On Avoiding the Union Bound When Answering Multiple Differentially Private Queries

Badih Ghazi BADIHGHAZI@GMAIL.COM
Ravi Kumar RAVI.K53@GMAIL.COM
Pasin Manurangsi PASIN@GOOGLE.COM
Google Research
Mountain View, CA.

Editors: Mikhail Belkin and Samory Kpotufe

Abstract

In this work, we study the problem of answering k queries with (ϵ, δ)-differential privacy, where each query has sensitivity one. We give an algorithm for this task that achieves an expected ℓ_∞ error bound of $O(\frac{1}{\epsilon}\sqrt{k \log \frac{1}{\delta}})$, which is known to be tight (Steinke and Ullman, 2016).

A very recent work by Dagan and Kur (2020) provides a similar result, albeit via a completely different approach. One difference between our work and theirs is that our guarantee holds even when $\delta < 2^{-\Omega(k/(\log k)^8)}$ whereas theirs does not apply in this case. On the other hand, the algorithm of Dagan and Kur (2020) has a remarkable advantage that the ℓ_∞ error bound of $O(\frac{1}{\epsilon}\sqrt{k \log \frac{1}{\delta}})$ holds not only in expectation but always (i.e., with probability one) while we can only get a high probability (or expected) guarantee on the error.

Keywords: Differential privacy, Multiple queries, ℓ_∞ error.

1. Introduction

One of the most ubiquitous—as well as one of the first—differentially private (DP) algorithm is the Laplace mechanism (Dwork et al., 2006b) where, to answer some query q on a sensitive dataset X, we simply compute the true answer $q(X)$ and then add to it a noise term sampled from the Laplace distribution, where the parameter of the distribution is calibrated to the desired privacy level and the sensitivity of q. For ϵ-DP and when the query q has sensitivity at most one, this algorithm yields an expected error of $O(\frac{1}{\epsilon})$, which is known to be tight (Ghosh et al., 2012).

In real-world applications, however, it is rarely the case that only a single query is performed on the dataset X. A more realistic scenario is when we are given multiple queries $q_1, \ldots, q_k$ to the dataset and are asked to compute private answers $a_1, \ldots, a_k$ to these queries. While there are several measures of error that can be used, one of the most common is the ℓ_∞ error (aka maximum error), which is defined as $\max_{i \in [k]} |q_i(X) - a_i|$. Note that a special case of this scenario is privately computing the mean of vectors or privately releasing the one-way marginals of a table, which is a basic primitive used in machine learning tasks such as gradient estimation and hypothesis testing. Private mean estimation has been extensively studied in the ℓ_2 error (Kamath et al., 2020; Bun et al., 2019; Kamath et al., 2019; Bun and Steinke, 2019; Karwa and Vadhan, 2018; Biswas et al., 2020; Feldman and Steinke, 2018) and in the ℓ_∞ error (Steinke and Ullman, 2016; Girgis et al., 2021).

When the Laplace mechanism is applied in this multiple query setting, the privacy budget has to be split over the k queries, i.e., each query has a budget of $\frac{\epsilon}{k}$. However, this does *not* result in an ℓ_∞

error of $O(\frac{k}{\epsilon})$ because one has to apply a union bound over all the k queries, which instead results in the expected ℓ_∞ error of $O(\frac{k \log k}{\epsilon})$. Remarkably, Steinke and Ullman (2016) showed that this bound is *not* tight, by giving an algorithm with expected ℓ_∞ error of $O(\frac{k}{\epsilon})$. In other words, their algorithm "avoids the union bound" in the error. Furthermore, this error is known to be asymptotically tight for ϵ-DP (Hardt and Talwar, 2010).

For (ϵ, δ)-DP algorithms (Dwork et al., 2006a), the situation is more complicated. For simplicity, throughout the paper, we use the following definition:

$$\mathsf{err}_{k,\epsilon,\delta} := \frac{1}{\epsilon}\sqrt{k \log \frac{1}{\delta}}.$$

It is known that the expected ℓ_∞ error must be at least $\Omega(\mathsf{err}_{k,\epsilon,\delta})$ for any[1] $k^{-O(1)} \geq \delta \geq 2^{-\Omega(k/\epsilon)}$ (Steinke and Ullman, 2016). However, the Laplace mechanism, together with the advanced composition theorem (Dwork et al., 2010b), only gives a bound of $O(\mathsf{err}_{k,\epsilon,\delta} \cdot \log k)$, where the $\log k$ factor once again comes from applying the union bound over all k coordinates. The Gaussian mechanism (see, e.g., Dwork and Roth, 2014) gives an improved bound of $O(\mathsf{err}_{k,\epsilon,\delta} \cdot \sqrt{\log k})$ due to a better tail behavior of the noise distribution. Steinke and Ullman (2016) once again showed that this is not optimal, by giving an algorithm with expected ℓ_∞ error of only $O(\mathsf{err}_{k,\epsilon,\delta} \cdot \sqrt{\log \log k})$. This has recently been improved by Ganesh and Zhao (2020) to $O(\mathsf{err}_{k,\epsilon,\delta} \cdot \sqrt{\log \log \log k}))$. Even more recently, Dagan and Kur (2020) reduce this expected ℓ_∞ error to the optimal $O(\mathsf{err}_{k,\epsilon,\delta})$ although their algorithm only works when δ is at least $2^{-\Omega(k/(\log k)^8)}$, thereby leaving open the question in the case $2^{-\Omega(k)} \leq \delta \leq 2^{-\Omega(k/(\log k)^8)}$.

