

Privately Estimating Black-Box Statistics

Günter F. Steinke*

Thomas Steinke†

Abstract

Standard techniques for differentially private estimation, such as Laplace or Gaussian noise addition, require guaranteed bounds on the sensitivity of the estimator in question. But such sensitivity bounds are often large or simply unknown. Thus we seek differentially private methods that can be applied to arbitrary black-box functions. A handful of such techniques exist, but all are either inefficient in their use of data or require evaluating the function on exponentially many inputs. In this work we present a scheme that trades off between statistical efficiency (i.e., how much data is needed) and oracle efficiency (i.e., the number of evaluations). We also present lower bounds showing the near-optimality of our scheme.

Contents

1	Introduction	2	4	Our Algorithm	16
1.1	Our Contributions	3	4.1	Pure & Concentrated DP Variants	19
1.2	Our Techniques	5	5	Lower Bound	20
2	Related Work	6	6	Discussion	24
2.1	Alternatives to Global Sensitivity	6	6.1	Interpretation	25
2.2	Down-Local Algorithms	8	6.2	Limitations & Further Work . . .	25
2.3	Lower Bounds	9	References	28	
2.4	Miscellaneous	9	A	Shifted Inverse Mechanism	36
3	Preliminaries	9	A.1	Pure DP – Theorem 3.3	37
3.1	Notation	9	A.2	Approximate DP – Theorem 3.4	37
3.2	Differential Privacy	10	A.3	Concentrated DP – Theorem 3.5	38
3.3	Shifted Inverse Mechanism	11	A.4	Noisy Binary Search	38
3.4	Covering Designs	12			

*University of Canterbury gunter.steinke@canterbury.ac.nz

†Google DeepMind steinke@google.com

1 Introduction

Differential privacy [DMNS06] provides a mathematical framework for measuring and controlling the leakage of sensitive information via computations on a private dataset. Given a function f that we wish to evaluate on a private dataset x , the simplest and best-known method for ensuring differential privacy is to add Laplace or Gaussian noise – e.g., $M(x) = f(x) + \xi$, where $\xi \leftarrow \text{Laplace}(\Delta_f/\varepsilon)$ is random noise scaled according to the privacy parameter ε and the (global) sensitivity of f . The *global sensitivity* of f is given by $\Delta_f := \sup_{x,x'} |f(x) - f(x')|$, where the supremum is over all pairs of inputs differing only by the addition or removal of one person’s data.

In many cases, the global sensitivity of the function we want to evaluate is large (or even infinite), or simply unknown. In practice, the function may be given to us as a “black box” – that is, we can only evaluate the function as a “oracle” and not inspect its inner workings, or it may be presented as a piece of untrusted code that is too complicated to analyse. In these cases, we cannot rely on the standard noise addition approach to ensure differential privacy. Many methods have been proposed for privately evaluating functions that go beyond noise addition and are applicable to functions with high (global) sensitivity; see Section 2 for a brief survey. However, all of these methods have various drawbacks (which we briefly discuss next) that have limited practical adoption: Either they require evaluating the function many times (or they require some a priori structural knowledge about the function), or they are statistically inefficient.

Most methods for evaluating functions with high global sensitivity still rely on being able to compute or bound some property of the function, such as smooth sensitivity [NRS07] or distance to instability [DL09], instead of global sensitivity. Hence these methods require either careful analysis of the function or evaluating of the function over a large fraction of its domain (i.e., they may require evaluating the function on exponentially many – or even infinitely many – inputs). This makes these methods impractical in the black-box setting.

A further limitation of the aforementioned methods is that they require evaluating the function on arbitrary inputs that may not correspond to any realistic data. This may “break” the function in the sense that the function may be well-behaved on real data, but could produce arbitrary values if even one input datapoint is corrupted. For example, changing one input can change the mean of a dataset arbitrarily. That is to say, the *local* sensitivity of the function may be large, which implies that, e.g., the smooth sensitivity will also be large. This limitation can be circumvented by using *down-local* algorithms (as we do); down-local algorithms only evaluate the function on (subsets of) the given input [CD20; FDY22; KL23; LRSS25]. However, most down-local algorithms still require evaluating a black-box function on exponentially many subsets of the input.

There is one method that does not have the aforementioned limitations: The sample-and-aggregate framework of Nissim, Raskhodnikova, and Smith [NRS07] only requires evaluating the function on a small number of inputs (with each input consisting only of “real” data) and does not require any structural assumptions about the function, which makes it appealing in practice [PAECT17; PSMRTE18]. The catch is that this method is statistically inefficient in terms of its use of data. That is, if we start with a dataset of size n , then sample-and-

aggregate evaluates the function on datasets of size $O(\varepsilon n)$. (Specifically, the sample-and-aggregate framework partitions the dataset into $O(1/\varepsilon)$ equal-sized parts.) Very roughly, the final accuracy of our private estimate given a dataset of size n is only as good as a non-private estimate on a dataset of size $O(\varepsilon n)$. Thus, if the privacy parameter ε is small, then sample-and-aggregate suffers a significant cost in terms of statistical accuracy.

1.1 Our Contributions

In this article, we examine the tradeoff between statistical efficiency (i.e., how much data is needed to estimate a statistic) and oracle query complexity (i.e., how many times do we need to evaluate the function).

Our main result is a differentially private algorithm which takes a real-valued black-box function f and a private dataset x and evaluates the function on multiple subsets of the input dataset and then outputs an estimate y for the value of the function. Our algorithm interpolates between sample-and-aggregate [NRS07] and more recent computationally inefficient approaches [CD20; FDY22; LRSS25, etc.]

Statistical View on Accuracy: Our work differs from much of the prior work in how we quantify the accuracy of our estimate. Namely, most prior work attempts to ensure $y \approx f(x)$. However, this goal can be too narrow. In many settings, the input x consists of independent and identically distributed (i.i.d.) samples from some distribution $\mathcal{D}$ and our goal is to estimate properties of the distribution $\mathcal{D}$ rather than of the sample x . Thus we take a statistical view of accuracy. Namely, we assume that the input x consists of i.i.d. samples and the function f returns a good estimate with high probability when it is given enough i.i.d. samples; our algorithm then also produces a good estimate.

Theorem 1.1 (Main Result). *Let $\mathcal{Y} \subseteq \mathbb{R}$ be finite and let $\mathcal{X}$ be arbitrary; denote $\mathcal{X}^* = \bigcup_{n \in \mathbb{N}} \mathcal{X}^n$. Let $\varepsilon, \delta > 0$ and $n, m, t \in \mathbb{N}$ satisfy*

$$n \geq m \geq t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|)). \quad (1.1)$$

*Let*¹

$$k = C(n, m, t) < \frac{\binom{n}{t}}{\binom{m}{t}} \left(1 + \log \binom{m}{t} \right) + 1 \leq O \left(\left(\frac{n-t+1}{m-t+1} \right)^t \cdot m \right). \quad (1.2)$$

Then, for all $f : \mathcal{X}^ \rightarrow \mathcal{Y}$, there exists an algorithm $M^f : \mathcal{X}^n \rightarrow \mathcal{Y}$ with the following properties.*

- **Privacy:** M^f is (ε, δ) -differentially private.
- **Statistical Accuracy:** Let $\mathcal{D}$ be an arbitrary probability distribution on $\mathcal{X}$. Suppose $\mathbb{P}_{X \leftarrow \mathcal{D}^{n-m}} [|f(X) - \nu| \geq 1 - \beta] \leq \alpha$. Then $\mathbb{P}_{X \leftarrow \mathcal{D}^n} [|M^f(X) - \nu| \geq \alpha] \geq 1 - k\beta$.

¹We precisely define $C(n, m, t)$ later in Definition 3.6; for now the given upper bound suffices.

- **Oracle Efficiency:** On input $x \in \mathcal{X}^n$, $M^f(x)$ selects k subsets of x , each of size $n - m$, and evaluates f on those subsets; other than these k evaluations, $M^f(x)$ does not depend on either f or x .

Before continuing we make some remarks interpreting Theorem 1.1:

1. Informally, the statistical accuracy guarantee says that, with n private samples, we can get the same accuracy as we could get with $n - m$ non-private samples. (There is an additional factor k blowup in the failure probability, but this is secondary.) Intuitively, the parameter m is the number of samples “wasted” to ensure differential privacy.
2. Making m smaller translates to better statistical accuracy, but it increases k . Making m larger makes k smaller, which means the algorithm requires fewer evaluations of the function f – i.e., lower oracle complexity. This tradeoff is the key phenomenon we study. In particular, the following are three points on the tradeoff curve:
 - (a) Setting $m = \frac{t}{t+1}n$ yields $n - m = \frac{n}{t+1}$ and $k = t + 1$,² which corresponds to sample-and-aggregate [NRS07]. That is, the cost of privacy is a *multiplicative* factor of t in the sample complexity – i.e., with n private samples we get accuracy comparable to $n - m = \Theta(n/t)$ non-private samples. This is the most computationally efficient instantiation of our result.
 - (b) Setting $m = t$ yields $k = \binom{n}{t}$, which corresponds to the results of Linder, Raskhodnikova, Smith, and Steinke [LRSS25]. In this setting, the cost of privacy is an *additive* t samples, at the expense of the number of evaluations k being exponential in t . This is the most statistically efficient instantiation of our result.
 - (c) The above points (a and b) are the extremes on the tradeoff curve; now we consider a setting that interpolates between these: Setting $m = \frac{tn}{t+c}$ yields $k = O(t^c)$. (Here $c \geq 1$ is an appropriate integer.) Relative to (a) sample-and-aggregate, this increases the number of data points $n - m = \frac{cn}{t+c}$ available for each evaluation by a factor of almost c – i.e., we go from $n - m = \Theta(n/t)$ to $n - m = \Theta(cn/t)$ – at the expense of a polynomial increase in the number of evaluations k . This parameter setting is likely of practical interest.
3. Note that the accuracy parameters α and β are not inputs to the algorithm. In a sense, our algorithm “automatically adjusts” to the difficulty of the problem.
4. The value t in Equation 1.1 depends on the privacy parameters ε, δ and on the size of the output space $\mathcal{Y}$. The dependence on the privacy parameters $t \geq \frac{\log(1/\delta)}{\varepsilon}$ is essentially the best we could hope for (see the lower bound below). And the dependence on the size of the output space is extremely mild; $\log^*$ denotes the iterated logarithm, which grows extremely slowly.

²This value of $k = C(n, \frac{t}{t+1}n, t) \leq t + 1$ is tighter than the upper bound given in the theorem statement; see Equation 3.27.

5. Our algorithm has an oracle efficiency guarantee, but not an overall computational efficiency guarantee. That is, we bound the computational cost related to evaluating the function, but not the cost of choosing the subsets and processing the values. See Section 6.2 for further discussion of these limitations.

We show that the number of evaluations of the oracle function f (i.e., k in Theorem 1.1) is roughly optimal:

Theorem 1.2 (Lower Bound). *Let $M^f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ be a randomised algorithm that makes at most k queries to an oracle $f : \mathbb{Z}^* \rightarrow \mathbb{Z}$. Suppose that, for every oracle f , the algorithm M^f satisfies (ε, δ) -differential privacy and the following. Let $\mathcal{D}$ be an arbitrary distribution on $\mathbb{Z}$ and let $\nu \in \mathbb{Z}$. If $\mathbb{P}_{X \leftarrow \mathcal{D}^{n-m}}[f(X) = \nu] \geq 0.999$, then $\mathbb{P}_{X \leftarrow \mathcal{D}^n}[|M^f(X) - \nu| \leq 1] \geq 1/2$. Then we must have*

$$k \geq \frac{\binom{n}{t}}{\binom{m}{t}} \cdot \Omega(1) \quad \text{with} \quad t = \Theta(1/\varepsilon) \quad (1.3)$$

and, simultaneously,

$$k \geq \frac{\binom{n}{t}}{\binom{m}{t}} \cdot \Omega(\delta^{0.01}) \quad \text{with} \quad t = \Theta(\log(1/\delta)/\varepsilon), \quad (1.4)$$

assuming $\delta \leq (\varepsilon/10)^{1.1}$.

Contrasting the lower bounds in Equations 1.3 and 1.4 with the upper bound in Equation 1.2, we see that the combinatorial term $k \approx \frac{\binom{n}{t}}{\binom{m}{t}}$ is present in both the upper and lower bounds. There are some additional factors, but these are relatively minor. The main difference is that the upper bound uses $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$, while the lower bound sets $t = \Omega(1/\varepsilon)$ or $t = \Omega(\log(1/\delta)/\varepsilon)$. Thus there is a multiplicative gap in the parameter t depending on the size of the output space. This is potentially significant, since this multiplicative factor affects the number of queries k in an exponential fashion. We remark that some dependence on the size of the range $\mathcal{Y}$ is known to be necessary for statistical estimation [BNSV15; ALMM19]. Thus, while the $\exp(O(\log^* |\mathcal{Y}|))$ term in the upper bound could potentially be improved, it cannot be removed entirely.

1.2 Our Techniques

Our algorithm can be viewed as an extension of the sample-and-aggregate paradigm [NRS07]. Namely, we evaluate the function on subsets of the input and then we aggregate those values in a way that ensures differential privacy. The novelty is in how we choose the subsets and how we aggregate the values.

Our algorithm has two main technical ingredients: First we use a combinatorial object known as a covering design or a Turán system to select k overlapping subsets of the input on which to evaluate the function. Second, we use a variant of the shifted inverse mechanism of Fang, Dong, and Yi [FDY22] to aggregate the values in a differentially private manner.

The property of the covering design is that if t out of the n input datapoints are corrupted, then at least one of the k subsets on which we evaluate the function will not contain any corrupted datapoints. (Note that this property holds without knowing which datapoints are corrupted.) Intuitively, (ε, δ) -differential privacy requires robustness to $t = O(\log(1/\delta)/\varepsilon)$ corruptions. And the covering design ensures this level of robustness, but only in the weak sense that one out of k values is uncorrupted.

It only remains to aggregate the values in a way that this form of robustness translates into differential privacy. Computing a differentially private mean or median of the k values does not suffice, since a single datapoint could affect a majority of the values. This is where the shifted inverse mechanism fits in.

To illustrate how the shifted inverse mechanism works, consider the special case with a binary output space $\mathcal{Y} = \{0, 1\}$. (The general case can, with some loss, be reduced to this case.) Now we ask “how many input datapoints would I need to remove so that all of the remaining output values are all 0?” (That is, if we remove a datapoint, then all of the output values that depend on that datapoint are removed.) If all of the output values are 0, then the answer to the query is 0 – i.e., no datapoints need to be removed. If all of the output values are 1, then the answer to the query is at least t , by the properties of the covering design. By construction, this query has sensitivity 1; thus the answer to the query can be approximated in a differentially private manner by adding Laplace or Gaussian noise. As long as t is large enough, we can accurately distinguish between the case where all of the output values are 0 and the case where all of them are 1. (When some values are 0 and some are 1, the outcome is indeterminate.) Assuming each value is individually accurate with high probability, the aggregated value will also be accurate with high probability.

Our lower bound mirrors the intuition for our algorithm. That is, (ε, δ) -differential privacy requires robustness to $t = O(\log(1/\delta)/\varepsilon)$ corruptions. To be precise, we perform a packing argument [HT10]. That is, we use group privacy to argue that changing t values cannot totally change the output of the algorithm. We restrict the algorithm to evaluating the function on subsets of the input of size $n - m$. (We can force this restriction using standard tricks, such as making the function fail when given an input that isn’t of this form.) Now the only way for the algorithm to satisfy the given level of robustness is for it to query enough sets so that at least one of them contains none of the t corrupted points. Roughly, this implies that the subsets queried by the algorithm must form a covering design. The lower bound then follows.

2 Related Work

2.1 Alternatives to Global Sensitivity

Adding Laplace or Gaussian noise scaled to global sensitivity has been the standard approach to ensure differential privacy since its inception [DMNS06; DKMMN06]. And this approach is surprisingly versatile. However, there has been a long line of work seeking methods that circumvent the limitations of global sensitivity, to which the present article adds. We now

briefly survey the most closely related approaches.

