Differentially Private Set Union

Sivakanth Gopi [1] **Pankaj Gulhane** [1] **Janardhan Kulkarni** [1] **Judy Hanwen Shen** [1 2] **Milad Shokouhi** [1]
Sergey Yekhanin [1]

Abstract

We study the basic operation of set union in the global model of differential privacy. In this problem, we are given a universe U of items, possibly of infinite size, and a database D of users. Each user i contributes a subset $W_i \subseteq U$ of items. We want an (ε,δ)-differentially private Algorithm which outputs a subset $S \subset \cup_i W_i$ such that the size of S is as large as possible. The problem arises in countless real world applications, and is particularly ubiquitous in natural language processing (NLP) applications. For example, discovering words, sentences, n-grams etc., from private text data belonging to users is an instance of the set union problem. In this paper we design new algorithms for this problem that significantly outperform the best known algorithms.

1. Introduction

Natural language models for applications such as suggested replies for e-mails and dialog systems rely on the discovery of n-grams and sentences (Hu et al., 2014; Kannan et al., 2016; Chen et al., 2019; Deb et al., 2019). Words and phrases used for training come from individuals, who may be left vulnerable if personal information is revealed. For example, a model could generate a sentence which reveals personal information of the users on which it was trained (Carlini et al., 2019). Therefore, algorithms that allow the public release of the words, n-grams, and sentences obtained from user text while preserving privacy are desirable. Additional applications of this problem include the release of search queries and keys in SQL queries (Korolova et al., 2009; Wilson et al., 2020). While other privacy definitions are common in practice, guaranteeing differential privacy, introduced in the seminal work of Dwork et al. (2006), ensures users the strongest preservation of privacy. In this paper we consider user level privacy.

Definition 1.1 (Differential Privacy (Dwork et al., 2006))**.** A randomized algorithm $\mathcal{A}$ is (ε,δ)-differentially private if for any two neighboring databases D and D', which differ in exactly the data pertaining to a single user, and for all sets $\mathcal{S}$ of possible outputs:

$$\Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta.$$

An algorithm satisfying differential privacy (DP) guarantees that its output does not change by much if a single user is either added or removed from the dataset. Moreover, the guarantee holds regardless of how the output of the algorithm is used downstream. Therefore, items (e.g. n-grams) produced using a DP algorithm can be used in other applications without any privacy concerns. Since its introduction a decade ago (Dwork et al., 2006), differential privacy has become the de facto notion of privacy in statistical analysis and machine learning, with a vast body of research work (see Dwork et al. (2014) and Vadhan (2017) for surveys) and growing acceptance in industry. Differential privacy is deployed in many industries, including Apple (Apple, 2017), Google (Erlingsson et al., 2014; Bittau et al., 2017), Microsoft (Ding et al., 2017), Mozilla (Avent et al., 2017), and the US Census Bureau (Abowd, 2016; Kuo et al., 2018).

The vocabulary extraction and n-gram discovery problems mentioned above, as well as many commonly studied problems (Korolova et al., 2009; Wilson et al., 2020) can be abstracted as a set union which leads to the following problem.

Problem 1.1 (Differentially Private Set Union (DPSU))**.** Let U be some universe of items, possibly of unbounded size. Suppose we are given a database D of users where each user i has a subset $W_i \subseteq U$. We want an (ε,δ)-differentially private Algorithm A which outputs a subset $S \subseteq \cup_i W_i$ such that the size of S is as large as possible.

As the universe of items can be unbounded, as in our motivating examples, it is not clear how to apply the exponential mechanism (McSherry & Talwar, 2007) to DPSU. Intuitively, when the universe is unbounded, an algorithm which

[1]Microsoft [2]Work done as part of the Microsoft AI Residency Program. Correspondence to: Sivakanth Gopi <sigopi@microsoft.com>, Janardhan Kulkarni <jakul@microsoft.com>, Judy Hanwen Shen <hashe@microsoft.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

outputs items outside the true union doesn't lead to any gains in privacy. And in many applications, it is essential that we output a subset of the true union. Therefore, in the definition of DPSU, we impose the condition that the output is a subset of the true union. It is not hard to show that there are no non-trivial $(\varepsilon, 0)$-DP DPSU algorithms, so we in this paper we will only study (ε, δ)-DP algorithms for $\delta > 0$.

Existing algorithms [1] for this problem (Korolova et al., 2009; Wilson et al., 2020) collect a bounded number of items from each user, build a histogram of these items, and disclose the items whose noisy counts fall above a certain threshold. In these algorithms, the contribution of each user is always *independent* from the identity of items held by other users, resulting in a wasteful aggregation process, where some items' counts could be far above the threshold. Since the goal is to release as large a set as possible rather than to release accurate counts of each item, there could be more efficient ways to allocate the weight to users' items. We

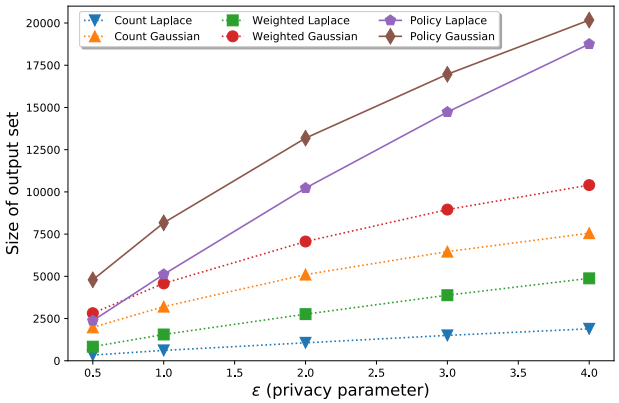

Figure 1. Size of the set output by our proposed algorithms POLICY LAPLACE and POLICY GAUSSIAN compared to natural generalizations of previously known algorithms for various values of privacy parameter ε and $\delta = \exp(-10)$. The error bars are negligible.

deviate from the previous methods by allowing users to contribute their items in a *dependent fashion*, guided by an *update policy*. In our algorithms, proving privacy is more delicate as some update policies can result in histograms with unbounded sensitivity. We prove a meta-theorem to show that update policies with certain *contractive properties* would result in differentially private algorithms. The main contributions of the paper are:

- Guided by our meta-theorems, we introduce two new algorithms called POLICY LAPLACE and POLICY GAUSSIAN for the DPSU problem. Both of them run in *linear time* and only require a single pass over the users' data.
- Using a Reddit dataset, we demonstrate that our algorithms significantly improve the size of DP set union even when compared to natural generalizations of the existing mechanisms for this problem (see Figure 1).