1.1. Our Contributions

In this work, we resolve the question of Steinke and Ullman (2016) by presenting an (ϵ, δ)-DP algorithm with tight expected ℓ_∞ error for any $0.5 \geq \delta > 0$, including the regime $2^{-\Omega(k)} \leq \delta \leq 2^{-\Omega(k/(\log k)^8)}$ not covered by Dagan and Kur (2020). Our main result is the following.

Theorem 1 *For any $k \in \mathbb{N}, \epsilon \in (0, 1]$ and $\delta \in (0, 0.5]$, there exists an (ϵ, δ)-DP algorithm that can answer k queries, each of sensitivity at most one, with expected ℓ_∞ error $O(\mathsf{err}_{k,\epsilon,\delta})$.*

Differences from Dagan and Kur (2020). We stress that the techniques used in our work and Dagan and Kur (2020) are completely different. Specifically, they arrived at their result by designing a new noise distribution and analyzing the algorithm that adds such independent noise to each query's answer. On the other hand, our approach, which is detailed in the next section, is based on the *sparse vector technique* (Dwork et al., 2009; Hardt and Rothblum, 2010; Roth and Roughgarden, 2010; Dwork et al., 2010a), similar to that of Steinke and Ullman (2016).

In terms of the guarantees, we reiterate that our algorithm works for any $\delta \leq 0.5$, whereas the current analysis of the algorithm of Dagan and Kur (2020) does not apply for $\delta \leq 2^{-\Omega(k/(\log k)^8)}$. On the other hand, the algorithm of Dagan and Kur (2020) has a remarkable advantage that the ℓ_∞ error bound of $O(\mathsf{err}_{k,\epsilon,\delta})$ holds not only in expectation but always (i.e., with probability one). In contrast, we can only get a high probability guarantee that the ℓ_∞ error does not exceed this bound (see Theorem 13).

1. Note that the lower bound on δ is necessary, as the ϵ-DP algorithm mentioned in the previous paragraph already yields an $O(\frac{k}{\epsilon})$ expected ℓ_∞ error.

1.2. Proof Overview

In this section, we describe the high-level technical ideas of our algorithm. We will sometimes be informal here, but all the details will be formalized in subsequent sections.

Our algorithm is inspired by the work of Steinke and Ullman (2016). Their algorithm works by first adding Gaussian noise to the queries. Then, they use the so-called *sparse vector technique* (Dwork et al., 2009; Hardt and Rothblum, 2010; Roth and Roughgarden, 2010; Dwork et al., 2010a) to "correct" the answers that are too far away from the true answers. The procedure they employed in this correction step is encapsulated in the following theorem; its proof can be found, e.g., in Dwork and Roth (2014).[2]

Theorem 2 *For every $k \geq 1, c_{\mathtt{sv}} \leq k, \epsilon_{\mathtt{sv}}, \delta_{\mathtt{sv}}, \beta_{\mathtt{sv}} > 0$, and*

$$\alpha_{\mathtt{sv}} \geq O\left(\mathsf{err}_{c_{\mathtt{sv}},\epsilon_{\mathtt{sv}},\delta_{\mathtt{sv}}} \cdot \log \frac{k}{\beta_{\mathtt{sv}}}\right),$$

there exists an $(\epsilon_{\mathtt{sv}}, \delta_{\mathtt{sv}})$-DP algorithm that takes as input queries $g_1, \ldots, g_k$ each of sensitivity one and if there are at most $c_{\mathtt{sv}}$ indices $i \in [k]$ such that $|g_i(X)| > \alpha_{\mathtt{sv}}/2$, then, with probability at least $1 - \beta_{\mathtt{sv}}$, it answers all the queries with ℓ_∞ error no more than $\alpha_{\mathtt{sv}}$.

Note here that g_i should be thought of as the difference between the true answer q_i and the estimate output in the first step. The above algorithm can be used to "correct" the g_i's that are too large, if there are not too many of them. Specifically, notice that the error $\alpha_{\mathtt{sv}}$ is $O(\mathsf{err}_{k,\epsilon,\delta})$ only when the number of "very incorrect" answers $c_{\mathtt{sv}}$ is at most $O(k/\log^2 k)$. This is indeed the reason why Steinke and Ullman (2016) achieve an error of $O(\mathsf{err}_{k,\epsilon,\delta} \cdot \sqrt{\log \log k})$, as they need to ensure (using the tail bound for Gaussian noise) that at most $O(k/\log^2 k)$ coordinates are "very incorrect".

This brings us to the main technical question we explore in this work: can we still apply the correction procedure when $\omega(k/\log^2 k)$ coordinates are "very incorrect"? How about even at $\Omega(k)$? In other words, can we apply a sparse vector-based correction in the *dense* regime?

In a specific sense, we show that this is possible, by carefully applying the sparse vector technique iteratively and ensuring that (with high probability) some progress is made each time.

To be more specific, we have to understand how the $\mathrm{poly}\log(k)$ factor appears in the first place. Roughly speaking, the main primitive used in Theorem 2 is the following $\mathrm{AboveThreshold}$ algorithm, which allows us to identify a single "incorrect" coordinate.

Algorithm 1: $\mathrm{AboveThreshold}^{\epsilon}_{T}$.