Nissim, Raskhodnikova, and Smith [NRS07] introduced two methods that go beyond global sensitivity – smooth sensitivity and sample-and-aggregate (which we discuss later in this section). Whereas global sensitivity depends only on the function and not on the dataset, **smooth sensitivity** is a dataset-dependent value. We can achieve differential privacy by adding noise that scales with the smooth sensitivity and the smooth sensitivity may be lower than the global sensitivity. In slightly more detail: Smooth sensitivity seeks to approximate the *local sensitivity* – i.e., how much can the function change by adding or removing one person’s data to or from the actual dataset at hand. For example, the median of a set of real numbers has unbounded global sensitivity, but its local sensitivity is bounded by the gap between the median value and the two values immediately before or after the median value in sorted order; indeed the local sensitivity of the median could even be zero if the median value is repeated multiple times. Ideally, we could add noise scaled to the local sensitivity, rather than to the global sensitivity, but the scale of the noise could itself be sensitive. Smooth sensitivity circumvents this issue by upper bounding the local sensitivity in a way that is itself low sensitivity (in a multiplicative, rather than additive, sense). A major disadvantage of the smooth sensitivity approach is that it is often challenging to compute the smooth sensitivity, as it is still a global property of the function. In general, to compute the smooth sensitivity, we need to know the value of the function over its entire domain.

Dwork and Lei [DL09] introduced the **propose-test-release** framework. Like smooth sensitivity, this framework seeks to exploit low local sensitivity. This framework begins a priori with a proposed upper bound on the local sensitivity. Then it performs a test to check whether or not this bound is correct. If the test passes, then it releases the value with noise scaled according to the bound. As long as the test is unlikely to yield a false positive, this guarantees differential privacy. A general recipe for the test step is to measure the distance (in terms of adding or removing people) from the given dataset to the nearest dataset with local sensitivity higher than the proposed bound. This distance is inherently low-sensitivity and so can be estimated privately. If the distance is large enough, the test will pass. Dwork and Lei [DL09] made the connection between differential privacy and robust statistics, which later work expanded upon [KLSU19; BS19; AMB19; KSU20; BAM20; LKKO21; BGSUZ21; GKMN21; TCKMS22; HKM22; LKO22; AL22; SV22; GH22; KMV22; AUZ23; AKTVZ23; LJKOS23; CHLLN23; Kam24; BZ25].

The **inverse sensitivity mechanism** [AD20; MMNW11; JS13; Ste23a] provides a loss function with low global sensitivity for any function of interest; the exponential mechanism [MT07] can then be applied to estimate the value of interest. The idea is simple: Given a function f , a dataset x , and a value y , define the loss $\ell(x, y) = \min\{|x \setminus x'| + |x' \setminus x| : f(x') = y\}$ to be the least number of elements of x that need to be added or removed in order the change the function value to y . Clearly, $\ell(x, y) = 0$ if and only if $y = f(x)$. Furthermore, if the local sensitivity of f at x is small, then $\ell(x, y) \leq 1$ implies $y \approx f(x)$. More generally, if f is appropriately well-behaved near x , then any approximate minimiser y of the loss $\ell(x, y)$ must be a good approximation to $f(x)$.

Another approach is to replace the function of interest with a function that has low

global sensitivity and which still provides a good approximation to the original function. For example, when computing a sum, we might clip the values to ensure that they are bounded; if the clipping threshold is chosen appropriately, this ensures low global sensitivity and doesn’t change the value of the function too much. In general, **Lipschitz extensions** provide low-sensitivity approximating functions that can be used in the differentially private setting [KNRS13; BBDS13; RS15; RS16b; RS16a].

Limitations: All of the aforementioned methods suffer from computational intractability. They are only practical in special cases where we can analytically compute the relevant bounds. The underlying reason for this is that the quantities of interest all involve some universal quantification over datasets. In general, implementing these methods requires enumerating exponentially many inputs – or even infinitely many. In particular, they are not practical for functions that are given to us as a black box.

2.2 Down-Local Algorithms

Recent work [CD20; FDY22; KL23; LRSS25] has sought to overcome the aforementioned limitations by devising algorithms that are **down-local** in the sense that, given a function f and a dataset x , the algorithm only evaluates f on subsets of x , rather than on arbitrary points in its domain.

In particular, down-local algorithms only evaluate the function on “real” data; this is an added benefit, since many functions might “break” when given arbitrary inputs. For example, even the mean has infinite *local* sensitivity, when we allow unbounded inputs, even though it may be well-behaved for real inputs.

Cummings and Durfee [CD20] effectively construct a Lipschitz extension by evaluating the function on all subsets of the input. This gives runtime which is “only” exponential in the number of input data points, and does not depend on the size of the function’s domain. (In special cases, like the mean and median, they give polynomial-time algorithms.)

Fang, Dong, and Yi [FDY22] presented the **shifted inverse mechanism**, which is a down-local version of the inverse sensitivity mechanism. However, their method only applies to monotone functions; this restriction was removed by Linder, Raskhodnikova, Smith, and Steinke [LRSS25]. Our results are based on this approach.

Kohli and Laskowski [KL23] present an algorithm (which they call **TAHOE**) that is, roughly, a down-local version of propose-test-release.

Sample-and-aggregate [NRS07] is closely related to our approach. In its simplest form,³ sample-and-aggregate partitions the dataset into smaller subdatasets, evaluates the function of interest on each subdataset, and then aggregates the function values in a differentially private manner (e.g., using smooth sensitivity applied to the aggregation function). A single person’s data will be in only one of the subdatasets and so a single person can only affect one of the values. The advantage of the sample and aggregate approach is that

³More generally, sample-and-aggregate allows overlapping subdatasets, but then the aggregator must handle the fact that a single person’s data may affect multiple function values.

requires no structural knowledge about the function of interest and is computationally efficient. (Intuitively, it pushes the privacy analysis onto the aggregation function, rather than the function we want to evaluate.) The downside is that we evaluate the function on the smaller subdatasets. This can lead to significant loss in accuracy relative to (non-privately) evaluating the function on the whole dataset. Our algorithm addresses the downside of sample and aggregate by allowing us to evaluate the function on larger subdatasets, at the expense of requiring us to evaluate on more of these subdatasets.

2.3 Lower Bounds

Linder, Raskhodnikova, Smith, and Steinke [LRSS25] prove lower bounds on both locality and query complexity, which are similar in spirit to our Theorem 1.2. (Locality refers to $|x \setminus x'|$ where x is the input and $x' \subseteq x$ is the subset on which the function is evaluated.) The main difference between our lower bound and their query complexity lower bound is the notion of accuracy. Theorem 1.2 assumes a statistical accuracy guarantee – that is, if $\mathbb{P}_{X \leftarrow \mathcal{D}^{n-m}}[f(X) = \nu] \geq 0.999$, then $\mathbb{P}_{X \leftarrow \mathcal{D}^n}[|M^f(X) - \nu| \leq 1] \geq \frac{1}{2}$ for arbitrary $\mathcal{D}$. In contrast, they [LRSS25, Theorem 6.1] assume an accuracy guarantee of the form $\mathbb{P}[|M(x) - f(x)| \leq \alpha] \geq 1 - \beta$ for arbitrary x under the promise that f is Lipschitz. Thus these results are formally incomparable.

2.4 Miscellaneous

Our algorithms rely on combinatorial objects known as covering designs. Combinatorial designs appear in many places. Notably, Park, Asodeh, and Lee [PAL24] used balanced incomplete block designs to develop minimax-optimal locally differentially private algorithms. Furthermore, Gentle [Gen25] showed that these combinatorial designs are in fact necessary to achieve optimality.

3 Preliminaries

3.1 Notation

For a natural number $n \in \mathbb{N}$, we denote $[n] := \{1, 2, \dots, n\}$. We use $\log$ to denote the natural logarithm. Throughout, we will let $\mathcal{X}$ denote the set of possible input data points. Then $\mathcal{X}^n$ denotes tuples of length n and $\mathcal{X}^* := \bigcup_{n \in \mathbb{N}} \mathcal{X}^n$ denotes tuples of arbitrary length.

We treat tuples as sets (and we use set notation), but we also maintain consistent indexing of the elements. This should be intuitive, but to be completely formal, below we define the set notation that we use on tuples. The reader should skip this subsection and only refer back if there is any confusion.

We assume that there is a special “null” element $\perp \in \mathcal{X}$. Informally, $\perp$ represents a missing element when the tuple is viewed as a set. (And we assume that $\perp$ is not in the

support of the data distribution $\mathcal{D}$.) For tuples $x, x' \in \mathcal{X}^n$, we define the following set notations:

1. The size of x is the number of non-null elements:

$$|x| := |\{i \in [n] : x_i \neq \perp\}|. \quad (3.1)$$

2. A subset corresponds to replacing elements with nulls:

$$x' \subseteq x \iff \forall i \in [n] (x'_i = x_i \vee x'_i = \perp). \quad (3.2)$$

3. Intersections and differences are given by

$$\forall i \in [n] \quad (x \cap x')_i = \begin{cases} x_i & \text{if } x_i = x'_i \\ \perp & \text{if } x_i \neq x'_i \end{cases} \quad (3.3)$$

and

$$\forall i \in [n] \quad (x \setminus x')_i = \begin{cases} \perp & \text{if } x_i = x'_i \\ x_i & \text{if } x_i \neq x'_i \end{cases}. \quad (3.4)$$

We have $x \cap x' \subseteq x$, $x \cap x' \subseteq x'$, and $x \setminus x' \subseteq x$. Also, $|x \setminus x'| + |x' \setminus x| = |\{i \in [n] : x_i \neq x'_i = \perp \vee \perp = x_i \neq x'_i\}| + 2|\{i \in [n] : \perp \neq x_i \neq x'_i \neq \perp\}|$.

4. Given a set of indices $S \subseteq [n]$, define $x_S \in \mathcal{X}^n$ by

$$\forall i \in [n] \quad (x_S)_i = \begin{cases} x_i & \text{if } i \in S \\ \perp & \text{if } i \notin S \end{cases}. \quad (3.5)$$

Note that $x_S \subseteq x$ for all S and, assuming $|x| = n$, we have $|x_S| = |S|$.

We work with functions $f : \mathcal{X}^* \rightarrow \mathcal{Y}$ and we assume that null values are equivalent to removing elements from the tuple entirely. That is, for all $n \in \mathbb{N}$, $x \in \mathcal{X}^n$, and $i \in [n+1]$, if $x' = (x_1, x_2, \dots, x_{i-1}, \perp, x_i, x_{i+1}, \dots, x_n) \in \mathcal{X}^{n+1}$, then $f(x) = f(x')$.

3.2 Differential Privacy

We say that $x, x' \in \mathcal{X}^n$ are *neighbouring* if $|x \setminus x'| + |x' \setminus x| = 1$. Equivalently, $x, x' \in \mathcal{X}^n$ are *neighbouring* if there exists $i \in [n]$ such that $x_i = \perp$ or $x'_i = \perp$ and, for all $j \in [n] \setminus \{i\}$, we have $x_j = x'_j$. Informally, neighbouring inputs differ by the addition or removal of one element, which corresponds to one person's data.

Definition 3.1 (Differential Privacy [DMNS06; DKMMN06]). *A randomised algorithm $M : \mathcal{X}^n \rightarrow \mathcal{Y}$ satisfies (ε, δ) -differential privacy if, for all neighbouring $x, x' \in \mathcal{X}^n$ and all measurable $V \subseteq \mathcal{Y}$,*

$$\mathbb{P}[M(x) \in V] \leq e^\varepsilon \mathbb{P}[M(x') \in V] + \delta. \quad (3.6)$$

“Pure differential privacy” (or “pointwise differential privacy”) refers to the setting where $\delta = 0$. In contrast “approximate differential privacy” refers to the setting where $\delta > 0$.

Differential privacy satisfies many useful properties. One is postprocessing – applying an arbitrary function to the output of a differentially private algorithm still results in a differentially private output, with no loss in parameters. The other property we use is group privacy:

Lemma 3.2 (Group privacy). *Suppose $M : \mathcal{X}^n \rightarrow \mathcal{Y}$ is (ε, δ) -differentially private. Suppose $x, x' \in \mathcal{X}^n$ are distance $t = |x \setminus x'| + |x' \setminus x|$ apart. Then, for all measurable $V \subseteq \mathcal{Y}$,*

$$\mathbb{P}[M(x) \in V] \leq e^{t\varepsilon} \mathbb{P}[M(x') \in V] + \frac{e^{t\varepsilon} - 1}{e^\varepsilon - 1} \delta. \quad (3.7)$$

Note that we do not allow *replacement* of one person’s data between neighbours – this would instead be group privacy at distance $t = 2$.

3.3 Shifted Inverse Mechanism

The basis of our algorithm is the shifted inverse mechanism of Fang, Dong, and Yi [FDY22]. For completeness, we review this algorithm in Appendix A.

Theorem 3.3 (Shifted Inverse Mechanism – Pure DP). *Let $g : \mathcal{X}^* \rightarrow \mathcal{Y}$ be monotone – i.e., $x' \subseteq x \implies g(x') \leq g(x)$ – where $\mathcal{Y} \subseteq \mathbb{R}$ is finite. Let $\varepsilon, \beta > 0$. Then there exists a $(\varepsilon, 0)$ -differentially private $M : \mathcal{X}^* \rightarrow \mathcal{Y}$ such that, for all $x \in \mathcal{X}^*$, we have*

$$\mathbb{P}_M[g(x) \geq M(x) \geq \min\{g(x') : x' \subseteq x, |x'| \geq |x| - t\}] \geq 1 - \beta, \quad (3.8)$$

where $t = 2 \left\lceil \frac{2}{\varepsilon} \log \left(\frac{|\mathcal{Y}|}{\beta} \right) \right\rceil$. Furthermore, $M(x)$ only depends on the values $g(x')$ for $x' \subseteq x$.

We use a variant of the algorithm satisfying approximate differential privacy [LRSS25; Ste23b]:

Theorem 3.4 (Shifted Inverse Mechanism – Approx DP). *Let $g : \mathcal{X}^* \rightarrow \mathcal{Y}$ be monotone – i.e., $x' \subseteq x \implies g(x') \leq g(x)$ – where $\mathcal{Y} \subseteq \mathbb{R}$ is finite. Let $\varepsilon, \delta > 0$. Then there exists a (ε, δ) -differentially private $M : \mathcal{X}^* \rightarrow \mathcal{Y}$ such that, for all $x \in \mathcal{X}^*$, we have*

$$\mathbb{P}_M[g(x) \geq M(x) \geq \min\{g(x') : x' \subseteq x, |x'| \geq |x| - t\}] = 1, \quad (3.9)$$

where $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$ and $\log^*$ denotes the iterated logarithm.⁴ Furthermore, $M(x)$ only depends on the values $g(x')$ for $x' \subseteq x$.

We also consider a variant that satisfies Concentrated Differential Privacy [DR16; BS16] or Gaussian Differential Privacy [DRS22]:

⁴The iterated logarithm is an extremely slow-growing function. It is the inverse of the exponential tower function, which satisfies the recurrence $\text{tower}(n+1) = 2^{\text{tower}(n)}$.

Theorem 3.5 (Shifted Inverse Mechanism – zCDP/GDP). *Let $g : \mathcal{X}^* \rightarrow \mathcal{Y}$ be monotone – i.e., $x' \subseteq x \implies g(x') \leq g(x)$ – where $\mathcal{Y} \subseteq \mathbb{R}$ is finite. Let $\rho, \beta > 0$. Then there exists $M : \mathcal{X}^* \rightarrow \mathcal{Y}$ satisfying ρ -zCDP and $\sqrt{2\rho}$ -GDP such that, for all $x \in \mathcal{X}^*$, we have*

$$\mathbb{P}_M [g(x) \geq M(x) \geq \min \{g(x') : x' \subseteq x, |x'| \geq |x| - t\}] \geq 1 - \beta, \quad (3.10)$$

where $t = O(\sqrt{\log(|\mathcal{Y}|/\beta)/\rho})$. Furthermore, $M(x)$ only depends on the values $g(x')$ for $x' \subseteq x$.

