Complexity of Differentially Private Selection via Federated APIs

Hilal Asi
Apple Inc.
hilal.asi94@gmail.com

Vitaly Feldman
Apple Inc.
vitaly.edu@gmail.com

Jelani Nelson*
Anthropic
minilek@alum.mit.edu

Huy L. Nguyen
Northeastern University
huylenguyen@gmail.com

Kunal Talwar
Apple Inc.
kunal@kunaltalwar.org

Samson Zhou
Texas A&M University
samsonzhou@gmail.com

Abstract

Differential privacy (DP) has become the standard for privacy-preserving data analysis, yet its practical implementation in federated or distributed settings often faces a significant utility gap compared to the central model. While existing frameworks, including local, shuffle, and secure aggregation models, offer vetted primitives, they can incur high error for important tasks such as selection. Our first contribution is to show that this is inherent in the aggregation model: any private selection algorithm relying on the standard noisy aggregation primitive requires $\Omega(\sqrt{d}/\varepsilon)$ samples. To address this limitation, we propose adding a new primitive that implements a version of the *Sparse Vector Technique* (SVT) and argue that it has simple implementations under various trust models. We demonstrate its power by developing an ε-differentially private algorithm for the selection problem. For selection in dimension d, our algorithm achieves an additive error of $O\left(\frac{\log(dn)}{\varepsilon}\right)$, matching that of the central model when n is polynomially bounded in d. Furthermore, we show that this can be done using a near-optimal expected number $O(d \log^* d)$ of SVT queries. Together with existing lower bounds for the shuffle model, this result establishes an exponential separation between the SVT model and shuffle or aggregation-based models. We demonstrate the model's broad applicability to diverse tasks such as F_p loss minimization, heavy-hitter identification, and sparse mean estimation.

1 Introduction

Differential privacy (DP) [DMNS17, DR+14] has emerged as the gold standard for privacy-preserving data analysis, offering a mathematically rigorous framework that limits the risk of disclosing the data of any individual entity. Informally, a randomized algorithm is differentially private if its output distribution is nearly unchanged when any individual record is modified or removed. This ensures that adversaries learn little about any one participant, even with auxiliary information.

Despite its mathematical elegance, the practical implementation and verification of differential privacy can be challenging [BO13, BDG+13, GHH+13, BGA+14, BFG+16, BGG+16, ZK17]. Because privacy guarantees are probabilistic and often counterintuitive, even subtle coding errors or floating-point vulnerabilities can lead to privacy leakage [Mir12]. This has led the community toward primitives that avoid floating-point numbers [CKS22, BV19]. More broadly, verifying that an

*Author contributed to the project while at UC Berkeley.

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

implementation of a differentially private algorithm is correct can be challenging, and even testing or auditing for privacy requires considerable effort [BGD+18, GFHY23].

To mitigate these risks, the field has moved toward a modular "API-centric" approach. Rather than requiring developers to implement complex mechanisms from scratch, privacy frameworks, dating from early systems like PINQ [McS10] and Airavat [RSK+10] to modern efforts like OpenDP and Google's differential privacy building blocks, rely on a small set of vetted primitives. These primitives are the "building blocks" for private algorithms; by thoroughly auditing a limited number of functions, such as the Laplace mechanism or standard composition theorems, one can build complex pipelines while mathematically tracking the total privacy loss.

In federated or distributed architectures, implementing such primitives is harder, and thus this modularity is even more critical. Specifically, primitives must be implemented under a variety of operational and security constraints. The move toward these standardized, vetted interfaces has led to the adoption of large-scale DP frameworks [Tea20, WZL+20], which provide high-level query languages backed by reliable primitives. However, as the ecosystem of these frameworks expands, it becomes vital to understand the relative power of various primitives.

Previous work has successfully formalized specific trust models, such as the local [War65, EGS03, KLN+11] and shuffle [BEM+17, CSU+19, EFM+19] privacy models. More broadly, a privacy framework can treat simple privacy primitives as interchangeable building blocks and support a set of privacy-accounting methods. A complex algorithm can be built by combining primitives, each run on all or a subset of the data, while privacy accounting upper bounds the algorithm's privacy cost. As an example, an aggregation system allows for algorithms that apply a function $f : \mathcal{X} \to \mathbb{R}^d$ to each individual data point and output a noisy aggregate $\tilde{f}(x_1, \ldots, x_n) = \sum_i \text{clip}_1(f(x_i)) + \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$, for a user-specified noise parameter σ, where clip_1 is the standard function that truncates the input to have ℓ_2 norm at most 1. A Rényi accountant in this setting would accumulate privacy cost in terms of Rényi DP [Mir17]. One can then use such a system to implement differentially private gradient descent to train a machine learning model. Enhancing the primitive and accounting to allow for privacy amplification by sampling further enables DP stochastic gradient descent [BST14, ACG+16]. Aggregation, potentially with sampling, can support a rich class of algorithms, and such systems have been proposed under various assumptions [DKM+06, BIK+17a, RCWH+20, FES+17, RNFH19, RZHP20, TWM+24]. In this work, we study the limits of the Aggregation API, and propose an additional primitive to help circumvent some of these limitations.

2 Our Contributions

Despite the wide applicability of the Aggregation API, there are important tasks that are not known to be implementable accurately using aggregation alone. Foremost among these is perhaps the selection problem, where the best-known algorithms using aggregation incur significant overhead compared to the optimal central-DP bounds: while centrally one can do selection using $O(\log d/\varepsilon)$ samples, existing implementations using aggregation require $\Omega(\sqrt{d}/\varepsilon)$ samples. While this sample complexity is provably required (sometimes under additional assumptions) for weaker APIs such as local and shuffle DP [DJW13, CSU+19, CU21, CGKM21], no existing lower bounds exclude a better way to do selection using an aggregation API. Our first result is a strong lower bound ruling out this possibility. We show that *any* selection algorithm using an aggregation API requires sample complexity $\Omega(\sqrt{d}/\varepsilon)$ if it guarantees (ε, δ)-DP for any implementation of the aggregation API and can identify a dimension whose average score is $1/3$ more than all the others.

Theorem 2.1. *[Informal] Any $(O(1), O(1/\operatorname{poly}(d)))$-DP algorithm for selection from d candidates using aggregation queries alone requires $\Omega(\sqrt{d})$ samples.*

The main advantage of this lower bound over most existing lower bounds for selection in restricted models is that we allow algorithms that use queries adaptively. Adaptivity is crucial for implementing most standard ML algorithms and is known to be necessary for some learning problems [DF20]. We prove this result specifically for aggregation with Gaussian noise addition, with the privacy guarantee derived by composition of the guarantees for individual queries (Theorem 3.4). This implies that our lower bound applies to any API with a comparable privacy-utility trade-off, which is known to be asymptotically optimal for L_2 error. We observe that such an aggregation oracle can be implemented

via distributed noise addition [DKM+06] while satisfying weaker privacy guarantees. We then prove a lower bound for such protocols using the information-theoretic techniques from [BGM+16, DR19].

To address these limitations, we propose adding a new *Sparse Vector Technique* (SVT) primitive. This primitive implements a special case of the sparse vector technique of [DNR+09], and suffices to implement a large class of private algorithms. By integrating SVT as a fundamental primitive, our approach expands the expressive power of DP APIs, improves accuracy guarantees, and supports a wider range of applications.

Informally, the SVT primitive implements the AboveThreshold subroutine for additive queries. We first formally define the SVT primitive. This technique is designed to privately process a potentially long sequence of queries by revealing information only about those whose true values exceed a specified threshold. A key innovation of SVT is that by careful correlation of the noise, the incurred privacy cost grows with the number of initialized SVT streams, and is independent of the number of queries issued. In our proposed primitive, each query is restricted to be a sum of bounded contributions from individual users. These are often referred to as additive or counting queries in the context of SVT, and decomposable score functions in the context of selection. Given n users with data points $x_1, \ldots, x_n \in \mathcal{X}$, a query is specified by a function $q_j : \mathcal{X} \to [0, 1]$ and a threshold τ_j. Up to noise, the query asks if $\sum_{i=1}^{n} q_j(x_i) \geq \tau_j$, and a Yes answer closes the current stream. [2] Because SVT needs to be stateful, we wrap this functionality in an *object* that can be created once and queried multiple times with different (q_j, τ_j) values chosen potentially in an online fashion. As in SVT, the API ensures that No answers do not incur an additional per-query cost, and each query is correctly answered up to a small additive error.

This primitive has the advantage of simplicity: each query is similar to an aggregation query with one additional threshold parameter. This simplicity allows for efficient and modular implementations of this primitive. While the precise implementation will vary with the trust model, we discuss several options in Section 4.

At the same time, we show that this primitive is quite powerful. SVT by itself has several applications and one can naturally implement any algorithm that builds on SVT for additive queries in this model (e.g., Private Multiplicative Weights [HR10]). We next argue that selection (the task solved by the exponential mechanism [MT07]) can be implemented using the SVT API for any decomposable score function. Recall that given a dataset $(x_1, \ldots, x_n) \in \mathcal{X}^n$, and d score functions $s_1, \ldots, s_d : \mathcal{X}^n \to \mathbb{R}$, the goal is to select the candidate $k \in [d]$ that (approximately) maximizes the score. We restrict ourselves to additive score functions of the form $s_j(x_1, \ldots, x_n) = \sum_{i=1}^{n} f_j(x_i)$ for some $f_j : \mathcal{X} \to [0, 1]$. We say an output candidate j achieves additive error α if $s_j(x_1, \ldots, x_n) \geq \max_{k \in [d]} s_k(x_1, \ldots, x_n) - \alpha$.

The reader may immediately notice that if one knew the target value $\max_{k \in [d]} s_k(x_1, \ldots, x_n)$, then a sequential pass over the d score functions with that target value minus the SVT accuracy margin as a threshold yields a candidate with additive error at most twice that margin. This gives a mechanism with error $O(\log d/\varepsilon)$ that matches that of the exponential mechanism, and is known to be optimal. Without knowing the target value, the natural approach is to start with a large guess for this threshold, and gradually reduce it. This algorithm (Algorithm 7) yields the optimal error with $O(d(n+1))$ SVT queries. It is easy to see that any algorithm must make $\Omega(d)$ queries. When n is large, the algorithm above incurs significant overhead in the number of queries. This leads to the natural question of whether one can reduce the number of queries to be closer to linear in d. We show that this is indeed possible: we give a new algorithm that achieves the optimal $O(\log d/\varepsilon)$ additive error (when n is polynomial in d) and makes only $O(d \log^* d)$ queries to the SVT API in expectation.

Theorem 2.2. *Let $\varepsilon \in (0, 1]$. There exists an ε-pure DP protocol that solves the selection problem for additive scores up to expected additive error $O\left(\frac{\log(dn)}{\varepsilon}\right)$, using $O(d \log^* n + \log^3 n)$ oracle queries to the SVT API in expectation.*

When the number of users n is upper bounded by a polynomial in d, the error of our protocol matches the optimal additive error of $O\left(\frac{\log d}{\varepsilon}\right)$ achievable in the central model of privacy, and the query complexity matches the d lower bound up to an $O(\log^* d)$ factor. We then show a number of

[2] Symmetrically, one may consider a below-threshold version of such a query. More generally, one can consider a BetweenThresholds query [BSU17] with two thresholds, where the stream closes when the query answer is between the two thresholds.

applications of Theorem 2.2, specifically, to F_p losses, heavy hitters with categories, multiplicative weights with the exponential mechanism, and sparse mean estimation.

Our lower and upper bounds for selection extend to other natural tasks. Combined with known results for the shuffle model, our new upper bounds in the SVT model, and our lower bounds in the aggregation model emphasize the value of the SVT primitive. As we show in Figure 1, our new API allows for accurate selection, where aggregation and shuffle models must incur large errors.

	SecAgg	Shuffle DP	SVT	Central DP
Aggregation	$\frac{\sqrt{d}}{\varepsilon}$	$\frac{\sqrt{d}}{\varepsilon}$	$\frac{\sqrt{d}}{\varepsilon}$	$\frac{\sqrt{d}}{\varepsilon}$
Selection	$\frac{\sqrt{d}}{\varepsilon}$	$\frac{\sqrt{d}}{\varepsilon}$	$\frac{\log d}{\varepsilon}$	$\frac{\log d}{\varepsilon}$

Fig. 1: Comparison of the additive error achievable for the different APIs in the federated setting, up to constant factors, for n users. In the aggregation row each user holds a vector of ℓ_2 norm at most 1 and the error is the ℓ_2 error of the sum. In the selection row each user holds scores in $[0,1]^d$ and the error is the additive error in the sum score; the entry $\sqrt{d}/\varepsilon$ indicates that error $o(n)$ is not achievable unless $n \geq \Omega(\sqrt{d}/\varepsilon)$. The entries for SVT and central DP are upper bounds (ε-DP, Theorem 2.2 for SVT with n polynomial in d). The entries for SecAgg and shuffle DP are lower bounds: for algorithms that access the data through an aggregation API (Theorem 3.4), and for robust or single-round shuffle protocols (Theorem A.3 and Theorem A.4).

Other related work. Our lower bounds for aggregation APIs are most closely related to statistical query (SQ) lower bounds in learning theory [BFJ+94, FGR+17]. An SQ oracle is an oracle that for a query q and data distribution P is guaranteed to return a value within some prescribed tolerance τ of the expectation $\mathbf{E}_{x\sim P}[q(x)]$ (the returned value may be chosen arbitrarily by the oracle). It is known that learning a parity function over k coordinates requires $\Omega(2^k\tau^2)$ statistical queries of tolerance τ (e.g. [Fel14]). Learning a parity function reduces to selection in dimension 2^k and thus this also gives a lower bound for selection. These lower bounds do not apply to our setting as we answer queries relative to a given dataset (and not the distribution).

SQ lower bounds are known to imply lower bounds for the local model of privacy with a pure-DP randomizer [KLN+11]. This reduction is also based on a simulation of a randomizer using an SQ oracle and, as a result, does not imply any lower bounds for the setting where the number of queries is comparable to d. A direct lower bound of $\Omega(d/\varepsilon^2)$ samples for ε-DP local *non-interactive* protocols was given by [DJW18]. A similar lower bound on non-interactive local-DP protocols was given by [Ull18] for sparse Boolean mean estimation. More recently, a lower bound of d/ε^2 for sparse Gaussian mean estimation in the local DP model that allows interactivity and (ε,δ) and 1-Renyi-DP (or KL-private) randomizers was given by [DR19].

Several recent works [CU21, CGKM21, BHNS20] provide lower bounds for private selection in variants of the multi-message shuffle model. All of these apply only to non-interactive (or one-round shuffling) settings. In fact, [BHNS20] show that a two-round multi-message protocol can simulate arbitrary secure multi-party protocols. The lower bound of [CU21] is the same as ours, $\Omega(\sqrt{d}/\varepsilon)$, and is based on somewhat related ideas. The lower bounds in [CGKM21, BHNS20] are of the form d/m, where m is the number of messages sent by each user, and thus are not directly comparable to ours. The question of implementing selection using simpler APIs has also been raised as an open question for *low-sensitivity* queries in [Ste25], and recent work [LM25] has shown nearly-optimal algorithms for that model. Arbitrary low-sensitivity queries can, however, be challenging to implement, and hence we focus on additive queries in this work.

3 Lower Bounds for Private Selection via Noisy Aggregates

The problem of private selection and its specific instantiations have been studied in a number of distributed models of computation. In this section, we review the known lower bounds that are most relevant to our setting. We then give a new lower bound for algorithms that rely on noisy aggregation. Specifically, we study algorithms whose *only* access to the dataset is through the noisy aggregation oracle defined in Definition 3.1, i.e., through noisy values of $\sum_{i\in[n]}\phi(x_i)$. Within that constraint, the algorithm is entirely unrestricted: it may issue unboundedly many adaptively chosen queries $\phi : \mathcal{X} \to \mathbb{R}^k$ with an arbitrary output dimension k. Such queries are referred to as high-dimensional linear or statistical queries.

Our lower bounds apply to algorithms that receive answers perturbed by standard Gaussian noise. This is both the most commonly used method in practice and the method that achieves an asymptotically tight privacy-utility trade-off (for expected squared L_2 error). Consequently, our lower bounds apply to any algorithm whose privacy guarantee is derived from black-box aggregation guarantees (which encompasses essentially all deployed aggregation APIs), since beating it would require beating the optimal trade-off for aggregation itself. We further remark that aggregation is often combined with subsampling (using privacy amplification results for subsampling). While such results improve the communication and computation efficiency of the implementation, they do not improve the privacy-utility trade-off.