Input: $g_1, \ldots, g_k$; X

begin
- $\rho \leftarrow \mathrm{Lap}(2/\epsilon)$
- **for** $i = 1, \ldots, k$ **do**
  - $\nu_i \leftarrow \mathrm{Lap}(4/\epsilon)$
  - **if** $g_i(X) + \nu_i \geq T + \rho$ **then**
    - **return** i

Its privacy guarantee is well-known (see, e.g., Dwork and Roth, 2014, Theorem 3.23):

2. See also (Ganesh and Zhao, 2020, Theorem 18) for a more detailed explanation.

Theorem 3 *If each of $g_1, \dots, g_k$ has sensitivity at most one, Algorithm 1 is ϵ-DP.*

It turns out that the $\log k$ factor in $\alpha_{\mathtt{sv}}$ comes from a rather extreme situation: suppose that $g_1, \dots, g_{k/2}$ are the "correct" coordinates, e.g., $g_1 = \cdots = g_{k/2} = 0$. To make sure that we do not output these coordinates we have to make sure that *all of* $\nu_1, \dots, \nu_{k/2}$ are smaller than the threshold. Thus, the threshold has to be at least $\Omega(\log k)$. We end by noting that the $\sqrt{c_{\mathtt{sv}} \log \frac{1}{\delta_{\mathtt{sv}}}}$ factor in $\alpha_{\mathtt{sv}}$ then shows up because of $c_{\mathtt{sv}}$-fold advanced composition (see Theorem 6).

Now, the above example is extreme. In fact, suppose that there is a γ fraction of the coordinates that we would like to correct. If we randomly permute the coordinates, at least one of these coordinates will appear, with a constant probability, in the first $1/\gamma$ coordinates. As a result, we only have to ensure that $\rho, \nu_1, \dots, \nu_{1/\gamma}$ are small, meaning that we should be able to get away with a threshold of $\log \frac{1}{\gamma}$ instead of $\log k$.

Our algorithm formalizes this idea. Specifically, it works in stages. In stage ℓ, we have a target number m_ℓ of items that we would like to "correct". This number will be (slowly) geometrically decreasing. The ϵ's for (the permuted version of) the $\mathrm{AboveThreshold}$ algorithm in each stage on the other hand geometrically increase, but slower than m_ℓ so that the entire algorithm in the end remains (ϵ, δ)-DP.

The actual analysis is more involved than the above outline, because some of the "correction" operations can also flip a "correct" coordinate to an "incorrect" one, and as such we have to track this number as well in order to ensure that we make progress. Another technical point is that while we can analyze the algorithm until no "incorrect" coordinates remain at all, it turns out to be more complicated as we need finer concentration inequalities. Instead, we analyze our iterative algorithm until the number of "incorrect" coordinates is sufficiently small that we can apply Theorem 2. Finally, even after doing so, it only gives a high probability bound over the ℓ_∞ error, so we then devise a simple extension that roughly takes the best of the current output and the output of another application of the Gaussian mechanism, which helps us bound the expected ℓ_∞ error, which eventually yields Theorem 1.

Organization. We provide necessary background in Section 2. Then, we start by analyzing the "permuted" variant of the $\mathrm{AboveThreshold}$ algorithm in Section 3. We continue in Section 4 by presenting our iterative correction algorithm and give an upper bound on the number of "incorrect" coordinates. In Section 5, we use this together with Theorem 2 to obtain a high-probability ℓ_∞ error bound. We then use this to obtain the expected error bound in Section 6. Finally, we discuss some open questions in Section 7.

2. Preliminaries

Definition 4 (Differential Privacy (Dwork et al., 2006b,a)) *For $\epsilon, \delta \geq 0$, we say that an algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private (or (ϵ, δ)-DP for short) if the following holds for any set S of outputs and any neighboring datasets X, X':*

$$\mathbf{Pr}[\mathcal{A}(X) \in S] \leq e^\epsilon \cdot \mathbf{Pr}[\mathcal{A}(X') \in S] + \delta.$$

When $\delta = 0$, we may simply say that the algorithm is ϵ-DP.

The *sensitivity* of a real-valued function f is defined to be $\max_{X,X'} |f(X) - f(X')|$ where the maximum is taken over all pairs of neighboring datasets X, X'. For the purpose of this work, it

does not matter how the neighboring relationship is defined; as long as we measure the sensitivity and differential privacy under the same neighboring notion, the results hold.

For brevity, we will assume henceforth that any query considered has sensitivity at most one and we may not state this assumption explicitly. Note that all results extend to the case where the sensitivity is bounded by Δ, with the (necessary) multiplicative Δ factor in the expected error.

We also recall the following composition theorems for DP:

Theorem 5 (Basic Composition Theorem) *An algorithm that applies a sequence of (ϵ_1, δ_1)-DP, $\ldots, (\epsilon_m, \delta_m)$-DP (possibly adaptive) algorithms is $(\epsilon_1 + \cdots + \epsilon_m, \delta_1 + \cdots + \delta_m)$-DP.*

Theorem 6 (Advanced Composition Theorem (Dwork et al., 2010b)) *An algorithm that applies an ϵ-DP (possibly adaptive) algorithm m times is (ϵ', δ')-DP for any $\delta' > 0$ with*

$$\epsilon' = \sqrt{2m \log \frac{1}{\delta'}} \cdot \epsilon + m\epsilon(e^\epsilon - 1).$$

3. Permuted AboveThreshold Algorithm

We start by analyzing the variant of the $\mathrm{AboveThreshold}$ algorithm where the coordinates are randomly permuted, as presented in Algorithm 2.

Algorithm 2: $\mathrm{PermutedAboveThreshold}^{\epsilon}_{T}$.