To the best of our knowledge, Theorem 3.5 is novel (although it follows from known techniques); thus we discuss it in more detail in Appendix A.

3.4 Covering Designs

Our algorithm also depends on a combinatorial object which is known as a covering design.

Definition 3.6 (Covering Design). *Given $n, m, t \in \mathbb{N}$, $t \leq m \leq n$, a (n, m, t) -covering design of size k is a collection of sets $S_1, S_2, \dots, S_k \subseteq [n]$ each of size $|S_i| = m$ with the property that, for every $T \subseteq [n]$ of size $|T| \leq t$, there exists $i \in [k]$ such that $T \subseteq S_i$. We let $C(n, m, t)$ denote the smallest k for which a (n, m, t) -covering design of size k exists.*

Covering designs are equivalent to what is known as Turán systems [Sid95]. To be precise, if $S_1, S_2, \dots, S_k$ is a (n, m, t) -covering design, then $[n] \setminus S_1, [n] \setminus S_2, \dots, [n] \setminus S_k$ is a $(n, n - t, n - m)$ -Turán system and vice versa. Such a Turán system has the property that for every $T \subseteq [n]$ of size $|T| = n - t$, there exists $i \in [k]$ such that $T \supseteq [n] \setminus S_i$.

In general, we do not have optimal constructions or even existential results for covering designs. However, the following result gives reasonable bounds.

Proposition 3.7. *For all $n, m, t \in \mathbb{N}$ with $n \geq m \geq t$ we have*

$$\frac{\binom{n}{t}}{\binom{m}{t}} \leq C(n, m, t) < \frac{\binom{n}{t}}{\binom{m}{t}} \left(1 + \log \binom{m}{t} \right) + 1. \quad (3.11)$$

Proof. The lower bound is due to Schönheim [Sch64] and the upper bound is due to Erdős and Spencer [ES74, Theorem 13.4]. Both proofs rely on the probabilistic method. We recap both proofs for completeness.

First the lower bound: We claim that $C(n, m, t) \geq \frac{n}{m} C(n - 1, m - 1, t - 1)$ for all $n \geq m \geq t \geq 1$.⁵ Induction then gives

$$C(n, m, t) \geq \frac{n}{m} C(n - 1, m - 1, t - 1) \geq \frac{n}{m} \frac{n - 1}{m - 1} C(n - 2, m - 2, t - 2) \geq \dots \geq \prod_{i=0}^{t-1} \frac{n - i}{m - i} = \frac{\binom{n}{t}}{\binom{m}{t}}. \quad (3.12)$$

⁵Here we define $C(n, m, 0) = 1$ to make the induction work for $t = 1$. (A $(n, m, 0)$ -covering design needs one set $S_1 \subseteq [n]$ that “contains” the empty set $\emptyset \subseteq S_1$.)

The claim follows from the following argument. Let $S_1, S_2, \dots, S_k \subseteq [n]$ be a (n, m, t) -covering design of size $k = C(n, m, t)$. Pick $U \in [n]$ uniformly at random. For each $i \in [k]$, if $U \in S_i$, we remove the element U and call the new set $\widehat{S}_i = S_i \setminus \{U\}$; if $U \notin S_i$, we discard S_i . Next we renumber $[n] \setminus \{U\}$ to map to $[n-1]$ and reindex so that $\widehat{S}_1, \dots, \widehat{S}_{\widehat{k}}$ excludes the discarded indices i with $U \notin S_i$. Now $\widehat{S}_1, \dots, \widehat{S}_{\widehat{k}} \subseteq [n-1]$ is a $(n-1, m-1, t-1)$ -covering design. In particular, any $\widehat{T} \subseteq [n-1]$ of size $|\widehat{T}| \leq t-1$ can be extended to $T \subseteq [n]$ with $|T| = |\widehat{T}| + 1$ (by adding $U \in T$) such that there exists $i \in [k]$ with $T \subseteq S_i$, which implies $\widehat{T} \subseteq \widehat{S}_i$. The size $\widehat{k}$ of the new covering design depends on U . Specifically, $\widehat{k} = |\{i \in [k] : U \in S_i\}|$. Since U is uniformly random, we have $\mathbb{E}[\widehat{k}] = \sum_{i \in [k]} \mathbb{P}[U \in S_i] = \sum_{i \in [k]} \frac{|S_i|}{n} = \frac{m}{n}k$. There must exist a fixed choice of U such that $\widehat{k} \leq \frac{m}{n}k$, which rearranges to $C(n, m, t) = k \geq \frac{n}{m}\widehat{k} \geq \frac{n}{m}C(n-1, m-1, t-1)$, as required.

Second the upper bound: Let $S_1, S_2, \dots, S_{k_1} \subseteq [n]$ be independent and uniformly random of size $|S_i| = m$ for each $i \in [k_1]$. For any fixed $T \subseteq [n]$ of size $|T| = t$, we have

$$\mathbb{P}[\nexists i \in [k_1] \text{ } T \subseteq S_i] = \prod_{i \in [k_1]} (1 - \mathbb{P}[T \subseteq S_i]) = \left(1 - \frac{\binom{n-t}{m-t}}{\binom{n}{m}}\right)^{k_1}. \quad (3.13)$$

Let K_2 be the number of sets $T \subseteq [n]$ of size $|T| = t$ such that there is no $i \in [k_1]$ with $T \subseteq S_i$. We have

$$\mathbb{E}[K_2] = \binom{n}{t} \mathbb{P}[\nexists i \in [k_1] \text{ } T \subseteq S_i] = \binom{n}{t} \left(1 - \frac{\binom{n-t}{m-t}}{\binom{n}{m}}\right)^{k_1}, \quad (3.14)$$

where $T \subseteq [n]$ with $|T| = t$ is arbitrary.⁶ Now we can create a (n, m, t) -covering design of size $k_1 + K_2$ by taking $S_1, \dots, S_{k_1}$ and, for each uncovered $T \subseteq [n]$ of size $|T| = t$, adding an additional set that covers it. There must exist a fixed choice of the randomness such that $K_2 \leq \lfloor \mathbb{E}[K_2] \rfloor$. Thus we have

$$\begin{aligned} C(n, m, t) &\leq k_1 + \left\lceil \binom{n}{t} \left(1 - \frac{\binom{n-t}{m-t}}{\binom{n}{m}}\right)^{k_1} \right\rceil \\ &= k_1 + \left\lceil \binom{n}{t} \left(1 - \frac{\binom{m}{t}}{\binom{n}{t}}\right)^{k_1} \right\rceil \quad (\text{simplifying}) \\ &\leq k_1 + \left\lceil \binom{n}{t} \exp\left(-k_1 \frac{\binom{m}{t}}{\binom{n}{t}}\right) \right\rceil. \quad (1 - x \leq \exp(-x)) \end{aligned} \quad (3.15)$$

⁶If $K_2 = 0$, then $S_1, \dots, S_{k_1}$ form a covering design (i.e., there is no need to add more sets). Since $K_2 \geq 0$ is an integer, if $\mathbb{E}[K_2] < 1$, then $K_2 = 0$ happens with nonzero probability; this already proves $C(n, m, t) \leq \frac{\binom{n}{t}}{\binom{m}{t}} \log \binom{n}{t}$ [CP96]. Going beyond existential results, if $\mathbb{E}[K_2]$ is small (e.g., ≤ 0.01), then $S_1, \dots, S_{k_1}$ form a covering design with high probability (e.g., ≥ 0.99).

Setting $k_1 = \left\lceil \frac{\binom{n}{t}}{\binom{m}{t}} \log \binom{m}{t} \right\rceil$ yields

$$C(n, m, t) \leq \left\lceil \frac{\binom{n}{t}}{\binom{m}{t}} \log \binom{m}{t} \right\rceil + \left\lfloor \frac{\binom{n}{t}}{\binom{m}{t}} \right\rfloor < \frac{\binom{n}{t}}{\binom{m}{t}} \left(\log \binom{m}{t} + 1 \right) + 1, \quad (3.16)$$

which is the desired result.⁷ □

We remark that the lower bound in Proposition 3.7 can be improved: Noting that $C(n, m, t)$ must be an integer, the inductive step can be improved to $C(n, m, t) \geq \left\lceil \frac{n}{m} C(n-1, m-1, t-1) \right\rceil$ for all $n \geq m \geq t \geq 1$. Equation 3.12 then becomes

$$C(n, m, t) \geq \left\lceil \frac{n}{m} C(n-1, m-1, t-1) \right\rceil \geq \left\lceil \frac{n}{m} \left\lceil \frac{n-1}{m-1} \left\lceil \frac{n-2}{m-2} \left[\cdots \left\lceil \frac{n-t+1}{m-t+1} \right\rceil \cdots \right] \right\rceil \right\rceil \right\rceil \quad (3.17)$$

We also state (looser) bounds without binomial coefficients:

Corollary 3.8. *For all $n, m, t \in \mathbb{N}$ with $n \geq m \geq t > 1$ we have*

$$\begin{aligned} \left(\frac{n}{m} \right)^t &\leq \left(\frac{2n-t+1}{2m-t+1} \right)^t \leq C(n, m, t) < \left(\frac{n}{m} \cdot \frac{n-t+1}{m-t+1} \right)^{t/2} \cdot \min \left\{ \begin{array}{l} 1+t+t \log(m/t), \\ 1+m \log 2, \\ 1+t \log m \end{array} \right\} + 1 \\ &\leq \left(\frac{n-t+1}{m-t+1} \right)^t \cdot \min\{1+m, 1+t \log m\} + 1. \end{aligned} \quad (3.18)$$

Proof. Proposition 3.7 states that

$$\frac{\binom{n}{t}}{\binom{m}{t}} \leq C(n, m, t) \leq \frac{\binom{n}{t}}{\binom{m}{t}} \left(1 + \log \binom{m}{t} \right) + 1. \quad (3.19)$$

We have $\binom{m}{t} \leq \min\{(em/t)^t, 2^m, m^t\}$. It only remains to bound

$$\frac{\binom{n}{t}}{\binom{m}{t}} = \prod_{i=0}^{t-1} \frac{n-i}{m-i} = \exp \left(\sum_{i=0}^{t-1} f(i) \right), \quad (3.20)$$

where we define $f(x) = \log \left(\frac{n-x}{m-x} \right) = \log(n-x) - \log(m-x)$. For $0 \leq x < m \leq n$, we have $f'(x) = \frac{-1}{n-x} - \frac{-1}{m-x} = \frac{n-m}{(m-x)(n-x)} \geq 0$ and $f''(x) = \frac{-1}{(n-x)^2} - \frac{-1}{(m-x)^2} = \frac{(n-x)^2 - (m-x)^2}{(n-x)^2(m-x)^2} =$

⁷Erdős and Spencer [ES74] state the result without the +1 at the end, but it is not clear to us how they obtain this result. Their proof ignores the need to round k_1 to an integer and they leave fixing this issue as an exercise.

$\frac{(n-m)((n-x)+(m-x))}{(n-x)^2(m-x)^2} \geq 0$. Thus f is an increasing and convex function on $[0, m)$. Using $\log(\frac{n}{m}) = f(0) \leq f(i) \leq f(t-1) = \log(\frac{n-t+1}{m-t+1})$ gives

$$\left(\frac{n}{m}\right)^t \leq \frac{\binom{n}{t}}{\binom{m}{t}} = \exp\left(\sum_{i=0}^{t-1} f(i)\right) \leq \left(\frac{n-t+1}{m-t+1}\right)^t. \quad (3.21)$$

Jensen's inequality gives

$$f\left(\frac{t-1}{2}\right) = f\left(\frac{1}{t} \sum_{i=0}^{t-1} i\right) \leq \frac{1}{t} \sum_{i=0}^{t-1} f(i) \leq \frac{1}{t} \sum_{i=0}^{t-1} \frac{(t-1-i)f(0) + if(t-1)}{t-1} = \frac{f(0) + f(t-1)}{2}, \quad (3.22)$$

which rearranges to

$$\left(\frac{n - \frac{t-1}{2}}{m - \frac{t-1}{2}}\right)^t \leq \frac{\binom{n}{t}}{\binom{m}{t}} = \exp\left(\sum_{i=0}^{t-1} f(i)\right) \leq \left(\frac{n}{m} \cdot \frac{n-t+1}{m-t+1}\right)^{t/2}, \quad (3.23)$$

as required. $\square$

Roughly speaking, the lower bound in Proposition 3.7 is tighter than the upper bound. When the lower bound is exactly tight, the covering design is known as a Steiner system (which satisfies the stricter property that each $T \subseteq [n]$ of size $|T| = t$ is contained in *exactly* one S_i , rather than *at least* one). Results on the existence of Steiner systems [Kee14; Kee24] imply that the lower bound is exactly tight infinitely often. More specifically, for any fixed integers $m \geq t \geq 1$, there exists n_0 such that $C(n, m, t) = \frac{\binom{n}{t}}{\binom{m}{t}}$ for all $n \geq n_0$ satisfying $\frac{\binom{n-i+1}{t-i+1}}{\binom{m-i+1}{t-i+1}} \in \mathbb{Z}$ for all $i \in [t]$. More generally, for any fixed integers $m \geq t \geq 1$, we have [Rö85]

$$C(n, m, t) \leq (1 + o(1)) \frac{\binom{n}{t}}{\binom{m}{t}} \quad \text{as } n \rightarrow \infty. \quad (3.24)$$

In particular, the lower bound is tight at the extreme choices of m : When $m = n$, we have $C(n, n, t) = 1$. And, when $m = t$, we have

$$C(n, t, t) = \binom{n}{t}. \quad (3.25)$$

The setting where m, t are fixed and $n \rightarrow \infty$ is of interest for our work. However, we are more interested in the setting where the ratio n/m is constant and $n, m \rightarrow \infty$. A simple result that helps grapple with this setting is

$$\forall \ell, n, m, t \in \mathbb{N} \quad C(\ell n, \ell m, t) \leq C(n, m, t). \quad (3.26)$$

Equation 3.26 follows by partitioning $[\ell n]$ into n chunks of size ℓ and taking a (n, m, t) -covering design and applying it to the chunks instead of individual points. Combining Equations 3.25 and 3.26 gives

$$C(n, \frac{t}{t+1}n, t) = C(\ell(t+1), \ell t, t) \leq C(t+1, t, t) = \binom{t+1}{t} = t+1, \quad (3.27)$$

where $\ell = \frac{n}{t+1}$ is assumed to be an integer.

4 Our Algorithm

Algorithm 4.1 Differentially Private Black-box Estimator

procedure ESTIMATE($f : \mathcal{X}^n \rightarrow \mathcal{Y}, x \in \mathcal{X}^n, \varepsilon > 0, \delta > 0, S_1, \dots, S_k \subseteq [n]$)

Let $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$ as in Theorem 3.4.

Assert that $S_1, S_2, \dots, S_k \subseteq [n]$ is a (n, m, t) -covering design (Definition 3.6).

▷ $n \geq m \geq t, k \geq C(n, m, t)$.

Compute $f(x_{[n] \setminus S_i})$ for each $i \in [k]$.

▷ Only values of f we need.

Define $g : \mathcal{X}^n \rightarrow \mathcal{Y}$ by

$$g(x') := \max\{f(x'_{[n] \setminus S_i}) : i \in [k], |x'_{[n] \setminus S_i}| = n - |S_i|\}. \quad (4.1)$$

▷ Define $\max \emptyset := \min \mathcal{Y}$.

Let M be the Shifted Inverse Mechanism from Theorem 3.4 applied to g .

return $M(x) \in \mathcal{Y}$.

end procedure

Our algorithm is specified in Algorithm 4.1. We can see from the algorithm description that the shifted inverse mechanism (§3.3) is the main ingredient. In terms of the analysis, we must check three things:

- (i) The function g is monotone, as required for the shifted inverse mechanism.
- (ii) The k evaluations of f suffice for all computations.
- (iii) The accuracy guarantee of the shifted inverse mechanism translates to the desired accuracy of our algorithm.