We start by formally defining the Gaussian Aggregation oracle. For simplicity of presentation, we assume that each query given to the oracle has L_2 sensitivity at most 2 under replacement adjacency, since its per-record output is in $\mathcal{B}_2^k(1) = \{x \in \mathbb{R}^k \mid \|x\|_2 \leq 1\}$. Queries with other bounds on sensitivity can be implemented by scaling the noise parameter of the oracle and then scaling the result back.

Definition 3.1. *For the input dataset $S = (x_1, \ldots, x_n)$, let $\mathcal{G}(S)$ be the oracle parameterized by S that, given a query $\phi\colon \mathcal{X} \to \mathcal{B}_2^k(1)$ and parameter $\sigma > 0$, returns a fresh sample from $\mathcal{N}(\phi(S), \sigma^2 I_k)$, where $\phi(S) = \sum_{i\in[n]} \phi(x_i)$.*

The privacy properties of a single answer to such a query are well-known. Most conveniently for applications involving composition, a single answer satisfies $2/\sigma^2$-zCDP [BS16]. We assume that privacy bounds for an algorithm with access to the Gaussian Aggregation oracle are derived directly from (advanced) composition of individual privacy guarantees. Following an analogous notion in [JMNR19], we refer to such privacy guarantees as *compositional*.

Definition 3.2. *Let $\mathcal{A}$ be an algorithm that is given access to $\mathcal{G}(S)$ and asks t (adaptively chosen) queries with noise scales $\sigma_1, \ldots, \sigma_t$. We say that $\mathcal{A}$ satisfies* compositional *ρ-zCDP if the sequence of zCDP parameters of individual queries $\rho_1 = 2/\sigma_1^2, \ldots, \rho_t = 2/\sigma_t^2$ satisfies $\sum_{i\leq t} \rho_i \leq \rho$.*

We remark that, for simplicity, in this definition the noise scales are chosen non-adaptively, since that is required by the standard composition results for Renyi DP [Mir17, BS16], and standard applications such as DP-SGD do not need adaptive choice of noise scales. However, using fully adaptive composition results for RDP with a Renyi filter [FZ21], we can also allow the noise scales to be chosen adaptively (with the filter ensuring the condition $\sum_{i\leq t} \rho_i \leq \rho$).

Our lower bound for selection for d score functions will apply to an easier problem of distinguishing the case where all score functions have mean $1/2$ from the case in which one of the functions has mean 1. Indeed, the index returned by a selection algorithm can be verified using a single aggregation query. Formally, we assume that each user i is given a d-dimensional Boolean vector X^i and the j-th score function is simply the coordinate j of that vector (denoted by X^i_j).

Problem 3.3 (Sparse Boolean distinguishing)**.** *For $Z \in \{1, 2\}$ and $V \in [d]$ we define a distribution $D_{Z,V}$ over $\{0, 1\}^d$. When $Z = 2$, $D_{Z,V}$ is uniform on $\{0, 1\}^d$. When $Z = 1$, $D_{Z,V}$ is uniform on $\{x \in \{0, 1\}^d : x_V = 1\}$. Each user i is given an independent sample X^i from $D_{Z,V}$. The goal is to estimate Z for Z chosen uniformly in $\{1, 2\}$ and V chosen uniformly in $[d]$. An algorithm (or protocol) is $2/3$-accurate if it produces an estimate $\widehat{Z}$ such that $\Pr[\widehat{Z} = Z] \geq 2/3$, where the probability is under all the random choices of the problem and the algorithm.*

We now state our main lower bound for this problem.

Theorem 3.4. *Every $2/3$-accurate algorithm for Sparse Boolean distinguishing (Problem 3.3) that is given access to $\mathcal{G}(S)$ and satisfies compositional ρ-zCDP must have $n = \Omega(\sqrt{d/\rho})$ samples.*

The key idea of our proof is to use the fact that the Gaussian Aggregation oracle can be implemented via distributed noise addition [DKM+06]. The transcript of this implementation satisfies privacy guarantees that are exactly n times worse (in terms of zCDP parameters) than the guarantees satisfied by the aggregates since each user adds Gaussian noise at scale $\sigma/\sqrt{n}$. Equivalently, Gaussian aggregation with zCDP parameter ρ can be implemented using a *local* $n\rho$-zCDP protocol. By applying zCDP composition to the local zCDP parameters of each query implementation, we get a direct simulation of any algorithm that has access to $\mathcal{G}$ and satisfies compositional zCDP.

Lemma 3.5 ([DKM+06])**.** *Let $\mathcal{A}$ be an algorithm that is given access to $\mathcal{G}(S)$ and satisfies* compositional *ρ-zCDP. Then there exists a local $n\rho$-zCDP protocol Dist-$\mathcal{A}$ that simulates $\mathcal{A}$.*

Therefore it suffices to prove lower bounds for local zCDP protocols. Lower bounds for such protocols that allow adaptive queries were first established by Duchi and Rogers [DR19], building on the framework of Braverman et al. [BGM+16]. In particular, they show that constant-error 1-sparse Gaussian mean estimation requires $\Omega(d/\rho_{\text{local}})$ samples, where ρ_{local} is the RDP parameter of order 1 (or KL) of the protocol. This result can be used to derive lower bounds for selection by reducing the 1-sparse Gaussian mean estimation to selection. Such a reduction will, however, incur an additional logarithmic dependence on d due to the need to bound the range of score functions. Our lower bound for Sparse Boolean distinguishing avoids this logarithmic factor, is simpler, and is essentially self-contained. We defer the proof, which may be of independent interest, to Appendix C.

We also note that ρ-zCDP implies $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP for every $\delta > 0$ [BS16]. Thus, if an algorithm derives an (ε, δ)-DP guarantee with $\varepsilon \leq 1$ from its compositional zCDP bound (as is done by the Rényi accountants used in practice), then $\rho \leq \varepsilon^2/(4\log(1/\delta))$ and Theorem 3.4 gives $n = \Omega(\sqrt{d\log(1/\delta)}/\varepsilon)$.

4 The SVT Primitive

We start by defining the expected behavior of our proposed SVT primitive. Informally, it takes as input a sequence of linear queries and thresholds, and up to a small additive error, the oracle outputs YES if $f(x_1, \ldots, x_n) \geq \tau$, and outputs NO otherwise. Crucially, one initialized stream has a fixed privacy cost, independently of the number of NO answers. To support online and adaptive queries, we allow the queries to be asked one at a time after initializing an SVT stream. We discuss both a batch version, which may be simpler to implement (Appendix D.1), and a more general version that implements the BETWEENTHRESHOLD algorithm of [BSU17] (Appendix D.2).

Initialize: Takes as input a privacy parameter ε, and creates the required internal state.

Query: Takes as input a function $f : \mathcal{X} \to [0, 1]$ and a threshold τ, informally asking whether $\sum_{i=1}^{n} f(x_i) \geq \tau$. Outputs YES or NO. If it outputs YES, the stream is closed and does not accept any more queries.

(α, β)**-Accuracy Property:** We say a response is α-accurate if the following hold: (a) If for a query (f, τ), $\sum_{i=1}^{n} f(x_i) \geq \tau + \alpha$, the output is YES. (b) If for a query (f, τ), $\sum_{i=1}^{n} f(x_i) \leq \tau - \alpha$, the output is NO. We say an SVT API is (α, β)-accurate for a specific query limit if all its responses are α-accurate except with probability β.

Using such a primitive, one can design compositionally differentially private algorithms in a natural way, where the incurred privacy cost is computed by applying a composition theorem to the privacy cost of individual API calls. In the upper bounds presented in this work, we use basic composition for pure-DP selection and advanced composition for approximate-DP applications [DR+14].

One way to implement this primitive is to use the ABOVETHRESHOLD algorithm, which satisfies ε-DP, and gives us (α, β)-accuracy for any sequence of K queries, with $\alpha = O(\log \frac{2(K+1)}{\beta}/\varepsilon)$. This algorithm is originally due to [DNR+09], and the formal description below is equivalent to the version in [DR+14].

Algorithm 1 SVTInit

Input: Privacy parameter ε.
Output: Create an SVT object S

1: $S.n \leftarrow \text{Lap}\left(\frac{2}{\varepsilon}\right)$ ▷This internal variable is private to the object.
2: $S.\varepsilon \leftarrow \varepsilon$
3: $S.Done \leftarrow False.$
4: Incur privacy cost ε, and return handle to S.

The following proofs can be found in [DR+14].

Theorem 4.1. *SVT satisfies ε-differential privacy.*

Theorem 4.2. *For any sequence of functions $f_1, \ldots, f_K : \mathcal{X} \to [0, 1]$, and thresholds $\tau_1, \ldots, \tau_K$, the output of SVT is (α, β)-accurate for any $\beta \in (0, 1)$ and $K \geq 1$, with $\alpha = \frac{8}{\varepsilon} \log \frac{2K}{\beta}$.*

Algorithm 2 SVTQuery

Input: Handle S, Function $f : \mathcal{X} \to \mathbb{R}$, Threshold $\tau \in \mathbb{R}$. Private input $\mathbf{x}$
Output: Output YES or NO.

if not $S.Done$ **then**
  Compute $f(\mathbf{x}) = \sum_i \texttt{clip}_{[0,1]}(f(x_i))$.
  Sample u from Lap $\left(\frac{4}{S.\varepsilon}\right)$.
  if $f(\mathbf{x}) + u > \tau + S.n$ **then**
    Return YES
    $S.Done \leftarrow True$
  else
    Return NO
else
  Return ERROR.

The clipping step in Algorithm 2 ensures that every contribution lies in $[0, 1]$ and hence that the privacy guarantee holds for arbitrary functions f. We state the accuracy guarantee only for functions with range $[0, 1]$, for which clipping has no effect.

We describe various design choices for our formalization of the SVT primitive. First, our choice of an SVT-based API over a dedicated selection primitive merits justification. Our goal is to propose a primitive that is relatively simple to implement while still capturing the power used in most commonly used DP techniques. While selection is a more common task, it is not clear that it allows for the implementation of SVT, so SVT is arguably a more general primitive. Regardless, abstracting and understanding the relative power of different natural primitives is an important goal in itself.

Second, as defined, the ABOVETHRESHOLD algorithm is well-defined for arbitrary sensitivity-1 queries, and indeed the privacy and utility guarantees in Theorem 4.1 and Theorem 4.2 hold in that generality. Our choice of restricting the functions $f_j(\cdot)$ to be sums $\sum_i f_j(x_i)$ presents a tradeoff. On the one hand, we believe that implementation of the primitive in a federated setting would be significantly easier under this additivity restriction. On the other hand, we will argue that a large class of differentially private algorithms of interest can be implemented even with this restriction. We detail these two arguments in Section 4 and Section 5.

We can also define a "batch" version of the SVT primitive, where a sequence of queries is given in advance and the index of the first YES, if any, is returned. This may be simpler to implement, as it does not need to retain internal state between API calls. In most settings, the batch and online versions are equivalent, except that the online version may be more communication efficient. One could also imagine algorithms that run multiple such SVT primitives in parallel, potentially with different parameters, and where composition could be analyzed using the concurrent composition theorems of [VW21, Lyu22, VZ23].

Implementation. The implementation of an SVT primitive will be a function of the specific trust model. A few different approaches have been studied for aggregation in the federated setting and we argue that these can accommodate the SVT primitive with relative ease.

The easiest approach is perhaps to use Trusted Execution Environments (TEEs), which have been proposed as an approach to implementing private federated algorithms in [JCM+24, ERB+24]. The simplicity of the SVT primitive makes it fairly well-suited for a TEE implementation, as it may be easy to verify the correctness of an implementation. While TEEs can in principle be used to implement general selection or even run an arbitrary central DP algorithm, the value of a minimal primitive like SVT is that it only needs to be audited and analyzed once. Running arbitrary analyst code inside the enclave makes the privacy guarantees depend on unaudited code, and both side-channel vulnerabilities and privacy budget state-continuity attacks [JCM+24] worsen as the trusted code base grows. In contrast, SVT's trusted state is a single scalar and its logic spans just a few lines of code, which is why we consider it a realistic and highly secure basis for solving more general problems.

A different approach to federated aggregation is to use the two-server or Prio model [CB17]. There is a large body of work on this model, and differentially private aggregation in this setup has been studied by [TWM+24]. In the two-party settings, noise addition requires some care: there are

multiple algorithms for collaborative noise addition, e.g., Franzese et al. [FFG+25], though a simple approach is for both servers to independently add scaled noise. Existing aggregation approaches already allow the two servers to very efficiently construct secret shares of the $f_j(\mathbf{x})$ values, as well as validate that each contributed $f_j(x_i)$ is in a bounded range. Distributed noise generation for γ and ν_j and the comparison can easily be implemented using garbled circuits and other standard techniques; indeed [BBCC+25] implement these ingredients in the same model.

An alternative approach, potentially combining trust models, is to use a TEE for the comparison part. This has the advantage that the input to the TEE, namely the secret shares of $f_j(\mathbf{x})$'s, does not scale with the population size n. Finally, one may consider an MPC approach to securely computing (shares of) the $f_j(\mathbf{x})$ values (e.g. [BIK+17b]). There are many approaches to secure noise generation in the MPC model, and the comparison operation can be implemented using general MPC. In all of these approaches, we exploit the fact that the computation of $f_j(\mathbf{x})$'s can be efficiently implemented using existing aggregation protocols, leaving a constant-size input to the comparison operator rather than one that grows with n. The noising and comparison operations are now a relatively simple circuit, which can be implemented using standard MPC tools.

Finally, we remark that SVT is significantly easier to implement securely than a general selection API. General selection requires arbitrary score functions as input, whereas each SVT query deals with a single score function at a time. This allows for much simpler cryptographic implementations; for example, if garbled circuits were used to implement selection, then Argmax would need to be securely solved over an arbitrary number of candidates. Indeed, [DKN+24] noted that "the bottleneck in the secure computation protocol is the comparisons required to compute argmax." Each candidate set size would require a different circuit, further making verification and auditing harder. Moreover, the secure part of the computation invariably adds non-trivial "constant" overheads. Because SVT processes a single score function at a time and returns a boolean threshold comparison, it allows the analyst to perform the Argmax outside of the cryptographic boundary, resulting in significant computational savings.

5 Selection in the SVT Model

In this section we give differentially private algorithms for the selection problem in the SVT model. We first describe a simple binary search over the range of the scores, which uses $O(d \log n)$ SVT queries but loses a factor of $\log n$ in the error. We then show that a descending linear scan achieves the optimal error using a single SVT stream but $O(dn)$ queries (Algorithm 7), and that allocating the privacy budget geometrically across the rounds of the binary search removes the $\log n$ factor from the error while keeping $O(d \log n)$ queries (Algorithm 8). Finally, in Appendix G.3 we combine sampling with a filtering pass to obtain the same error using $O(d \log^* n + \log^3 n)$ queries in expectation (Algorithm 9). All algorithms access the data only through SVT streams. Thus their privacy follows from Theorem 4.1 and basic composition. We analyze their accuracy on the event that all SVT answers are accurate. Throughout this section every score is a sum of per-user values in $[0, 1]$, as required by the SVT primitive. Scores with per-user values in $[0, \Delta]$ can be divided by Δ, which multiplies the error bounds by Δ.

The algorithm appears in full in Algorithm 3 and can be informally interpreted as a binary search over the set of possible scores for the index with the maximum score. We show in Appendix F that this approach can be used to achieve the following formal guarantees:

Theorem 5.1. *For $d \geq 2$, $n \geq 1$, $0 < \varepsilon \leq 1$, and $\beta \in (0, 1)$, Algorithm 3, with an equal share of the privacy budget for each SVT stream, is ε-DP and has additive error $O(\log(2n) \log(2d \log(2n)/\beta)/\varepsilon)$ with probability at least $1 - \beta$, using $O(d \log(2n))$ queries.*

Corollary 5.2. *Given an oracle $\mathbb{O}$ that is (α, β)-accurate for $\alpha = O\left(\frac{\log d}{\varepsilon}\right)$ and $\beta = \frac{1}{\text{poly}(d)}$, there exists an ε-DP algorithm that performs approximate maximization for decomposable score functions.*

Unfortunately, the additive error of this binary search approach is $O\left(\frac{\log d \log n}{\varepsilon}\right)$ rather than the optimal error of $O\left(\frac{\log d}{\varepsilon}\right)$ achievable in the central setting. This is because the binary search procedure may need to split the privacy budget small enough to invoke the SVT mechanism $O(\log n)$ times. Thus one may ask whether it is possible to achieve the optimal additive error of $O\left(\frac{\log d}{\varepsilon}\right)$.

