries

Definition 1 (Distance Between Distributions) Let $\mathcal{D}, \mathcal{D}'$ be probability distributions (possibly joint distributions) over domain Ω_{n}. We define the total variation distance between two distributions as

$$d_{\mathtt{TV}}(D, D') = \frac{1}{2}||\mathcal{D} - \mathcal{D}'||_1 = \max_{S \subseteq \Omega_{\mathsf{n}}} |\mathcal{D}(S) - \mathcal{D}'(S)| \ . \tag{1}$$

Definition 2 (Sample Access) We say an algorithm $\mathcal{A}$ has sample access to a distribution $\mathcal{D}$ if it has access to an oracle that it can invoke to generate independent samples from $\mathcal{D}$. We remark that the oracle is memoryless and sequential invocations of the oracle are equivalent to drawing independent samples from $\mathcal{D}$. We denote the output of an algorithm $\mathcal{A}$ with input x (explicit) and sample access to $\mathcal{D}$ by $\mathcal{A}^{\mathcal{D}}(x)$.

Definition 3 (Negligible Functions And Noticeable Functions) A function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible iff $\forall c \in \mathbb{N}$, there $\exists n_c \in \mathbb{N}$, such that $\forall n > n_c$, $\mu(n) \leq n^{-c}$. Any function *not* negligible in n is called *noticeable* in n.

Definition 4 (Statistical Distinguishablity) Fix security parameter $n \in \mathbb{N}$. We say two probability ensembles $\{X_n\}_{n \in \mathbb{N}}$ and $\{Y_n\}_{n \in \mathbb{N}}$ are statistically indistinguishable if there exists a negligible function μ such that, for all sufficiently large n's it holds that

$$d_{\mathtt{TV}}(X_n, Y_n) \leq \mu(n)$$

We denote two statistically indistinguishable distributions as $\{X_n\}_{n \in \mathbb{N}} \overset{s}{\equiv} \{Y_n\}_{n \in \mathbb{N}}$. If we have $\mu(n) = 0$, we say the two distributions are perfectly indistinguishable and we denote it as $\{X_n\}_{n \in \mathbb{N}} \equiv \{Y_n\}_{n \in \mathbb{N}}$.

Definition 5 (Computational Indistinguishability) Fix security parameter $n \in \mathbb{N}$. Let $\{X_n\}_{n \in \mathbb{N}}$ and $\{Y_n\}_{n \in \mathbb{N}}$ be probability distributions over $\{0,1\}^{\mathtt{poly}(n)}$. If for all non-uniform PPT Turing machines D ("distinguishers"), there exists a negligible function μ, such that for every $n \in \mathbb{N}$

$$\Big|\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]\Big| \leq \mu(n)$$

We say that $\{X_n\}_{n \in N}$ and $\{Y_n\}_{n \in \mathbb{N}}$ are computationally indistinguishable and denote it as $\{X_n\}_{n \in \mathbb{N}} \overset{c}{\equiv} \{Y_n\}_{n \in \mathbb{N}}$.

Definition 6 (Information Theoretic Pure DP) Let $\kappa \in \mathbb{N}$ be the security parameter. Fix $n \in \mathbb{N}, \varepsilon \geq 0$ and $\delta(\kappa)$ as a negligible function. Let $\mathcal{Q} = \{f : \mathcal{X}^n \to \mathcal{Y}\}$ denote a family of functions whose output we wish to make differentially private. Further assume that the L_1 norm is well defined on $\mathcal{X}$ and $\mathcal{Y}$. Define a mechanism $\mathtt{M} : \mathcal{X}^n \times \mathcal{Q} \to \Delta(\mathcal{Y})$ that takes as input n client inputs $X = (x_1, \ldots, x_n)$ and a function $f \in \mathcal{Q}$ and constructs a distribution $\mathtt{M}(X, f) \in \Delta(\mathcal{Y})$. It then outputs an oracle $\mathcal{O}_{\mathtt{M}(X,f)}$ that provides sample access to $\mathtt{M}(X, f)$. $\mathtt{M}$ satisfies ε, δ differential privacy if for *every* two neighboring datasets $X \sim X'$ such that $||X - X'||_1 = 1$ and for *every* query $f \in \mathcal{Q}$ we have *for all* $T \subseteq \mathcal{Y}$

$$\Pr_{Y \xleftarrow{\$} \mathtt{M}(X,f)}[Y \in T] \leq e^{\epsilon} \Pr_{Y \xleftarrow{\$} \mathtt{M}(X',f)}[Y \in T] + \delta(\kappa) \tag{2}$$