We address these claims in the following two lemmata. But first we provide some intuition for our choice of the function g :

Ideally, we want $g = f$, but we need g to be monotone. A natural way to monotone f is to set $g(x') = \max\{f(\tilde{x}) : \tilde{x} \subseteq x'\}$. For example, if $f(x) = \sum_i x_i$ is the sum, then the corresponding monotone g would simply be the sum over positive terms $g(x) = \sum_i \max\{x_i, 0\}$. The main issue with this monotone g requires

evaluating f exponentially many times. We fix this issue by only evaluating f on carefully-chosen subsets of the input. This is where the covering design $S_1, \dots, S_k$ enters the picture. Setting $g(x') = \max\{f(x'_{[n] \setminus S_i}) : i \in [k]\}$ *almost* works – the subtlety is that the shifted inverse mechanism evaluates $g(x')$ for many $x' \subseteq x$. We add the restriction $|x'_{[n] \setminus S_i}| = n - |S_i|$ to address this subtlety and ensure that evaluating $f(x_{[n] \setminus S_i})$ for $i \in [k]$ suffices to compute $g(x')$ for all $x' \subseteq x$.

Next: Why choose the subsets to form a covering design? Monotonicity (and therefore privacy) holds for any choice of subsets. And for oracle efficiency we just want to minimize the number of subsets k . The last requirement is statistical accuracy. For this we want each individual element in the maximum $f(x'_{[n] \setminus S_i})$ to be statistically accurate, which means we want $[n] \setminus S_i$ to be large, so we want S_i to be small. Finally, the shifted inverse mechanism's accuracy guarantee requires g to have low down-sensitivity.⁸ This translates to the covering requirement – each small $T \subseteq [n]$ must be contained in some S_i .

Lemma 4.1. *The function g defined by Equation 4.1 in Algorithm 4.1 is monotone – i.e., $x' \subseteq x \implies g(x') \leq g(x)$. Furthermore, all values $g(x')$ for $x' \subseteq x$ can be computed from the values $f(x_{[n] \setminus S_i})$ for $i \in [k]$.*

Proof. Recall $g : \mathcal{X}^n \rightarrow \mathcal{Y}$ is given by

$$g(x') := \max\{f(x'_{[n] \setminus S_i}) : i \in [k], |x'_{[n] \setminus S_i}| = n - |S_i|\}, \quad (4.1)$$

where we set $\max \emptyset = \min \mathcal{Y}$ to cover the corner case. The requirement $|x'_{[n] \setminus S_i}| = n - |S_i|$ is equivalent to requiring that there are no null values in $x'_{[n] \setminus S_i}$ – i.e., $\forall j \in [n] \setminus S_i, x'_j \neq \perp$. (Recall how size is defined in Equation 3.1.)

Thus, if we remove an element from the input, then this removes from the maximum all function values that depend on the removed input. That is, if we remove, say, the j -th element x'_j from the input x' by replacing it with $\perp$, then this removes from the maximum all indices $i \in [k]$ such that $j \notin S_i$. Removing elements can only decrease the maximum, which implies the monotonicity of g .

For $x' \subseteq x$ and $i \in [k]$, if $|x'_{[n] \setminus S_i}| = n - |S_i|$, then $x'_{[n] \setminus S_i} = x_{[n] \setminus S_i}$. Thus, if $x' \subseteq x$, then $g(x') = \max\{f(x_{[n] \setminus S_i}) : i \in [k], |x'_{[n] \setminus S_i}| = n - |S_i|\}$. In other words, for any $x' \subseteq x$, we can compute the value $g(x')$ from the k values $f(x_{[n] \setminus S_i})$ for $i \in [k]$, as required. (See Section 6.2 for further discussion about computing g from f .) $\square$

Lemma 4.2. *Let g , t , and $S_1, \dots, S_k$ be as in Algorithm 4.1. Let $x, x' \in \mathcal{X}^n$ with $x' \subseteq x$ with $|x'| \geq n - t$ and $|x| = n$. Then*

$$\max\{f(x_{[n] \setminus S_i}) : i \in [k]\} \geq g(x) \quad \text{and} \quad g(x') \geq \min\{f(x_{[n] \setminus S_i}) : i \in [k]\}. \quad (4.2)$$

Proof. The first part of the claim is in fact an equality: $\max\{f(x_{[n] \setminus S_i}) : i \in [k]\} = g(x)$ and follows from the definition of g (Equation 4.1) and assumption $|x| = n$, which implies

⁸We avoid the formalism of down sensitivity. To be precise, the t -down sensitivity of f at x is $\text{DS}_f^t(x) := \max\{|f(x) - f(x')| : x' \subseteq x, |x \setminus x'| \leq t\}$.

$|x_{[n] \setminus S_i}| = n - |S_i| = n - m$ for all $i \in [k]$. The second part of the claim relies on the fact that $S_1, \dots, S_k$ is a (n, m, t) -covering (Definition 3.6). Let $T \subseteq [n]$ be such that $x' = x_{[n] \setminus T}$ and $|T| = t$. Then, by definition, there exists some $i \in [k]$ with $T \subseteq S_i$. It follows that $x'_{[n] \setminus S_i} = x_{[n] \setminus S_i}$ and $|x'_{[n] \setminus S_i}| = |x_{[n] \setminus S_i}| = n - m$; hence, $g(x') = \max\{f(x'_{[n] \setminus S_i}) : i \in [k], |x'_{[n] \setminus S_i}| = n - m\} \geq f(x_{[n] \setminus S_i})$, as required. $\square$

Combining Lemmas 4.2 and 4.1 with the guarantee of the shifted inverse mechanism in Theorem 3.4 and an optimal covering design, gives us the following guarantee.

Theorem 4.3 (Main Result – General Version). *Let $f : \mathcal{X}^n \rightarrow \mathcal{Y}$ with $\mathcal{Y} \subseteq \mathbb{R}$ finite. Let $\varepsilon, \delta > 0$. Let $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$ as in Theorem 3.4. Let $m \in \mathbb{N}$ satisfy $n \geq m \geq t$. Let $M : \mathcal{X}^n \rightarrow \mathcal{Y}$ be ESTIMATE from Algorithm 4.1 instantiated with f, ε, δ and a (n, m, t) -covering design $S_1, \dots, S_k \subseteq [n]$ of size k . Then we have the following properties.*

- **Privacy:** M is (ε, δ) -differentially private.
- **Accuracy:** For any input $x \in \mathcal{X}^n$ of size $|x| = n$,

$$\max\{f(x_{[n] \setminus S_i}) : i \in [k]\} \geq M(x) \geq \min\{f(x_{[n] \setminus S_i}) : i \in [k]\}. \quad (4.3)$$

- **Oracle Efficiency:** $M(x)$ only depends on the k values $f(x_{[n] \setminus S_i})$ for $i \in [k]$.

Proof. Privacy follows from the privacy guarantee of the shifted inverse mechanism (Theorem 3.4) and postprocessing; this requires g to be monotone, which is guaranteed by the first part of Lemma 4.1. Efficiency follows from the second part of Lemma 4.1 – for all $x' \subseteq x$, $g(x')$ is determined by the k values $f(x_{[n] \setminus S_i})$ for $i \in [k]$ and the shifted inverse mechanism only accesses the input by evaluating $g(x')$ with $x' \subseteq x$. The accuracy guarantee of the shifted inverse mechanism (Theorem 3.4) is that

$$g(x) \geq M(x) \geq \min\{g(x') : x' \subseteq x, |x'| \geq |x| - t\}. \quad (4.4)$$

By Lemma 4.2, $\max\{f(x_{[n] \setminus S_i}) : i \in [k]\} \geq g(x)$ and

$$\min\{g(x') : x' \subseteq x, |x'| \geq |x| - t\} \geq \min\{f(x_{[n] \setminus S_i}) : i \in [k]\}. \quad (4.5)$$

Combining the bounds yields the accuracy guarantee and completes the proof. $\square$

Theorem 1.1 in the introduction is a simplification of Theorem 4.3.

Proof of Theorem 1.1. The algorithm M^f promised by Theorem 1.1 is ESTIMATE from Algorithm 4.1 instantiated with an optimal covering design i.e. $k = C(n, m, t)$. The bounds on k in Theorem 1.1 follow from Proposition 3.7 and Corollary 3.8. The privacy and oracle efficiency guarantees of Theorem 1.1 follows immediately from those of Theorem 4.3. It only remains to translate the accuracy guarantee: Suppose we have an input $X \in \mathcal{X}^n$ of size $|X| = n$ that consists of n independent samples from an (unknown) distribution $\mathcal{D}$. For each $i \in [k]$, the subset of the input $X_{[n] \setminus S_i}$ corresponds to $n - m$ independent samples from $\mathcal{D}$,

since $|S_i| = m$. Theorem 1.1 assumes that $\mathbb{P}_{X \leftarrow \mathcal{D}^{n-m}} [|f(X) - \nu| \leq \alpha] \geq 1 - \beta$ for some value ν (where ν depends on $\mathcal{D}$). Thus, by a union bound, $\mathbb{P}_{X \leftarrow \mathcal{D}^n} [\forall i \in [k] \ |f(X_{[n] \setminus S_i}) - \nu| \leq \alpha] \geq 1 - k\beta$. From Theorem 4.3, we have

$$\max\{f(X_{[n] \setminus S_i}) : i \in [k]\} \geq M(X) \geq \min\{f(X_{[n] \setminus S_i}) : i \in [k]\}. \quad (4.6)$$

It follows that $\mathbb{P}_{X \leftarrow \mathcal{D}^n} [|M(X) - \nu| \leq \alpha] \geq 1 - k\beta$, as required. $\square$

4.1 Pure & Concentrated DP Variants

Theorem 4.3 is stated for approximate differential privacy. We also state results for the pure and concentrated variants of differential privacy. These results follow by applying the relevant versions of the shifted inverse mechanism (§3.3). However, this requires us to introduce an added failure probability in the mechanism.

Theorem 4.4 (Main Result – Pure DP Version). *Let $f : \mathcal{X}^n \rightarrow \mathcal{Y}$ with $\mathcal{Y} \subseteq \mathbb{R}$ finite. Let $\varepsilon, \beta > 0$. Let $t = 2 \left\lceil \frac{2}{\varepsilon} \log \left(\frac{|\mathcal{Y}|}{\beta} \right) \right\rceil$ as in Theorem 3.3. Let $m \in \mathbb{N}$ satisfy $n \geq m \geq t$ and let $k \geq C(n, m, t)$. Then there exists $M : \mathcal{X}^n \rightarrow \mathcal{Y}$ with the following properties.*

- **Privacy:** M is $(\varepsilon, 0)$ -differentially private.
- **Accuracy:** For any input $x \in \mathcal{X}^n$ of size $|x| = n$,

$$\mathbb{P}_M [\max\{f(x_{[n] \setminus S_i}) : i \in [k]\} \geq M(x) \geq \min\{f(x_{[n] \setminus S_i}) : i \in [k]\}] \geq 1 - \beta. \quad (4.7)$$

- **Oracle Efficiency:** $M(x)$ only depends on the k values $f(x_{[n] \setminus S_i})$ for $i \in [k]$.

Theorem 4.5 (Main Result – Concentrated DP Version). *Let $f : \mathcal{X}^n \rightarrow \mathcal{Y}$ with $\mathcal{Y} \subseteq \mathbb{R}$ finite. Let $\rho, \beta > 0$. Let $t = O \left(\sqrt{\frac{1}{\rho} \log \left(\frac{|\mathcal{Y}|}{\beta} \right)} \right)$ as in Theorem 3.5. Let $m \in \mathbb{N}$ satisfy $n \geq m \geq t$ and let $k \geq C(n, m, t)$. Then there exists $M : \mathcal{X}^n \rightarrow \mathcal{Y}$ with the following properties.*

- **Privacy:** M is ρ -zCDP and $\sqrt{2\rho}$ -GDP.
- **Accuracy:** For any input $x \in \mathcal{X}^n$ of size $|x| = n$,

$$\mathbb{P}_M [\max\{f(x_{[n] \setminus S_i}) : i \in [k]\} \geq M(x) \geq \min\{f(x_{[n] \setminus S_i}) : i \in [k]\}] \geq 1 - \beta. \quad (4.8)$$

- **Oracle Efficiency:** $M(x)$ only depends on the k values $f(x_{[n] \setminus S_i})$ for $i \in [k]$.

To guarantee (ε, δ) -differential privacy, it suffices [Ste22, Remark 15] to have ρ -zCDP with

$$\rho = \frac{\varepsilon^2}{4 \log(1/\delta) + 4\varepsilon}. \quad (4.9)$$

Substituting this bound into Theorem 4.5 gives

$$t = O\left(\frac{1}{\varepsilon} \sqrt{(\log(1/\delta) + \varepsilon) \cdot \log\left(\frac{|\mathcal{Y}|}{\beta}\right)}\right). \quad (4.10)$$

We can compare this bound to $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$ in Theorem 4.3. In particular, if the output space $\mathcal{Y}$ is not too large and we can tolerate a reasonable failure probability β , then $\sqrt{\log(|\mathcal{Y}|/\beta)}$ is not much larger than $\exp(O(\log^* |\mathcal{Y}|))$, which means that the dominant difference between the bounds is that Theorem 4.3 has a $\log(1/\delta)$ term, where Theorem 4.5 gives $\sqrt{\log(1/\delta)}$. That is to say, in a reasonable parameter regime, Theorem 4.5 is better than Theorem 4.3. (This comparison, of course, depends on the constants hidden by the big-O notation.)

Furthermore, if the desired privacy failure probability δ is sufficiently small, then the bound of $t = O\left(\frac{1}{\varepsilon} \log(|\mathcal{Y}|/\beta)\right)$ from Theorem 4.4 – which is independent of δ – may dominate the bounds from Theorems 4.3 and 4.5.

5 Lower Bound

Now we prove our lower bound which shows the near-optimality of our upper bound.

Theorem 5.1 (Lower Bound – General Version). *Let $M^f : \mathbb{Z}^n \rightarrow \mathbb{Z}$ be a randomised algorithm that makes at most k queries to an oracle $f : \mathbb{Z}^* \rightarrow \mathbb{Z}$. Let $\gamma > 0$ be fixed. Suppose that, for every oracle f , the algorithm M^f satisfies (ε, δ) -differential privacy and the following. Let $\mathcal{D}$ be an arbitrary distribution on $\mathbb{Z}$ and let $\nu \in \mathbb{Z}$. If $\mathbb{P}_{X \leftarrow \mathcal{D}^{n-m}}[f(X) = \nu] \geq 1 - \gamma$, then $\mathbb{P}_{X \leftarrow \mathcal{D}^n}[|M^f(X) - \nu| \leq 1] \geq 1/2$. Then we must have*

$$k \geq \frac{\binom{n}{t}}{\binom{m}{t}} \cdot \left(\frac{1}{2} e^{-2t\varepsilon} - \frac{\delta}{e^\varepsilon - 1} \right) \quad (5.1)$$

for all integers $t \geq 0$.

Theorem 1.2 in the introduction is attained by substituting either $t = 1/\varepsilon$ or $t = \log(1/\delta)/200\varepsilon$ into Theorem 5.1.

We begin by giving some intuition for the lower bound: Consider the function $f : \mathbb{Z}^* \rightarrow \mathbb{Z}$ being the max function and consider the data distribution $\mathcal{D}$ to be a point mass on 0. Clearly, f always evaluates to 0 on input from this distribution and thus our differentially private algorithm, given n samples from this distribution, should output something near 0 with high probability. Assume, for now, that M is restricted to evaluating f on subsets of its input of the appropriate size. Now suppose $t \approx \frac{1}{\varepsilon} \log(1/\delta)$ out of the n samples are corrupted to be 1 instead of 0. If each subset that M evaluates f on includes at least one corrupted sample, then each evaluation will return something near 1 and so M should output 1 with high probability. This sets up a contradiction with group privacy, since we have two inputs

that only differ by t replacements on which the outputs are very different. The only way to avoid the contradiction is for the algorithm to query enough sets so that at least one of them doesn't contain any corrupted samples – in a sense, the subsets queried must form a covering design – and that's the lower bound.