Algorithm 3 Private binary search

Input: Oracle $\mathbb{O}$, initial bounds $[\ell, u] = [0, n]$ for non-negative scores, accuracy $\alpha \geq 1$; preserve the last positive index across recursive calls.
Output: Index j corresponding to approximately highest noisy score $s(v_j)$

$m \leftarrow \frac{u+\ell}{2}$
Query oracle $\mathbb{O}$ for index from $1, 2, \ldots, d$ with threshold m.
If returned index $j < d$, query oracle $\mathbb{O}$ for index from $j + 1, \ldots, d$ with threshold m.
if exactly one index in $[d]$ returned **then**
    Return that index ▷Only one positive answer; its score is within 2α of the maximum.
else if two indices returned **then**
    $\ell \leftarrow m - \alpha$ ▷Multiple scores $\geq m$
else
    $u \leftarrow m + \alpha$ ▷All noisy scores $\leq m$
if $u - \ell > 3\alpha$ **then**
    Recurse to find index for upper and lower limits $[\ell, u]$
else
    Return the last returned index, or arbitrary index if no indices returned.

Optimal additive error. To that end, we observe that a simple linear scan can achieve an optimal additive error. Namely, the protocol simply starts from the maximum possible score of an index and repeatedly calls the SVT API. For each query where the SVT API does not return a coordinate, the protocol decreases the guess for the maximum possible score of an index, repeating until the guess is within the typical bounds for Laplacian noise added to the score of the maximum coordinate. Since this approach will report only a single coordinate from the SVT API, it spends less privacy budget than the binary search approach, giving the following guarantees:

Theorem 5.3. *For $n \geq 1$, $d \geq 2$, and $\beta \in (0, 1)$, Algorithm 7 with $[\ell, u] = [0, n]$ and a single ε-DP SVT stream is an ε-DP selection algorithm with additive error $O(1 + \log(d(n+1)/\beta)/\varepsilon)$ with probability at least $1 - \beta$, using $O(d(n+1))$ queries.*

The downside is that to achieve the additive error $O(1 + \log(d(n+1)/\beta)/\varepsilon)$, the linear scan requires using $O(nd)$ queries. By comparison, binary search requires $O(d \log n)$ queries but has additive error $O(\log d \log n/\varepsilon)$. Therefore, a natural question is whether we can achieve near-optimal additive error with low query complexity.

Near-linear algorithm. Next, we summarize a more sophisticated algorithm that achieves the same optimal additive error while using only $O(d \log^* n + \log^3 n)$ SVT queries in expectation (Algorithm 9). Like the binary search with geometrically allocated budgets (Algorithm 8), it maintains an interval that contains the maximum score, together with a witness index whose score is at least the lower end of the interval. However, instead of halving the interval in every round (at the cost of d queries per round), it proceeds in *phases*. Each phase shrinks an interval of length $x\alpha$ to one of length $O(\alpha \log^3 x)$ and makes $O(d + \log^3 x)$ queries in expectation, where α is the target error (up to a constant factor). Thus $O(\log^* n)$ phases reduce the length to $O(\alpha)$. We then complete the selection by running the binary search with geometrically allocated budgets for a constant number of rounds.

In each phase we run a *filtering pass*, in which we query every candidate at most once against a fixed threshold. We then need to search only the candidates that answer YES. At the same time, we have to start a new SVT stream after every YES answer. Thus each candidate that passes the filter adds the privacy cost of one stream. To keep the number of such candidates small, we sample each candidate with probability $1/t$, for $t = \lceil \log_2 x \rceil$, and set the threshold slightly above an estimate of the maximum score in the sample. Except with probability e^{-t}, fewer than t^2 candidates have scores above the maximum score in the sample. Since the sample has about d/t candidates, we can estimate its maximum score by a binary search with t rounds using about d queries in expectation. We search the fewer than t^2 candidates that pass the filter in the same way, using at most t^3 queries.

We run both binary searches of a phase with privacy budget ε/t^3 and give the same budget to every stream of the filtering pass. Thus the new interval has length $O(\alpha t^3) = O(\alpha \log^3 x)$. With at most t^2 streams in the filtering pass, the total budget of a phase is $O(\varepsilon/t)$. Since the value of t at least halves from one phase to the next, the sum of these budgets over all phases is dominated by the budget of

the last phase. We need the privacy guarantee to hold even when some SVT answers are inaccurate. Therefore we fix the parameters of all phases in advance. We also cap the filtering pass at t^2 YES answers and fall back to searching all d candidates when the cap is reached (on the event that all SVT answers are accurate, this happens with probability at most e^{-t}). We prove in Appendix G.3 that this algorithm satisfies the following guarantees:

Theorem 5.4. *Let $n \geq 2$, $d \geq 1$, $\varepsilon \in (0, 1]$ and $\beta \in (0, 1/d]$. Algorithm 9 is ε-differentially private and, with probability at least $1 - \beta$, it outputs an index $j^\star$ such that*

$$\max_{j \in [d]} s_j(D) - s_{j^\star}(D) \leq \frac{776}{\varepsilon} \log \frac{384 d \log_2^3(2n)}{\beta} = O\left(\frac{1}{\varepsilon} \log \frac{d \log n}{\beta}\right).$$

The expected number of SVT queries made by the algorithm is $O(d \log^* n + \log^3 n)$. In particular, for $\beta = 1/(dn)$ the expected additive error is $O\left(\frac{\log(dn)}{\varepsilon}\right)$, and if $d \geq \log^3 n$ then the expected number of queries is $O(d \log^* d)$.

6 Applications in the SVT Model

Because private selection is a basic building block in data analysis, our near-linear optimal selection algorithm yields central-model-like performance across a range of downstream tasks. We highlight four representative applications where the SVT primitive improves over standard federated approaches; formal statements, algorithms, and proofs appear in Appendix B.

F_p loss minimization. Our framework also naturally extends to settings where the score for a coordinate $i \in [d]$ is given by an F_p moment or an ℓ_p norm over users. Each user $j \in [n]$ locally computes $w_i^{(j)} = (v_i^{(j)})^p$, so the global score reduces to a standard additive query. This enables our SVT API to privately identify the highest-loss coordinates with optimal additive error of $O\left(\frac{b^p \log d}{\varepsilon}\right)$ when all residuals satisfy $|v_i^{(j)}| \leq b$. This formulation captures several standard robust loss measures, including squared loss (F_2), absolute deviation (ℓ_1), and higher-order moments (F_p for $p > 2$), which act as smooth proxies for worst-case ℓ_∞ error. In federated learning, this allows systems to privately identify coordinate bottlenecks without scaling noise with the total number of parameters.

Heavy-hitters with categories. Consider n users, each holding an r-sparse binary vector over d categories, e.g., trending topics or products, and suppose we want to identify the top k heavy-hitters. Since a single user can affect up to r categories, the global L_1 sensitivity is r, and standard ε-DP histograms incur $\Omega(r)$ error per coordinate. Our SVT-based approach avoids this dependence on global sensitivity. Each category count has unit sensitivity, allowing us to query counts directly and iteratively recover the top k items. Applying composition over the k selections gives additive error $O(k \log d)$, eliminating the dependence on r.

MWEM for synthetic data. MWEM [HLM12] is a standard algorithm for privately generating synthetic datasets that approximately answer a set of queries Q over a domain $\mathcal{D}$, iteratively using the exponential mechanism to select a query $q \in Q$ with the largest error under the current synthetic data, and then updating the synthetic distribution. Our API directly implements this selection step in the distributed setting via decomposable score functions. As a result, we can run MWEM with utility matching the central model: maximum query error $\preceq n^{2/3} \left(\frac{\log |\mathcal{D}| \log |Q|}{\varepsilon}\right)^{1/3}$ under pure ε-DP, improving to $\preceq \sqrt{\frac{n}{\varepsilon}} \cdot \mathrm{polylog}\left(|\mathcal{D}|, |Q|, \frac{1}{\delta}\right)$ under (ε, δ)-DP.

Sparse mean estimation. Finally, consider estimating an unknown k-sparse mean vector $\mu \in \mathbb{R}^d$ ($k \ll d$) from n samples. A standard approach computes the empirical mean and keeps only the largest k coordinates in magnitude. Since entries may be unbounded, we first clip coordinate values to $O(\log(nd))$ and then use our SVT API to iteratively identify the top k coordinates, recovering the support of μ. This gives an (ε, δ)-DP estimator with additive error $O\left(\sqrt{k} \cdot \mathrm{polylog}(nd)\right)$ per non-zero coordinate, improving over the $O\left(\frac{\sqrt{d}}{\varepsilon}\right)$ error of standard aggregation by removing dependence on d.

Acknowledgments and Disclosure of Funding

Jelani Nelson's contribution to this project while at UC Berkeley was supported by NSF grant CCF-1951384, ONR grant N00014-18-1-2562, and ONR DORECG award N00014-17-1-2127. Huy L. Nguyen was supported in part by NSF CCF-2311649. Samson Zhou was supported in part by NSF CCF-2335411.

References

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[BBCC+25] Borja Balle, James Bell-Clark, Albert Cheu, Adria Gascon, Jonathan Katz, Mariana Raykova, Phillipp Schoppmann, and Thomas Steinke. Hash-prune-invert: Improved differentially private heavy-hitter detection in the two-server model. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2903–2918, 2025.

[BDG+13] Gilles Barthe, George Danezis, Benjamin Grégoire, César Kunz, and Santiago Zanella-Béguelin. Verified computational differential privacy with applications to smart metering. In *2013 IEEE 26th Computer Security Foundations Symposium*, pages 287–301, 2013.

[BEM+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

[BFG+16] Gilles Barthe, Noémie Fong, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Advanced probabilistic couplings for differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 55–67, 2016.

[BFJ+94] Avrim Blum, Merrick L. Furst, Jeffrey C. Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 253–262, 1994.

[BGA+14] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, César Kunz, and Pierre-Yves Strub. Proving differential privacy in hoare logic. In *IEEE 27th Computer Security Foundations Symposium, CSF*, pages 411–424, 2014.

[BGD+18] Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin T. Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 508–524, 2018.

[BGG+16] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 749–758, 2016.

[BGM+16] Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1011–1020, 2016.

[BHNS20] Amos Beimel, Iftach Haitner, Kobbi Nissim, and Uri Stemmer. On the round complexity of the shuffle model. In *Theory of Cryptography Conference*, pages 683–712. Springer, 2020.

[BIK+17a] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[BIK+17b] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[BO13] Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP, Proceedings, Part II*, pages 49–60, 2013.

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of cryptography conference*, pages 635–658. Springer, 2016.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, page 464–473, USA, 2014. IEEE Computer Society.

[BSU17] Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: the price of online queries in differential privacy. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, page 1306–1325, USA, 2017. Society for Industrial and Applied Mathematics.

[BV19] Victor Balcer and Salil Vadhan. Differential privacy on finite computers. *Journal of Privacy and Confidentiality*, 9(2), Sep. 2019.

[CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, pages 259–282, 2017.

[CGKM21] Lijie Chen, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. On distributed differential privacy and counting distinct elements. In *12th Innovations in Theoretical Computer Science Conference, ITCS*, pages 56:1–56:18, 2021.

[CKS22] Clement Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Journal of Privacy and Confidentiality*, 12(1), Jul. 2022.

[CSU+19] Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part I*, pages 375–403, 2019.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2 edition, 2006.

[CU21] Albert Cheu and Jonathan R. Ullman. The limits of pan privacy and shuffle privacy for learning and estimation. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1081–1094, 2021.

[DF20] Yuval Dagan and Vitaly Feldman. Interaction is necessary for distributed learning with privacy or communication constraints. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 450–462, 2020.

[DJW13] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 429–438, 2013.

[DJW18] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, pages 486–503, 2006.

[DKN+24] Ivan Damgård, Hannah Keller, Boel Nelson, Claudio Orlandi, and Rasmus Pagh. Differentially private selection from secure distributed computing. In Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, {Hady W.} Lauw, and {Roy Ka-Wei} Lee, editors, *WWW 2024 - Proceedings of the ACM Web Conference*, pages 1103–1114, United States, May 2024. Association for Computing Machinery. 33rd ACM Web Conference, WWW 2024 ; Conference date: 13-05-2024 Through 17-05-2024.

[DMNS17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3):17–51, 2017.

[DNR+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 381–390, 2009.

[DR+14] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3–4):211–407, 2014.

[DR19] John Duchi and Ryan Rogers. Lower bounds for locally private estimation via communication complexity. In *Conference on Learning Theory*, pages 1161–1191. PMLR, 2019.

[EFM+19] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2468–2479, 2019.

[EGS03] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 211–222, 2003.

[ERB+24] Hubert Eichner, Daniel Ramage, Kallista Bonawitz, Dzmitry Huba, Tiziano Santoro, Brett McLarnon, Timon Van Overveldt, Nova Fallen, Peter Kairouz, Albert Cheu, Katharine , Daly, Adria Gascon, Marco Gruteser, and Brendan McMahan. Confidential federated computations. *arXiv preprint arXiv:2404.10764*, 2024.

[Fel14] Vitaly Feldman. Open problem: The statistical query complexity of learning sparse halfspaces. In *Proceedings of The 27th Conference on Learning Theory, COLT*, pages 1283–1289, 2014.

[FES+17] David Froelicher, Patricia Egger, João Sá Sousa, Jean Louis Raisaro, Zhicong Huang, Christian Mouchet, Bryan Ford, and Jean-Pierre Hubaux. UnLynx: A decentralized system for privacy-conscious data sharing. In *Proceedings on Privacy Enhancing Technologies*, pages 232—-250, 2017.

[FFG+25] Olive Franzese, Congyu Fang, Radhika Garg, Xiao Wang, Somesh Jha, Nicolas Papernot, and Adam Dziedzic. Secure noise sampling for differentially private collaborative learning. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 4649–4663, 2025.

[FGR+17] Vitaly Feldman, Elena Grigorescu, Lev Reyzin, Santosh S Vempala, and Ying Xiao. Statistical algorithms and a lower bound for detecting planted cliques. *Journal of the ACM (JACM)*, 64(2):1–37, 2017.

[FS17] Vitaly Feldman and Thomas Steinke. Generalization for adaptively-chosen estimators via stable median. In *Conference on learning theory*, pages 728–757. PMLR, 2017.

[FZ21] Vitaly Feldman and Tijana Zrnic. Individual privacy accounting via a renyi filter. *Advances in Neural Information Processing Systems*, 34:28080–28091, 2021.

[GFHY23] Ji Guan, Wang Fang, Mingyu Huang, and Mingsheng Ying. Detecting violations of differential privacy for quantum algorithms. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 2277–2291, 2023.

[GHH+13] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL*, pages 357–370, 2013.

[HLM12] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings*, pages 2348–2356, 2012.

[HR10] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 61–70, 2010.

[Jay09] T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of and. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, page 562–573. Springer-Verlag, 2009.

[JCM+24] Jiankai Jin, Chitchanok Chuengsatiansup, Toby Murray, Benjamin I. P. Rubinstein, Yuval Yarom, and Olga Ohrimenko. Elephants do not forget: Differential privacy with state continuity for privacy budget. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1909–1923, New York, NY, USA, 2024. Association for Computing Machinery.

[JMNR19] Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The role of interactivity in local differential privacy. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 94–105. IEEE, 2019.

[KLN+11] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

[LM25] Ethan Leeman and Pasin Manurangsi. Nearly-optimal private selection via gaussian mechanism, 2025.

[Lyu22] Xin Lyu. Composition theorems for interactive differential privacy. *Advances in Neural Information Processing Systems*, 35:9700–9712, 2022.

[McS10] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, 2010.

[Mir12] Ilya Mironov. On significance of the least significant bits for differential privacy. In *the ACM Conference on Computer and Communications Security, CCS'12*, pages 650–661, 2012.

[Mir17] Ilya Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF*, pages 263–275, 2017.

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

[RCWH+20] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Cryptε: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 603–619, New York, NY, USA, 2020. Association for Computing Machinery.

[RNFH19] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. Honeycrisp: Large-scale differentially private aggregation without a trusted core. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 196–210, New York, NY, USA, 2019. Association for Computing Machinery.

[RSK+10] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, pages 297–312, 2010.

[RZHP20] Edo Roth, Hengchu Zhang, Andreas Haeberlen, and Benjamin C. Pierce. Orchard: Differentially private analytics at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 1065–1081. USENIX Association, November 2020.

[Ste25] Thomas Steinke. Open problem: Selection via low-sensitivity queries. DifferentialPrivacy.org, 05 2025. https://differentialprivacy.org/open-problem-selection/.

[Tea20] The OpenDP Team. The opendp white paper, 2020.