Input: $f_1, \ldots, f_k; X$

1 **begin**

2 $\quad \pi \leftarrow$ random permutation on $[k]$

3 $\quad i \leftarrow \mathrm{AboveThreshold}^{\epsilon}_{T}(f_{\pi(1)}, \ldots, f_{\pi(k)}; X)$

4 $\quad$ **return** $\pi^{-1}(i)$

Since the only step in Algorithm 2 that depends on the input dataset is Step 3, we can apply the privacy guarantee of $\mathrm{AboveThreshold}$ (from Theorem 3) to arrive at the following similar guarantee for $\mathrm{PermutedAboveThreshold}$.

Observation 7 *If each of $f_1, \ldots, f_k$ has sensitivity at most one, Algorithm 2 is ϵ-DP.*

Next, we analyze its utility. Let i^* be the output index of $\mathrm{PermutedAboveThreshold}^{\epsilon}_{T}$. Our goal here is to ensure that, with some non-trivial constant probability, $f_{i^*}(X) \geq T + w$ for some parameter $w > 0$. Similar to the known analyses of the vanilla $\mathrm{AboveThreshold}$ algorithm, we will have to assume that there are not too many "bad" coordinates i with $f_i(X) \in (T - w, T + w)$. The difference in our analysis below is that we additionally assume that there are many, i.e., $\gamma \cdot k$ "good" coordinates i that satisfy $f_i(X) \geq T + w$. This turns out to help us reduce the w parameter; specifically, we get w to be as small as $O(\frac{1}{\epsilon} \log \frac{1}{\gamma})$, comparing to the vanilla analysis that would have required w to be at least $O(\frac{1}{\epsilon} \log k)$.

Lemma 8 *Let $\gamma, w > 0$ be any real numbers. Define $I_{\mathrm{good}} := \{i \in [k] \mid f_i(X) \geq T + w\}$ and $I_{\mathrm{bad}} := \{i \in [k] \mid f_i(X) \in (T - w, T + w)\}$. Suppose that the following conditions all hold:*

(i) $|I_{\text{good}}| \geq \gamma \cdot k$.

(ii) $w \geq 8 \cdot \frac{1}{\epsilon} \cdot \log \frac{400}{\gamma}$.

(iii) $|I_{\text{good}}| \geq 2 \cdot |I_{\text{bad}}|$.

Then, $\mathbf{Pr}_{i^*}[i^* \in I_{\text{good}}] \geq 0.55$.

Proof For a permutation π, let $j_{\text{good}} \in [k]$ denote the smallest index such that $\pi(j_{\text{good}}) \in I_{\text{good}}$. Similarly, let $j_{\text{bad}} \in [k]$ denote the smallest index such that $\pi(j_{\text{bad}}) \in I_{\text{bad}}$. Furthermore, let us define the following three events:

- Event E_{before}: $j_{\text{good}} < j_{\text{bad}}$.
- Event $E_{\text{small-index}}$: $j_{\text{good}} \leq 5/\gamma$.
- Event $E_{\text{small-noise}}$: $|\rho|, |\nu_1|, \ldots, |\nu_{\lceil 5/\gamma \rceil}| < w/2$ (where $\rho, \nu_1, \ldots, \nu_{\lceil 5/\gamma \rceil}$ are the random variables sampled in the call to Algorithm 1).

It is simple to see that, when the three events occur together, we have that $i^* = \pi(j_{\text{good}}) \in I_{\text{good}}$ as desired. Furthermore, we may bound the probability of each event as follows:

- Event E_{before}: this happens with probability exactly $\frac{|I_{\text{good}}|}{|I_{\text{good}}|+|I_{\text{bad}}|}$, which is at least 2/3 from condition (iii).
- Event $E_{\text{small-index}}$: the probability that this event does *not* occur is at most

$$\left(1 - \frac{|I_{\text{good}}|}{k}\right)^{\lfloor 5/\gamma \rfloor} \leq (1-\gamma)^{4/\gamma} \leq e^{-4} \leq 0.02,$$

  where the first inequality follows from condition (i).
- Event $E_{\text{small-noise}}$: Notice that each of $|\rho|, |\nu_1|, \ldots, |\nu_{\lceil 5/\gamma \rceil}|$ is at least $w/2$ with probability at most

$$2\exp\left(-\frac{w/2}{4/\epsilon}\right) \leq 2\left(\frac{\gamma}{400}\right) = \frac{\gamma}{200},$$

  where the first inequality follows from condition (ii). From a union bound, we have that $E_{\text{small-noise}}$ holds with probability at least 0.95.

Applying a union bound over all the three events, we can conclude that they all simultaneously hold with probability at least 0.55. This concludes our proof. ■

4. Iterative Sparse Vector Algorithm

Our iterative version of the correction algorithm via the sparse vector technique is presented in Algorithm 3. As stated earlier, the algorithm performs the correction in multiple stages. In stage ℓ, we use ϵ_ℓ to denote the privacy parameter for each correction, m_ℓ to denote the number of corrections made, and T_ℓ to denote the threshold used. These parameters will be set below. Before doing so, let us state the guarantee that this algorithm achieves:

Theorem 9 *For any $k \in \mathbb{N}$, $\epsilon \in (0, 1]$ and $\delta \in (0, 0.5]$, there exists an (ϵ, δ)-DP algorithm that given queries $q_1, \ldots, q_k$ each of sensitivity at most one, outputs $a_1, \ldots, a_k$ that satisfy*

$$|\{i \in [k] \mid |q_i - a_i| > O(\mathsf{err}_{k,\epsilon,\delta})\}| \leq O(k/(\log k)^{10}),$$

with probability $1 - 2^{-\Omega(k/(\log k)^{10})}$.