Our algorithms are being productized in industry to make a basic subroutine in an NLP application differentially private.

1.1. Base Line Algorithms

To understand the DPSU problem better, let us start with the simplest case we can solve by known techniques. Define $\Delta_0 = \max_i |W_i|$. Suppose $\Delta_0 = 1$. This special case can be solved using the algorithms in (Korolova et al., 2009; Wilson et al., 2020). Their algorithm works as follows: Construct a histogram on $\cup_i W_i$ (the set of items in a database D) where the count of each item is the number of sets it belongs to. Then add Laplace noise or Gaussian noise to the counts of each item. Finally, release only those items whose noisy histogram counts are above a certain *threshold* ρ. It is not hard to prove that if the threshold is set sufficiently high, then the algorithm is (ε, δ)-DP.

In many applications, however, Δ_0 is much greater than 1. For example, in the n-gram discovery problem, each user holds a large set of n-grams not just 1. A straight-forward extension of the histogram algorithm for $\Delta_0 > 1$ is to upper bound the ℓ_1-sensitivity by Δ_0 (and ℓ_2-sensitivity by $\sqrt{\Delta_0}$), and then add some appropriate amount of Laplace noise (or Gaussian noise) based on sensitivity. The threshold ρ has to be set based on Δ_0. The Laplace noise based algorithm was also the approach considered in (Korolova et al., 2009; Wilson et al., 2020). This approach has the following drawback. Suppose a significant fraction of users have sets of size smaller than Δ_0. Then constructing a histogram based on *counts* of the items results in *wastage of sensitivity budget*. A user i with $|W_i| < \Delta_0$ can increment the count of items in W_i by any vector $v \in \mathbb{R}^{W_i}$ as long as one can ensure that ℓ_1 sensitivity is bounded by Δ_0 (or ℓ_2 sensitivity is bounded by $\sqrt{\Delta_0}$ if adding Gaussian noise). Consider the following natural generalization of Laplace and Gaussian mechanisms to create a *weighted histogram of elements*. A weighted histogram over a domain U is any map $H : U \to \mathbb{R}$. For an item $u \in U$, $H(u)$ is called the weight of u. In the rest of the paper, the term histogram should be interpreted as weighted histogram. Each user i updates the weight of each item $u \in W_i$ using the rule: $H[u] := H[u] + (\Delta_0/|W_i|)^{1/p}$ for $p = 1, 2$. It is not hard to see that ℓ_p-sensitivity of this weighted histogram is still $\Delta_0^{1/p}$. Adding Laplace noise (for $p = 1$) or Gaussian noise (for $p = 2$) to each item of the weighted histogram, and releasing only those items above an appropriately calibrated

[1] They don't study the DPSU problem as defined in this paper. Their goal is to output approximate counts of as many items as possible in $\cup_i W_i$.

threshold will lead to differentially private output. We call these algorithms as WEIGHTED LAPLACE and WEIGHTED GAUSSIAN, they will be used as benchmarks to compare against our new algorithms.

1.2. Our Techniques

The WEIGHTED LAPLACE and WEIGHTED GAUSSIAN mechanisms described above can be thought of trying to solve the following variant of a *Knapsack* problem. Here each item $u \in U$ is a bin and we gain a profit of 1 if the total weight of the item in the weighted histogram constructed is more than the threshold. Each user can increment the weight of elements $u \in W_i$ using an *update policy* ϕ which is defined as follows.

Definition 1.2 (Update policy)**.** An update policy is a map $\phi : \mathbb{R}^U \times 2^U \to \mathbb{R}^U$ such that $\text{supp}(\phi(H, W) - H) \subset W$, i.e., ϕ can only update the weights of items in W. And the i^{th} user updates H to $\phi(H, W_i)$. Since W_i is typically understood from context, we will write $\phi(H)$ instead of $\phi(H, W_i)$ for simplicity.

In this framework, the main technical challenge is the following:

How to design update policies such that the sensitivity of the resulting weighted histogram is small while maximizing the number of bins that are full?

Note that bounding sensitivity requires that $\|\phi(H, W) - H\|_{\ell_p} \leq C$ for some constant C i.e. each user has an ℓ_p-budget of C and can increase the weights of items in their set by an ℓ_p-distance of at most C. By scaling, WLOG we can assume that $C = 1$. Note that having a larger value of Δ_0 should help in filling more bins as users have more choice in how they can use their budget to increment the weight of items.

In this paper, we consider algorithms which *iteratively* construct the weighted histogram. That is, in our algorithms, we consider users in a random order, and each user updates the weighted histogram using the update policy ϕ. Algorithm 1 is a meta-algorithm for DP set union, and all our subsequent algorithms follow this framework.

If the update policy is such that it increments the weights of items independent of other users (as done in WEIGHTED LAPLACE and WEIGHTED GAUSSIAN), then it is not hard to see that sensitivity of H can be bounded by 1; that is, by the budget of each user. However, if some item is already way above the threshold ρ, then it doesn't make much sense to waste the limited budget on that item. So the users can choose a clever *update policy* to distribute their budget among the W_i items based on the current weights.