[TWM+24] Kunal Talwar, Shan Wang, Audra McMillan, Vitaly Feldman, Pansy Bansal, Bailey Basile, Aine Cahill, Yi Sheng Chan, Mike Chatzidakis, Junye Chen, Oliver R. A. Chick, Mona Chitnis, Suman Ganta, Yusuf Goren, Filip Granqvist, Kristine Guo, Frederic Jacobs, Omid Javidbakht, Albert Liu, Richard Low, Dan Mascenik, Steve Myers, David Park, Wonhee Park, Gianni Parsa, Tommy Pauly, Christian Priebe, Rehan Rishi, Guy N. Rothblum, Congzheng Song, Linmao Song, Karl Tarbe, Sebastian Vogt, Shundong Zhou, Vojta Jina, Michael Scaria, and Luke Winstrom. Samplable anonymous aggregation for private federated data analysis. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 2859–2873, New York, NY, USA, 2024. Association for Computing Machinery.

[Ull18] Jonathan Ullman. Tight lower bounds for locally differentially private selection. *arXiv preprint arXiv:1802.02638*, 2018.

[VW21] Salil Vadhan and Tianhao Wang. Concurrent composition of differential privacy. In *Theory of Cryptography Conference*, pages 582–604. Springer, 2021.

[VZ23] Salil P. Vadhan and Wanrong Zhang. Concurrent composition theorems for differential privacy. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC*, pages 507–519, 2023.

[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.

[WZL+20] Royce J. Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private SQL with bounded user contribution. *Proc. Priv. Enhancing Technol.*, 2020(2):230–250, 2020.

[ZK17] Danfeng Zhang and Daniel Kifer. Lightdp: towards automating differential privacy proofs. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL*, pages 888–901, 2017.

A Preliminaries

We use the notation $[n]$ for a positive integer $n > 0$ to denote the set $\{1, 2, \ldots, n\}$. For functions $f(n), g(n)$, we sometimes use the notation $f(n) \preceq g(n)$ in conjunction with $f(n) = O(g(n))$, i.e., that there exists a constant $C \geq 0$ such that $f(x) \leq C \cdot g(x)$ for all $x \geq n$ for a sufficiently large n.

Differential Privacy [DMNS17]

Definition A.1. *Let M be a randomized algorithm mapping a dataset $\mathbf{x} \in \mathcal{X}^n$ to a range $\mathcal{R}$. We say two datasets $\mathbf{x}, \mathbf{x}'$ are neighboring if $x_i = x'_i$ for all but one of the n coordinates. The mechanism M is (ε, δ)-differentially private if for any pair of neighboring datasets $\mathbf{x}, \mathbf{x}'$, and for any event $E \subseteq \mathcal{R}$, it is the case that*

$$\Pr[M(\mathbf{x}) \in E] \leq e^{\varepsilon} \cdot \Pr[M(\mathbf{x}') \in E] + \delta.$$

When an algorithm is $(\varepsilon, 0)$-differentially private, we often call it ε-differentially private.

Shuffle model. In the shuffle model, each individual user first randomizes their own data into one or more messages. These messages are then sent to a secure shuffler, which aggregates and randomly permutes all user messages. The shuffle model considers an attacker who only sees the collection of messages after the permutation and thus, the set of shuffled messages must satisfy differential privacy. Equivalently, an attacker can only observe the histogram of the messages. Formally, we have the following definition:

Definition A.2 (Shuffle model of differential privacy)**.** *In the shuffle model of privacy, a protocol Π consists of three randomized algorithms:*

- *A randomizer $\Pi_R : \mathcal{X} \to \mathcal{Y}^*$ that maps data to vectors, possibly of variable length, which denotes the number of messages sent. The protocol is said to be single-message if the probability of sending a single message is 1 on all inputs; otherwise it is said to be multi-message.*
- *A shuffler $\Pi_S : \mathcal{Y}^* \to \mathcal{Y}^*$ that applies a uniformly random permutation to all messages.*
- *An analyzer $\Pi_A : \mathcal{Y}^* \to O$ that performs computation on a set of messages (after permutation).*

Since the shuffler is the same across all protocols, we use $\Pi = (\Pi_R, \Pi_A)$ to denote each shuffle protocol Π, so that on an input $x \in \mathcal{X}^n$, the output is $\Pi(x) = \Pi_A(\Pi_S(\Pi_R(x_1), \ldots, \Pi_R(x_n)))$. Hence, we denote the output of the shuffler by $(\Pi_S \circ \Pi_R^n)(x) := \Pi_S(\Pi_R(x_1), \ldots, \Pi_R(x_n))$.

We remark on two existing lower bounds for selection in the shuffle model of privacy. We first recall the following lower bound by [CGKM21]:

Theorem A.3 (Theorem 1.9 in [CGKM21])**.** *Any (ε, δ)-shuffle DP protocol that allows each user to send up to m messages and solves the selection problem on d-dimensional inputs with additive error $\frac{1}{10}$ in the average scores, with constant success probability, must use $\Omega\left(\frac{d}{m}\right)$ samples, for any $\varepsilon = O(1)$ and $\delta = o\left(\frac{1}{d}\right)$.*

Theorem A.4 (Theorem 7.3 in [CU21])**.** *Any (ε, δ)-shuffle DP protocol that remains private when at least a $\frac{1}{3}$-fraction of users follow the protocol and solves the selection problem on d-dimensional inputs with additive error α in the population means and success probability at least $99/100$ must use $\Omega\left(\frac{\sqrt{d}}{\alpha\varepsilon}\right)$ samples for $\delta \log \frac{d}{\delta} \ll \frac{\alpha^2\varepsilon^2}{d}$.*

B Applications in the SVT Model

Because private selection appears in many settings, Theorem 2.2 has a wide range of applications. In this section, we present several examples demonstrating how the SVT primitive substantially improves upon existing models.

F_p losses. We first note that our framework extends naturally to the setting where the score function of a coordinate $i \in [d]$ is either the F_p moment or the L_p norm of the individual components. For $p \geq 1$, in the first case we have $s(i) = \sum_{j\in[n]} |v_i^{(j)}|^p$, and in the second case we have $s(i) = \left(\sum_{j\in[n]} |v_i^{(j)}|^p\right)^{1/p}$. Observe that each user $j \in [n]$ can compute the vector $w^{(j)}$, defined so that $w_i^{(j)} = |v_i^{(j)}|^p$ for all $i \in [d]$. Then the F_p score is $s(i) = \sum_{j\in[n]} w_i^{(j)}$. If all the residuals satisfy $|v_i^{(j)}| \leq b$, then every $w_i^{(j)}$ lies in $[0, b^p]$ and the users can run our selection algorithm on the vectors $w^{(1)}/b^p, \ldots, w^{(n)}/b^p$, thereby achieving additive error $O\left(\frac{b^p \log d}{\varepsilon}\right)$ for the F_p score (when n is polynomial in d). Since $(y-a)^{1/p} \geq y^{1/p} - a^{1/p}$ for all $0 \leq a \leq y$, an additive error a for the F_p score implies additive error at most $a^{1/p}$ for the L_p score. This formulation captures a number of standard loss measures, including:

- *Squared Euclidean loss, i.e., F_2 loss:* If $v_i^{(j)}$ represents a residual or prediction error, then $s(i) = \sum_{j\in[n]} (v_i^{(j)})^2$ measures the total squared error at coordinate i. As the score function is the squared ℓ_2 norm of the vector $(v_i^{(1)}, \ldots, v_i^{(n)})$, it penalizes large deviations more heavily than small ones and thus selection based on $s(i)$ identifies coordinates where the error vector has large length. This loss is widely used in regression, least-squares estimation, and many machine learning tasks.
- *Absolute deviation, i.e., F_1 or L_1 loss:* Taking $p = 1$ results in the score function $s(i) = \sum_{j\in[n]} |v_i^{(j)}|$, a standard robust alternative to squared loss. The L_1 loss is less sensitive to outliers and is commonly used in robust estimation and sparse recovery when occasional large deviations are expected.
- *Higher-order moments, i.e., F_p loss for $p > 2$:* For larger p, the score $s(i) = \sum_{j\in[n]} |v_i^{(j)}|^p$ emphasizes rare but large deviations. The corresponding L_p score $s(i) = \left(\sum_{j\in[n]} |v_i^{(j)}|^p\right)^{1/p}$ serves as a smooth proxy for the ℓ_∞ loss $s_\infty(i) = \max_{j\in[n]} |v_i^{(j)}|$, which captures worst-case error. This is useful in applications requiring uniform accuracy or robustness guarantees, while using finite p retains computational feasibility in many applications. For instance, in distributed or federated learning, one might want to identify coordinates where the largest per-user errors occur, ensuring that no single coordinate becomes a bottleneck or source of extreme error. Similarly, in sensor networks or traffic monitoring, large F_p or L_p scores correspond to coordinates with unusually high variability or spikes, enabling anomaly detection and more robust decision-making.

Heavy-hitters with categories. In many real-world datasets, items are associated with multiple categories, forming sparse binary vectors where each item is represented by a vector indicating its presence across various categories. The goal is to identify the top k categories that contain the most items. This problem arises in a wide range of applications. For instance, in e-commerce, determining the most popular product categories can inform inventory management and marketing strategies. In social media, identifying trending topics or hashtags helps improve content curation and user engagement. Similarly, in healthcare, analyzing patient data to find prevalent disease categories can aid in resource allocation and treatment planning. Efficiently solving this problem is particularly relevant in these applications where both accuracy and privacy are important.

Formally, suppose there are n items, which can each be mapped to up to r categories in a universe of d categories. In other words, there are n vectors that are each r-sparse binary vectors $v^{(1)}, \ldots, v^{(n)} \in \{0,1\}^d$ such that $\|v^{(i)}\|_0 \leq r$ for all $i \in [n]$. The goal is to find the top k categories with the largest number of items. Specifically, the goal is to find the largest k coordinates of the vector $v = \sum_{i\in[n]} v^{(i)} \in \mathbb{R}^d$.

We first observe that because each item can be in up to r categories, the L_1 sensitivity of the count vector v is $2r$ (when one item is replaced). Thus the standard approach of releasing all d counts with Laplace noise calibrated to this sensitivity adds noise of scale $2r/\varepsilon$ to every count, and the resulting error in the largest counts is $\Theta(r \log d/\varepsilon)$ with constant probability. On the other hand, each individual count changes by at most 1 when an item is replaced. Therefore we can apply our

selection algorithm directly to the score functions $f_j(v^{(i)}) = v_j^{(i)}$ and avoid the dependence on r. Applying Theorem 5.4 k times, each time with privacy budget ε/k, failure probability $1/d$ and with the previously selected categories removed from the candidate set, we obtain the following guarantee.

Theorem B.1. *Let $\varepsilon \in (0, 1]$, $n \geq 2$ and $k \leq d$. There is an ε-DP algorithm in the SVT model that outputs distinct categories $j_1, \ldots, j_k$ such that, with probability at least $1 - k/d$, for every $t \in [k]$ the count v_{j_t} is at least the t-th largest count minus $O\left(\frac{k \log(d \log n)}{\varepsilon}\right)$. The algorithm makes $O(k(d \log^* n + \log^3 n))$ SVT queries in expectation.*

Proof. In round t the candidate set excludes only the $t - 1$ previously selected categories and thus its maximum count is at least the t-th largest count overall. By Theorem 5.4 with budget ε/k and failure probability $1/d$, with probability at least $1 - 1/d$ the selected category has count within $O(k \log(d \log n)/\varepsilon)$ of this maximum. The claim follows by a union bound over the k rounds and basic composition. □

Multiplicative weights and exponential mechanism (MWEM). We next consider MWEM, a framework introduced by [HLM12] based on multiplicative weights and the exponential mechanism, for privately producing synthetic datasets that accurately answer any set of queries on a protected dataset. In particular, we assume there exists a set Q of queries q, where each query $q : \mathcal{D} \to [-1, +1]$ is a function from the record domain $\mathcal{D}$ to the interval $[-1, +1]$, and extended to datasets A by

$$q(A) = \sum_{x \in \mathcal{D}} q(x) \cdot A(x).$$

The goal is to produce a synthetic dataset B such that $q(B) \approx q(A)$ for all $q \in Q$.

We first recall a standard approach toward achieving this goal in the central model of differential privacy, as depicted in Algorithm 10. The algorithm, known as MWEM, maintains a synthetic dataset that is initialized uniformly over the domain $\mathcal{D}$ and is iteratively refined. At each iteration, it privately selects a query whose current synthetic answer has the largest deviation from the true dataset, measures the true answer with added Laplace noise for privacy, and updates the synthetic dataset multiplicatively to better match this noisy measurement. After T iterations, the resulting averaged synthetic dataset A approximates the answers to all queries in Q while preserving differential privacy.

Theorem B.2. *For any private dataset B with n records, finite query set Q, privacy parameter $\varepsilon \in (0, 1]$ and $\beta \in (0, 1)$, there exists an ε-differentially private algorithm in the SVT model that releases a synthetic dataset A such that with probability at least $1 - \beta$, we have*

$$\max_{q \in Q} |q(A) - q(B)| \preceq n^{2/3} \cdot \left(\frac{\log |\mathcal{D}| \cdot \log(n|Q|/\beta)}{\varepsilon}\right)^{1/3}.$$

In fact, we can achieve better utility bounds if we permit (ε, δ)-differential privacy for $0 < \varepsilon \leq 1$ and $\delta \in (0, 1/2]$, by a similar argument, allowing additional privacy loss per time step and instead using advanced composition: each of the $2T$ private steps can use privacy parameter $\Theta\left(\varepsilon/\sqrt{T \log(1/\delta)}\right)$, which replaces the factor T in η' by $O(\sqrt{T \log(1/\delta)})$, and balancing the two terms of Lemma H.6 then gives $T \asymp n\varepsilon\sqrt{\log |\mathcal{D}|/\log(1/\delta)}/\log(n|Q|/\beta)$:

Theorem B.3. *For any private dataset B with n records, query set Q, privacy parameters $0 < \varepsilon \leq 1$ and $\delta \in (0, 1/2]$, and $\beta \in (0, 1)$, there exists an (ε, δ)-differentially private algorithm in the SVT model that releases a synthetic dataset A such that with probability at least $1 - \beta$, we have*

$$\max_{q \in Q} |q(A) - q(B)| \preceq \sqrt{\frac{n}{\varepsilon}} \cdot \operatorname{polylog}\left(|\mathcal{D}|, |Q|, n, \frac{1}{\delta}, \frac{1}{\beta}\right).$$

Sparse mean estimation. Let $X_1, \ldots, X_n \in \mathbb{R}^d$ be independent samples from $\mathcal{N}(\mu, I_d)$, where the mean μ is k-sparse (that is, $\|\mu\|_0 \leq k$ for some $k \ll d$) and satisfies $\|\mu\|_\infty \leq 1$. The goal is to estimate μ. Without privacy, the standard estimator computes the empirical mean and zeroes out all but its k entries of largest magnitude, which gives ℓ_2 error $O(\sqrt{k \log(d)/n})$ with high probability.

We assume Gaussian data and a known bound on the entries of μ for simplicity of presentation. They allow us to clip the entries of the samples to a bounded range that, with high probability, does not change the samples. Some bound on the range of μ is necessary for this since the SVT API only accepts bounded contributions. Both assumptions can be relaxed using standard techniques (for example, by first computing a coarse private estimate of the range of every coordinate).

Our estimator follows the non-private one. We first privately select k coordinates whose empirical sums have the largest magnitude, and then privately estimate the sums of the selected coordinates. Both steps use only sums of bounded functions of the samples, which we obtain by clipping every entry to the interval $[-B, B]$ for $B = 1 + \sqrt{2\log(8nd/\beta)}$, where β is the failure probability. We denote the clipped samples by $\widetilde{X}_1, \ldots, \widetilde{X}_n$.

To select coordinates we use the $2d$ score functions $f_{j,\sigma}(\widetilde{X}_i) = \frac{B+\sigma\widetilde{X}_{ij}}{2B} \in [0,1]$ for $j \in [d]$ and $\sigma \in \{-1,+1\}$, so that $\max_\sigma \sum_i f_{j,\sigma}(\widetilde{X}_i) = \frac{n}{2} + \frac{1}{2B}\left|\sum_i \widetilde{X}_{ij}\right|$. We select k coordinates in k rounds. In every round we run Algorithm 9 with privacy budget $\varepsilon/(2k)$ and failure probability $\min\{\beta/(4k), 1/(2d)\}$, and then remove both score functions of the selected coordinate from the candidate set. We denote the set of selected coordinates by $\widehat{S}$. For every $j \in \widehat{S}$, we then estimate $\sum_i \widetilde{X}_{ij}$ by applying Corollary G.5 to the function $g_j(\widetilde{X}_i) = \frac{B+\widetilde{X}_{ij}}{2B}$ with privacy budget $\varepsilon/(2k)$ and failure probability $\beta/(4k)$. If $\hat{g}_j$ is the midpoint of the returned interval, we output $\widehat{\mu}_j = \frac{B(2\hat{g}_j - n)}{n}$. For $j \notin \widehat{S}$ we output $\widehat{\mu}_j = 0$. By basic composition, this algorithm is ε-differentially private.