Notice that this guarantee does not give the ℓ_∞ error bound yet, as there can still be as many as $O(k/(\log k)^{10})$ coordinates that have error larger than the desired bound of $O(\mathsf{err}_{k,\epsilon,\delta})$. However, as we will see in the next section, this already suffices for us to apply Theorem 2 at the end and get a high probability bound on the ℓ_∞ error.

We remark that it is possible to select parameters in such a way that the final application of Theorem 2 is not needed, i.e., by adding one more stage that essentially imitates Theorem 2. Nonetheless, since this does not seem to help clarify the analysis, we choose to not include it.

Algorithm 3: $\mathrm{IterativeSVT}^{\epsilon_1,\ldots,\epsilon_L}_{m_1,\ldots,m_L,T_1,\ldots,T_L}$.

Input: $q_1, \ldots, q_k$

begin
- $(a_1, \ldots, a_k) \leftarrow (\infty, \ldots, \infty)$
- **for** $\ell = 1, \ldots, L$ **do**
  - **for** $j = 1, \ldots, m_\ell$ **do**
    - $i^* \leftarrow \mathrm{PermutedAboveThreshold}^{0.5\epsilon_\ell}_{T_\ell}(|q_1 - a_1|, \ldots, |q_k - a_k|)$
    - $a_{i^*} \leftarrow q_{i^*}(X) + \mathrm{Lap}(2/\epsilon_\ell)$
- **return** $(a_1, \ldots, a_k)$

Our selection of parameters is as follows:

- $\kappa = 0.9$ and $\lambda = 0.95$
- $L = \lceil 10 \log_{1/\kappa} \log k \rceil$
- $\epsilon_0 = \frac{\epsilon}{1000\sqrt{\log(1/\delta)}}$

For $\ell \geq 1$, we define

- $m_\ell = \kappa^\ell \cdot k$
- $\epsilon_\ell = \frac{\epsilon_0}{\sqrt{k}} \left(\frac{1}{\sqrt{\ell \lambda^\ell}} \right)$

- $w_\ell = \frac{100 \log(500k/m_\ell)}{\epsilon_\ell}$
- $T_\ell = 4(w_1 + \cdots + w_{\ell-1}) + 3w_\ell + 2w_{\ell+1}$

Finally, we define $T_0 = 2w_1$ and $w_0 = 0$.

Throughout this section, we always assume that the parameters are as specified above and we will not mention this again. We also assume that m_ℓ defined above is an integer for every $\ell \in [L]$. This is without loss of generality since we may simply replace k with $k' := 10^{\lceil \log_{10} k \rceil}$ where $q_{k+1}, \ldots, q_{k'}$ are constants; when k is sufficiently large, this ensures that m_ℓ is an integer for all $\ell \in [L]$.

The proof of Theorem 9 is broken down into two parts: the privacy proof and the utility proof.

4.1. Privacy Analysis

We will start by proving the privacy guarantee of the algorithm.

Theorem 10 (Privacy Guarantee) *For $\epsilon \in (0, 1]$ and $\delta \in (0, 0.5]$, Algorithm 3 is (ϵ, δ)-DP.*

Proof From Observation 7 and the privacy of the Laplace mechanism, we can conclude that a single execution of Lines 5 and 6 is ϵ_ℓ-DP. Hence, for a fixed outer iteration $\ell \in [L]$, we may apply advanced composition (Theorem 6) to conclude that it is $(\epsilon'_\ell, \delta'_\ell)$-DP where $\delta'_\ell = 0.5^\ell \cdot \delta$ and

$$\begin{aligned}
\epsilon'_\ell &= \sqrt{2m_\ell \log(2^\ell/\delta)} \cdot \epsilon_\ell + m_\ell \epsilon_\ell (e^{\epsilon_\ell} - 1) \\
&\le \sqrt{2m_\ell \log(2^\ell/\delta)} \cdot \epsilon_\ell + 2m_\ell \epsilon_\ell^2 \\
&= \sqrt{2m_\ell \log(2^\ell/\delta)} \cdot \frac{\epsilon_0}{\sqrt{k}} \left(\frac{1}{\sqrt{\ell \cdot \lambda^\ell}} \right) + 2m_\ell \left(\frac{\epsilon_0}{\sqrt{k}} \left(\frac{1}{\sqrt{\ell \cdot \lambda^\ell}} \right) \right)^2 \\
&\le \epsilon_0 \left(\sqrt{\frac{2m_\ell}{\lambda^\ell k} \cdot \frac{\log(2^\ell/\delta)}{\ell}} + \frac{2m_\ell}{\lambda^\ell k} \right) \\
&\le 4\epsilon_0 \left(\sqrt{(\kappa/\lambda)^\ell \log(1/\delta)} \right) \\
&\le \frac{\epsilon}{200} \cdot (\kappa/\lambda)^{\ell/2}.
\end{aligned}$$

Finally, we apply basic composition (Theorem 5) over all $\ell \in [L]$, which implies that the entire algorithm is (ϵ', δ')-DP for

$$\epsilon' = \sum_{\ell \in [L]} \epsilon'_\ell = \sum_{\ell \in [L]} \frac{\epsilon}{200} \cdot (\kappa/\lambda)^{\ell/2} \le \frac{\epsilon}{200(1 - (\kappa/\lambda)^{0.5})} \le \epsilon,$$

and

$$\delta' = \sum_{\ell \in [L]} \delta'_\ell = \sum_{\ell \in [L]} \left(0.5^\ell \delta \right) \le \delta,$$

as desired. ■

4.2. Utility Analysis

We will next prove the utility guarantee, as restated below.