Note that if a policy is such that updates of a user depend on other users, it can be quite tricky to bound the sensitivity of the resulting weighted histogram. To illustrate this, consider for example the *greedy* update policy. Each user i can use his budget of 1 to fill the bins that is closest to the threshold among the bins $u \in W_i$. If an item already reached the threshold, the user can spend his remaining budget incrementing the weight of next bin that is closest to the threshold and so on. Note that from our Knapsack problem analogy this seems to be a good way to maximize the number of bins filled. However such a greedy policy can have very large sensitivity (see supplementary material for an example), and hence won't lead to any reasonable DP algorithm. So, the main contribution of the paper is

Algorithm 1 High level meta algorithm for DP Set Union

Input: D: Database of n users where each user i has some subset $W_i \subset U$
ρ: threshold
Noise: Noise distribution ($\mathsf{Lap}(0, \lambda)$ or $\mathcal{N}(0, \sigma^2)$)
Output: S: A subset of $\cup_i W_i$
Build weighted histogram H supported over $\cup_i W_i$ using Algorithm 2. $S = \emptyset$ (empty set)
for $u \in \cup_i W_i$ **do**
  $\hat{H}[u] \leftarrow H[u] +$ Noise
  if $\hat{H}[u] > \rho$ **then**
    $S \leftarrow S \cup \{u\}$
  end if
end for
Output S

Algorithm 2 High level meta algorithm for building weighted histogram using a given update policy

Input: D: Database of n users where each user i has some subset $W_i \subset U$
Δ_0: maximum contribution parameter
hash: A random hash function which maps user ids into some large domain without collisions
ϕ: Update policy for a user to update the weights of items in their set
Output: H: A weighted histogram in $\mathbb{R}^{\cup_i W_i}$
$H = \{\}$ (empty histogram)
Sort users into $\text{User}_1, \text{User}_2, \ldots, \text{User}_n$ by sorting the hash values of their user ids under the hash function hash
for $i = 1$ **to** n **do**
  $W_i \leftarrow$ set of User_i
  if $|W_i| > \Delta_0$ **then**
    $W_i' \leftarrow$ Randomly choose Δ_0 items from W_i
  else
    $W_i' \leftarrow W_i$
  end if
  Update $H[u]$ for each $u \in W_i'$ using update policy ϕ
end for
Output H

in showing policies which help maximize the number of items that are filled while keeping the sensitivity low. In particular, we define a general class of ℓ_p-contractive update policies and show that they produce weighted histograms with bounded ℓ_p-sensitivity.

Definition 1.3 (ℓ_p-contractive update policy)**.** We say that an update policy ϕ is ℓ_p-contractive if there exists a subset $\mathcal{I}$ (called the invariant subset for ϕ) of pairs of weighted histograms which are at an ℓ_p distance of at most 1, i.e.,

$$\mathcal{I} \subset \left\{(H_1, H_2) : \|H_1 - H_2\|_{\ell_p} \leq 1\right\}$$

such that the following conditions hold.

1. (Invariance) $(H_1, H_2) \in \mathcal{I} \Rightarrow (\phi(H_1), \phi(H_2)) \in \mathcal{I}$.[2]
2. $(\phi(H), H) \in \mathcal{I}$ for all H.

Property (2) of Definition 1.3 requires that the update policy can change the histogram by an ℓ_p distance of at most 1 (budget of a user).

Theorem 1.1 (Contractivity implies bounded sensitivity)**.** Suppose ϕ is an update policy which is ℓ_p-contractive over some invariant subset $\mathcal{I}$. Then the histogram output by Algorithm 2 has ℓ_p-sensitivity bounded by 1.

We prove Theorem 1.1 in Section 2. Once we have bounded ℓ_p-sensitivity, we can get a DP Set Union algorithm with some additional technical work.

Theorem 1.2. (Informal: Bounded sensitivity implies DP) For $p \in \{1, 2\}$, if the ℓ_p-sensitivity of the weighted histogram output by Algorithm 2 is bounded, then Algorithm 1 for DP Set Union can be made (ε, δ)-differentially private by appropriately choosing the noise distribution (Noise) and threshold (ρ).

The main contribution of the paper is two new algorithms guided by Theorem 1.1. The first algorithm, which we call POLICY LAPLACE, uses update policy that is ℓ_1-contractive. The second algorithm, which we call POLICY GAUSSIAN, uses update policy that is ℓ_2-contractive. Finally we show that our algorithms significantly outperform the weighted policies.

At a very high-level, the role of contractivity in our algorithms is indeed similar to its role in the recent elegant work of Feldman et al. (2018). They show that if an iterative algorithm is contractive in each step, then adding Gaussian noise in each iteration will lead to strong privacy amplification. In particular, users who make updates early on will enjoy much better privacy guarantees. However their framework is not applicable in our setting, because their algorithm requires adding noise to the count of every item in every iteration; this will lead to unbounded growth of counts and items which belong to only a single user can also get output which violates privacy.

[2]Note that property (1) is a slightly weaker requirement than the usual notion of ℓ_p-contractivity which requires $\|\phi(H_1) - \phi(H_2)\|_{\ell_p} \leq \|H_1 - H_2\|_{\ell_p}$ for all H_1, H_2. Instead we require contraction only for $(H_1, H_2) \in \mathcal{I}$.

2. Contractivity implies DP algorithms

In this section, we show that if an update policy satisfies contractive property as in Definition 1.3, we can use it to develop a DP algorithm for DPSU. First we show that contractivity of update policy implies bounded sensitivity (Theorem 1.1), which in turn implies a DP Set Union algorithm by Theorem 1.2

We will first define sensitivity and update policy formally. Let $\mathcal{D}$ denote the collection of all databases. We say that D, D' are neighboring databases, denoted by $D \sim D'$, if they differ in exactly one user.

Definition 2.1. For $p \geq 0$, the ℓ_p-sensitivity of $f : \mathcal{D} \to \mathbb{R}^k$ is defined as $\sup_{D \sim D'} \|f(D) - f(D')\|_{\ell_p}$ where the supremum is over all neighboring databases D, D'.

Proof of Theorem 1.1. Let ϕ be an ℓ_p-contractive update policy with invariant subset $\mathcal{I}$. Consider two neighboring databases D_1 and D_2 where D_1 has one extra user compared to D_2. Let H_1 and H_2 denote the histograms built by Algorithm 1 using the update policy ϕ when the databases are D_1 and D_2 respectively.