Theorem B.4. *Let $\varepsilon \in (0,1]$, $\beta \in (0,1)$, $n \geq 2$ and $k \leq d$. The algorithm above is ε-DP in the SVT model, makes $O(k(d\log^* n + \log^3 n))$ SVT queries in expectation and, with probability at least $1-\beta$, its output satisfies*

$$\|\widehat{\mu}-\mu\|_\infty = O\left(\sqrt{\frac{\log(d/\beta)}{n}} + \frac{k\sqrt{\log(nd/\beta)}\,\log\frac{d\log n}{\beta}}{n\varepsilon}\right) \qquad \text{and} \qquad \|\widehat{\mu}-\mu\|_2 \leq \sqrt{2k}\,\|\widehat{\mu}-\mu\|_\infty.$$

For $\delta \in (0, 1/2]$, using advanced composition instead of basic composition, the factor k in the second term can be replaced by $O(\sqrt{k\log(1/\delta)})$ at the cost of (ε,δ)-DP.

The analysis follows the non-private one. With high probability no entry is clipped and all the empirical sums are within $A = \sqrt{2n\log(8d/\beta)}$ of their expectations. Therefore every selected coordinate is estimated with error $O(A)$ plus the error of the private estimation. A coordinate in the support of μ that is not selected must have a small mean, since a coordinate outside of the support, whose empirical sum has magnitude at most A, was selected instead.

Proof. Let $S_j = \sum_{i\in[n]} X_{ij}$. Since $S_j - n\mu_j \sim \mathcal{N}(0,n)$ and $X_{ij} - \mu_j \sim \mathcal{N}(0,1)$, standard Gaussian tail bounds and a union bound imply that, with probability at least $1-\beta/2$, $|S_j - n\mu_j| \leq A$ for all $j \in [d]$ and $\widetilde{X}_i = X_i$ for all $i \in [n]$. By Theorem 5.4 and a union bound over the k rounds, with probability at least $1-\beta/4$, the coordinate j_t selected in round t satisfies $|S_{j_t}| \geq |S_j| - E_1$ for every coordinate j that was not selected before round t, where $E_1 = O\left(\frac{kB}{\varepsilon}\log\frac{d\log n}{\beta}\right)$ is $2B$ times the error bound of Theorem 5.4. By Corollary G.5, with probability at least $1-\beta/4$ we have $|n\widehat{\mu}_j - S_j| \leq E_2$ for all $j \in \widehat{S}$, where $E_2 = O\left(\frac{kB}{\varepsilon}\log\frac{k\log n}{\beta}\right)$. We assume that all these events hold, which happens with probability at least $1-\beta$.

For $j \in \widehat{S}$ we have $|\widehat{\mu}_j - \mu_j| \leq (|n\widehat{\mu}_j - S_j| + |S_j - n\mu_j|)/n \leq (E_2 + A)/n$. Now let j^* be a coordinate in the support of μ that is not in $\widehat{S}$. Since $|\widehat{S}| = k \geq \|\mu\|_0$, some selected coordinate j' is outside of the support of μ, and j^* was a candidate in the round in which j' was selected. Therefore

$$A \geq |S_{j'}| \geq |S_{j^*}| - E_1 \geq n|\mu_{j^*}| - A - E_1,$$

that is, $|\widehat{\mu}_{j^*} - \mu_{j^*}| = |\mu_{j^*}| \leq (2A + E_1)/n$. All other coordinates satisfy $\widehat{\mu}_j = \mu_j = 0$. This gives the bound on $\|\widehat{\mu} - \mu\|_\infty$ since $B = O(\sqrt{\log(nd/\beta)})$. The bound on $\|\widehat{\mu} - \mu\|_2$ follows since $\widehat{\mu} - \mu$ is supported on at most $2k$ coordinates. Finally, the bound on the number of queries follows from Theorem 5.4 and Corollary G.5. □

C Proof of the Lower Bound for Selection with Gaussian Aggregation

Our lower bound relies on a lower bound for solving the problem in the standard setting of distributed multi-party protocols. Specifically, we use the standard number-in-hand blackboard communication model. Each user initially knows only its own input and its private randomness. The users communicate by writing messages to a common transcript, where each message may depend on the sender's input, the sender's private randomness, public randomness, and the previous transcript. The full transcript is denoted by M; we include public randomness and any randomness used for the final decision in M, so the output $\widehat{Z}$ is a function of M. Throughout this appendix, $B(2)$ denotes the uniform distribution on $\{1,2\}$.

Theorem C.1. *For every $2/3$-accurate protocol for Problem 3.3, its transcript M satisfies*

$$I\left(M; X^{\le n} \,\middle|\, V, Z\right) = \Omega(d).$$

Our proof follows the approach in [BGM+16], which established a lower bound for the sparse Gaussian mean estimation problem (see also [DR19]). The key lemma in their result is Jayram's Hellinger cut-and-paste inequality that we state below, where h denotes Hellinger distance. We use the normalization $h^2(P,Q) = \frac{1}{2}\int(\sqrt{p}-\sqrt{q})^2$ and natural logarithms in the KL divergence. With this normalization, for $R = (P+Q)/2$ we have $\mathrm{KL}(P\|R) \ge 2h^2(P,R)$ and therefore $\frac{1}{2}\mathrm{KL}(P\|R) + \frac{1}{2}\mathrm{KL}(Q\|R) \ge h^2(P,R) + h^2(Q,R) \ge \frac{1}{2}\left(h(P,R)+h(Q,R)\right)^2 \ge \frac{1}{2}h^2(P,Q)$, where the last step is the triangle inequality for h.

Lemma C.2 (Jayram's cut-and-paste inequality [Jay09, Theorem 7]; see also [BGM+16, Theorem A.1])**.** *There is an absolute constant $C > 0$ such that the following holds. Consider any n-user blackboard protocol. For each user i, let ν_1^i, ν_2^i be two distributions over that user's input space. For $r \in \{1,2\}^n$, let P_r be the transcript distribution when the users' inputs are sampled independently as $X^i \sim \nu_{r_i}^i$. Then, for any $a, b \in \{1,2\}^n$,*

$$\sum_{i=1}^{n} h^2\left(P_{a^{i\leftarrow b_i}}, P_a\right) \ge C \cdot h^2\left(P_b, P_a\right).$$

Proof of Theorem C.1. Let $X_j^{\le n} = (X_j^1, \ldots, X_j^n)$. We first relate the information in the randomly chosen coordinate to the information in the full input under $Z = 2$.

$$\begin{aligned} I\left(M; X_V^{\le n} \,\middle|\, V, Z=2\right) &= \frac{1}{d}\sum_{j=1}^{d} I\left(M; X_j^{\le n} \,\middle|\, V=j, Z=2\right) \\ &= \frac{1}{d}\sum_{j=1}^{d} I\left(M; X_j^{\le n} \,\middle|\, Z=2\right) \\ &\le \frac{1}{d} I\left(M; X^{\le n} \,\middle|\, Z=2\right). \end{aligned} \tag{1}$$

The second equality follows from the fact that when $Z = 2$, the transcript and inputs are independent of V. The inequality follows from the standard chain rule for mutual information [CT06, Section 2.5]: if $Y_1, \ldots, Y_m$ are mutually independent conditioned on W, then $\sum_{\ell=1}^{m} I\left(Y_\ell; M \,|\, W\right) \le I\left(Y_1, \ldots, Y_m; M \,|\, W\right)$. We apply this with the independent coordinates $X_1^{\le n}, \ldots, X_d^{\le n}$ conditioned on $Z = 2$.

Applying the same property to the independent user coordinates $X_V^1, \ldots, X_V^n$ conditioned on $(V, Z = 2)$,

$$I\left(M; X_V^{\le n} \,\middle|\, V, Z=2\right) \ge \sum_{i=1}^{n} I\left(M; X_V^i \,\middle|\, V, Z=2\right). \tag{2}$$

Next fix a user $i \in \{1, \ldots, n\}$ and a coordinate $V = j$, and analyze $I\left(M; X_V^i \,\middle|\, V=j, Z=2\right)$. Let b be a vector in $\{1,2\}^n$. Let M_b denote the distribution of the transcript M when, for every user k, the input coordinate X_j^k is sampled according to state b_k; that is, if $b_k = 1$ then $X_j^k = 1$, while if

$b_k = 2$ then X_j^k is sampled uniformly from $\{0,1\}$. All other coordinates are sampled independently and uniformly from $\{0,1\}$. Let D_b denote the corresponding distribution of the full inputs $X^{\le n}$.

First consider the setting where $Z_i \sim B(2)$, user i receives the input X_j^i sampled according to Z_i, and the remaining $n-1$ users still receive inputs sampled using $Z = 2$. Since all other coordinates are uniform independently of Z_i, we have the Markov chain $Z_i \to X_j^i \to M$ in this experiment. By data processing and the comparison between the Jensen-Shannon divergence and the Hellinger distance,

$$\begin{aligned}
&I_{Z_i\sim B(2),X^{\le n}\sim D_{\bar{2}^{i\leftarrow Z_i}}}\left(M; X_j^i \,\middle|\, V=j\right)\\
&\ge I_{Z_i\sim B(2),X^{\le n}\sim D_{\bar{2}^{i\leftarrow Z_i}}}\left(M; Z_i \,\middle|\, V=j\right)\\
&= \mathop{\mathbf{E}}_{z\sim B(2)}\left[\mathrm{KL}\left(M_{\bar{2}^{i\leftarrow z}} \,\middle\|\, \mathop{\mathbf{E}}_{Z_i} M_{\bar{2}^{i\leftarrow Z_i}}\right)\right]\\
&= \frac{1}{2}\mathrm{KL}\left(M_{\bar{2}^{i\leftarrow 1}} \,\middle\|\, \frac{1}{2}M_{\bar{2}^{i\leftarrow 1}} + \frac{1}{2}M_{\bar{2}}\right) + \frac{1}{2}\mathrm{KL}\left(M_{\bar{2}} \,\middle\|\, \frac{1}{2}M_{\bar{2}^{i\leftarrow 1}} + \frac{1}{2}M_{\bar{2}}\right)\\
&\ge \tfrac{1}{2}h^2(M_{\bar{2}^{i\leftarrow 1}}, M_{\bar{2}})\,.
\end{aligned} \tag{3}$$

Observe that, in the experiment with $Z_i \sim B(2)$, $\Pr\left[X_j^i = x \,\middle|\, Z_i = 2\right] \ge \frac{2}{3}\Pr\left[X_j^i = x\right]$ for all $x \in \{0,1\}$, so we can apply the following lemma.

Lemma C.3 ([BGM+16, Lemma 5])**.** *Suppose two distributions μ, μ' satisfy $\mu \ge c \cdot \mu'$. Let $\Pi(X)$ be a random function that only depends on X. If $X \sim \mu$ and $X' \sim \mu'$, then*

$$I(X; \Pi(X)) \ge c \cdot I(X'; \Pi(X')).$$

Applying this lemma gives

$$I_{X^{\le n}\sim D_{\bar{2}}}\left(M; X_j^i \,\middle|\, V=j, Z=2\right) \ge \frac{2}{3} I_{Z_i\sim B(2),X^{\le n}\sim D_{\bar{2}^{i\leftarrow Z_i}}}\left(M; X_j^i \,\middle|\, V=j\right). \tag{4}$$

We next use Lemma C.2 to pass from changing one user to changing all users. For the fixed coordinate j, apply Lemma C.2 with $a = \bar{2}$ and $b = \bar{1}$ to the product input distributions defining M_b. This gives

$$\sum_{i=1}^{n} h^2(M_{\bar{2}^{i\leftarrow 1}}, M_{\bar{2}}) \ge C \cdot h^2(M_{\bar{1}}, M_{\bar{2}}). \tag{5}$$

Taking expectation over random $V = j$ and combining the above inequalities, we get

$$\begin{aligned}
\mathop{\mathbf{E}}_{V}\left[C \cdot h^2(M_{\bar{1}}, M_{\bar{2}})\right] &\le \mathop{\mathbf{E}}_{V}\left[\sum_{i=1}^{n} h^2(M_{\bar{2}^{i\leftarrow 1}}, M_{\bar{2}})\right]\\
&\le 2\mathop{\mathbf{E}}_{V}\left[\sum_{i=1}^{n} I_{Z_i\sim B(2),X^{\le n}\sim D_{\bar{2}^{i\leftarrow Z_i}}}\left(M; X_V^i \,\middle|\, V\right)\right]\\
&\le 3\sum_{i=1}^{n} I\left(M; X_V^i \,\middle|\, V, Z=2\right)\\
&\le \frac{3}{d} I\left(M; X^{\le n} \,\middle|\, Z=2\right).
\end{aligned} \tag{6}$$

The first inequality in Equation 6 is Equation 5, averaged over the random coordinate V. The second inequality applies the one-coordinate Jensen-Shannon/Hellinger bound Equation 3 to each user i and coordinate V. The third inequality applies Lemma C.3 as in Equation 4: since X_V^i under $Z = 2$ dominates its distribution in the mixed experiment with $Z_i \sim B(2)$ by a factor $2/3$, the mixed mutual information is at most $\frac{3}{2}$ times the mutual information under $Z = 2$. The last inequality uses Equation 2 and Equation 1.

It remains to use the $2/3$-accuracy assumption. For a fixed coordinate j, let P_j be the distribution of M conditioned on $V = j$ and $Z = 1$, and let Q be the distribution of M conditioned on $Z = 2$. Under $Z = 2$, the transcript distribution is independent of V, so Q is also the distribution denoted

by $M_{\bar{2}}$ for every j, while P_j is the distribution denoted by $M_{\bar{1}}$ for that j. Let $\bar{P} = \mathbf{E}_V P_V$ be the transcript distribution conditioned on $Z = 1$. If $A = \{m : \widehat{Z}(m) = 1\}$, then $2/3$-accuracy gives

$$\bar{P}(A) - Q(A) \geq \frac{1}{3}, \tag{7}$$

and hence $\mathrm{TV}(\bar{P}, Q) \geq 1/3$. Therefore

$$\frac{1}{3} \leq \mathrm{TV}(\bar{P}, Q) \leq \mathop{\mathbf{E}}_{V} [\mathrm{TV}(P_V, Q)] \leq \sqrt{2} \mathop{\mathbf{E}}_{V} [h(P_V, Q)], \tag{8}$$

By Jensen's inequality this implies that

$$\mathop{\mathbf{E}}_{V} \left[h^2(M_{\bar{1}}, M_{\bar{2}})\right] = \Omega(1). \tag{9}$$

Combining Equation 6 and Equation 9 gives

$$I\left(M; X^{\leq n} \,\middle|\, Z = 2\right) = \Omega(d).$$

Finally, conditioned on $Z = 2$, the transcript M and input $X^{\leq n}$ are independent of V, so

$$I\left(M; X^{\leq n} \,\middle|\, V, Z = 2\right) = I\left(M; X^{\leq n} \,\middle|\, Z = 2\right).$$

Since $Z \sim B(2)$,

$$I\left(M; X^{\leq n} \,\middle|\, V, Z\right) \geq \frac{1}{2} I\left(M; X^{\leq n} \,\middle|\, V, Z = 2\right) = \Omega(d),$$

as claimed. □

To relate the lower bound in Theorem C.1 to private aggregation we recall that differential privacy limits mutual information.

For this purpose we will need only privacy with respect to KL divergence (or Renyi DP of order 1). We say that an algorithm $\mathcal{A} : \mathcal{X}^n \to Y$ is ρ-KL stable if for any pair of neighboring datasets S, S', we have $\mathrm{KL}(\mathcal{A}(S) \| \mathcal{A}(S')) \leq \rho$.

Lemma C.4 ([FS17])**.** *Assume that the transcript of a communication protocol viewed as the output of an algorithm acting on the users' inputs is ρ-KL stable. Then for any distribution D over $\mathcal{X}$, if the users' inputs $X^{\leq n}$ are sampled i.i.d. from D then we have*

$$I\left(M; X^{\leq n}\right) \leq \rho n.$$

We can now complete the proof of Theorem 3.4 using Lemma 3.5.