Definition 7 (Computational DP [MPRV09]) Fix $\kappa \in \mathbb{N}$ and $n \in \mathbb{N}$. Let $\varepsilon \geq 0$ and $\delta(\kappa) \leq \kappa^{-\omega(1)}$ be a negligible function, and let $\mathtt{M} = \{\mathtt{M}_\kappa : \mathcal{X}_\kappa^n \to \mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of randomised algorithms, where $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ can be represented by $\mathtt{poly}(\kappa)$-bit strings. We say that $\mathtt{M}$ is *computationally ε-differentially private* if for every non-uniform PPT Turing

machine ("distinguishers") D_z with auxiliary input z, for *every* query $f \in \mathcal{Q}$, and for *every* neighbouring dataset $X \sim X'$, *for all* $T \subseteq \mathcal{Y}_\kappa$ we have

$$\Pr_{Y \xleftarrow{\$} \mathtt{M}(X,f)}\Big[D_{X'}(Y \in T) = 1\Big] \leq e^{\epsilon} \cdot \Pr_{Y \xleftarrow{\$} \mathtt{M}(X',f)}\Big[D_{X'}(Y \in T) = 1\Big] + \delta(\kappa) \tag{3}$$

In the above definition, the distinguishing algorithm knows X' i.e., all but one input in X, and it still cannot distinguish between samples between $\mathtt{M}(X, f)$ and $\mathtt{M}(X', f)$.

Lemma 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 $\eta > 30$, $0 < \delta \leq o(\frac{1}{\eta})$ and let $Z \sim \mathtt{Binomial}(\eta, \frac{1}{2})$. Then $Z + Q(X)$ is an (ϵ, δ)-differentially private with $\epsilon = 10\sqrt{\frac{1}{\eta} \ln \frac{2}{\delta}}$.

Cryptographic Background

Definition 8 (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 9 (Commitments) Let $\kappa \in \mathbb{N}$ be the security parameter. A non-interactive commitment scheme consists of a pair of probabilistic polynomial time algorithms ($\mathtt{Setup}$, $\mathtt{Com}$). The setup algorithm $\mathtt{pp} \leftarrow \mathtt{Setup}(1^\kappa)$ generates public parameters $\mathtt{pp}$. Given a message space $\mathtt{M}_{\mathtt{pp}}$ and randomness space $\mathtt{R}_{\mathtt{pp}}$, the commitment algorithm $\mathtt{Com}_{\mathtt{pp}}$ defines a function $\mathtt{M}_{\mathtt{pp}} \times \mathtt{R}_{\mathtt{pp}} \to \mathtt{C}_{\mathtt{pp}}$ that maps a message to the commitment space $\mathtt{C}_{\mathtt{pp}}$ using the random space. For a message $x \in \mathtt{M}_{\mathtt{pp}}$, the algorithm samples $r_x \xleftarrow{R} \mathtt{R}_{\mathtt{pp}}$ and computes $c_x = \mathtt{Com}_{\mathtt{pp}}(x, r_x)$. When the context is clear, we will drop the subscript and write $\mathtt{Com}_{\mathtt{pp}}$ as $\mathtt{Com}$.

Definition 10 (Homomorphic Commitments) A homomorphic commitment scheme is a non-interactive commitment scheme such that $\mathtt{M}_{\mathtt{pp}}$ and $\mathtt{R}_{\mathtt{pp}}$ are fields (with $(+, \times)$) and $\mathtt{C}_{\mathtt{pp}}$ is an abelian groups with the $\otimes$ operator on which the discrete log problem (Definition 8) is hard, so that for all $x_1, x_2 \in \mathtt{M}_{\mathtt{pp}}$ and $r_1, r_2 \in \mathtt{R}_{\mathtt{pp}}$ we have

$$\mathtt{Com}(x_1, r_1) \otimes \mathtt{Com}(x_2, r_2) = \mathtt{Com}(x_1 + x_2, r_1 + r_2) \tag{4}$$

Throughout this paper, when we use a commitment scheme, we mean a non-interactive homomorphic commitment scheme with the following properties:

1. **Hiding:** A commitment c_x reveals no information about x and r_x to a computationally bounded adversary.
   Definition 11 (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 $\mu(\kappa) = 0$.
   $$\Pr\left[b = b' : \begin{array}{c}\mathtt{pp} \leftarrow \mathtt{Setup}(1^\kappa) \\ b \xleftarrow{R} \{0,1\}, r_{x_b} \xleftarrow{R} \mathtt{R}_{\mathtt{pp}} \\ (x_0, x_1) \in \mathcal{M}_{\mathtt{pp}}^2 \leftarrow \mathcal{A}(\mathtt{pp}) \\ c = \mathsf{Com}(x_b, r_{x_b}), b' = \mathcal{A}(\mathtt{pp}, c)\end{array}\right] \leq \mu(\kappa)$$
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 $\mathsf{Com}(x', r_{x'}) = c_x = \mathsf{Com}(x, r_x)$
   Definition 12 (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 $\mu(\kappa) = 0$.
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 $\Pi_{\mathtt{OR}}$ and say it securely computes the oracle $\mathcal{O}_{\mathtt{OR}}$, which computes if $c_x \in L_{\mathtt{Bit}}$
   $$L_{\mathtt{Bit}} = \{c_x : x \in \{0,1\} \wedge c_x = \mathsf{Com}(x, r_x)\} \tag{5}$$
   where for some $r_x \in \mathbb{Z}_q$.

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.

A.3 Interactive Proofs For Differential Privacy

Definition 13 (Interactive Proofs For DP) Fix $\kappa \in \mathbb{N}$ as the security parameter. Let $\mathtt{M}$ be a DP mechanism for computing $f : \mathcal{X}^n \to \mathcal{Y}$ using inputs $X = (x_1, \ldots, x_n)$ received from n distinct clients (input sources). Let $K \geq 1$ and $(X_1, \ldots, X_K)$ denote a partitioning/secret sharing of X such that there exists a public deterministic PPT reconstruction algorithm A, such that $X = A(X_1, \ldots, X_K)$.

An interactive proof for DP (IPDP) for $\mathtt{M}$ consists of a PPT algorithm $\mathtt{Setup}$ and a multi-prover (single if $K = 1$) interactive proof system Π. $\mathtt{pp} \xleftarrow{\$} \mathtt{Setup}(1^\kappa)$ describes the PPT algorithm that receives a unary representation of the security parameter as input and generates public parameters $\mathtt{pp}$. Π denotes a multi prover interactive proof system between $K \geq 1$ provers $\vec{\mathsf{P}} = (\mathsf{P}_1, \ldots, \mathsf{P}_K)$ and a single verifier V where for each $j \in [K]$, $(X_j, r_{\mathsf{P}_j}^{\vec{}}, \mathtt{pp})$ denotes the inputs for P_j and $(\vec{z}, \vec{r}_{\mathsf{V}}, \mathtt{pp})$ denote the verifier's input. Here where $r_{\mathsf{P}_j}^{\vec{}}$ and $r_{\mathsf{V}}^{\vec{}}$ denotes P_j's and the verifier's internal randomness respectively and $\vec{z}$ denotes the verifier's auxiliary input. The proof system Π is an interactive proof system for mechanism $\mathtt{M}$ if there exist negligible functions δ_c and δ_s in κ such that the following hold:

1. **Completeness:** Let $\Pi(\vec{\mathsf{P}}, \mathsf{V})$ denotes the induced output distribution from which the verifier samples the final output $y \xleftarrow{\$} \Pi(\vec{\mathsf{P}}, \mathsf{V})$ if the verifier and all provers are honest. If $\Pi(\vec{\mathsf{P}}, \mathsf{V}) = \mathtt{M}(X, f)$

$$\Pr\left[\mathtt{out}(\mathsf{V}, \vec{\mathsf{P}}) = 0 : \begin{array}{c} \mathtt{pp} \xleftarrow{\$} \mathtt{Setup}(1^\kappa) \\ \mathsf{P}_j \xleftarrow{\text{Input}} (X_k, \vec{r}_{\mathsf{P}_j}, \mathtt{pp}) \\ \mathsf{V} \xleftarrow{\text{Input}} (\vec{z}, \vec{r}_{\mathsf{V}}, \mathtt{pp}) \\ y \xleftarrow{\$} \Pi(\vec{\mathsf{P}}, \mathsf{V}) \end{array}\right] \leq \delta_c.$$