Say the extra user in D_1 has position t in the global ordering given by the hash function. Let H_1^{t-1} and H_2^{t-1} be the histograms after the first $t-1$ (according to the global order given by the hash function hash) users' data is added to the histogram. Therefore $H_1^{t-1} = H_2^{t-1}$. And the new user updates H_1^{t-1} to H_1^t. By property (2) in Definition 1.3 of ℓ_p-contractive policy, $(\phi(H_1^{t-1}), H_1^{t-1}) \in \mathcal{I}$. Since $\phi(H_1^{t-1}) = H_1^t$, we have $(H_1^t, H_1^{t-1}) = (H_1^t, H_2^{t-1}) \in \mathcal{I}$. The remaining users are now added to H_1^t, H_2^{t-1} in the same order. Note that we are using the fact that the users are sorted according some hash function and they contribute in that order (this is also needed to claim that $H_1^{t-1} = H_2^{t-1}$). Therefore, by property (1) in Definition 1.3 of ℓ_p-contractive policy, we get $(H_1, H_2) \in \mathcal{I}$. Since $\mathcal{I}$ only contains pairs with ℓ_p-distance at most 1, we have $\|H_1 - H_2\|_{\ell_p} \leq 1$. This proves that the histogram built by Algorithm 2 using ϕ has ℓ_p-sensitivity of at most 1. □

Above theorem implies that once we have a ℓ_p contractive update policy, we can appeal to Theorem 1.2 to design a DP algorithm for DPSU.

3. Policy Laplace Algorithm

In this section we will present an ℓ_1-contractive update policy called ℓ_1-DESCENT (Algorithm 3) and use it to obtain a DP Set Union algorithm called POLICY LAPLACE (Algorithm 4).

3.1. ℓ_1-DESCENT update policy

The policy is described in Algorithm 3. We will set some *cutoff* Γ above the threshold ρ to use in the update policy. Once the weight of an item ($H[u]$) crosses the cutoff, we don't want to increase it further. In this policy, each user starts with a budget of 1. The user uniformly increases $H[u]$ for each $u \in W_i'$ s.t. $H[u] < \Gamma$. Once some item's weight reaches Γ, the user stops increasing that item and keeps increasing the rest of the items until the budget of 1 is expended. To implement this efficiently, the Δ_0 items from each user are sorted based on distance to the cutoff. Beginning with the item whose weight is closest to the cutoff Γ (but still below the cutoff), say item u, we will add $\Gamma - H[u]$ (gap to cutoff for item u) to each of the items below the cutoff. This repeats until the user's budget of 1 has been expended.

This policy can also be interpreted as *gradient descent* to minimize the ℓ_1-distance between the current weighted histogram and the point $(\Gamma, \Gamma, \ldots, \Gamma)$, hence the name ℓ_1-DESCENT. Since the gradient vector is 1 in coordinates where the weight is below cutoff Γ and 0 in coordinates where the weight is Γ, the ℓ_1-DESCENT policy is moving in the direction of the gradient until it has moved a total ℓ_1-distance of at most 1.

3.2. POLICY LAPLACE

The POLICY LAPLACE algorithm (Algorithm 4) for DPSU uses the framework of the meta algorithm in Algorithm 1 using the update policy in Algorithm 3. Since the added noise is $\mathsf{Lap}(0, \lambda)$, which is centered at 0, we want to set the cutoff Γ in the update policy to be sufficiently above the threshold ρ. Thus we pick $\Gamma = \rho_{\mathsf{Lap}} + \alpha \cdot \lambda$ for some $\alpha > 0$. From our experiments, choosing $\alpha \in [2, 6]$ works best empirically. The parameters $\lambda, \rho_{\mathsf{Lap}}$ are set so as to achieve (ε, δ)-DP as shown in Theorem 3.1.

3.3. Privacy analysis of POLICY LAPLACE

In this section we will prove that the POLICY LAPLACE algorithm (Algorithm 4) is (ε, δ)-DP. By Theorem 1.2 and Theorem 1.1, it is enough to show that ℓ_1-DESCENT policy (Algorithm 3) is ℓ_1-contractive. For two histograms G_1, G_2, we write $G_1 \geq G_2$ if $G_1[u] \geq G_2[u]$ for each every item u. $G_1 \leq G_2$ is defined similarly.

Lemma 3.1. Let $\mathcal{I} = \{(G_1, G_2) : G_1 \geq G_2,\ \|G_1 - G_2\|_{\ell_1} \leq 1\}$. Then ℓ_1-DESCENT update policy is ℓ_1-contractive over the invariant subset $\mathcal{I}$.

Algorithm 3 ℓ_1-DESCENT update policy

Input: H: Current histogram
W: A subset of U of size at most Δ_0
Γ: cutoff parameter
Output: H: Updated histogram
// Build cost dictionary G
$G = \{\}$ // Empty dictionary
for $u \in W$ **do**
  if $H[u] < \Gamma$ **then**
    // Gap to cutoff for items below cutoff Γ
    $G[u] \leftarrow \Gamma - H[u]$
  end if
end for
budget $\leftarrow 1$ // Each user gets a total budget of 1
$K \leftarrow |G|$ // Number of items still under cutoff
// Sort in increasing order of the gap $\Gamma - H[u]$
$G \leftarrow \mathrm{sort}(G)$
// Let $u_1, u_2, \ldots, u_{|G|}$ be the sorted order
for $j = 1$ to $|G|$ **do**
  // Cost of increasing weights of remaining K items by $G[u_j]$
  cost = $G[u_j] \cdot K$
  if cost $\leq$ budget **then**
    for $\ell = j$ to $|G|$ **do**
      $H[u_\ell] \leftarrow H[u_\ell] + G[u_j]$
      // Gap to cutoff is reduced by $G[u_j]$
      $G[u_\ell] \leftarrow G[u_\ell] - G[u_j]$
    end for
    budget $\leftarrow$ budget - cost
    // u_j has reached cutoff, so decrease K by 1
    $K \leftarrow K - 1$
  else
    for $\ell = j$ to $|G|$ **do**
      // Update item weights by as much as remaining budget allows
      $H[u_\ell] \leftarrow H[u_\ell] + \frac{\text{budget}}{K}$
      break
    end for
  end if
end for

Proof. Let ϕ denote the ℓ_1-DESCENT update policy.