Proof of Theorem 3.4. Let $\mathcal{A}$ be a $2/3$-accurate algorithm for Problem 3.3 that has access to $\mathcal{G}(S)$ and satisfies compositional ρ-zCDP. By Lemma 3.5, there is a local $n\rho$-zCDP blackboard protocol Dist-$\mathcal{A}$ whose output has the same distribution as the output of $\mathcal{A}$ on every input dataset. In particular, when run on the random input $X^{\leq n}$ from Problem 3.3, Dist-$\mathcal{A}$ is also $2/3$-accurate. Let M denote its full transcript, including public randomness and the final decision randomness as above.

By Theorem C.1,

$$I\left(M; X^{\leq n} \,\middle|\, V, Z\right) = \Omega(d).$$

On the other hand, the local $n\rho$-zCDP guarantee implies that the transcript is $n\rho$-KL stable with respect to changing one user's input. Fix any $v \in [d]$ and $z \in \{1, 2\}$. Conditioned on $(V, Z) = (v, z)$, the users' inputs are i.i.d. samples from $D_{z,v}$. Hence Lemma C.4 gives

$$I\left(M; X^{\leq n} \,\middle|\, V = v, Z = z\right) \leq n^2 \rho.$$

Averaging over V and Z yields

$$I\left(M; X^{\leq n} \,\middle|\, V, Z\right) \leq n^2 \rho.$$

Combining the two inequalities gives $n^2\rho = \Omega(d)$, and therefore

$$n = \Omega\left(\sqrt{\frac{d}{\rho}}\right),$$

as claimed. □

D Variants of the SVT Primitive

D.1 A Batch Version

Here, we define a batch version of the SVT primitive, where one inputs a sequence of queries (f, τ) with the expectation that the SVT outputs the index of the first query where the answer is above the threshold.

Input: A sequence of functions $f_1, \ldots, f_K : \mathcal{X} \to [0,1]$; a sequence of thresholds $\tau_1, \ldots, \tau_K$; a privacy parameter ε.

Output: An output $o \in [K] \cup \{\perp\}$.

(α, β)**-Accuracy Property:** For $j \in [K]$, and the dataset $\mathbf{x} \in \mathcal{X}^n$, let $f_j(\mathbf{x}) = \sum_i \mathtt{f}_j(x_i)$. We say an output $o \in [K] \cup \{\perp\}$ is α-accurate on $\mathbf{x}$ if the following hold. If $o = \perp$, then $f_j(\mathbf{x}) \leq \tau_j + \alpha$ for all $j \in [K]$. If $o \in [K]$, then $f_j(\mathbf{x}) \leq \tau_j + \alpha$ for all $j < o$, and $f_o(\mathbf{x}) \geq \tau_o - \alpha$. We say an oracle is (α, β)-accurate if its output is α-accurate except with probability β.

As in the online case, this batch version can be implemented using AboveThreshold. This batch version has the advantage of not needing to retain internal state between API calls.

Algorithm 4 AboveThreshold

Input: Dataset $\mathbf{x} \in \mathcal{X}^n$. Functions $f_1, \ldots, f_K : \mathcal{X} \to [0,1]$; thresholds $\tau_1, \ldots, \tau_K$; privacy parameter ε.
Output: Output $o \in [K] \cup \{\perp\}$.

$\gamma \leftarrow \mathrm{Lap}\left(\frac{2}{\varepsilon}\right)$.
for $j = 1, 2, \ldots, K$ **do**
  $f_j(\mathbf{x}) \leftarrow \sum_i f_j(x_i)$.
  $\nu_j \leftarrow \mathrm{Lap}(\frac{4}{\varepsilon})$.
  if $f_j(\mathbf{x}) + \nu_j \geq \tau_j + \gamma$ **then**
    Return j
Return $\perp$

D.2 A More Powerful Variant

Next we present a more general primitive that may be amenable to similar implementation techniques as our simple SVT primitive.

Our more general primitive will allow the query to specify two thresholds (τ^L, τ^H) in addition to f, and get back information on whether the function $f(\mathbf{x})$ is well below τ^L (`Low`), well above τ^H (`High`), or in between (`Close`). The primitive can keep answering queries as long as the query answer is not `Close`. We show an implementation of this primitive below.

Algorithm 5 CompareToThresholdsInit

Input: Privacy parameters $\varepsilon, \delta \in (0,1)$.
Output: Create a CompareToThresholds Object CTT

$CTT.n \leftarrow \mathrm{Lap}\left(\frac{2}{\varepsilon}\right)$ ▷This internal variable is private to the object.
$CTT.\varepsilon = \varepsilon$
$CTT.gap = \frac{12}{\varepsilon}\left(\log\frac{10}{\varepsilon} + \log\frac{1}{\delta} + 1\right)$.
$CTT.Done \leftarrow False$.
Incur privacy cost (ε, δ), and return handle to CTT.

This variant satisfies (ε, δ)-DP and the privacy bound is proved in [BSU17].

E Missing Proofs from Section 3

Lemma 3.5 ([DKM+06])**.** *Let $\mathcal{A}$ be an algorithm that is given access to $\mathcal{G}(S)$ and satisfies* compositional *ρ-zCDP. Then there exists a local $n\rho$-zCDP protocol Dist-$\mathcal{A}$ that simulates $\mathcal{A}$.*

Algorithm 6 CompareToThresholdsQuery

Input: Handle CTT, Function $f : \mathcal{X} \to \mathbb{R}$, Thresholds $\tau^L, \tau^H \in \mathbb{R}$, private dataset $\mathbf{x}$
Output: Output `Low`, `High`, or `Close`.

if not $CTT.Done$ and $\tau^H - \tau^L \geq CTT.gap$ **then**
  Compute $f(\mathbf{x}) = \sum_i \texttt{clip}_{[0,1]}(f(x_i))$.
  Sample u from Lap $\left(\frac{6}{CTT.\varepsilon}\right)$.
  if $f(\mathbf{x}) + u > \tau^H - CTT.n$ **then**
    Return `High`
  if $f(\mathbf{x}) + u < \tau^L + CTT.n$ **then**
    Return `Low`
  $CTT.Done \leftarrow True$
  Return `Close`
else
  Return ERROR.

Proof. We provide a short, self-contained proof. In the distributed noise addition protocol of [DKM+06], to simulate the Gaussian aggregation oracle, which adds noise $\mathcal{N}(0, \sigma^2 I_k)$, each of the n users independently adds noise $\mathcal{N}(0, (\sigma^2/n) I_k)$ to their contribution. Because the sum of n independent $\mathcal{N}(0, \sigma^2/n)$ random variables is distributed exactly as $\mathcal{N}(0, \sigma^2 I_k)$, the aggregate of the users' messages exactly simulates drawing Gaussian noise $G(S)$. Thus for a query with sensitivity 2, a single user's message with noise variance σ^2/n achieves the desired overall noise distribution, and by composition, the protocol is locally $n\rho$-zCDP. □

F Missing Proofs from Section 5

We begin by recalling the SVT model. We will only need the batch version of SVT and up to a transformation of each user's input, it can be phrased as the following. Each of n users holds a private vector $v^{(i)} \in [0,1]^d$, and the analyst is interested in aggregate quantities of the form

$$v = \sum_{i=1}^{n} v^{(i)} \in \mathbb{R}^d.$$

In selection, given a score function $s : \mathbb{R} \to \mathbb{R}$ applied coordinate-wise, the goal is to identify a coordinate $j \in [d]$ whose score $s(v_j)$ is close to the maximum over all coordinates, while ensuring ε-differential privacy with respect to changes in any single user input. Crucially, the SVT primitive allows the analyst to query whether $s(v_j)$ lies above or below a threshold, and to incur privacy cost only when a threshold is crossed.

Our algorithm builds directly on this capability. At a high level, we design a private selection procedure that repeatedly performs noisy threshold comparisons to narrow down the range of possible values for the maximum score, and isolate a high-scoring coordinate. Rather than explicitly evaluating all scores with independent noise—as in the naive Laplace mechanism—the algorithm adaptively compares against a threshold using the SVT primitive. This approach allows us to limit privacy loss to the relatively small number of times the algorithm observes that a score exceeds a threshold, while ignoring the vast majority of coordinates whose scores are clearly suboptimal.

Concretely, the algorithm maintains a candidate interval for the maximum score and performs a logarithmic number of rounds. In each round, it picks the midpoint of the current interval where the maximum lies and scans the coordinates in their given order using the SVT primitive. Based on whether zero, one, or multiple coordinates cross the threshold, the algorithm refines its estimate of the maximum and continues.

If the current width is W, the next width is at most $W/2 + \alpha$. For $[\ell, u] = [0, n]$ and $\alpha \geq 1$, at most $R = \lceil \log_2(2n) \rceil + 1$ rounds suffice, each using at most two fresh SVT streams and at most d queries. Allocating $\varepsilon/(2R)$ to each stream makes the algorithm compositionally ε-DP.

Assuming that the SVT oracle is (α, β)-accurate, all answers in the at most $2R$ streams are correct up to an additive α except with probability $2R\beta$. There are three cases. When all d queries are below the

current guess m, it follows that the true maximum is below $m + \alpha$, so our recursive step is correct. When at least two queries are above the current guess m, it follows that the maximum is above $m - \alpha$, so once again the maximum score is in $[\ell, u]$. Finally, if there is exactly one score above m, the returned score is within 2α of the maximum, so we can stop. The recursion stops when $u - \ell$ is smaller than 3α, and the algorithm outputs the last positive index. Since the maximum score was guaranteed to be in $[\ell, u]$ and the last positive score was at least ℓ, it follows that the returned index has score within $O(\alpha)$ of the maximum. Since our SVT primitive gives $(O(\frac{\log(d/\beta)}{\varepsilon}), \beta)$-accurate answers when called with privacy parameter ε, we get the following bound.

Theorem F.1. *There exists an ε-DP algorithm that solves the selection problem using the SVT primitive with additive error $O\left(\frac{\log d \log n}{\varepsilon}\right)$.*

G Other Approaches for Selection

G.1 Linear Scan

Algorithm 7 Private linear scan

Input: Oracle $\mathbb{O}$, upper limit u, lower limit ℓ on score
Output: Index $j \in [d]$ with approximately maximum true score

```
for m ← u, u − 1, u − 2, . . . , ℓ do
    Query the same initialized SVT stream 𝕆 at threshold m for all indices
    if an index j ∈ [d] is returned then
        Return j                                              ▷End procedure
Return an arbitrary index in [d]
```

Assuming the score for each user is normalized to $[0, 1]$, we have the following guarantees:

Theorem 5.3. *For $n \geq 1$, $d \geq 2$, and $\beta \in (0, 1)$, Algorithm 7 with $[\ell, u] = [0, n]$ and a single ε-DP SVT stream is an ε-DP selection algorithm with additive error $O(1 + \log(d(n+1)/\beta)/\varepsilon)$ with probability at least $1 - \beta$, using $O(d(n + 1))$ queries.*

Proof. Privacy follows from the fact that we run exactly one SVT oracle, and stop once we get one score above the threshold. The accuracy bound follows from the fact that we use decreasing thresholds and the SVT oracle is α-accurate. Finally, note that since the score for each user is in $[0, 1]$, then the score for each index is contained within the interval $[0, n]$. Hence, there are $n + 1$ threshold levels when scanning the integer bounds from n through 0, each across d coordinates, for a total of $O(nd)$ queries. □

G.2 Binary Search

We now show that allocating the privacy budget geometrically across the rounds of the binary search removes the $\log n$ factor from the error while keeping $O(d \log n)$ queries. Indeed, the error introduced in a round of the binary search is halved in each of the subsequent rounds. Therefore the early rounds can use a much smaller privacy budget than the last ones. Specifically, in Algorithm 8 round t uses budget proportional to γ^{T-t} for $\gamma = 3/4$. Thus its contribution to the final error is proportional to $(2\gamma)^{-(T-t)}$ and the total error is dominated by the last round. We state the algorithm for an arbitrary set of candidates P, an upper bound u on the maximum score in P and a witness index whose score is at least a given lower bound ℓ, since we use it as a subroutine in Appendix G.3 and in Appendix B. We next analyze Algorithm 8. Let $\mathcal{E}$ denote the event that every oracle call satisfies its accuracy guarantee. Each of the T calls fails with probability at most $\frac{\beta}{T}$, and thus $\mathbf{Pr}[\mathcal{E}] \geq 1 - \beta$ by a union bound. We thus condition on the event $\mathcal{E}$ for the remainder of the utility analysis.

Lemma G.1. *For all $t \in [T + 1]$, we have $s_{j^\star}(D) \geq \ell_t$ and $S^\star \leq u_t$, where $S^\star = \max_{j \in P} s_j(D)$.*

Proof. We prove the claim by induction on t. For $t = 1$, we have $s_{j^\star}(D) = s_{j_0}(D) \geq \ell = \ell_1$ and $S^\star \leq u = u_1$ by the assumptions on the input. Suppose the inductive hypothesis holds at step t and let $m_t = (\ell_t + u_t)/2$. If no index is returned, then the oracle accuracy gives $S^\star \leq m_t + \alpha_t$ and thus

$$S^\star \leq \min\{u_t, m_t + \alpha_t\} = u_{t+1}.$$

Algorithm 8 Private geometric binary search (GEOMETRICBINARYSEARCH)

Input: Oracle $\mathbb{O}$, candidate set $P \subseteq [d]$, interval $[\ell, u]$ with $\max_{j \in P} s_j(D) \le u$, witness $j_0 \in [d]$ with $\ell \le s_{j_0}(D)$, number of rounds $T \ge 1$, privacy budget $\varepsilon > 0$, failure probability $\beta \in (0,1)$
Output: Index $j^\star$ and an interval $[\ell, u]$ such that $s_{j^\star}(D) \ge \ell$ and $\max_{j \in P} s_j(D) \le u$

$\gamma \leftarrow 3/4$, $Z \leftarrow \frac{1-\gamma^T}{1-\gamma}$, $j^\star \leftarrow j_0$
for $t = 1, \dots, T$ **do**
  $\varepsilon_t \leftarrow \frac{\varepsilon}{Z}\gamma^{T-t}$
  $\alpha_t \leftarrow \frac{32}{\varepsilon_t} \log\left(\frac{2|P|T}{\beta}\right)$
  $m \leftarrow \frac{\ell+u}{2}$
  Query oracle $\mathbb{O}$ on all candidates in P with threshold m, budget ε_t, and failure probability β/T
  if no index returned **then**
    $u \leftarrow \min\{u, m + \alpha_t\}$ ▷All noisy scores $\lesssim m$
  else(oracle returns index j)
    if $m - \alpha_t > \ell$ **then**
      $\ell \leftarrow m - \alpha_t$
      $j^\star \leftarrow j$
Return $(j^\star, \ell, u)$

We also have $\ell_{t+1} = \ell_t$.

On the other hand, if index j is returned, then $s_j(D) \ge m_t - \alpha_t$. If $m_t - \alpha_t > \ell_t$, the algorithm sets $\ell_{t+1} = m_t - \alpha_t$ and $j^\star = j$, so $s_{j^\star}(D) \ge \ell_{t+1}$. Otherwise, we again have $\ell_{t+1} = \ell_t$. Moreover, in both cases, we have $u_{t+1} = u_t$. Therefore, the inductive step is complete. □

We next bound the size of the search space, showing that it geometrically decreases at each step $t \in [T]$.

Lemma G.2. *Let $W_t = u_t - \ell_t$. Then for all $t \le T$, we have $W_{t+1} \le \frac{W_t}{2} + \alpha_t$.*

Proof. Since $m_t = \frac{\ell_t + u_t}{2}$, then we have

$$m_t - \ell_t = u_t - m_t = \frac{W_t}{2}.$$

Now, if no index is returned, then we have

$$W_{t+1} = u_{t+1} - \ell_t \le (m_t + \alpha_t) - \ell_t = \frac{W_t}{2} + \alpha_t.$$

Otherwise, if an index is returned, we have

$$W_{t+1} = u_t - \ell_{t+1} \le u_t - (m_t - \alpha_t) = \frac{W_t}{2} + \alpha_t.$$

□

The following theorem summarizes the guarantees of Algorithm 8.