The above proof sketch has a couple of holes that we must fill: (i) We must vary the correct answer to rule out an algorithm that simply “knows” that the correct answer is 0. (ii) We must randomise which inputs are corrupted to rule out an algorithm that “knows” which inputs to avoid. (iii) The algorithm could defeat this particular setup by evaluating f on small subsets, which are more likely to exclude all the corrupted samples, so we need to modify the function f to “fail” when given the wrong input size. (iv) The algorithm could also defeat this particular setup by evaluating the function on “fake” inputs, so we must rule this out too. (v) Finally, we need to be careful with the group privacy argument; we are effectively performing a “packing argument” [HT10].

Proof. Let ℓ be a sufficiently large integer.⁹ Let $S \subseteq [\ell^2]$ be uniformly random with size $|S| = \ell$. Define $\mathcal{D}_S$ to be the uniform distribution on S . Let $\nu \in \mathbb{Z} \setminus \{0\}$ be arbitrary. Let $f_{S,\nu,n-m} : \mathbb{Z}^* \rightarrow \mathbb{Z}$ be defined as

$$f_{S,\nu,n-m}(x) = \begin{cases} \nu & \text{if } |x| = n - m \text{ and } \forall i \ x_i \in S \text{ and } \forall i \neq j \ x_i \neq x_j \\ 0 & \text{if } |x| \neq n - m \text{ or } \exists i \ x_i \notin S \text{ or } \exists i \neq j \ x_i = x_j \end{cases}. \quad (5.2)$$

In words, $f_{S,\nu,n-m}(x) = \nu$ when the size of the input x is exactly $n - m$ (without repetitions) and all of the elements in x are inputs from S ; otherwise $f_{S,\nu,n-m}(x) = 0$.

By construction, $f_{S,\nu,n-m}(X) = \nu$ whenever $X \leftarrow \mathcal{D}_S^{n-m}$ and there are no collisions, which happens with probability

$$\mathbb{P}_{X \leftarrow \mathcal{D}_S^{n-m}} [f_{S,\nu,n-m}(X) = \nu] = \mathbb{P}_{X \leftarrow \mathcal{D}_S^{n-m}} [\forall i \neq j \ X_i \neq X_j] = \prod_{i=0}^{n-m-1} \frac{|S| - i}{|S|} \geq 1 - \frac{(n-m)^2}{2\ell} \geq 1 - \gamma, \quad (5.3)$$

assuming $\ell \geq n^2/2\gamma$. Hence, by assumption, $M^{f_{S,\nu,n-m}}(X) \in [\nu - 1, \nu + 1]$ with probability at least $1/2$ whenever $X \leftarrow \mathcal{D}_S^n$.

Now we define a new “corrupted” distribution $\mathcal{C}_{S,n,t}$ on $\mathcal{X}^n$ as follows. It consists of $n - t$ independent uniformly random samples from S and t independent uniformly random samples from $[\ell^2] \setminus S$ in an independently uniformly random order. By construction, there is a coupling between $\mathcal{C}_{S,n,t}$ and $\mathcal{D}_S^n$ such that they always differ by t replacements or $2t$ additions/removals – i.e., the ∞ -Wasserstein distance between the distributions is bounded by $2t$. Given S , it is easy to distinguish $\mathcal{C}_{S,n,t}$ from $\mathcal{D}_S^n$. However, if S is unknown, these distributions are indistinguishable (provided ℓ is sufficiently large). Our key claim is that, given a sample $\tilde{X} \leftarrow \mathcal{C}_{S,n,t}$ and oracle access to $f_{S,\nu,n-m}$ (and no additional information about S), it is hard to generate a query to the oracle that returns the value ν :

⁹We will eventually take $\ell \rightarrow \infty$.

Claim 5.2. Let $g : \mathbb{Z}^* \rightarrow \mathbb{Z}^*$ be a (possibly randomised) function. Let $S \subseteq [\ell^2]$ be uniformly random of size $|S| = \ell$. Let $\tilde{X} \leftarrow \mathcal{C}_{S,n,t}$ – that is, $\tilde{X}$ contains $n - t$ elements from S and t elements from $[\ell^2] \setminus S$ and is otherwise uniformly random. Let $Y = g(\tilde{X})$. Then

$$\mathbb{P}[|Y| = n - m \text{ and } \forall i Y_i \in S \text{ and } \forall i \neq j Y_i \neq Y_j] \leq \frac{\binom{m}{t}}{\binom{n}{t}} + \frac{n^2}{2\ell}, \quad (5.4)$$

assuming ℓ is sufficiently large and $0 \leq t \leq m \leq n$.

Proof. For simplicity, we split the analysis into two cases depending on whether or not $\tilde{X}$ has any collisions. Let E denote the event that there are collisions and let $\overline{E}$ denote the event that there are no collisions. The probability of no collisions is

$$\mathbb{P}_{\tilde{X} \leftarrow \mathcal{C}_{S,n,t}}[\overline{E}] = \mathbb{P}_{\tilde{X} \leftarrow \mathcal{C}_{S,n,t}}[\forall i \neq j \tilde{X}_i \neq \tilde{X}_j] \quad (5.5)$$

$$\begin{aligned} &= \left(\prod_{i=0}^{n-t-1} \frac{|S| - i}{|S|} \right) \left(\prod_{j=0}^{t-1} \frac{|[\ell^2] \setminus S| - j}{|[\ell^2] \setminus S|} \right) \\ &\geq 1 - \sum_{i=0}^{n-t-1} \frac{i}{\ell} - \sum_{j=0}^{t-1} \frac{j}{\ell^2 - \ell} \quad (\text{Bernoulli's inequality}) \\ &\geq 1 - \frac{(n-t)^2}{2\ell} - \frac{t^2}{2\ell(\ell-1)} \geq 1 - \frac{n^2}{2\ell}. \end{aligned} \quad (5.6)$$

Since the probability of a collision $\mathbb{P}[E] \leq n^2/2\ell$ is low (because ℓ is large), we focus on the case of no collisions.

Now we take a Bayesian perspective. Conditioned on $\tilde{X}$, the set S and the output Y are independent. We can consider a fixed Y ; and we can assume $|Y| = n - m$ and $\forall i \neq j Y_i \neq Y_j$. Given $\tilde{X}$ (with no collisions – i.e., $|\tilde{X}| = n$), all we know about S is that its intersection with $\tilde{X}$ has size exactly $|S \cap \tilde{X}| = n - t$; otherwise S is uniformly random with size $|S| = \ell$. Now we can directly calculate the probability that $\forall i Y_i \in S$ based on the intersection between Y and $\tilde{X}$, namely

$$\begin{aligned} \mathbb{P}_S[\forall i Y_i \in S | \overline{E}] &= \mathbb{P}_S[Y \subseteq S | \overline{E}] \\ &= \mathbb{P}_{S \cap \tilde{X}}[Y \cap \tilde{X} \subseteq S \cap \tilde{X} | \overline{E}] \mathbb{P}_{S \setminus \tilde{X}}[Y \setminus \tilde{X} \subseteq S \setminus \tilde{X} | \overline{E}] \\ &= \frac{\binom{n - |Y \cap \tilde{X}|}{n - t - |Y \cap \tilde{X}|}}{\binom{n}{n-t}} \frac{\binom{\ell^2 - n - |Y \setminus \tilde{X}|}{\ell - (n-t) - |Y \setminus \tilde{X}|}}{\binom{\ell^2 - n}{\ell - (n-t)}} \\ &= \frac{\binom{n - |Y \cap \tilde{X}|}{t}}{\binom{n}{t}} \frac{\binom{\ell^2 - n - |Y| + |Y \cap \tilde{X}|}{\ell^2 - \ell - t}}{\binom{\ell^2 - n}{\ell^2 - \ell - t}} =: h(|Y \cap \tilde{X}|). \end{aligned} \quad (5.7)$$

Next we show that the function h defined above is increasing: For $k < n - t \leq n$, we have

$$\begin{aligned} \frac{h(k+1)}{h(k)} &= \frac{\binom{n-(k+1)}{t} \binom{\ell^2 - \ell - |Y| + (k+1)}{\ell^2 - n - t}}{\binom{n-k}{t} \binom{\ell^2 - n - |Y| + k}{\ell^2 - \ell - t}} \\ &= \frac{(n-k-t)(\ell^2 - n - |Y| + k + 1)}{(n-k)(\ell + t - n + k + 1 - |Y|)} \\ &\geq 1, \end{aligned} \quad (5.8)$$

assuming ℓ is sufficiently large. Thus Equation 5.7 is maximised when $k = |Y \cap \tilde{X}| = |Y| = n - m$. Hence

$$\mathbb{P}_S [\forall i Y_i \in S | \overline{E}] = \mathbb{P}_S [Y \subseteq S | \overline{E}] \leq \frac{\binom{n-(n-m)}{t} \binom{\ell^2 - n}{\ell^2 - \ell - t}}{\binom{n}{t} \binom{\ell^2 - n}{\ell^2 - \ell - t}} = \frac{\binom{m}{t}}{\binom{n}{t}}. \quad (5.9)$$

The above analysis assumes no collisions in $\tilde{X}$. Rather than carefully analyzing the case with collisions, we use the naive bound $\mathbb{P}_S [\forall i Y_i \in S | E] \leq 1$ for this case. Combining with the bound of Equation 5.6 gives

$$\mathbb{P}_S [\forall i Y_i \in S] \leq \mathbb{P}_S [\forall i Y_i \in S | \overline{E}] \cdot \mathbb{P} [\overline{E}] + \mathbb{P}_S [\forall i Y_i \in S | E] \cdot \mathbb{P} [E] \leq \frac{\binom{m}{t}}{\binom{n}{t}} \cdot 1 + 1 \cdot \frac{n^2}{2\ell}, \quad (5.10)$$

as required. $\square$

It follows from Claim 5.2 and a union bound that if $M^{f_{S,\nu,n-m}}$ makes at most k queries to the oracle $f_{S,\nu,n-m}$, then the probability that the oracle ever returns ν is at most $k \cdot \left(\frac{\binom{m}{t}}{\binom{n}{t}} + \frac{n^2}{2\ell} \right)$. Hence we can almost simulate $M^{f_{S,\nu,n-m}}$ by running M with an oracle that always returns 0. Namely, for all $V \subseteq \mathbb{Z}$,

$$\mathbb{P}_{S, \tilde{X} \leftarrow \mathcal{C}_{S,n,t}} \left[M^{f_{S,\nu,n-m}}(\tilde{X}) \in V \right] \leq \mathbb{P}_{S, \tilde{X} \leftarrow \mathcal{C}_{S,n,t}} \left[M^0(\tilde{X}) \in V \right] + k \cdot \left(\frac{\binom{m}{t}}{\binom{n}{t}} + \frac{n^2}{2\ell} \right). \quad (5.11)$$

Note that the right hand side of Equation 5.11 does *not* depend on ν . Now we pick an arbitrary $\nu \in \mathbb{Z}$ such that

$$\mathbb{P}_{S, \tilde{X} \leftarrow \mathcal{C}_{S,n,t}} \left[M^0(\tilde{X}) \in [\nu - 1, \nu + 1] \right] \leq \frac{1}{\ell}. \quad (5.12)$$

By group privacy (Lemma 3.2), for all $V \subseteq Z$ and all $S \subseteq [\ell^2]$ with $|S| = \ell$, we have

$$\mathbb{P}_{X \leftarrow \mathcal{D}_S^n} \left[M^{f_{S,\nu,n-m}}(\tilde{X}) \in V \right] \leq e^{2t\varepsilon} \mathbb{P}_{\tilde{X} \leftarrow \mathcal{C}_{S,n,t}} \left[M^{f_{S,\nu,n-m}}(\tilde{X}) \in V \right] + \frac{e^{2t\varepsilon} - 1}{e^\varepsilon - 1} \delta. \quad (5.13)$$

(This relies on the fact that we can couple $X \leftarrow \mathcal{D}_S$ and $\tilde{X} \leftarrow \mathcal{C}_{S,n,t}$ such that $\mathbb{P}[|X \setminus \tilde{X}| = |\tilde{X} \setminus X| = t] = 1$.) By our accuracy assumption,

$$\mathbb{P}_{X \leftarrow \mathcal{D}_S^n} \left[M^{f_{S,\nu,n-m}}(\tilde{X}) \in [\nu - 1, \nu + 1] \right] \geq \frac{1}{2}. \quad (5.14)$$

Now we string together the above equations to obtain

$$\frac{1}{2} \leq e^{2t\varepsilon} \left(\frac{1}{\ell} + k \cdot \left(\frac{\binom{m}{t}}{\binom{n}{t}} + \frac{n^2}{2\ell} \right) \right) + \frac{e^{2t\varepsilon} - 1}{e^\varepsilon - 1} \delta. \quad (5.15)$$

At this point we can take the limit $\ell \rightarrow \infty$ to obtain

$$\frac{1}{2} \leq e^{2t\varepsilon} \cdot k \cdot \frac{\binom{m}{t}}{\binom{n}{t}} + \frac{e^{2t\varepsilon} - 1}{e^\varepsilon - 1} \delta \leq e^{2t\varepsilon} \cdot \left(k \cdot \frac{\binom{m}{t}}{\binom{n}{t}} + \frac{\delta}{e^\varepsilon - 1} \right), \quad (5.16)$$

which rearranges to

$$k \geq \frac{\binom{n}{t}}{\binom{m}{t}} \cdot \left(\frac{1}{2} e^{-2t\varepsilon} - \frac{\delta}{e^\varepsilon - 1} \right), \quad (5.17)$$

as required. $\square$

Our lower bound in Theorem 5.1 assumes the algorithm M has an infinite range $\mathcal{Y} = \mathbb{Z}$. In contrast, our algorithm (Theorem 4.3) assumes a finite range $\mathcal{Y}$ and has a dependency on its size, namely $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$. However, the proof of Theorem 5.1 assumes the algorithm M has range $\mathcal{Y} = [\ell^2]$ (and we take $\ell \rightarrow \infty$). Thus our lower bound could be extended to the setting with a finite range.

6 Discussion

To recap: Our main result (Theorem 1.1) provides a differentially private algorithm which takes a real-valued black-box function f and a private dataset x (consisting of i.i.d. samples from some distribution $\mathcal{D}$) and evaluates the function on k subsets of the input dataset and then outputs a statistical estimate $y \approx f(\mathcal{D}^{n-m})$ for the value of the function. Our result trades off between oracle efficiency (i.e., how many subsets we evaluate the function on) and statistical efficiency (i.e., the size of each of those subsets). We also prove a lower bound (Theorem 1.2) that shows that our upper bound is roughly optimal. Namely, the combinatorial term appearing in the oracle complexity $k \approx \frac{\binom{n}{t}}{\binom{m}{t}}$ is necessary, where t depends on the differential privacy parameters.

6.1 Interpretation

By varying the parameter m (with $n \geq m \geq t \approx \Theta(\frac{1}{\varepsilon} \log(1/\delta))$) our result interpolates between sample-and-aggregate [NRS07] and more recent results [FDY22; LRSS25]. While we believe that the entire tradeoff curve is interesting, we point out a few illustrative values along the curve in Table 6.1 to aid understanding.