We will first show property (2) of Definition 1.3. Let G be any weighted histogram and let $G' = \phi(G)$. Clearly $G' \geq G$ as the new user will never decrease the weight of any item. Moreover, the total change to the histogram is at most 1 in ℓ_1-distance. Therefore $\|G' - G\|_{\ell_1} \leq 1$. Therefore $(G', G) \in \mathcal{I}$.

We will now prove property (1) of Definition 1.3. Let $(G_1, G_2) \in \mathcal{I}$, i.e., $G_1 \geq G_2$ and $\|G_1 - G_2\|_{\ell_1} \leq 1$. Let $G_1' = \phi(G_1), G_2' = \phi(G_2)$. A new user can increase G_1 and G_2 by at most 1 in ℓ_1 distance. Let Γ be the cutoff

Algorithm 4 POLICY LAPLACE algorithm for DPSU

Input: D: Database of n users where each user has some subset $W \subset U$
Δ_0: maximum contribution parameter
(ε, δ): privacy parameters
α: parameter for setting cutoff
Output: S: A subset of $\cup_i W_i$
$\lambda \leftarrow 1/\varepsilon$ // Noise parameter in $\mathsf{Lap}(0, \lambda)$
// Threshold parameter
$\rho_{\mathsf{Lap}} \leftarrow \max_{1 \le t \le \Delta_0} \frac{1}{t} + \frac{1}{\varepsilon} \log\left(\frac{1}{2\left(1-(1-\delta)^{1/t}\right)}\right)$
$\Gamma \leftarrow \rho_{\mathsf{Lap}} + \alpha \cdot \lambda$ // Cutoff parameter for update policy
Run Algorithm 1 with $\mathsf{Noise} \sim \mathsf{Lap}(0, \lambda)$ and the ℓ_1-DESCENT update policy in Algorithm 3 to output S.

parameter in Algorithm 3. Let S be the set of Δ_0 items with the new user, therefore only the items in S will change in G_1', G_2'. WLOG, we can assume that the user changes both G_1 and G_2 by exactly total ℓ_1 distance of 1. Otherwise, in at least one of them all the items in S should reach the cutoff Γ. If this happens with G_1, then clearly $\Gamma = G_1'[u] \ge G_2'[u]$ for all $u \in S$. But it is easy to see that if this happens with G_2, then it should also happen with G_1 in which case $G_1'[u] = G_2'[u] = \Gamma$ for $u \in S$.

Imagine that at time $t = 0$, the user starts pushing mass continuously at a rate of 1 to both G_1, G_2 until the entire mass of 1 is sent, which happens at time $t = 1$. The mass flow is equally split among all the items which haven't yet crossed cutoff. Let G_1^t and G_2^t be the histograms at time t. Therefore $G_i^0 = G_i$ and $G_i^1 = G_i'$. We claim that $G_1^t \ge G_2^t$ implies that $\frac{dG_1^t[u]}{dt} \ge \frac{dG_2^t[u]}{dt}$ for all $u \in S$ s.t. $G_1^t[u] < \Gamma$. This is because the flow is split equally among items which didn't cross the cutoff, and there can only be more items in G_2^t which didn't cross the the cutoff when compared to G_1^t. And at time $t = 0$, we have $G_1^0 \ge G_2^0$. Therefore, we have $G_1^t \ge G_2^t$ for all $t \in [0, 1]$ and so $G_1' \ge G_2'$.

We will now prove ℓ_1-contraction. Let $C_i = \|G_i - G_i'\|_{\ell_1}$. By the discussion above, $C_1 \le C_2$ (either total mass flow is equal to 1 for both or all items in S will reach cutoff Γ in G_1 before this happens in G_2).

$$
\begin{aligned}
&\|G_1' - G_2'\|_{\ell_1} \\
&= \sum_{u \in S} G_1'[u] - \sum_{u \in S} G_2'[u] && \text{(Since } G_1' \ge G_2'\text{)} \\
&= \sum_{u \in S} G_1[u] - \sum_{u \in S} G_2[u] + C_1 - C_2 \\
&\le \sum_{u \in S} G_1[u] - \sum_{u \in S} G_2[u] && \text{(Since } C_1 \le C_2\text{)} \\
&= \|G_1 - G_2\|_{\ell_1} && \text{(Since } G_1 \ge G_2\text{)} \\
&\le 1.
\end{aligned}
$$

Therefore $(G_1', G_2') \in \mathcal{I}$ which proves property (2) of Definition 1.3. □

We now state a formal theorem which proves $(\varepsilon, \delta) - DP$ of POLICY LAPLACE algorithm[3].

Theorem 3.1. The POLICY LAPLACE algorithm (Algorithm 4) is (ε, δ)-DP when

$$\rho_{\mathsf{Lap}} \ge \max_{1 \le t \le \Delta_0} \frac{1}{t} + \frac{1}{\varepsilon} \log\left(\frac{1}{2\left(1-(1-\delta)^{1/t}\right)}\right).$$

4. Policy Gaussian Algorithm

In this section we will present an ℓ_2-contractive update policy called ℓ_2-DESCENT (Algorithm 5) and use it to obtain a DP Set Union algorithm called POLICY GAUSSIAN (Algorithm 4).

4.1. ℓ_2-DESCENT update policy

Similar to the Laplace update policy, we will set some *cutoff* Γ above the threshold ρ and once an item's count ($H[u]$) crosses the cutoff, we don't want to increase it further. In this policy, each user starts with a budget of 1. But now, the total change a user can make to the histogram can be at most 1 when measured in ℓ_2-norm (whereas in Laplace update policy we used ℓ_1-norm to measure change). In other words, sum of the squares of the changes that the user makes is at most 1. Since we want to get as close to the cutoff (Γ) as possible, the user moves the counts vector (restricted to the set W of Δ_0 items the user has) in the direction of the point $(\Gamma, \Gamma, \ldots, \Gamma)$ by an ℓ_2-distance of at most 1. This update policy is presented in Algorithm 5.