Theorem G.3. *Algorithm 8 is ε-differentially private and makes at most $|P|T$ SVT queries. Let $W = u - \ell$. With probability at least $1-\beta$, it returns $(j^\star, \ell', u')$ such that $s_{j^\star}(D) \ge \ell'$, $\max_{j \in P} s_j(D) \le u'$, and*

$$u' - \ell' \le \frac{W}{2^T} + \frac{384}{\varepsilon} \log\left(\frac{2|P|T}{\beta}\right).$$

In particular, for $P = [d]$, $[\ell, u] = [0, n]$, $j_0 = 1$ and $T = \max\{1, \lceil \log_2 n \rceil\}$, with probability at least $1 - \beta$ the returned index satisfies

$$S^\star - s_{j^\star}(D) \le 1 + \frac{384}{\varepsilon} \log\left(\frac{2d(1 + \log_2 n)}{\beta}\right).$$

Proof. Algorithm 8 accesses the data only through T oracle calls, and the call in round t has privacy budget ε_t and answers at most $|P|$ queries. By the choice of Z, $\sum_{t\in[T]} \varepsilon_t = \varepsilon$, and therefore privacy follows from basic composition. Conditioned on the event $\mathcal{E}$, we have

$$S^\star - s_{j^\star}(D) \le u_{T+1} - \ell_{T+1} = W_{T+1}.$$

By Lemma G.2, we have

$$W_{T+1} \le \frac{W_1}{2^T} + \sum_{t=1}^{T} \frac{\alpha_t}{2^{T-t}}.$$

Since $\alpha_t = \frac{32}{\varepsilon_t}\log\left(\frac{2|P|T}{\beta}\right) = \frac{32Z}{\varepsilon}\gamma^{-(T-t)}\log\left(\frac{2|P|T}{\beta}\right)$, we have

$$\sum_{t=1}^{T} \frac{\alpha_t}{2^{T-t}} = \frac{32Z}{\varepsilon}\log\left(\frac{2|P|T}{\beta}\right)\sum_{k=0}^{T-1}(2\gamma)^{-k} \le \frac{384}{\varepsilon}\log\left(\frac{2|P|T}{\beta}\right),$$

where we used that $Z \le \frac{1}{1-\gamma} = 4$ and $\sum_{k\ge 0}(2\gamma)^{-k} = 3$. This gives the first claim. For the second claim we note that $W_1 = n \le 2^T$ and $T \le 1 + \log_2 n$. □

Corollary G.4. *If $d \ge 2$, $0 < \varepsilon \le 1$, $\beta = d^{-c}$ for a fixed $c > 0$, and $n \le e^{\mathrm{poly}(d)}$, then Algorithm 8 with $P = [d]$, $[\ell, u] = [0, n]$, $j_0 = 1$ and $T = \max\{1, \lceil \log_2 n \rceil\}$ makes $O(d \log n)$ SVT queries and, with probability at least $1 - \beta$, returns $j^\star$ such that*

$$S^\star - s_{j^\star}(D) = O\left(\frac{\log d}{\varepsilon}\right).$$

Running Algorithm 8 on a single candidate gives a simple way to privately estimate a sum using the SVT API, which we use in Appendix B.

Corollary G.5. *Let $f : \mathcal{X} \to [0, 1]$, $\varepsilon > 0$ and $\beta \in (0, 1)$, and let $T = \max\{1, \lceil \log_2 n \rceil\}$. There is an ε-differentially private algorithm that makes T SVT queries and, with probability at least $1 - \beta$, outputs an interval of length at most $1 + \frac{384}{\varepsilon}\log\frac{2T}{\beta}$ that contains $\sum_{i\in[n]} f(x_i)$.*

Proof. We run Algorithm 8 with the single candidate f, the interval $[0, n]$, the witness f (whose score is nonnegative) and T rounds. By Theorem G.3, the returned interval $[\ell', u']$ contains $\sum_{i\in[n]} f(x_i)$ and has length at most $n/2^T + \frac{384}{\varepsilon}\log\frac{2T}{\beta}$. □

G.3 Near-Linear Algorithm

We now describe an algorithm that achieves the error of Algorithm 8 using only $O(d \log^* n + \log^3 n)$ SVT queries in expectation. For simplicity of notation we write s_j for $s_j(D)$ and $S^\star = \max_{j\in[d]} s_j$.

Like Algorithm 8, our algorithm maintains an interval $[\ell, u]$ that contains $S^\star$ together with a witness index j_w whose score is at least ℓ. Algorithm 8 halves the length of this interval in each round and queries all d candidates in every round. Our algorithm instead proceeds in *phases*, each of which reduces the length of the interval from $x\alpha$ to $O(\alpha \log^3 x)$ using $O(d + \log^3 x)$ queries in expectation, where α is (up to a constant factor) the target error. Since the length decreases so rapidly, $O(\log^* n)$ phases suffice to reduce it to $O(\alpha)$, and a final run of Algorithm 8 with a constant number of rounds completes the selection.

The main step of a phase is a *filtering pass* in which we query every candidate once against a fixed threshold τ. Every candidate that answers NO has score at most τ (up to the accuracy of the SVT). Thus it remains to search the set Y of candidates that answer YES, which is cheap when Y is small. At the same time, each YES answer closes the SVT stream and therefore the privacy cost of a filtering pass grows linearly with $|Y|$. Thus we need a threshold that only few candidates exceed. To construct this threshold, we first form a random sample P by including each candidate independently with probability $1/t$, where $t = \lceil \log_2 x \rceil$. We estimate the maximum score in P by running Algorithm 8 on P with t rounds, which makes at most $|P|t$ queries, or about d queries in expectation. We then set τ slightly above this estimate, with enough slack to account for both the estimation error and the SVT error, and run the filtering pass. When all SVT answers are accurate, only candidates whose scores

exceed the maximum score in P can answer YES. The number of such candidates is dominated by a geometric random variable with mean t and thus it is less than t^2 except with probability e^{-t}. When $|Y| < t^2$, we search Y using Algorithm 8 with t rounds, which makes at most t^3 queries.

We use privacy budget ε/t^3 for every stream of the filtering pass and for each of the two runs of Algorithm 8 in a phase. Therefore the accuracy of the SVT answers, and also the length of the new interval, is $O(\alpha t^3) = O(\alpha \log^3 x)$. The filtering pass initializes at most t^2 streams and thus a phase uses budget $O(\varepsilon/t)$ in total. This budget at least doubles from one phase to the next and thus the total budget of all phases is dominated by the last one. Two additional details ensure that the privacy guarantee holds on every execution, and not only when all SVT answers are accurate. First, the parameters of every phase are fixed in advance and do not depend on the data. Second, the filtering pass stops after t^2 YES answers and we then search all d candidates instead of Y. This makes dt queries but happens with probability at most e^{-t} when all SVT answers are accurate. In Algorithm 9, GEOMETRICBINARYSEARCH$(P, [\ell, u], j, T, \varepsilon, \beta)$ denotes a run of Algorithm 8 on the candidate set P with the interval $[\ell, u]$, the witness j, T rounds, privacy budget ε and failure probability β.

Algorithm 9 Private selection by sampling and filtering

Input: Oracle $\mathbb{O}$, candidates $[d]$ with scores in $[0, n]$, privacy budget $\varepsilon \in (0, 1]$, failure probability $\beta \in (0, 1)$

Output: Index $j^\star$ with approximately maximum score

$N \leftarrow 3\log_2^2(2n)$, $\beta' \leftarrow \beta/N$, $\alpha \leftarrow \frac{8}{\varepsilon} \log \frac{384d \log_2^3(2n)}{\beta}$
$\ell \leftarrow 0$, $u \leftarrow n$, $j_w \leftarrow 1$, $x \leftarrow n/\alpha$
while $x \geq 2^{64}$ **do** ▷one phase
  $t \leftarrow \lceil \log_2 x \rceil$, $\varepsilon' \leftarrow \varepsilon/t^3$, $a \leftarrow t^3\alpha$
  Sample $P' \subseteq [d]$ by including each candidate independently with probability $1/t$, and let $P \leftarrow P' \cup \{j_w\}$
  $(j_P, \ell_P, \cdot) \leftarrow$ GEOMETRICBINARYSEARCH$(P, [\ell, u], j_w, t, \varepsilon', \beta')$
  $\tau \leftarrow \ell_P + 50a$, $Y \leftarrow \emptyset$, and initialize an SVT stream with budget ε' ▷filtering pass
  for $j = 1, \ldots, d$ **do**
    if the query (f_j, τ) to the current stream returns YES **then**
      $Y \leftarrow Y \cup \{j\}$
      if $|Y| = t^2$ **then break**, **else** initialize a new SVT stream with budget ε'
  $C \leftarrow [d]$ if $|Y| = t^2$, and $C \leftarrow Y \cup \{j_P\}$ otherwise
  $(j_w, \ell, u') \leftarrow$ GEOMETRICBINARYSEARCH$(C, [\ell_P, u], j_P, t, \varepsilon', \beta')$
  $u \leftarrow \min\{u, \max\{u', \tau + a\}\}$, $x \leftarrow 64t^3$
$(j^\star, \cdot, \cdot) \leftarrow$ GEOMETRICBINARYSEARCH$([d], [\ell, u], j_w, 64, \varepsilon/2, \beta')$
return $j^\star$

We now analyze Algorithm 9. Let k denote the number of phases and, for $i < k$, let x_i and $t_i = \lceil \log_2 x_i \rceil$ denote the values of x and t in phase i (starting with $i = 0$). We also denote by x_k the value of x after the last phase. Note that these values depend only on n, d, ε and β. By definition, $x_{i+1} = 64t_i^3$ and $t_i \geq 64$ for every $i < k$. This implies that $t_{i+1} \leq 7 + 3\log_2 t_i \leq t_i/2$. Therefore

$$\sum_{i<k} \frac{1}{t_i} \leq \frac{2}{t_{k-1}} \leq \frac{1}{32}. \tag{10}$$

Further, it is easy to see that two consecutive phases reduce any sufficiently large value of x to at most $\log_2 x$ and thus $k = O(\log^* n)$.

We first show that the algorithm is private, which follows from basic composition since the budgets of all phases are fixed in advance.

Lemma G.6. *Algorithm 9 is ε-differentially private.*

Proof. The algorithm accesses the data only through SVT streams, either in a filtering pass or in a run of Algorithm 8. In phase i each of the two runs of Algorithm 8 uses budget ε/t_i^3, and the filtering pass initializes at most t_i^2 streams with budget ε/t_i^3 each. Thus phase i uses budget at most $(t_i^2 + 2)\varepsilon/t_i^3 \leq 2\varepsilon/t_i$. By Equation 10, all phases together use budget at most $\varepsilon/16$. The final run of Algorithm 8 uses budget $\varepsilon/2$ and therefore the claim follows from basic composition. □

Next we analyze the accuracy of the algorithm. Every run of Algorithm 8 is made with failure probability $\beta' = \beta/N$. Every stream of a filtering pass answers at most d queries and thus, by Theorem 4.2, it is $\frac{8}{\varepsilon'}\log\frac{2dN}{\beta}$-accurate except with probability β/N. The total number of runs of Algorithm 8 and streams of filtering passes is at most $1 + \sum_{i<k}(t_i^2 + 2) \le N$, since $t_{i+1} \le t_i/2$ and $t_0 \le \log_2(2n)$ (as $\alpha \ge 1$). Let $\mathcal{E}$ denote the event that every run of Algorithm 8 satisfies the guarantee of Theorem G.3 and every stream of a filtering pass is accurate as above. By a union bound, $\Pr[\mathcal{E}] \ge 1 - \beta$. By our choice of α, on the event $\mathcal{E}$ every stream of the filtering pass in phase i is $t_i^3\alpha$-accurate, and every run of Algorithm 8 in phase i with input interval $[\ell', u']$ returns an interval of length at most $(u' - \ell')/2^{t_i} + 48t_i^3\alpha$. Similarly, the final run returns an interval of length at most $(u - \ell)/2^{64} + 96\alpha$.

We now show that on the event $\mathcal{E}$ every phase maintains the interval and the witness, and reduces the length of the interval as intended. The proof formalizes the description above. The threshold of the filtering pass exceeds the maximum score in P by more than the accuracy of the stream. Thus only candidates whose scores exceed this maximum can answer YES. Every candidate that answers NO is certified to have score at most $\tau + a$, and the run of Algorithm 8 on C certifies the scores of the remaining candidates.

Lemma G.7. *On the event $\mathcal{E}$, for every $i \le k$, at the beginning of phase i (or of the final run of Algorithm 8 if $i = k$) we have $S^\star \le u$, $s_{j_w} \ge \ell$ and $u - \ell \le x_i\alpha$. Moreover, the filtering pass of phase $i < k$ stops after t_i^2 YES answers only if at least t_i^2 candidates have score larger than $\max_{j\in P} s_j$.*

Proof. We prove the claim by induction on i. For $i = 0$ it holds since all scores lie in $[0, n]$ and $x_0 = n/\alpha$. Now consider phase $i < k$ and let $t = t_i$ and $a = t^3\alpha$. By the inductive hypothesis, the run of Algorithm 8 on P returns $\ell_P \ge \ell$ (the lower end of the interval never decreases in Algorithm 8) such that $s_{j_P} \ge \ell_P$ and $S_P^\star = \max_{j\in P} s_j \le \ell_P + (u - \ell)/2^t + 48a \le \ell_P + 49a$. Therefore every candidate j with $s_j \le S_P^\star$ satisfies $s_j \le \tau - a$ and answers NO in the filtering pass. This proves the second claim since every candidate is queried at most once. Further, every candidate $j \notin Y$ that is queried in the filtering pass answers NO and thus satisfies $s_j \le \tau + a$.

Let (j', ℓ', u') denote the output of the run of Algorithm 8 on C, so that the new values are $j_w = j'$, $\ell = \ell'$ and $u = \min\{u, \max\{u', \tau + a\}\}$. This run starts from the interval $[\ell_P, u]$ with the witness j_P and thus $s_{j'} \ge \ell'$, $S_C^\star = \max_{j\in C} s_j \le u'$ and $u' - \ell' \le 49a$. If the filtering pass does not stop early then every candidate outside of C answered NO and thus $S^\star \le \max\{u', \tau + a\}$. Otherwise $C = [d]$ and $S^\star = S_C^\star \le u'$. In both cases the new value of u is at least $S^\star$. The length of the new interval is at most $\max\{u' - \ell', \tau + a - \ell'\}$. If $Y \ne \emptyset$ then some candidate in C answered YES. Thus $S_C^\star \ge \tau - a$ and $\ell' \ge S_C^\star - 49a \ge \tau - 50a$. If $Y = \emptyset$ then $\ell' \ge \ell_P = \tau - 50a$. In both cases the new interval has length at most $51a \le x_{i+1}\alpha$. □

Finally, we bound the expected number of queries. The run of Algorithm 8 on P makes about d queries in expectation and the filtering pass makes at most d queries. The run on C makes at most t^3 queries, unless the filtering pass stops early. By Lemma G.7, on the event $\mathcal{E}$ this happens only if none of the t^2 candidates with the largest scores is in the sample, which has probability at most e^{-t}.

Lemma G.8. *If $\beta \le 1/d$ then the expected number of SVT queries made by Algorithm 9 is at most $3dk + 2\log_2^3(2n) + 64d = O(d\log^* n + \log^3 n)$.*

Proof. Consider phase $i < k$ and let $t = t_i$. The run of Algorithm 8 on P makes $|P|t$ queries and $\mathbf{E}[|P|] \le 1 + d/t$. The filtering pass makes at most d queries. The run on C makes at most t^3 queries if the filtering pass does not stop early, and dt queries otherwise. Let $j_1, \ldots, j_d$ denote the candidates sorted by their scores in decreasing order (with ties broken arbitrarily). If P' includes one of $j_1, \ldots, j_{t^2}$ then fewer than t^2 candidates have score larger than $\max_{j\in P} s_j$. By Lemma G.7, this implies that the filtering pass stops early only if $\mathcal{E}$ does not hold or P' includes none of $j_1, \ldots, j_{t^2}$, which happens with probability at most $\beta + (1 - 1/t)^{t^2} \le \beta + e^{-t}$. Therefore the expected number of queries in phase i is at most

$$(1 + d/t)t + d + t^3 + (\beta + e^{-t})dt \le 3d + t^3 + 2t,$$

where we used that $\beta d \le 1$ and $te^{-t} \le 1$. Summing over the phases and using that $t_{i+1} \le t_i/2$ and $t_0 \le \log_2(2n)$, we get that all phases together make at most $3dk + 2\log_2^3(2n)$ queries in expectation. The final run of Algorithm 8 makes $64d$ queries. □

By Lemma G.7, on the event $\mathcal{E}$ the final run of Algorithm 8 starts from an interval of length $u - \ell \le x_k\alpha < 2^{64}\alpha$ that contains $S^\star$, with a witness whose score is at least ℓ. Therefore it returns $j^\star$ such that $S^\star - s_{j^\star} \le \alpha + 96\alpha = 97\alpha$. Together with Lemmas G.6 and G.8, this gives the following more general version of Theorem 2.2.