Theorem 11 (Utility Guarantee) *With probability at least* $1-2^{-\Omega(k/(\log k)^{10})}$*, the output* $(a_1, \ldots, a_k)$ *of Algorithm 3 satisfies*

$$|\{i \in [k] \mid |q_i - a_i| > O(\mathsf{err}_{k,\epsilon,\delta})\}| \leq O(k/(\log k)^{10}).$$

Our utility analysis crucially relies on tracking the set of indices i such that $|q_i - a_i|$ is above a certain threshold. Specifically, for every $\ell \in [L], j \in [m_\ell]$ and $t \in \{0, \ldots, L\}$, we define $I_t^{\ell,j}$ to be the set of indices $i = 1, \ldots, k$ such that, after the (ℓ, j)th iteration, $|q_i - a_i| \geq T_t + w_t$. For notational convenience, we define $I_t^{\ell} := I_t^{\ell,m_\ell}$, $I_t^0 := [k]$, $I_t^{\ell,0} := I_t^{\ell-1}$, and $\tau_\ell := T_\ell + w_\ell$.

The high-level idea of the proof is to consider two cases, based on whether the number of indices at the end of the $(\ell-1)$th iteration whose errors exceed τ_ℓ, i.e., $|I_\ell^{\ell-1}|$, is small. Now, if this is already small (i.e., noticeably smaller than m_ℓ), then we can use a concentration inequality to show that the number of additional indices that are "flipped" from below τ_ℓ to above τ_ℓ is small; from this, we can conclude that $|I_\ell^\ell|$ is small. On the other hand, if $|I_\ell^{\ell-1}|$ is large, then we know that after the $(\ell-1)$th iteration the number of indices whose errors belong to $(\tau_{\ell-1}, \tau_\ell)$ is small. Roughly speaking, this allows us to apply Lemma 8, which ensures that a significant fraction of selected coordinates indeed have errors at least τ_ℓ. This means that $|I_\ell^\ell|$ must be significantly smaller than $|I_\ell^{\ell-1}|$, which ultimately gives us the desired bound in the second case.

We will show that with high probability $|I_\ell^\ell|$ is small, as formalized in our main lemma below.

Lemma 12 *Let* $\ell \in [L]$*. Conditioned on* $|I_{\ell-1}^{\ell-1}| \leq 2m_{\ell-1}$*, we have that*

$$\mathbf{Pr}[|I_\ell^\ell| \leq 2m_\ell] \geq 1 - 2^{-\Omega(m_\ell)}.$$

Proof Consider the ℓth (outer) iteration of the algorithm. Let Z_j denote the Laplace random variable drawn on Line 6 in the jth inner iteration. Notice that from our setting of parameters $\epsilon_\ell, \tau_{\ell-1}$, and w_ℓ, we have

$$\mathbf{Pr}[|Z_j| \geq \tau_{\ell-1}] \leq \mathbf{Pr}[|Z_j| \geq w_\ell] \leq 0.0009.$$

From this and from the independence of the Z_j's, we may apply the Chernoff bound, which implies that the following holds with probability at least $1 - 2^{-\Omega(m_\ell)}$:

$$|\{j \in [m_\ell] \mid Z_j \geq \tau_{\ell-1}\}| \leq 0.001 m_\ell. \tag{1}$$

Thus, we may hence forth assume that (1) holds.

We will next consider two cases:

1. Case I: $|I_\ell^{\ell-1}| \leq 1.999 m_\ell$. From (1), it follows that $|I_\ell^\ell| \leq |I_\ell^{\ell-1}| + 0.001 m_\ell \leq 2m_\ell$ as desired.

2. Case II: $|I_\ell^{\ell-1}| > 1.999 m_\ell$.

   In this case, we would like to apply Lemma 8. To do this, we will check that each of condition of Lemma 8 is satisfied with $T = T_\ell, w = w_\ell, \gamma = 0.9 m_\ell / k$, and $\epsilon = \epsilon_\ell$.

We can bound the number of "good" indices i such that $|q_i - a_i| \geq \tau_\ell$ by

$$|I_\ell^{\ell,j}| \geq |I_\ell^{\ell-1}| - j \geq 1.999m_\ell - m_\ell \geq 0.999m_\ell, \tag{2}$$

which is at least $\gamma \cdot k$ as desired.

The second condition of Lemma 8 holds simply by our choice of w_ℓ.

Finally, let us bound the number of indices i such that $a_i \in [\tau_{\ell-1}, \tau_\ell)$ as follows. Notice that

$$|I_{\ell-1}^{\ell-1} \setminus I_\ell^{\ell-1}| \leq m_{\ell-1} - 0.999m_\ell = (1/\kappa - 1.999)m_\ell \leq 0.23m_\ell,$$

where the first inequality follows from our assumption on $I_{\ell-1}^{\ell-1}$. We may now use (1) to conclude that, for any $j \in [m_\ell]$, we have

$$|I_{\ell-1}^{\ell,j} \setminus I_\ell^{\ell,j}| \leq 0.23m_\ell + 0.001m_\ell \leq 0.3m_\ell. \tag{3}$$

In other words, in the ℓth outer loop, the number of indices i with $a_i \in [\tau_{\ell-1}, \tau_\ell)$ is always at most $0.3m_\ell$. Since $T_\ell = \tau_\ell - w_\ell \geq \tau_{\ell-1} + w_\ell$, this also implies that the number of "bad" indices i with $a_i \in [T_\ell - w_\ell, T_\ell + w_\ell)$ is at most $0.3m_\ell$. Together with (2), this implies that the last condition of Lemma 8 holds.