This policy can also be interpreted as *gradient descent* to minimize the ℓ_2-distance between the current weighted histogram and the point $(\Gamma, \Gamma, \ldots, \Gamma)$, hence the name ℓ_2-DESCENT. Since the gradient vector is in the direction of the line joining the current point and $(\Gamma, \Gamma, \ldots, \Gamma)$, the ℓ_2-DESCENT policy is moving the current histogram towards $(\Gamma, \Gamma, \ldots, \Gamma)$ by an ℓ_2-distance of at most 1.

4.2. POLICY GAUSSIAN

The POLICY GAUSSIAN algorithm (Algorithm 6) for DPSU uses the framework of the meta algorithm in Algorithm 1 using the Gaussian update policy (Algorithm 5). Since the added noise is $\mathcal{N}(0, \sigma^2)$ which is centered at 0, we want to set the cutoff Γ in the update policy to be sufficiently above (but not too high above) the threshold ρ_{Gauss}. Thus we pick $\Gamma = \rho_{\mathsf{Gauss}} + \alpha \cdot \sigma$ for some $\alpha > 0$. From our experiments, choosing $\alpha \in [2, 6]$ empirically yields these best results.

[3]The proof for this theorem can be found in the supplementary material.

Algorithm 5 ℓ_2-DESCENT update policy

Input: H: Current histogram
W: A subset of U of size at most Δ_0
Γ: cutoff parameter
Output: H: Updated histogram
$G = \{\}$ // Empty dictionary
for $u \in W$ **do**
  // G is the vector joining $H|_W$ to $(\Gamma, \Gamma, \ldots, \Gamma)$
  $G[u] \leftarrow \Gamma - H[u]$
end for
// ℓ_2-distance between $H|_W$ and $(\Gamma, \Gamma, \ldots, \Gamma)$
$Z \leftarrow \left(\sum_{u \in W} G[u]^2\right)^{1/2}$
// If $Z \leq 1$, then the user moves $H|_W$ to $(\Gamma, \Gamma, \ldots, \Gamma)$. Else, move $H|_W$ in the direction of $(\Gamma, \Gamma, \ldots, \Gamma)$ by an ℓ_2-distance of at most 1
if $Z < 1$ **then**
  for $u \in W$ **do**
    $H[u] \leftarrow \Gamma$
  end for
else
  for $u \in W$ **do**
    $H[u] \leftarrow H[u] + \frac{G[u]}{Z}$
  end for
end if

The parameters $\sigma, \rho_{\mathsf{Gauss}}$ are set so as to achieve (ε, δ)-DP as shown in Theorem 4.1. $\Phi(\cdot)$ is the cumulative density function of standard Gaussian distribution and $\Phi^{-1}(\cdot)$ is its inverse.

Algorithm 6 POLICY GAUSSIAN algorithm for DPSU

Input: D: Database of n users where each user has some subset $W \subset U$
Δ_0: maximum contribution parameter
(ε, δ): privacy parameters
α: parameter for setting cutoff
Output: S: A subset of $\cup_i W_i$
// Standard deviation in Gaussian noise
$\sigma \leftarrow \min\left\{\sigma : \Phi\left(\frac{1}{2\sigma} - \varepsilon\sigma\right) - e^{\varepsilon}\Phi\left(-\frac{1}{2\sigma} - \varepsilon\sigma\right) \leq \frac{\delta}{2}\right\}$
// Threshold parameter
$\rho_{\mathsf{Gauss}} \leftarrow \max_{1 \leq t \leq \Delta_0}\left(\frac{1}{\sqrt{t}} + \sigma\Phi^{-1}\left(\left(1 - \frac{\delta}{2}\right)^{1/t}\right)\right)$
$\Gamma \leftarrow \rho_{\mathsf{Lap}} + \alpha \cdot \sigma$ // Cutoff parameter for update policy
Run Algorithm 1 with $\mathsf{Noise} \sim \mathcal{N}(0, \sigma^2)$ and the ℓ_2-DESCENT update policy in Algorithm 5 to output S.

To find $\min\left\{\sigma : \Phi\left(\frac{1}{2\sigma} - \varepsilon\sigma\right) - e^{\varepsilon}\Phi\left(-\frac{1}{2\sigma} - \varepsilon\sigma\right) \leq \frac{\delta}{2}\right\}$, one can use binary search because $\Phi\left(\frac{1}{2\sigma} - \varepsilon\sigma\right) - e^{\varepsilon}\Phi\left(-\frac{1}{2\sigma} - \varepsilon\sigma\right)$ is a decreasing function of σ. An efficient and robust implementation of this binary search can be found in (Balle & Wang, 2018).

4.3. Privacy analysis of POLICY GAUSSIAN

In this section we will prove that the POLICY GAUSSIAN algorithm (Algorithm 6) is (ε, δ)-DP. By Theorem 1.2 and Theorem 1.1, it is enough to show ℓ_2-contractivity of ℓ_2-DESCENT update policy. We will need a simple plane geometry lemma for this. A proof can be found in the supplementary material.

Lemma 4.1. Let A, B, C denote the vertices of a triangle in the Euclidean plane. If $|AB| > 1$, let B' be the point on the side AB which is at a distance of 1 from B and if $|AB| \leq 1$, define $B' = A$. C' is defined similarly. Then $|B'C'| \leq |BC|$.

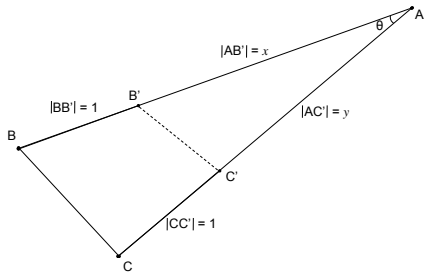

Figure 2. Geometric illustration of Lemma 4.1 when $|AB|, |AC| > 1$. The lemma implies that $|B'C'| \leq |BC|$.

Lemma 4.2. Let $\mathcal{I} = \{(G_1, G_2) : \|G_1 - G_2\|_{\ell_2} \leq 1\}$. Then the ℓ_2-DESCENT update policy is ℓ_2-contractive over the invariant set $\mathcal{I}$.

Proof. Let ϕ denote the ℓ_2-DESCENT policy. Property (2) of Definition 1.3 follows easily because each new user can only change a weighted histogram by an ℓ_2-distance of at most 1.