Theorem 5.4. *Let $n \ge 2$, $d \ge 1$, $\varepsilon \in (0, 1]$ and $\beta \in (0, 1/d]$. Algorithm 9 is ε-differentially private and, with probability at least $1 - \beta$, it outputs an index $j^\star$ such that*

$$\max_{j\in[d]} s_j(D) - s_{j^\star}(D) \le \frac{776}{\varepsilon} \log \frac{384d \log_2^3(2n)}{\beta} = O\left(\frac{1}{\varepsilon} \log \frac{d \log n}{\beta}\right).$$

The expected number of SVT queries made by the algorithm is $O(d \log^* n + \log^3 n)$. In particular, for $\beta = 1/(dn)$ the expected additive error is $O\left(\frac{\log(dn)}{\varepsilon}\right)$, and if $d \ge \log^3 n$ then the expected number of queries is $O(d \log^* d)$.

H Missing Proofs from Appendix B

Algorithm 10 MWEM Algorithm [HLM12]

Input: Private dataset B with n records and d features, query set Q, number T of iterations, privacy budget $\varepsilon > 0$

Output: Synthetic histogram A of total mass n over $\mathcal{D}$

1: Initialize A_0 as a uniform distribution over the domain $\mathcal{D}$, scaled by n

2: **for** $t = 1$ to T **do**

3: Use exponential mechanism to privately select the query $q_t \in Q$ that maximizes the score function $|q(B) - q(A_{t-1})|$, with privacy budget $\frac{\varepsilon}{2T}$

4: $m_t \leftarrow q_t(B) + \mathrm{Lap}\left(\frac{4T}{\varepsilon}\right)$

5: Let A_t be n times the distribution whose entries satisfy

$$A_t(x) \propto A_{t-1}(x) \times \exp\left(q_t(x) \times \frac{m_t - q_t(A_{t-1})}{2n}\right)$$

6: **Return** $A = \frac{1}{T}\sum_{t\in[T]} A_t$ ▷The average of the T updated synthetic histograms

Observe that the key challenge to implementing Algorithm 10 in distributed models is implementing the exponential mechanism to privately select the query $q_t \in Q$ that maximizes the score function $|q(B) - q(A_{t-1})|$. Suppose there exists a mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ so that there exists a bijection between queries $q \in Q$ and coordinates $k \in [d']$ so that the query q can be decomposed as the sum $q(B) = \sum_{i\in[n]}(f(v_i))_k$. However, this does not quite allow us to sample using score function $|q(B) - q(A_{t-1})|$ due to the term $q(A_{t-1})$. Thus, for the embedded dimension d', we use the public offset vector $w \in \mathbb{R}^{d'}$ so that for each query q, there exists $k \in [d']$ such that

$$q(B) - q(A_{t-1}) = w_k + \sum_{i\in[n]} (f(v_i))_k.$$

For example, consider the class of two-way marginal queries, defined as follows. A two-way marginal query $q_{(a,b)}$ is specified by a pair of attributes $a, b \in [d]$, and returns the number of records in the dataset whose attribute a and attribute b are both equal to 1, i.e., $q_{(a,b)}(B) = \sum_{i\in[n]} \mathbf{1}[v_i(a) = v_i(b) = 1]$. We define the embedding as follows. Let each user record $v_i \in \{0, 1\}^d$ encode its attribute values, and let the embedded dimension d' index all pairs (a, b), so that $d' = O\left(d^2\right)$. Define $f(v_i) \in \mathbb{R}^{d'}$ by

$$(f(v_i))_{(a,b)} = \mathbf{1}[v_i(a) = v_i(b) = 1].$$

Then for any two-way marginal query $q_{(a,b)}$,

$$q_{(a,b)}(B) = \sum_{i\in[n]} (f(v_i))_{(a,b)}.$$

Moreover, since A_{t-1} is fixed at round t, we can introduce a public offset vector $w \in \mathbb{R}^{d'}$ defined by $w_{(a,b)} = -q_{(a,b)}(A_{t-1})$, so that

$$q_{(a,b)}(B) - q_{(a,b)}(A_{t-1}) = w_{(a,b)} + \sum_{i \in [n]} (f(v_i))_{(a,b)}$$

Thus, for two-way marginal queries, the signed MWEM discrepancy can be realized as a coordinate of an aggregated embedded vector plus a public offset. After analyzing Algorithm 10, we show how this representation allows us to implement the selection step in the SVT model, using approximate selection in place of the exponential mechanism.

We define $\Delta_t = \max_{q \in Q} |q(A_{t-1}) - q(B)|$ and, for a confidence parameter $\beta \in (0,1)$, $\eta = \frac{8T}{\varepsilon} \ln \frac{2T|Q|}{\beta}$.

Lemma H.1. *With probability at least $1-\beta$, we have for all $t \in [T]$,*

$$|q_t(A_{t-1}) - q_t(B)| \geq \Delta_t - \eta, \qquad |m_t - q_t(B)| \leq \eta.$$

Proof. Since $q(x) \in [-1,+1]$, replacing one record changes $q(B)$, and hence the score $|q(B) - q(A_{t-1})|$, by at most 2 (conditioned on the preceding transcript, A_{t-1} is fixed). By the standard utility guarantee of the exponential mechanism [MT07, DR+14], with privacy parameter $\frac{\varepsilon}{2T}$ and score sensitivity 2, the selected query q_t satisfies $|q_t(A_{t-1}) - q_t(B)| \geq \Delta_t - \frac{8T}{\varepsilon}\left(\ln|Q| + \ln\frac{2T}{\beta}\right) = \Delta_t - \eta$ with probability at least $1 - \frac{\beta}{2T}$. Further, $m_t - q_t(B) \sim \mathrm{Lap}\left(\frac{4T}{\varepsilon}\right)$ and therefore $|m_t - q_t(B)| \leq \frac{4T}{\varepsilon} \ln \frac{2T}{\beta} \leq \eta$ with probability at least $1 - \frac{\beta}{2T}$. The claim follows by a union bound over the $2T$ events. □

We next define the relative entropy by

$$\Phi_t = \sum_{x \in \mathcal{D}} \frac{B(x)}{n} \cdot \log \frac{B(x)}{A_t(x)}.$$

We utilize the following two standard properties of relative entropy:

Fact H.2 (Non-negativity of relative entropy)**.** *[HLM12] $\Phi_t \geq 0$ for all $t \in [T]$*

Fact H.3 (Jensen's inequality for relative entropy)**.** *[HLM12] $\Phi_0 \leq \log|\mathcal{D}|$*

We also have the following well-known fact for the exponential function:

Fact H.4. *[HLM12] For all $x \in [-1,1]$, we have $\exp(x) \leq 1 + x + x^2$.*

To bound the error $|q_t(A) - q_t(B)|$ in each round, we show that the relative entropy decreases by some amount in each round.

Lemma H.5. *For each $t \in [T]$, we have*

$$\Phi_{t-1} - \Phi_t \geq \left(\frac{q_t(A_{t-1}) - q_t(B)}{2n}\right)^2 - \left(\frac{m_t - q_t(B)}{2n}\right)^2.$$

Proof. By definition, we have

$$\Phi_{t-1} - \Phi_t = \sum_{x \in \mathcal{D}} \frac{B(x)}{n} \log \frac{A_t(x)}{A_{t-1}(x)}.$$

Let $\alpha_t := \frac{m_t - q_t(A_{t-1})}{2n}$ and β_t be the normalization factor at step t. Then we have $\frac{A_t(x)}{A_{t-1}(x)} = \frac{1}{\beta_t} \cdot \exp(q_t(x) \cdot \alpha_t)$ so that

$$\Phi_{t-1} - \Phi_t = \frac{\alpha_t \cdot q_t(B)}{n} - \log \beta_t.$$

Since we have the normalization factor

$$\beta_t = \sum_{x \in \mathcal{D}} \frac{A_{t-1}(x)}{n} \cdot \exp(q_t(x) \cdot \alpha_t),$$

Note that A_{t-1}/n is a probability distribution over $\mathcal{D}$ since $\sum_x A_{t-1}(x) = n$. Hoeffding's lemma, applied to $q_t(X) \in [-1, 1]$ for $X \sim A_{t-1}/n$, gives, for every real α_t,

$$\log \beta_t \leq \frac{\alpha_t q_t(A_{t-1})}{n} + \frac{\alpha_t^2}{2} \leq \frac{\alpha_t q_t(A_{t-1})}{n} + \alpha_t^2.$$

Therefore,

$$\Phi_{t-1} - \Phi_t \geq \frac{\alpha_t}{n} \cdot (q_t(B) - q_t(A_{t-1})) - \alpha_t^2.$$

The desired claim then holds from the definition of α_t. □

Lemma H.6. *For an integer $T \geq 1$ and $|\mathcal{D}| \geq 2$, on the simultaneous accuracy event of Lemma H.1,*

$$\max_{q \in Q} |q(A) - q(B)| \preceq n \cdot \sqrt{\frac{\log |\mathcal{D}|}{T}} + \eta.$$

Proof. Firstly, observe that since $\Delta_t = \max_{q \in Q} |q(A_{t-1}) - q(B)|$, then

$$\begin{aligned}\max_{q \in Q} |q(A) - q(B)| &\leq \frac{1}{T} \sum_{t=1}^{T} \Delta_t + \frac{1}{T} \max_{q \in Q} |q(A_T) - q(A_0)| \\ &\leq \frac{1}{T} \sum_{t=1}^{T} \Delta_t + \frac{2n}{T}.\end{aligned}$$

On the accuracy event of Lemma H.1 we have that $\Delta_t \leq |q_t(A_{t-1}) - q_t(B)| + \eta$ and $|m_t - q_t(B)| \leq \eta$ for all $t \in [T]$. By Lemma H.5, we have

$$\frac{1}{T} \sum_{t \in [T]} \Delta_t \preceq \left(\frac{n^2}{T} \cdot \sum_{t \in [T]} (\Phi_{t-1} - \Phi_t) + \eta^2 \right)^{1/2} + \eta.$$

By a telescoping argument, observe that $\sum_{t \in [T]} (\Phi_{t-1} - \Phi_t) = \Phi_0 - \Phi_T$. Then by Fact H.2 and Fact H.3, we have

$$\frac{1}{T} \sum_{t \in [T]} \Delta_t \preceq \left(\frac{n^2 \cdot \log(|\mathcal{D}|)}{T} + \eta^2 \right)^{1/2} + \eta.$$

Since $\sqrt{x + y} \leq \sqrt{x} + \sqrt{y}$ and $2n/T = O(n\sqrt{\log |\mathcal{D}|/T})$ for $|\mathcal{D}| \geq 2$ and $T \geq 1$, putting things together gives

$$\max_{q \in Q} |q(A) - q(B)| \preceq n \cdot \sqrt{\frac{\log |\mathcal{D}|}{T}} + \eta.$$

□

For $0 < \varepsilon \leq 1$, let $L = \ln \frac{2n|Q|}{\beta}$ and $T = \min \left\{ n, \left\lceil (n\varepsilon)^{2/3} (\log |\mathcal{D}|)^{1/3} L^{-2/3} \right\rceil \right\}$. If $\log |\mathcal{D}| > n$ then the bound below is larger than n and holds trivially. Otherwise T is the balancing value of the two terms in Lemma H.6 rounded up to an integer, $T \leq n$ and hence $\eta \leq \frac{8T}{\varepsilon} L$, and we obtain:

Corollary H.7. *For $0 < \varepsilon \leq 1$, with probability at least $1 - \beta$,*

$$\max_{q \in Q} |q(A) - q(B)| \preceq n^{2/3} \cdot \left(\frac{\log |\mathcal{D}| \ln(2n|Q|/\beta)}{\varepsilon} \right)^{1/3}.$$

Finally, we show that Algorithm 10 is differentially private.

Lemma H.8. *Algorithm 10 is ε-differentially private.*

Proof. The algorithm makes T invocations of the Exponential Mechanism with privacy parameter $\frac{\varepsilon}{2T}$, and T invocations of the Laplace Mechanism with privacy parameter $\frac{\varepsilon}{2T}$. By the basic composition of differential privacy, the overall algorithm is ε-differentially private. □

We now describe how to implement Algorithm 10 in the SVT model. The algorithm accesses the private dataset only to select the query q_t and to measure m_t. We implement the selection step using our selection algorithms and the measurement step using the estimation of a single sum (Corollary G.5). As in the example of two-way marginals above, the only complication is that the score of a query depends on the synthetic answer $q(A_{t-1})$. This answer is determined by the outputs of the previous rounds and is therefore public in round t. Thus it only shifts the thresholds of the SVT queries.

Specifically, for the selection step we use $2|Q|$ candidates, one for every query $q \in Q$ and sign $\sigma \in \{-1, +1\}$, with the functions $f_{q,\sigma}(x) = \frac{1+\sigma q(x)}{2} \in [0,1]$ and the public offsets $w_{q,\sigma} = \frac{n-\sigma q(A_{t-1})}{2} \in [0,n]$. The shifted score of candidate (q,σ) is then

$$\sum_{i\in[n]} f_{q,\sigma}(x_i) + w_{q,\sigma} = n + \frac{\sigma\left(q(B) - q(A_{t-1})\right)}{2} \in [0,2n],$$

and the maximum of the shifted scores over all candidates is $n + \Delta_t/2$. Our selection algorithms access the scores only by comparing them with thresholds, and comparing the shifted score of candidate (q,σ) with a threshold τ is the SVT query $(f_{q,\sigma}, \tau - w_{q,\sigma})$. Therefore they apply to shifted scores, with the range $[0,n]$ of the scores replaced by $[0,2n]$. We run Algorithm 8 on all $2|Q|$ candidates with privacy budget $\frac{\varepsilon}{2T}$, failure probability $\frac{\beta}{2T}$ and $\lceil \log_2(2n) \rceil$ rounds. By Theorem G.3, with probability at least $1 - \frac{\beta}{2T}$ the selected query satisfies $|q_t(A_{t-1}) - q_t(B)| \geq \Delta_t - \eta'$, where $\eta' = O\left(\frac{T}{\varepsilon} \log \frac{T|Q| \log n}{\beta}\right)$.

For the measurement step we apply Corollary G.5 to the function $\frac{1+q_t(x)}{2}$ with privacy budget $\frac{\varepsilon}{2T}$ and failure probability $\frac{\beta}{2T}$, and set $m_t = 2\hat{g} - n$, where $\hat{g}$ is the midpoint of the returned interval. Then $|m_t - q_t(B)| \leq \eta'$ with probability at least $1 - \frac{\beta}{2T}$ (for a suitable constant in the definition of η'). Therefore Lemma H.1 holds with η' in place of η. The rest of the analysis uses only the two inequalities of Lemma H.1 and thus Lemma H.6 and the corollary above also hold with η' in place of η (for $T \leq n$, $\eta' = O\left(\frac{T}{\varepsilon} \log \frac{n|Q|}{\beta}\right)$). The resulting algorithm is ε-DP by basic composition of its $2T$ private steps. We remark that when $\frac{\beta}{2T} \leq \frac{1}{2|Q|}$, the selection step can also be implemented using Algorithm 9 (with $2n$ in place of n), which gives the same bound on η' up to a constant factor using $O(|Q| \log^* n + \log^3 n)$ SVT queries per round in expectation.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

(1) **Claims**

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

Answer: [Yes]

Justification: Yes, the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope. In particular, the claims made match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

(2) **Limitations**

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

Answer: [Yes]

Justification: Yes, the paper discusses relevant lower bounds that establish limitations on the techniques.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

(3) **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Yes, the paper provides the full set of assumptions and a complete and correct proof for all theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

(4) **Experimental result reproducibility**

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

Answer: [N/A]

Justification: The paper does not include experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

(5) **Open access to data and code**

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

Answer: [N/A]

Justification: The paper does not include experiments.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

(6) **Experimental setting/details**

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

Answer: [N/A]

Justification: The paper does not include experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

(7) **Experiment statistical significance**

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

Answer: [N/A]

Justification: The paper does not include experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

(8) **Experiments compute resources**

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

Answer: [N/A]

Justification: The paper does not include experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

(9) **Code of ethics**

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

Answer: [Yes]

Justification: Yes, the research conducted adheres to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

(10) **Broader impacts**

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

Answer: [N/A]

Justification: This work is foundational in nature and thus not tied to any specific application.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

(11) **Safeguards**

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

Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

(12) **Licenses for existing assets**

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

Answer: [N/A]

Justification: The paper does not use existing assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

(13) **New assets**

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

Answer: [N/A]

Justification: The paper does not release new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

(14) **Crowdsourcing and research with human subjects**

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

(16) **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.