	Subset size $n - m$ (larger is better)	Number of evaluations $k = C(n, m, t)$ (smaller is better)	Note
1	$\frac{n}{t+1}$	$t + 1$	Cf. sample-and-aggregate [NRS07]
2	$\frac{2n}{t+2}$	$\leq \frac{(t+2)(t+1)}{2} = O(t^2)$	(3.25, 3.26)
3	$\frac{cn}{t+c}$	$\leq \binom{t+c}{c} = O(t^c)$	$c \in \mathbb{N}$
4	$n - ct$	$\leq O\left(\left(\frac{n}{ct}\right)^t \cdot ct\right)$	$c \in \mathbb{N}$
5	$n - t$	$\binom{n}{t} = O(n/t)^t$	[Cf. FDY22; Ste23b; LRSS25]
6		$< \frac{\binom{n}{t}}{\binom{m}{t}} (1 + \log \binom{m}{t}) + 1$	general upper bound ($n \geq m \geq t$)

Table 6.1: Example parameter choices for Theorem 1.1. This shows the tradeoff between the number of evaluations of the function k (i.e., oracle complexity) and the size of the subsets on which we evaluate it (which determines statistical efficiency). Here $t = \frac{1}{\varepsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$ depends on the privacy parameters $\varepsilon, \delta > 0$ and the size of the range $\mathcal{Y}$ of the function.

Perhaps the most practically relevant instantiations of our result are given in Lines 2 and 3 of Table 6.1. Compared to Line 1, these show that we can increase the subset size by a constant factor while only suffering a polynomial blowup in the number of evaluations. In particular, we can roughly double the amount of data available in each evaluation at the expense of only a quadratic blowup in the number of evaluations.

On the other end of the tradeoff curve, we can compare Lines 4 and 5 of Table 6.1. The amount of data that is “sacrificed” for differential privacy increases by a factor of c . This decreases the number of evaluations by a multiplicative factor of c^t , which is significant, but the large n^t factor remains.

6.2 Limitations & Futher Work

As mentioned in the introduction (page 5, bullet point 5), the main limitation of our algorithm is that, while we bound the oracle complexity (i.e., the number of evaluations of the function), we do not account for the computational cost of choosing the subsets of the input on which to evaluate the function and of processing the values returned by the function. Note that in many cases evaluating the function can be quite expensive – e.g., in the PATE framework [PAEGT17; PSMRTE18], each function evaluation corresponds to training a machine learning model – thus it is reasonable to focus on minimizing the number of evaluations.

Choosing the subsets amounts to generating a covering design. As mentioned in Section 3.4, we do not have general-purpose optimal existential results for covering designs, let alone efficient constructions. (Although there are a lot of special-purpose constructions in the literature.) However, a (slightly suboptimal) covering design can be constructed (with high probability) by simply taking enough random subsets of the appropriate size (see the proof of Proposition 3.7 and Footnote 6). We must account for the failure probability, but otherwise this does give an efficient method for choosing the subsets.¹⁰

Next we consider the computational complexity of processing the function values. Recall Algorithm 4.1: We have a (n, m, t) -covering design $S_1, \dots, S_k \subseteq [n]$ and we evaluate $f(x_{[n] \setminus S_i})$ for each $i \in [k]$. From these values, we must compute

$$g(x') := \max\{f(x'_{[n] \setminus S_i}) : i \in [k], |x'_{[n] \setminus S_i}| = n - m\} \quad (6.1)$$

where $x' \subseteq x$. Then the shifted inverse mechanism (see Appendix A for details) computes

$$\ell(x, y) := \min\{|x \setminus \tilde{x}| : \tilde{x} \subseteq x, g(\tilde{x}) \leq y\}. \quad (6.2)$$

This task can be framed as a decision problem:

$$\begin{aligned} \ell(x, y) \leq v &\iff \exists \tilde{x} \subseteq x \quad |x \setminus \tilde{x}| \leq v \wedge g(\tilde{x}) \leq y \\ &\iff \exists T \subseteq [n] \quad |T| \leq v \wedge g(x_{[n] \setminus T}) \leq y \\ &\iff \exists T \subseteq [n] \quad |T| \leq v \wedge (\forall i \in [k] \quad |(x_{[n] \setminus T})_{[n] \setminus S_i}| = n - |S_i| \implies f(x_{[n] \setminus S_i}) \leq y) \\ &\iff \exists T \subseteq [n] \quad |T| \leq v \wedge (\forall i \in [k] \quad T \subseteq S_i \implies f(x_{[n] \setminus S_i}) \leq y) \\ &\iff \exists T \subseteq [n] \quad |T| \leq v \wedge (\forall i \in [k] \quad f(x_{[n] \setminus S_i}) > y \implies T \not\subseteq S_i) \\ &\iff \exists T \subseteq [n] \quad |T| \leq v \wedge (\forall i \in [k] \quad f(x_{[n] \setminus S_i}) > y \implies T \cap ([n] \setminus S_i) \neq \emptyset). \end{aligned} \quad (6.3)$$

In words, proving $\ell(x, y) \leq v$ is equivalent to finding a set T of at most v points such that at least one point lies in each set of the form $[n] \setminus S_i$ with $f(x_{[n] \setminus S_i}) > y$ and $i \in [k]$. That is, the set T is a “hitting set” for all the input sets corresponding to function values that are larger than y . In general, finding a small hitting set is equivalent to the set cover problem, which is NP-complete [Kar72].

The fact that processing the function values reduces to an NP-complete problem suggests that our algorithm cannot be made computationally efficient. However, this reduction is going the wrong way to make that suggestion formal – that is, it does *not* prove NP-hardness. Hope is not lost!

If the collection of sets $S_1, \dots, S_n$ is arbitrary and the function f and input x are arbitrary, then Equation 6.3 can give rise to arbitrary instances of the hitting set problem, which is NP-complete. Thus, to avoid NP-hardness, we need to rely on some of these parameters not being arbitrary.

¹⁰Verifying that a given collection of sets is a covering design is co-NP-hard. Thus this failure probability cannot easily be “checked away.” However, we remark that the privacy guarantee of our algorithm still holds if the sets do not form a covering design; namely Lemma 4.1 holds for any choice of subsets $S_1, \dots, S_k \subseteq [n]$.

In the black-box setting the function f is indeed arbitrary. The input x is i.i.d., but from an arbitrary distribution. However, the sets $S_1, \dots, S_n$ are *not* arbitrary; our algorithm can choose them, subject only to the constraint that they form a covering design. This points to an avenue for making the algorithm computationally efficient – construct a covering design with additional structural properties that make the decision problem in Equation 6.3 easy. We formulate this as an open problem:

Open Problem 6.1. *Construct a pair of algorithms **Gen** and **Eval** with the following properties.*

- *Given integers $n \geq m \geq t$, **Gen** produces a (n, m, t) -covering design (Definition 3.6) $S_1, \dots, S_k$ (and also outputs some context to pass to **Eval**). That is,*

$$(S_1, \dots, S_k, \text{context}) \leftarrow \text{Gen}(n, m, t), \quad (6.4)$$

where $S_1, \dots, S_k \subseteq [n]$ and $\forall i \in [k] |S_i| = m$ and $\forall T \subseteq [n] |T| \leq t \implies \exists i \in [k] T \subseteq S_i$.

- *The number of sets k should be not too large; ideally $k \leq C(n, m, t) \cdot \text{poly}(n)$, where $C(n, m, t)$ is the smallest possible k .*
- *Given $v \in [n]$ and $I \subseteq [k]$, **Eval** indicates whether the collection $[n] \setminus S_i$ for $i \in I$ has a hitting set of size $\leq v$. That is,*

$$\text{Eval}(v, I, \text{context}) = \text{true} \iff \exists T \subseteq [n] |T| \leq v \wedge \forall i \in I T \cap ([n] \setminus S_i) \neq \emptyset. \quad (6.5)$$

- *Both **Gen** and **Eval** should be computationally efficient for the parameter regime of interest. Ideally, their runtime should be polynomial in the parameter n and the covering size k . (Note that $k \geq C(n, m, t)$ may be exponential in n per Proposition 3.7.)*

*The algorithms **Gen** and **Eval** may be randomised; it suffices for the guarantees in Equations 6.4 and 6.5 to hold with high probability, although the dependence of k and the algorithms' runtimes on the failure probability should be polylogarithmic.*

Tying the open problem back to our application: Algorithm 4.1 would first call **Gen** to produce the sets $S_1, \dots, S_k$ (and **context**). Then it would evaluate $f(x_{[n] \setminus S_i})$ for each $i \in [k]$. Finally, it would run the shifted inverse mechanism (see Appendix A for details); this requires evaluating $\ell(x, y)$ for various values of y , which depends on f . Per Equations 6.3 and 6.5,

$$\ell(x, y) \leq v \iff \text{Eval}(v, I, \text{context}) = \text{true}, \quad \text{where } I = \{i \in [k] : f(x_{[n] \setminus S_i}) > y\}. \quad (6.6)$$

Thus $\ell(x, y)$ can be evaluated by calling **Eval** and performing binary search on $v \in \{0, 1, 2, \dots, n\}$. Overall, to implement Theorem 4.5 the runtime of Algorithm 4.1 would be dominated by the runtime of one call to **Gen**, plus k calls to f , plus $O(\log(n) \cdot \log |\mathcal{Y}|)$ calls to **Eval**.¹¹

¹¹The concentrated differentially private version of our algorithm in Theorem 4.5 performs binary search on $y \in \mathcal{Y}$ (see Appendix A.4 for details) and so we only evaluate $\ell(x, y)$ for $O(\log |\mathcal{Y}|)$ values of y . The other versions of our algorithm potentially require evaluating $\ell(x, y)$ for more values of y .

References

- [ABGIP25] A. Aamand, F. Boninsegna, A. Gentle, J. Imola, and R. Pagh. “Lightweight Protocols for Distributed Private Quantile Estimation”. In: *Forty-second International Conference on Machine Learning*. 2025. URL: <https://arxiv.org/abs/2502.02990> (cit. on p. 38).
- [AD20] H. Asi and J. C. Duchi. “Near instance-optimality in differential privacy”. In: (2020). URL: <https://arxiv.org/abs/2005.10630> (cit. on p. 7).
- [AKTVZ23] D. Alabi, P. K. Kothari, P. Tankala, P. Venkat, and F. Zhang. “Privately estimating a gaussian: Efficient, robust, and optimal”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 483–496. URL: <https://arxiv.org/abs/2212.08018> (cit. on p. 7).
- [AL22] H. Ashtiani and C. Liaw. “Private and polynomial time algorithms for learning gaussians and beyond”. In: *Conference on Learning Theory*. PMLR. 2022, pp. 1075–1076. URL: <https://arxiv.org/abs/2111.11320> (cit. on p. 7).
- [ALMM19] N. Alon, R. Livni, M. Malliaris, and S. Moran. “Private PAC learning implies finite Littlestone dimension”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 852–860. URL: <https://arxiv.org/abs/1806.00949> (cit. on pp. 5, 37).
- [AMB19] M. Avella-Medina and V.-E. Brunel. “Differentially private sub-gaussian location estimators”. In: (2019). URL: <https://arxiv.org/abs/1906.11923> (cit. on p. 7).
- [AUZ23] H. Asi, J. Ullman, and L. Zakynthinou. “From robustness to privacy and back”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 1121–1146. URL: <https://arxiv.org/abs/2302.01855> (cit. on p. 7).
- [BAM20] V.-E. Brunel and M. Avella-Medina. “Propose, test, release: Differentially private estimation with high probability”. In: (2020). URL: <https://arxiv.org/abs/2002.08774> (cit. on p. 7).
- [BBDS13] J. Blocki, A. Blum, A. Datta, and O. Sheffet. “Differentially private data analysis of social networks via restricted sensitivity”. In: *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 2013, pp. 87–96. URL: <https://arxiv.org/abs/1208.4586> (cit. on p. 8).
- [BDRS18] M. Bun, C. Dwork, G. N. Rothblum, and T. Steinke. “Composable and versatile privacy via truncated cdp”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 74–86. URL: <https://stein.ke/tcdp.pdf> (cit. on p. 38).
- [BGSUZ21] G. Brown, M. Gaboardi, A. Smith, J. Ullman, and L. Zakynthinou. “Covariance-aware private mean estimation without private covariance estimation”. In: *Advances in neural information processing systems* 34 (2021), pp. 7950–7964. URL: <https://arxiv.org/abs/2106.13329> (cit. on p. 7).

- [BLR13] A. Blum, K. Ligett, and A. Roth. “A learning theory approach to noninteractive database privacy”. In: *Journal of the ACM (JACM)* 60.2 (2013), pp. 1–25. URL: <https://arxiv.org/abs/1109.2229> (cit. on p. 38).
- [BNSV15] M. Bun, K. Nissim, U. Stemmer, and S. Vadhan. “Differentially private release and learning of threshold functions”. In: *2015 IEEE 56th annual symposium on foundations of computer science*. IEEE. 2015, pp. 634–649. URL: <https://arxiv.org/abs/1504.07553> (cit. on pp. 5, 37).
- [BS16] M. Bun and T. Steinke. “Concentrated differential privacy: Simplifications, extensions, and lower bounds”. In: *Theory of cryptography conference*. Springer. 2016, pp. 635–658. URL: <https://arxiv.org/abs/1605.02065> (cit. on pp. 11, 38, 40).
- [BS19] M. Bun and T. Steinke. “Average-case averages: Private algorithms for smooth sensitivity and mean estimation”. In: *Advances in Neural Information Processing Systems* 32 (2019). URL: <https://arxiv.org/abs/1906.02830> (cit. on p. 7).
- [BSU17] M. Bun, T. Steinke, and J. 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*. SIAM. 2017, pp. 1306–1325. URL: <https://arxiv.org/abs/1604.04618> (cit. on p. 38).
- [BZ25] G. Brown and L. Zakynthinou. “Tukey Depth Mechanisms for Practical Private Mean Estimation”. In: (2025). URL: <https://arxiv.org/abs/2502.18698> (cit. on p. 7).
- [CD20] R. Cummings and D. Durfee. “Individual sensitivity preprocessing for data privacy”. In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2020, pp. 528–547. URL: <https://arxiv.org/abs/1804.08645> (cit. on pp. 2, 3, 8).
- [CHLLN23] C. Canonne, S. B. Hopkins, J. Li, A. Liu, and S. Narayanan. “The full landscape of robust mean testing: Sharp separations between oblivious and adaptive contamination”. In: *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2023, pp. 2159–2168. URL: <https://arxiv.org/abs/2307.10273> (cit. on p. 7).
- [CLNSS23] E. Cohen, X. Lyu, J. Nelson, T. Sarlós, and U. Stemmer. “Optimal differentially private learning of thresholds and quasi-concave optimization”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 472–482. URL: <https://arxiv.org/abs/2211.06387> (cit. on p. 38).