We will now prove Property (1) of Definition 1.3. Let $(G_1, G_2) \in \mathcal{I}$, i.e., $\|G_1 - G_2\|_{\ell_2} \leq 1$. Let $G_1' = \phi(G_1)$ and $G_2' = \phi(G_2)$. A new user can increase G_1 and G_2 by at most 1 in ℓ_2 distance. Let Γ be the cutoff parameter in Algorithm 5. Let S be the set of Δ_0 items with the new user, therefore only the items in S will change in G_1', G_2'. Therefore we can just assume that G_1, G_2 are supported on S for the sake of the analysis. Algorithm 6 moves G_i towards $P = (\Gamma, \Gamma, \ldots, \Gamma)$ by an ℓ_2-distance of 1 (or to P if the distance to P is already lower than 1). We can restrict ourselves to the plane containing G_1, G_2, P (G_1', G_2' will also lie on the same plane). Now by Lemma 4.1, $\|G_1' - G_2'\|_{\ell_2} \leq \|G_1 - G_2\|_{\ell_2}$. □

We now state a formal theorem which proves $(\varepsilon, \delta) - DP$ of POLICY GAUSSIAN algorithm[4].

[4]The proof for this theorem can be found in the supplementary material.

Theorem 4.1. The POLICY GAUSSIAN algorithm (Algorithm 6) is (ε, δ)-DP if $\sigma, \rho_{\text{Gauss}}$ are chosen s.t.

$$\Phi\left(\frac{1}{2\sigma} - \varepsilon\sigma\right) - e^{\varepsilon}\Phi\left(-\frac{1}{2\sigma} - \varepsilon\sigma\right) \leq \frac{\delta}{2} \text{ and}$$

$$\rho_{\text{Gauss}} \geq \max_{1\leq t\leq \Delta_0}\left(\frac{1}{\sqrt{t}} + \sigma\Phi^{-1}\left(\left(1-\frac{\delta}{2}\right)^{1/t}\right)\right).$$

5. Experiments

While the algorithms we described generalize to many domains that involve the release of set union, our experiments will use a natural language dataset. In the context of n-gram release, D is a database of users where each user is associated with 1 or more Reddit posts and W_i is the set of unique n-grams used by each user. The goal is to output as large a subset of n-grams $\cup_i W_i$ as possible while providing (ε, δ)-differential privacy to each user. In our experiments we consider $n = 1$ (unigrams)[5].

5.1. Dataset

Our dataset is collected from the subreddit `r/AskReddit`. We take a sample of 15,000 posts from each month between January 2017 and December 2018. We filter out duplicate entries, removed posts, and deleted authors. For text preprocessing, we remove URLs and symbols, lowercase all words, and tokenize using `nltk.word_tokenize`. After preprocessing, we again filter out empty posts to arrive at a dataset of 373,983 posts from 223,388 users.

Similar to other natural language datasets, this corpus follows Zipf's law across users. The frequency of unigrams across users is inversely proportional to its rank of the unigram. The distribution of how many unigrams each user uses also follows a long tail distribution. While the top 10 users contribute 850-2000 unique unigrams, most users (93.1%) contribute less than 100 unique unigrams.

5.2. Results

For the problem of outputting the large possible set of unigrams, Table 1 and Figure 3, summarize the performance of DP set union algorithms for different values of Δ_0. The privacy parameters are $\varepsilon = 3$ and $\delta = \exp(-10)$. For each algorithm, we average the results of 5 different shuffles of user ordering and also include the standard deviation in Table 1. We compare our algorithms with baseline algorithms: COUNT LAPLACE, COUNT GAUSSIAN, WEIGHTED LAPLACE, and WEIGHTED GAUSSIAN discussed previously[6]. Our conclusions are as follows:

[5]Our code is made available here `https://github.com/heyyjudes/differentially-private-set-union`.

[6]Additional experiments with different α and ε values are included in the supplementary materials

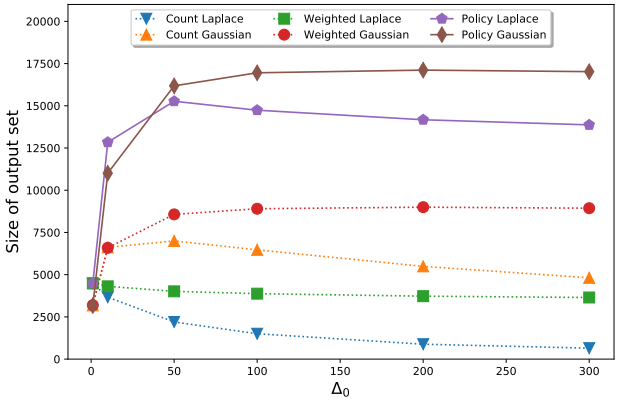

Figure 3. Count of unigrams released by set union algorithms averaged across 5 shuffles of user order. Privacy parameters are $\varepsilon = 3$ and $\delta = \exp(-10)$. The cutoff Γ is calculated using $\alpha = 5$.

- Our new algorithms POLICY LAPLACE and POLICY GAUSSIAN output a DP set union that is 2-4 times larger than output of weighted/count based algorithms. This holds for all values of $\varepsilon \geq 1$.
- To put the size of released set in context, we compare our new algorithms against the number of unigrams belonging to at least k users (See Table 2). For POLICY LAPLACE with $\Delta_0 = 100$, the size of the output set covers almost all unigrams (94.8%) when $k = 20$ and surpasses the size of the output set when $k \geq 25$. POLICY GAUSSIAN with $\Delta_0 = 100$ covers almost all unigrams (91.8%) when $k = 15$ and surpasses the size of the output set when $k \geq 18$. In other words, our algorithms (with $\varepsilon = 3$ and $\delta = \exp(-10)$) perform better than k-anonymity based algorithms for values of k around 20.