Thus, we may apply Lemma 8, which implies that $\mathbf{Pr}[a_{i^*} \geq \tau_\ell] \geq 0.5$ for each call on Line 5. Hence, by the Chernoff bound, we can conclude that, with probability $2^{-\Omega(m_\ell)}$, at least $0.4m_\ell$ of the i^*'s returned satisfy $a_{i^*} \geq \tau_\ell$. When this holds, it (together with (1)) implies that

$$\begin{aligned}
|I_\ell^\ell| &\leq |I_\ell^{\ell-1}| - 0.4m_\ell + 0.001m_\ell \\
&\leq |I_{\ell-1}^{\ell-1}| - 0.4m_\ell + 0.001m_\ell \\
&\leq 2m_{\ell-1} - 0.4m_\ell + 0.001m_\ell \\
&\leq 2m_\ell.
\end{aligned}$$

Hence, in both cases, we have $|I_\ell^\ell| \leq 2m_\ell$ with probability at least $1 - 2^{-\Omega(m_\ell)}$, as desired. ■

Theorem 11 now follows easily from the above lemma.

Proof [of Theorem 11] By applying Lemma 12 for each $\ell \in [L]$ and a union bound, we have that

$$\mathbf{Pr}[|I_L^L| \geq 2m_L] \leq 1 - \sum_{\ell \in [L]} 2^{-\Omega(m_\ell)} \leq 1 - 2^{-\Omega(k/(\log k)^{10})},$$

where the latter follows from $m_1 \geq \cdots \geq m_L = \Theta(k/(\log k)^{10})$. Finally, notice that

$$\begin{aligned}
\tau_L &\leq 4\left(\sum_{\ell \in [L+1]} w_\ell\right) \\
&= 4\left(\sum_{\ell \in [L+1]} \frac{100\log(500k/m_\ell)}{\epsilon_\ell}\right)
\end{aligned}$$

$$
\begin{aligned}
&= 4\frac{\sqrt{k}}{\epsilon_0}\left(\sum_{\ell\in[L+1]} 100\log(500(1/\kappa)^\ell)\cdot\sqrt{\ell\lambda^\ell}\right)\\
&\leq O(\sqrt{k}/\epsilon_0)\cdot\left(\sum_{\ell\in[L+1]}\ell^{3/2}\cdot\lambda^{\ell/2}\right)\\
&= O(\sqrt{k}/\epsilon_0)\\
&= O(\mathsf{err}_{k,\epsilon,\delta}).
\end{aligned}
$$

This means that $I_L^L = \{i\in[k] \mid |q_i - a_i| > O(\mathsf{err}_{k,\epsilon,\delta})\}$ as desired. ■

5. Obtaining High Probability Error Bound

In this section, we use our bound in the previous section together with Theorem 2 to obtain the following high probability ℓ_∞ error guarantee:

Theorem 13 *For any $k\in\mathbb{N}, \epsilon\in(0,1]$ and $\delta\in(0,0.5]$, there exists an (ϵ,δ)-DP algorithm that can provide answers to k queries such that, with probability $1-O(1/k^{10})$, the ℓ_∞ error is $O(\mathsf{err}_{k,\epsilon,\delta})$.*

Proof First, we apply the $(\epsilon/2,\delta/2)$-DP algorithm from Theorem 9 to get the answers $a_1,\ldots,a_k$ to the queries. From Theorem 9, there exist constants $C_1, C_2 > 0$ such that, with probability $2^{-\Omega(k/(\log k)^{10})}$, we have that

$$|\{i\in[k] \mid |q_i - a_i| > C_1\cdot\mathsf{err}_{k,\epsilon,\delta}\}| \leq C_2\cdot k/(\log k)^{10}.$$

We then apply Theorem 2 with $\epsilon_{\mathtt{sv}} = \epsilon/2, \delta_{\mathtt{sv}} = \delta/2, \beta_{\mathtt{sv}} = 1/k^{10}, c_{\mathtt{sv}} = C_2\cdot k/(\log k)^{10}, g_i = q_i - a_i$, and let $\alpha_{\mathtt{sv}} = 2C_1\cdot\mathsf{err}_{k,\epsilon,\delta}$. Let $b_1,\ldots,b_k$ be its output. Our algorithm then outputs $a_1+b_1,\ldots,a_k+b_k$.

It is simple to verify that the condition on $\alpha_{\mathtt{sv}}$ in Theorem 2 holds for any sufficiently large k. As a result, Theorem 2 implies that, with probability $1-O(1/k^{10})$, the ℓ_∞ error is $O(\mathsf{err}_{k,\epsilon,\delta})$. ■

6. From High Probability to Expected Error Bound

Finally, we will transform our high probability error bound into an expected error bound (Theorem 1). To do so, we need the following well-known theorem, which follows from the Gaussian mechanism (see, e.g., Dwork and Roth, 2014; Steinke and Ullman, 2016).

Theorem 14 (Gaussian Mechanism) *For any $k\in\mathbb{N}, \epsilon_g\in(0,1]$ and $\delta_g\in(0,0.5]$, there exists an (ϵ_g,δ_g)-DP algorithm that can answer k queries such that, for any $t>0$, the ℓ_∞ error is at least $O(t\cdot\mathsf{err}_{k,\epsilon_g,\delta_g})$ with probability at most $k\cdot e^{-\Omega(t^2)}$.*

Proof [of Theorem 1] The entire algorithm is as follows:

- First, we apply $(\epsilon/3,\delta/3)$-DP algorithm from Theorem 13 to get the answers $a_1,\ldots,a_k$ to the input queries.