- [CP96] H.-C. Chang and N. Prabhu. “Set covering number for a finite set”. In: *Bulletin of the Australian Mathematical Society* 53.2 (1996), pp. 267–269. URL: <https://www.cambridge.org/core/journals/bulletin-of-the-australian-mathematical-society/article/set-covering-number-for-a-finite-set/C7BD03A71F5236BC0ACEA1672BD96F8> (cit. on p. 13).
- [DK22] Y. Dagan and G. Kur. “A bounded-noise mechanism for differential privacy”. In: *Conference on Learning Theory*. PMLR, 2022, pp. 625–661. URL: <https://arxiv.org/abs/2012.03817> (cit. on p. 39).
- [DKMMN06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. “Our data, ourselves: Privacy via distributed noise generation”. In: *Advances in Cryptology–EUROCRYPT*. 2006, pp. 486–503. URL: <https://www.iacr.org/archive/eurocrypt2006/40040493/40040493.pdf> (cit. on pp. 6, 10).
- [DL09] C. Dwork and J. Lei. “Differential privacy and robust statistics”. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 371–380. URL: <https://www.stat.cmu.edu/~jinglei/dl09.pdf> (cit. on pp. 2, 7).
- [DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *Proc. of the Third Conf. on Theory of Cryptography (TCC)*. 2006, pp. 265–284. URL: http://dx.doi.org/10.1007/11681878_14 (cit. on pp. 2, 6, 10).
- [DR14] C. Dwork and A. Roth. “The algorithmic foundations of differential privacy”. In: *Foundations and trends® in theoretical computer science* 9.3–4 (2014), pp. 211–407. URL: <https://www.cis.upenn.edu/~aaroht/Papers/privacybook.pdf> (cit. on p. 37).
- [DR16] C. Dwork and G. N. Rothblum. “Concentrated differential privacy”. In: *arXiv preprint arXiv:1603.01887* (2016). URL: <https://arxiv.org/abs/1603.01887> (cit. on pp. 11, 38).
- [DRS22] J. Dong, A. Roth, and W. J. Su. “Gaussian differential privacy”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 84.1 (2022), pp. 3–37. URL: <https://arxiv.org/abs/1905.02383> (cit. on pp. 11, 40).
- [ES74] P. Erdős and J. Spencer. *Probabilistic methods in combinatorics*. Akadémiai Kiadó, 1974 (cit. on pp. 12, 14).
- [FDY22] J. Fang, W. Dong, and K. Yi. “Shifted Inverse: A General Mechanism for Monotonic Functions under User Differential Privacy”. In: *Proceedings of the SIGSAC Conference on Computer and Communications Security, CCS*. ACM, 2022, pp. 1009–1022. URL: <https://doi.org/10.1145/3548606.3560567> (cit. on pp. 2, 3, 5, 8, 11, 25, 36).
- [FS17] V. Feldman and T. Steinke. “Generalization for adaptively-chosen estimators via stable median”. In: *Conference on learning theory*. PMLR, 2017, pp. 728–757. URL: <https://arxiv.org/abs/1706.05069> (cit. on p. 38).

- [Gen25] A. Gentle. “Necessity of Block Designs for Optimal Locally Private Distribution Estimation”. In: (2025). URL: <https://arxiv.org/abs/2508.05110> (cit. on p. 9).
- [GH22] K. Georgiev and S. Hopkins. “Privacy induces robustness: Information-computation gaps and sparse mean estimation”. In: *Advances in neural information processing systems* 35 (2022), pp. 6829–6842. URL: <https://arxiv.org/abs/2211.00724> (cit. on p. 7).
- [GKM21] B. Ghazi, R. Kumar, and P. Manurangsi. “On avoiding the union bound when answering multiple differentially private queries”. In: *Conference on Learning Theory*. PMLR. 2021, pp. 2133–2146. URL: <https://arxiv.org/abs/2012.09116> (cit. on p. 39).
- [GKMN21] B. Ghazi, R. Kumar, P. Manurangsi, and T. Nguyen. “Robust and private learning of halfspaces”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1603–1611. URL: <https://arxiv.org/abs/2011.14580> (cit. on p. 7).
- [GP24] L. Greta and E. Price. “Sharp Noisy Binary Search with Monotonic Probabilities”. In: *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2024, pp. 75–1. URL: <https://arxiv.org/abs/2311.00840> (cit. on pp. 39, 40).
- [GZ21] A. Ganesh and J. Zhao. “Privately Answering Counting Queries with Generalized Gaussian Mechanisms”. In: *2nd Symposium on Foundations of Responsible Computing*. 2021. URL: <https://arxiv.org/abs/2010.01457> (cit. on p. 39).
- [HKM22] S. B. Hopkins, G. Kamath, and M. Majid. “Efficient mean estimation with pure differential privacy via a sum-of-squares exponential mechanism”. In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 1406–1417. URL: <https://arxiv.org/abs/2111.12981> (cit. on p. 7).
- [HT10] M. Hardt and K. Talwar. “On the geometry of differential privacy”. In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010, pp. 705–714. URL: <https://arxiv.org/abs/0907.3754> (cit. on pp. 6, 21).
- [JS13] A. Johnson and V. Shmatikov. “Privacy-preserving data exploration in genome-wide association studies”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 1079–1087. URL: <https://dl.acm.org/doi/abs/10.1145/2487575.2487687> (cit. on p. 7).
- [Kam24] G. Kamath. “The broader landscape of robustness in algorithmic statistics”. In: (2024). URL: <https://arxiv.org/abs/2412.02670> (cit. on p. 7).

- [Kar72] R. M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. URL: https://doi.org/10.1007/978-1-4684-2001-2_9 (cit. on p. 26).
- [Kee14] P. Keevash. “The existence of designs”. In: (2014). URL: <https://arxiv.org/abs/1401.3665> (cit. on p. 15).
- [Kee24] P. Keevash. “A short proof of the existence of designs”. In: (2024). URL: <https://arxiv.org/abs/2411.18291> (cit. on p. 15).
- [KK07] R. M. Karp and R. Kleinberg. “Noisy binary search and its applications”. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 881–890. URL: <https://www.cs.cornell.edu/~rdk/papers/karpr2.pdf> (cit. on p. 39).
- [KL23] N. Kohli and P. Laskowski. “Differential privacy for black-box statistical analyses”. In: *Proceedings on Privacy Enhancing Technologies* (2023). URL: <https://petsymposium.org/popets/2023/popets-2023-0089.php> (cit. on pp. 2, 8).
- [KLSU19] G. Kamath, J. Li, V. Singhal, and J. Ullman. “Privately learning high-dimensional distributions”. In: *Conference on Learning Theory*. PMLR. 2019, pp. 1853–1902. URL: <https://arxiv.org/abs/1805.00216> (cit. on p. 7).
- [KMRSSU25] G. Kamath, A. Mouzakis, M. Regehr, V. Singhal, T. Steinke, and J. Ullman. “A bias-accuracy-privacy trilemma for statistical estimation”. In: *Journal of the American Statistical Association* (2025), pp. 1–12. URL: <https://arxiv.org/abs/2301.13334> (cit. on p. 36).
- [KMV22] P. Kothari, P. Manurangsi, and A. Velingker. “Private robust estimation by stabilizing convex relaxations”. In: *Conference on Learning Theory*. PMLR. 2022, pp. 723–777. URL: <https://arxiv.org/abs/2112.03548> (cit. on p. 7).
- [KNRS13] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. “Analyzing graphs with node differential privacy”. In: *Theory of Cryptography Conference*. Springer. 2013, pp. 457–476. URL: <https://iacr.org/archive/tcc2013/77850455/77850455.pdf> (cit. on p. 8).
- [KSU20] G. Kamath, V. Singhal, and J. Ullman. “Private mean estimation of heavy-tailed distributions”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 2204–2235. URL: <https://arxiv.org/abs/2002.09464> (cit. on p. 7).

- [LJKOS23] X. Liu, P. Jain, W. Kong, S. Oh, and A. S. Suggala. “Near optimal private and robust linear regression”. In: *arXiv preprint arXiv:2301.13273* (2023). URL: <https://arxiv.org/abs/2301.13273> (cit. on p. 7).
- [LKKO21] X. Liu, W. Kong, S. Kakade, and S. Oh. “Robust and differentially private mean estimation”. In: *Advances in neural information processing systems* 34 (2021), pp. 3887–3901. URL: <https://arxiv.org/abs/2102.09159> (cit. on p. 7).
- [LKO22] X. Liu, W. Kong, and S. Oh. “Differential privacy and robust statistics in high dimensions”. In: *Conference on Learning Theory*. PMLR. 2022, pp. 1167–1246. URL: <https://arxiv.org/abs/2111.06578> (cit. on p. 7).
- [LRSS25] E. Linder, S. Raskhodnikova, A. Smith, and T. Steinke. “Privately Evaluating Untrusted Black-Box Functions”. In: *ACM Symposium on Theory of Computing (STOC)*. 2025. URL: <https://arxiv.org/abs/2503.19268> (cit. on pp. 2–4, 8, 9, 11, 25, 36).
- [LS25] X. Lyu and T. Steinke. *Differentially Private Algorithms that Never Fail*. DifferentialPrivacy.org. <https://differentialprivacy.org/fail-prob/>. Mar. 2025 (cit. on p. 38).
- [LT19] J. Liu and K. Talwar. “Private selection from private candidates”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 298–309. URL: <https://arxiv.org/abs/1811.07971> (cit. on p. 38).
- [MMNW11] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright. “Pan-private algorithms via statistics on sketches”. In: *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2011, pp. 37–48. URL: <https://dl.acm.org/doi/abs/10.1145/1989284.1989290> (cit. on p. 7).
- [MT07] F. McSherry and K. Talwar. “Mechanism design via differential privacy”. In: *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS’07)*. IEEE. 2007, pp. 94–103. URL: <https://ieeexplore.ieee.org/document/4389483> (cit. on pp. 7, 37).
- [NRS07] K. Nissim, S. Raskhodnikova, and A. Smith. “Smooth sensitivity and sampling in private data analysis”. In: *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. STOC ’07. San Diego, California, USA: Association for Computing Machinery, 2007, 75–84. ISBN: 9781595936318. URL: <https://doi.org/10.1145/1250790.1250803> (cit. on pp. 2–5, 7, 8, 25).
- [PAEGT17] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. “Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data”. In: *International Conference on Learning Representations*. 2017. URL: <https://arxiv.org/abs/1610.05755> (cit. on pp. 2, 25).

- [PAL24] H.-Y. Park, S. Asodeh, and S.-H. Lee. “Exactly minimax-optimal locally differentially private sampling”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 10274–10319. URL: <https://arxiv.org/abs/2410.22699> (cit. on p. 9).
- [PS22] N. Papernot and T. Steinke. “Hyperparameter Tuning with Renyi Differential Privacy”. In: *International Conference on Learning Representations*. 2022. URL: <https://arxiv.org/abs/2110.03620> (cit. on p. 38).
- [PSMRTE18] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson. “Scalable Private Learning with PATE”. In: *International Conference on Learning Representations*. 2018. URL: <https://arxiv.org/abs/1802.08908> (cit. on pp. 2, 25).
- [RS15] S. Raskhodnikova and A. Smith. “Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions”. In: (2015). URL: <https://arxiv.org/abs/1504.07912> (cit. on p. 8).
- [RS16a] S. Raskhodnikova and A. Smith. “Differentially private analysis of graphs”. In: *Encyclopedia of Algorithms*. Springer, 2016, pp. 543–547. URL: https://doi.org/10.1007/978-1-4939-2864-4_549 (cit. on p. 8).
- [RS16b] S. Raskhodnikova and A. Smith. “Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism”. In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2016, pp. 495–504. URL: <https://par.nsf.gov/servlets/purl/10092293> (cit. on p. 8).
- [Rö85] V. Rödl. “On a Packing and Covering Problem”. In: *European Journal of Combinatorics* 6.1 (1985), pp. 69–78. ISSN: 0195-6698. URL: <https://www.sciencedirect.com/science/article/pii/S0195669885800238> (cit. on p. 15).
- [Sch64] J. Schönheim. “On coverings.” In: *Pacific Journal of Mathematics* 14.4 (1964), pp. 1405–1411. URL: <https://projecteuclid.org/journals/pacific-journal-of-mathematics/volume-14/issue-4/On-coverings/pjm/1103033815.full> (cit. on p. 12).
- [SDGHMS21] A. Smith, J. Drechsler, I. Globus-Harris, A. McMillan, and J. Sarathy. “Non-parametric differentially private confidence intervals for the median”. In: (2021). URL: <https://arxiv.org/abs/2106.10333> (cit. on p. 38).
- [Sid95] A. Sidorenko. “What we know and what we do not know about Turán numbers”. In: *Graphs and Combinatorics* 11 (June 1995), pp. 179–199. URL: <https://link.springer.com/article/10.1007/BF01929486> (cit. on p. 12).
- [Ste22] T. Steinke. “Composition of differential privacy & privacy amplification by subsampling”. In: (2022). URL: <https://arxiv.org/abs/2210.00597> (cit. on pp. 19, 40).

- [Ste23a] T. Steinke. *Beyond Global Sensitivity via Inverse Sensitivity*. DifferentialPrivacy.org. <https://differentialprivacy.org/inverse-sensitivity/>. Sept. 2023 (cit. on p. 7).
- [Ste23b] T. Steinke. *Beyond Local Sensitivity via Down Sensitivity*. DifferentialPrivacy.org. Sept. 2023. URL: <https://differentialprivacy.org/down-sensitivity/> (cit. on pp. 11, 25, 36).
- [SU17] T. Steinke and J. Ullman. “Between Pure and Approximate Differential Privacy”. In: *Journal of Privacy and Confidentiality* 7.2 (2017). URL: <https://journalprivacyconfidentiality.org/index.php/jpc/article/view/648> (cit. on p. 39).
- [SV22] J. Sarathy and S. Vadhan. “Analyzing the differentially private theil-sen estimator for simple linear regression”. In: (2022). URL: <https://arxiv.org/abs/2207.13289> (cit. on p. 7).
- [TCKMS22] E. Tsfadia, E. Cohen, H. Kaplan, Y. Mansour, and U. Stemmer. “Friendlycore: Practical differentially private aggregation”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 21828–21863. URL: <https://arxiv.org/abs/2110.10132> (cit. on p. 7).

A Shifted Inverse Mechanism

In Section 3.3, we stated the properties of the shifted inverse mechanism of Fang, Dong, and Yi [FDY22], which is integral to our algorithm. We now briefly review how this algorithm works, following the presentation of Steinke [Ste23b].

The key idea behind the shifted inverse mechanism is the following transformation from an arbitrary monotone function to a low-sensitivity loss function. The benefit of this transformation is that low-sensitivity functions are something we know how to work with in a differentially private manner.

Proposition A.1 ([FDY22, Lemma 4.1], [Ste23b, Proposition 3], [LRSS25, Lemma 3.4]). *Let $g : \mathcal{X}^* \rightarrow \mathbb{R}$ be monotone – i.e., $x' \subseteq x \implies g(x') \leq g(x)$. Define $\ell : \mathcal{X}^* \times \mathbb{R} \rightarrow \mathbb{Z} \cup \{\infty\}$ by*

$$\ell(x, y) := \min\{|x \setminus \tilde{x}| : \tilde{x} \subseteq x, g(\tilde{x}) \leq y\}. \quad (\text{A.1})$$

Then ℓ has sensitivity 1 in its first argument. That is, $|\ell(x, y) - \ell(x', y)| \leq |x \setminus x'| + |x' \setminus x|$ for all $x, x' \in \mathcal{X}^n$ and all $y \in \mathbb{R}$ with $y \geq g(\emptyset)$.

The transformation (A.1) is invertible, namely

$$g(x) = \min\{y \in \mathbb{R} : \ell(x, y) = 0\} \quad (\text{A.2})$$

for all $x \in \mathcal{X}^*$. Note that $\ell(x, y) \geq 0$ is a decreasing function of y .

Essentially, the shifted inverse mechanism works by performing this inversion (A.2). Of course, under the constraint of differential privacy, we can only approximate $\ell(x, y)$ and hence we can only approximately perform the inversion. Performing this inversion is roughly equivalent to computing a differentially private approximate median, which is a well-studied task.

To be precise, the shifted inverse mechanism finds y satisfying two conditions:

- First, $\ell(x, y) \leq t$ for some tolerance $t > 0$, which is equivalent to $y \geq \min\{g(x') : x' \subseteq x, |x'| \geq |x| - t\}$.
- Informally, we want $\ell(x, y) > 0$, which is equivalent to $y < g(x)$. Formally, define

$$\bar{\ell}(x, y) := \min\{\ell(x, y - \eta) : \eta > 0\} = \min\{|x \setminus \tilde{x}| : \tilde{x} \subseteq x, g(\tilde{x}) < y\}. \quad (\text{A.3})$$

Then our second condition is $\bar{\ell}(x, y) > 0$, which is equivalent to $y \leq g(x)$. Note that $\bar{\ell}$ has the same properties as ℓ , namely it has sensitivity 1 in its first argument and is decreasing in its second argument.

Note that the shifted inverse mechanism *underestimates* $g(x)$. Some bias in differentially private estimation is inherent [KMRSSU25]; the negative direction of the bias arises from the fact that we are looking at down-local algorithms (i.e., we are removing some input elements) and from the monotonicity of g .

There are several differentially private ways to approximately implement the inversion (A.2), which we review next. These lead to the various forms of the shifted inverse mechanism in Section 3.3.