5.2.1. SELECTING HYPERPARAMETERS WHILE MAINTAINING PRIVACY

As can be seen from Table 1 the Δ_0 resulting in the largest output set varies by algorithm. Since most users in our dataset possess less than 300 unique unigrams, it is not surprising that the largest output set can be achieved with $\Delta_0 < 300$. However, running our algorithms for different values of Δ_0 and selecting the best output will result in a higher value of ε. There are several ways to find the best value of Δ_0 (or any other tunable parameter): 1) using prior knowledge of the data 2) running the algorithms on a small sample of the data to find the best parameters, and discarding that sample. 3) finally, one could also run all the algorithms in parallel and choose the best performing one. Here we will have to account for the loss in privacy budget; see (Liu & Talwar, 2019) for example.

Table 1. Count of unigrams released by various set union algorithms. Results are averaged across 5 shuffles of user order. The best results for each algorithm are in bold. The privacy parameters are $\varepsilon = 3$ and $\delta = \exp(-10)$. $\alpha = 5$ is chosen for the cutoff parameter Γ.

Δ_0	**1**	**10**	**50**	**100**	**200**	**300**
COUNT LAPLACE	**4484** $\pm$ 32	3666 $\pm$ 7	2199 $\pm$ 8	1502 $\pm$ 14	882 $\pm$ 4	647 $\pm$ 4
COUNT GAUSSIAN	3179 $\pm$ 15	6616 $\pm$ 18	**6998** $\pm$ 23	6470 $\pm$ 12	5492 $\pm$ 14	4813 $\pm$ 14
WEIGHTED LAPLACE	**4479** $\pm$ 26	4309 $\pm$ 15	4012 $\pm$ 10	3875 $\pm$ 9	3726 $\pm$ 17	3648 $\pm$ 12
WEIGHTED GAUSSIAN	3194 $\pm$ 11	6591 $\pm$ 18	8570 $\pm$ 14	8904 $\pm$ 24	**8996** $\pm$ 30	8936 $\pm$ 12
POLICY LAPLACE	4482 $\pm$ 21	12840 $\pm$ 28	**15268** $\pm$ 10	14739 $\pm$ 23	14173 $\pm$ 25	13870 $\pm$ 23
POLICY GAUSSIAN	3169 $\pm$ 13	11010 $\pm$ 15	16181 $\pm$ 33	16954 $\pm$ 58	**17113** $\pm$ 16	17022 $\pm$ 57

Table 2. This table shows the total number of unigrams that at least k users possess ($|\mathcal{S}_k|$) and the percentage coverage of this total by POLICY LAPLACE ($|\mathcal{S}_{PL}| = 14739$) and POLICY GAUSSIAN ($|\mathcal{S}_{PG}| = 16954$) for $\Delta_0 = 100$.

k	$\lvert\mathcal{S}_k\rvert$	% COVERAGE POLICY LAPLACE	% COVERAGE POLICY GAUSSIAN
5	34699	24.5%	48.9%
10	23471	62.8%	72.2%
15	18461	79.8%	91.8%
18	16612	88.7%	102.1%
20	15550	94.8%	109.0%
25	13638	108.1%	124.3%

6. Conclusion

We initiated the study of differentially private set union problem, which has many real-world applications. We designed better algorithms for the problem using the notion of contractive update policy as a guiding principle to preserve privacy. From a set of empirical experiments on a Reddit natural language dataset, we demonstrate that our policy algorithms release a larger set-union than previous algorithms.

One immediate question is to give theoretical guarantees on the size of set union produced by our algorithms. A more interesting and significantly challenging question is to design instance optimal algorithms for the problem. Given the ubiquitous nature of this problem, we believe that it is a worthwhile direction to explore.

References

Abowd, J. M. The challenge of scientific reproducibility and privacy protection for statistical agencies. Technical report, Census Scientific Advisory Committee, 2016.

Apple, D. P. T. Learning with privacy at scale. Technical report, Apple, 2017.

Avent, B., Korolova, A., Zeber, D., Hovden, T., and Livshits, B. Blender: enabling local search with a hybrid differential privacy model. In *Proc. of the 26th USENIX Security Symposium*, pp. 747–764, 2017.

Balle, B. and Wang, Y.-X. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pp. 403–412, 2018.

Bittau, A., Erlingsson, U., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pp. 441–459, 2017.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 267–284, 2019.

Chen, M. X., Lee, B. N., Bansal, G., Cao, Y., Zhang, S., Lu, J., Tsay, J., Wang, Y., Dai, A. M., Chen, Z., and et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2287–2295, 2019.

Deb, B., Bailey, P., and Shokouhi, M. Diversifying reply suggestions using a matching-conditional variational autoencoder. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Ding, B., Kulkarni, J., and Yekhanin, S. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pp. 3574–3583, 2017.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pp. 265–284. Springer, 2006.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Erlingsson, Ú., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067. ACM, 2014.

Feldman, V., Mironov, I., Talwar, K., and Thakurta, A. Privacy amplification by iteration. In Thorup, M. (ed.), *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pp. 521–532. IEEE Computer Society, 2018.

Hu, B., Lu, Z., Li, H., and Chen, Q. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pp. 2042–2050, 2014.

Kannan, A., Kurach, K., Ravi, S., Kaufmann, T., Tomkins, A., Miklos, B., Corrado, G., Lukacs, L., Ganea, M., Young, P., et al. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 955–964, 2016.

Korolova, A., Kenthapadi, K., Mishra, N., and Ntoulas, A. Releasing search queries and clicks privately. In *Proceedings of the 18th international conference on World wide web*, pp. 171–180, 2009.

Kuo, Y.-H., Chiu, C.-C., Kifer, D., Hay, M., and Machanavajjhala, A. Differentially private hierarchical group size estimation. *arXiv preprint arXiv:1804.00370*, 2018.

Liu, J. and Talwar, K. Private selection from private candidates. In Charikar, M. and Cohen, E. (eds.), *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 298–309. ACM, 2019. doi: 10.1145/3313276.3316377. URL https://doi.org/10.1145/3313276.3316377.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pp. 94–103. IEEE Computer Society, 2007. doi: 10.1109/FOCS.2007.41. URL https://doi.org/10.1109/FOCS.2007.41.

Vadhan, S. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pp. 347–450. Springer, 2017.

Wilson, R., Zhang, C. Y., Lam, W., Desfontaines, D., Simmons-Marengo, D., and Gipson, B. Differentially private SQL with bounded user contribution. 2020.