- Secondly, we apply $(\epsilon/3, \delta/3)$-DP algorithm from Theorem 13 to get the answers $b_1, \ldots, b_k$ to the input queries.
- Then, we apply $(\epsilon/3, \delta/3)$-DP algorithm from Theorem 14 to the queries $|q_1 - a_1|, \ldots, |q_k - a_k|$ to get the answers $c_1, \ldots, c_k$
- If $\max\{c_1, \ldots, c_k\} \le k^{10} \cdot \mathsf{err}_{k,\epsilon,\delta}$, then we output $(a_1, \ldots, a_k)$.
- Otherwise, if $\max\{c_1, \ldots, c_k\} > k^{10} \cdot \mathsf{err}_{k,\epsilon,\delta}$, then we output $(b_1, \ldots, b_k)$.

By basic composition (Theorem 5), the entire algorithm is (ϵ, δ)-DP as desired.

We next analyze the expected ℓ_∞ error of the algorithm. To do this, notice first that, regardless of $a_1, \ldots, a_k$, the ℓ_∞ error of the output is at most the sum of $k^{10} \cdot \mathsf{err}_{k,\epsilon,\delta}$ and the two ℓ_∞ errors of the two runs of the Gaussian mechanism. As such, we still have that the ℓ_∞ error of the entire algorithm is at least $k^{10} \cdot \mathsf{err}_{k,\epsilon,\delta} + O(t \cdot \mathsf{err}_{k,\epsilon,\delta})$ with probability at most $k \cdot e^{-\Omega(t^2)}$.

Furthermore, the guarantee from Theorem 13 ensures that the ℓ_∞ error of the answers $a_1, \ldots, a_k$ is at most $O(\mathsf{err}_{k,\epsilon,\delta})$ with probability $1 - O(1/k^{10})$. When this event holds, we may apply the tail bound from Theorem 14, which implies that the output of the entire algorithm has error $h := O(\mathsf{err}_{k,\epsilon,\delta})$ with probability $1 - O(1/k^{10})$.

Let υ denote the ℓ_∞ error of the entire algorithm. Combining the bounds from the previous two paragaphs, we get

$$\begin{aligned}
\mathbb{E}[\upsilon] &= \int_0^\infty \mathbf{Pr}[\upsilon > x]dx \\
&= \int_0^h \mathbf{Pr}[\upsilon > x]dx + \int_h^{2k^{10}\cdot\mathsf{err}_{k,\epsilon,\delta}} \mathbf{Pr}[\upsilon > x]dx + \int_{2k^{10}\cdot\mathsf{err}_{k,\epsilon,\delta}}^\infty \mathbf{Pr}[\upsilon > x]dx \\
&\le h + O(1/k^{10}) \cdot (2k^{10} \cdot \mathsf{err}_{k,\epsilon,\delta}) + O(\mathsf{err}_{k,\epsilon,\delta}) \cdot \int_{k^{10}}^\infty e^{-\Omega(t^2)}dt \\
&\le O(\mathsf{err}_{k,\epsilon,\delta}),
\end{aligned}$$

which concludes our proof. ■

7. Conclusions and Open Questions

In this work, we give an (ϵ, δ)-DP algorithm that can answer k queries, each of sensitivity one, with ℓ_∞ error $O(\mathsf{err}_{k,\epsilon,\delta})$. This resolves the question posed by Steinke and Ullman (2016).

An immediate open question is if one can get the best of both our work and that of Dagan and Kur (2020), namely, to devise an (ϵ, δ)-DP algorithm for answering k queries (of sensitivity one) such that the error is always (i.e., with probability one) $O(\mathsf{err}_{k,\epsilon,\delta})$ for any value of $\delta > 0$.

Finally, we note that the main focus of this line of works has been mostly on the theoretical front; it would be interesting to achieve a practical algorithm for which the advantage of $\Theta(\sqrt{\log k})$ can be actually observed compared to the widely used Gaussian mechanism.

References

Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. Coinpress: Practical private mean and covariance estimation. In *NeurIPS*, 2020.

Mark Bun and Thomas Steinke. Average-case averages: Private algorithms for smooth sensitivity and mean estimation. In *NeurIPS*, pages 181–191, 2019.

Mark Bun, Gautam Kamath, Thomas Steinke, and Zhiwei Steven Wu. Private hypothesis selection. In *NeurIPS*, pages 156–167, 2019.

Yuval Dagan and Gil Kur. A bounded-noise mechanism for differential privacy. *CoRR*, abs/2012.03817, 2020.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006b.

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010a.

Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010b.

Vitaly Feldman and Thomas Steinke. Calibrating noise to variance in adaptive data analysis. In *COLT*, pages 535–544, 2018.

Arun Ganesh and Jiazheng Zhao. Privately answering counting queries with generalized Gaussian mechanisms. *CoRR*, (abs/2010.01457), 2020.

Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SICOMP*, 41(6):1673–1693, 2012.

Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *AISTATS*, 2021.

Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

Gautam Kamath, Jerry Li, Vikrant Singhal, and Jonathan Ullman. Privately learning high-dimensional distributions. In *COLT*, pages 1853–1902, 2019.

Gautam Kamath, Vikrant Singhal, and Jonathan Ullman. Private mean estimation of heavy-tailed distributions. In *COLT*, pages 2204–2235, 2020.

Vishesh Karwa and Salil Vadhan. Finite sample differentially private confidence intervals. In *ITCS*, pages 44:1–44:9, 2018.

Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.

Thomas Steinke and Jonathan R. Ullman. Between pure and approximate differential privacy. *J. Priv. Confidentiality*, 7(2), 2016.