2. **Soundness:** Let α denote a noticeable function in κ. For any subset, $I \subseteq [K]$, let $\mathcal{A}(I, \vec{\mathsf{P}})$ denote a subset provers indexed by I, that are corrupted by an adversary $\mathcal{A}$, and $\widetilde{\mathsf{P}}$ denote the set of honest provers not indexed by I. Let $\mathcal{D} = \Pi(\widetilde{\mathsf{P}} \cup \mathcal{A}(I, \vec{\mathsf{P}}), \mathsf{V})$ denote the induced output distribution from which the verifier samples the final output $y \xleftarrow{\$} \mathcal{D}$. Then, if $d_{\mathtt{TV}}(\mathcal{D}, \mathtt{M}(X, f)) > \alpha(\kappa)$

$$\Pr\left[\mathtt{out}(\mathsf{V}, \widetilde{\mathsf{P}} \cup \mathcal{A}(I, \vec{\mathsf{P}})) = 1 : \begin{array}{c} \mathtt{pp} \xleftarrow{\$} \mathtt{Setup}(1^\kappa) \\ \mathsf{P}_j \xleftarrow{\text{Input}} (X_k, \vec{r}_{\mathsf{P}_j}, \mathtt{pp}) \\ \mathsf{V} \xleftarrow{\text{Input}} (\vec{z}, \vec{r}_{\mathsf{V}}, \mathtt{pp}) \\ y \xleftarrow{\$} \mathcal{D} \end{array}\right] \leq \delta_s.$$

3. **Zero Knowledge**: Let $\mathcal{A}$ denote an adversary that knows all but one of the inputs of X (denoted as X'). Let $\hat{\mathsf{V}}$ denote an arbitrary verifier strategy that is corrupted along with any proper subset $I \subset [K]$ of provers by $\mathcal{A}$. Let $\mathcal{A}(I, \vec{\mathsf{P}})$ denote the collection of corrupted provers, indexed by I, and $\widetilde{\mathsf{P}}$ denote the set of honest provers. Let $\text{view}\left[\Pi\left(\widetilde{\mathsf{P}} \cup \mathcal{A}(I, \vec{\mathsf{P}})), \hat{\mathsf{V}}\right)\right]$ be the joint distribution[12] of messages received by $\mathcal{A}$ and induced output distribution $\mathcal{D}$ during the execution of Π in the presence of corrupted parties. There exists a PPT algorithm called $\mathsf{Sim}^{(\hat{\mathsf{V}}, \vec{\mathsf{P}^*}, \mathcal{D})}$ with black box access to $\hat{\mathsf{V}}$ and $\vec{\mathsf{P}^*}$ (via $\mathcal{A}$) and sample access to $\mathcal{D}$, such that if $\mathcal{D} = \mathtt{M}(X, Q)$

$$\text{view}_{\mathcal{A}}\left[\Pi\left(\widetilde{\mathsf{P}} \cup \mathcal{A}(I, \vec{\mathsf{P}}), \hat{\mathsf{V}}\right)\right] \stackrel{s}{\equiv} \mathsf{Sim}^{(\mathcal{A}, \mathcal{D})}(\vec{r}_{\mathsf{V}}, \vec{z})$$

Definition 14 (View of an interactive protocol) Let $\Pi(\vec{\mathsf{P}}, \mathsf{V})$ be an interactive proof system for differential privacy. Let $\mathcal{A}$ denote an adversary that controls an arbitrary verifier strategy $\hat{\mathsf{V}}$ and up to $K - 1$ provers. The view of the verifier $\mathcal{A}$ defined as

$$\text{view}_{\mathcal{A}}[\Pi(\mathsf{P}, \mathsf{V})] := (m_1, m_2, \ldots, m_{t_\Pi}; r_{\mathcal{A}}; y),$$

where $m_1, \ldots m_{t_\Pi}$ are the messages $\mathcal{A}$ receives from the honest prover's P, $r_{\mathcal{A}}$ denotes $\mathcal{A}$'s local coins, which it uses to set $\hat{\mathsf{V}}$ and the dishonest prover's random tape. z denotes $\mathcal{A}$'s auxiliary input and y denotes a samples drawn from the induced output distribution $\mathcal{D}$.

[12] As the output of $\mathtt{M}$ is random, 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.