Note that, for the differentially private inversion to work, we must restrict to a finite search space [BNSV15; ALMM19]. Thus we assume that the underlying monotone function g has a finite range $\mathcal{Y} \subseteq \mathbb{R}$.

A.1 Pure DP – Theorem 3.3

The simplest implementation of the shifted inverse mechanism is to apply the exponential mechanism [MT07]. Roughly, we want to find $y \in \mathcal{Y}$ such that $0 < \ell(x, y) < 2\tau$, where $\tau > 0$ is an appropriately-chosen offset. The exponential mechanism can do this by minimising $|\ell(x, y) - \tau|$, which still has sensitivity 1.

However, ℓ is not a continuous function, so there may not exist *any* y such that $0 < \ell(x, y) < 2\tau$. Thus we need to be a bit more careful: For $\tau \in \mathbb{N}$, define

$$\widehat{\ell}_\tau(x, y) := \max\{\ell(x, y) - \tau, \tau - \bar{\ell}(x, y)\}, \quad (\text{A.4})$$

where $\ell(x, y) := \min\{|x \setminus \tilde{x}| : \tilde{x} \subseteq x, g(\tilde{x}) \leq y\}$ and $\bar{\ell}(x, y) := \min\{|x \setminus \tilde{x}| : \tilde{x} \subseteq x, g(\tilde{x}) < y\}$ are as in Equations A.1 and A.3. Intuitively, $\widehat{\ell}_\tau(x, y) \approx |\ell(x, y) - \tau|$, but we are guaranteed that there exists *some* y_* such that $\widehat{\ell}_\tau(x, y_*) \leq 0$, namely for $y_* = \min\{g(x') : x' \subseteq x, |x \setminus x'| = \tau\}$. Since $\widehat{\ell}_\tau$ has sensitivity 1 in its first parameter, we can apply the exponential mechanism: That is,

$$\forall x \in \mathcal{X}^* \ \forall y \in \mathcal{Y} \quad \mathbb{P}[M(x) = y] = \frac{\exp\left(-\frac{\varepsilon}{2}\widehat{\ell}_\tau(x, y)\right)}{\sum_{\hat{y} \in \mathcal{Y}} \exp\left(-\frac{\varepsilon}{2}\widehat{\ell}_\tau(x, \hat{y})\right)} \quad (\text{A.5})$$

defines a $(\varepsilon, 0)$ -differentially private algorithm $M : \mathcal{X}^* \rightarrow \mathcal{Y}$. The exponential mechanism guarantees that [DR14, Theorem 3.11]

$$\mathbb{P}_M \left[\widehat{\ell}_\tau(x, M(x)) < \min_{y \in \mathcal{Y}} \widehat{\ell}_\tau(x, y) + \frac{2}{\varepsilon} \log(|\mathcal{Y}|/\beta) \right] \geq 1 - \beta. \quad (\text{A.6})$$

Since $\min_{y \in \mathcal{Y}} \widehat{\ell}_\tau(x, y) \leq 0$, setting $\tau = \lceil \frac{2}{\varepsilon} \log(|\mathcal{Y}|/\beta) \rceil$ yields the conclusion that, with probability at least $1 - \beta$, we have $\tau - \bar{\ell}(x, M(x)) < \frac{2}{\varepsilon} \log(|\mathcal{Y}|/\beta)$ and $\ell(x, M(x)) - \tau < \frac{2}{\varepsilon} \log(|\mathcal{Y}|/\beta)$. Now $\tau - \bar{\ell}(x, M(x)) < \frac{2}{\varepsilon} \log(|\mathcal{Y}|/\beta)$ implies $\bar{\ell}(x, M(x)) > 0$, which implies $M(x) \leq g(x)$. Next $\ell(x, M(x)) - \tau < \frac{2}{\varepsilon} \log(|\mathcal{Y}|/\beta)$ implies $\ell(x, M(x)) \leq 2\tau$, which implies $M(x) \geq \min\{g(x') : x' \subseteq x, |x'| \geq |x| - 2\tau\}$, as required.

A.2 Approximate DP – Theorem 3.4

The main limitation of the exponential mechanism is its logarithmic dependence on the size of the search space $\mathcal{Y}$ – i.e., $t = O(\log |\mathcal{Y}|)$. We can improve this by relaxing to approximate differential privacy (i.e., (ε, δ) -differential privacy with $\delta > 0$) and by exploiting the fact

that $\ell(x, y)$ is a decreasing function of y . (The exponential mechanism does not exploit this structure.) At this point we can apply sophisticated algorithms from the literature.

To summarize, we have $\ell(x, y)$, which is sensitivity-1 in the private input x and decreasing in the other input y , and our goal is to privately find $y_i \in \mathcal{Y}$ (where $\mathcal{Y} = \{y_1 < y_2 < \dots < y_{|\mathcal{Y}|}\} \subseteq \mathbb{R}$) such that $\ell(x, y_i) \leq t$ and $\ell(x, y_{i-1}) > 0$, where y_{i-1} is the element in $\mathcal{Y}$ that is immediately before y_i in sorted order. This formulation is exactly the *generalized interior point problem* [BDRS18]. Bun, Dwork, Rothblum, and Steinke [BDRS18] sketch¹² an algorithm for this task with $t = \frac{1}{\epsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$, as required.

Alternatively, we can formulate the problem as minimising a quasi-convex function: The function $\hat{\ell}_\tau(x, y)$ from Equation A.4 has sensitivity 1 in the private input x and, in terms of the other input y , it is quasi-convex (i.e., decreasing-then-increasing). And our goal is to find an approximate minimiser $y \in \mathcal{Y}$. This is the formulation¹³ of Cohen, Lyu, Nelson, Sarlós, and Stemmer [CLNSS23] and they provide an algorithm to find an approximate minimiser with excess loss $t = \frac{1}{\epsilon} \log(1/\delta) \exp(O(\log^* |\mathcal{Y}|))$, as required.

A.3 Concentrated DP – Theorem 3.5

The aforementioned sophisticated algorithms achieve a very good dependence on the size of the output space $\mathcal{Y}$, namely $t = \exp(O(\log^* |\mathcal{Y}|))$. The iterated logarithm is a function that is constant for all practical purposes; although it is, in theory, unbounded. However, these sophisticated algorithms are quite complicated and, to the best of our knowledge, have never been implemented. Furthermore, approximate differential privacy (i.e., $\delta > 0$) can be undesirable, since it permits arbitrary privacy failures with probability $\leq \delta$. Thus we also consider concentrated differential privacy [DR16; BS16], which is a relaxation of pure differential privacy that does not permit arbitrary privacy failures; it also tends to correspond to more practical algorithms.

A natural algorithm for performing the inversion (A.2) is binary search. To make this differentially private, we must use some form of *noisy* binary search, which we discuss next. Noisy binary search gives us a $t = O(\sqrt{\log |\mathcal{Y}|})$ dependence on the size of the output space $\mathcal{Y}$. To the best of our knowledge, the specific guarantee in Theorem 3.5 does not appear in the literature; hence we give more detail.

A.4 Noisy Binary Search

Noisy binary search appears frequently in the differential privacy literature [BLR13; BSU17; FS17; SDGHMS21; ABGIP25].

¹²Unfortunately, they do not provide a theorem statement for approximate differential privacy.

¹³There are some additional differences between their formulation and our formulation. E.g., they maximise a quasi-concave function; we minimise a quasi-convex function. Their statement [CLNSS23, Theorem 4.2] only guarantees success with probability $\geq 9/10$; this can be increased to 1 either by modifying their algorithm or using generic reductions [LT19; PS22; LS25].

Problem Statement: The setting is that we have a function $\ell : \mathcal{X}^* \times \mathcal{Y} \rightarrow \mathbb{R}$ that is low sensitivity in its first argument and decreasing in its second argument. That is, $|\ell(x, y) - \ell(x', y)| \leq |x \setminus x'| + |x' \setminus x|$ and $y \leq y' \implies \ell(x, y) \geq \ell(x, y')$. Informally, the goal is, given a private input x , to find y such that $\ell(x, y) \approx \tau$ for some target value τ . To make this tractable we must restrict the search space $\mathcal{Y} \subseteq \mathbb{R}$ to be finite; namely, let $y_1 \leq y_2 \leq \dots \leq y_{|\mathcal{Y}|}$ be a sorted enumeration of the search space $\mathcal{Y}$. To be precise, our goal is to find an index i such that $\ell(x, y_{i-1}) > \tau - \eta$ and $\ell(x, y_i) < \tau + \eta$, where i ranges from $i = 1$ (in which case we define $\ell(x, y_{i-1}) = \infty$) to $i = |\mathcal{Y}| + 1$ (in which case we define $\ell(x, y_i) = -\infty$). Here $\eta > 0$ is some tolerance. The output i must be differentially private in terms of the input x .

Naïve Solution: We can find an appropriate index i via binary search using $\log_2 |\mathcal{Y}|$ steps, each time comparing the function value $\ell(x, y_i)$ to the target value τ . To ensure concentrated differential privacy, we add Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the function value $\ell(x, y_i)$. This means that each comparison may be incorrect. The parameter $\eta > 0$ allows us to tolerate some amount of error. By composition over $\log_2 |\mathcal{Y}|$ steps, the scale of the noise grows as $\sigma = O(\sqrt{\log |\mathcal{Y}|})$. Naïvely, we must take a union bound over all $O(\log |\mathcal{Y}|)$ steps, which adds a $\sqrt{\log \log |\mathcal{Y}|}$ factor to our error guarantee – i.e., $\eta = O(\sigma \cdot \sqrt{\log \log |\mathcal{Y}|}) = O(\sqrt{\log |\mathcal{Y}| \cdot \log \log |\mathcal{Y}|})$. (This is because the maximum of k Gaussians is $\sqrt{\log k}$ standard deviations above the mean.) Fortunately, we can remove this asymptotic factor.

There are two ways to reduce this union bound factor: We could add non-independent noise that is carefully tailored to reduce the probability of one value being large [SU17; GZ21; DK22; GKM21]. Alternatively, we can modify the binary search procedure itself to be noise-tolerant.

Binary Search Over Biased Coins: There is a rich literature on noise-tolerant versions of binary search, although most of this work considers settings that are not directly relevant to our setting.

Karp and Kleinberg [KK07] consider a setting in which there are m biased coins. The coins are sorted by bias, but otherwise the biases are unknown other than what we can learn by flipping the coins. The goal is to find a nearly unbiased coin by flipping the coins as few times as possible.¹⁴ Our setting can be reduced to their setting.

Karp and Kleinberg [KK07] give an algorithm that flips $O(\log m)$ coins and has a constant probability of success. (Obviously, the probability of success can be boosted by repetition.) Greta and Price [GP24] give an improved algorithm with better constants and high-probability success bounds.

¹⁴Each coin can be flipped multiple times and the outcomes are all independent. If no unbiased coin exists, the goal is to find a successive pair of coins whose biases are approximately on opposite sides of the unbiased threshold. In general, we can search for an arbitrary bias (instead of an unbiased coin).

Theorem A.2 ([GP24, Theorem 1.1]). *Let $\beta, \gamma \in (0, 1/4)$ and $m \in \mathbb{N}$. Then there exists*

$$q = \frac{\log_2(m) + O((\log(m))^{2/3}(\log(1/\beta))^{1/3} + \log(1/\beta))}{1 - H(1/2 - \gamma)} \quad (\text{A.7})$$

and an algorithm with the following properties. Here $H(p) := p \log_2(1/p) + (1-p) \log_2(1/(1-p))$ is the binary entropy function.

Let $0 = p_0 \leq p_1 \leq p_2 \leq \dots \leq p_m \leq p_{m+1} = 1$. The algorithm has access to an oracle that, given an index $i \in [m]$, returns an independent sample from $\text{Bernoulli}(p_i)$; otherwise the algorithm does not have access to $p_1, \dots, p_m$. The algorithm makes q queries to this oracle and, with probability at least $1 - \beta$, returns $i \in [m+1]$ with $[p_{i-1}, p_i] \cap (1/2 - \gamma, 1/2 + \gamma) \neq \emptyset$.

We can simplify the bound to $q \leq O(\log(m/\beta)/\gamma^2)$. Gretta and Price [GP24] state a more general form of the result in which the target bias of $1/2$ can be set differently. The above special case of their result suffices for our application.

In our setting, the oracle returns a noisy value $V = \ell(x, y_i) + \mathcal{N}(0, \sigma^2)$. We can then compare this noisy value to the target threshold τ . The binary indicator variable $\mathbb{I}[V > \tau]$ is then a Bernoulli random variable (i.e., a coin flip) with expectation

$$\mathbb{P}[\ell(x, y_i) + \mathcal{N}(0, \sigma^2) > \tau] = \mathbb{P}\left[\mathcal{N}(0, 1) > \frac{1}{\sigma}(\tau - \ell(x, y_i))\right] = \frac{1}{2} + \Theta\left(\frac{\ell(x, y_i) - \tau}{\sigma}\right), \quad (\text{A.8})$$

where the asymptotic expression holds for $\ell(x, y_i) \approx \tau$. Thresholding a noisy real value to a binary indicator loses information, but this suffices for our application. An avenue for future work is to improve noisy binary search to fully exploit the information given by noisy values.

Proposition A.3 (DP Binary Search). *Let $\ell : \mathcal{X}^* \times [m] \rightarrow \mathbb{R}$ have sensitivity 1 in its first argument and be a decreasing function of its second argument – i.e., $|\ell(x, y) - \ell(x', y)| \leq |x \setminus x'| + |x' \setminus x|$ and $y \leq y' \implies \ell(x, y) \geq \ell(x, y')$. Let $\rho, \beta > 0$ and $\tau \in \mathbb{R}$. Then there exists $\eta = O(\sqrt{\log(m/\beta)/\rho})$ and a ρ -zCDP [BS16] and $\sqrt{2\rho}$ -GDP [DRS22] algorithm $M : \mathcal{X}^* \rightarrow [m+1]$ with the following property. For all $x \in \mathcal{X}^*$, with probability at least $1 - \beta$, we have $\ell(x, M(x)) < \tau + \eta$ and $\ell(x, M(x) - 1) > \tau - \eta$, where we define $\ell(x, 0) = \infty$ and $\ell(x, m+1) = -\infty$.*

Proposition A.3 follows from Theorem A.2 and basic properties of differential privacy (namely, composition and the Gaussian mechanism) [Ste22]. The algorithm asks $q = O(\log(m/\beta)/\gamma^2)$ queries in the form of an index $i \in [m]$ and gets answers in the form of samples $\mathcal{N}(\ell(x, y_i), \sigma^2)$. To ensure ρ -zCDP and $\sqrt{2\rho}$ -GDP we set $\sigma = \sqrt{q/2\rho}$. Per Equation A.8, we can set $\gamma = \Theta(\frac{\eta}{\sigma})$ to translate between the magnitude σ of the noise added, the tolerance γ in the coin bias, and the tolerance η in the values. We set the tolerance in the coin biases γ to some constant (e.g., $\gamma = 1/5$). Thus $\eta = \Theta(\sigma) = \Theta(\sqrt{q/\rho}) = \Theta(\sqrt{\log(m/\beta)/\rho})$.

Theorem 3.5 follows by combining Propositions A.1 and A.3. Proposition A.1 gives a low-sensitivity/decreasing loss function. Proposition A.3 gives a ρ -zCDP algorithm for approximately inverting it. Setting $\tau = \eta$ in Proposition A.3 means, with probability at least $1 - \beta$, we get an index i such that $\ell(x, y_{i-1}) > 0$ and $\ell(x, y_i) < t = \tau + \eta = O(\sqrt{\log(|\mathcal{Y}|/\beta)/\rho})$. The guarantee $\ell(x, y_{i-1}) > 0$ entails $y_i \leq g(x)$, while $\ell(x, y_i) < t$ entails $y_i \geq \min\{g(x') : x' \leq x, |x'| \geq |x| - t\}$, as required.