Differential Privacy for Black-Box Statistical Analyses

Nitin Kohli
UC Berkeley Center for Effective Global Action
Berkeley, CA, USA
nitin.kohli@berkeley.edu

Paul Laskowski
UC Berkeley School of Information
Berkeley, CA, USA
paul@ischool.berkeley.edu

ABSTRACT

We formalize a notion of a privacy wrapper, defined as an algorithm that can take an arbitrary and untrusted script and produce an output with differential privacy guarantees. Our novel privacy wrapper, named TAHOE, incorporates two design ideas: a type of stability under subsetting, and randomization over subset size. We show that TAHOE imposes differential privacy for every possible script. When the data alphabet is finite and small enough, TAHOE can be practically run on a single computer. Performance simulations show that TAHOE has greater accuracy than a benchmark algorithm based on a subsample-and-aggregate approach for certain scenarios and parameter values.

KEYWORDS

Differential Privacy, Untrusted Code, Black Box, Statistics

1 INTRODUCTION

Consider a scenario involving a holder of personal data and a researcher. The researcher has written an analysis script and would like to apply it to the data in order to gain insight. The data holder is concerned with the privacy of individuals in the data, and may not necessarily trust the researcher. Furthermore, the data holder may have contractual obligations that prevent them from sharing information about individuals in the data. The researcher gives the data holder their analysis script, but the data holder does not have the resources or expertise to analyze the code for privacy threats. Instead, the data holder desires a simple way to add privacy guarantees to the researcher's script, without looking inside it, so that the output can be safely returned. This motivates the central question of this study:

> What privacy guarantees can be added to a statistical analysis script from an untrusted source, treating it in a black box fashion?

Although many organizations are interested in increasing researcher access to data, the costs given current technology can be considerable. For example, the US Census Bureau has a program to allow researchers to interact with raw data. To maintain privacy, however, the Bureau maintains a set of secure locations around the country and puts all applicants through a background check.[1] In another example, Facebook and Social Science One teamed up to make data about shared URLs available to researchers using differential privacy [18]. However, to work with the most sensitive data, researchers must be pre-approved by a university review board, pass an application process, and undergo monitoring - even at the level of individual keystrokes.[2]

In contrast with existing approaches, the goal of this study is to enable a researcher to work with private data in an automatic fashion, without the need for screening procedures to establish trust. We will refer to a system that achieves this as a *privacy wrapper*. Akin to a function wrapper in programming, the idea is to write an algorithm that mediates all interaction with the researcher's script, producing output that is based on the behavior of the script, while also yielding strong privacy guarantees.

1.1 Considerations for Untrusted Code

Running code from untrusted sources introduces special challenges to the data holder. To borrow terms from the security literature, some researchers that submit scripts may be *honest*, meaning they abide by the intended use of the system. Others may be considered *malicious*, meaning that their true goal is to extract private information from the data. A lack of trust suggests that all researchers must be considered potentially malicious.

A malicious researcher has a variety of techniques to extract private information from a dataset. Suppose a researcher is looking for information about a target individual, t. The researcher could write a script that hides t's income in the trailing digits of the output. They could code the script so the sum of the digits in the output is even if t is in the data, and odd otherwise. They could make the script deliberately fail if a certain medical diagnosis appears in t's records. Providing further challenges, a malicious researcher may *obfuscate* their code, making it difficult to determine its true purpose. Obfuscation may be performed by hand, or a variety of automated obfuscation tools may be used by researchers without special expertise [2, 20].

Following common practice, the data holder might hope to apply differential privacy by adding noise related to the researcher script's *sensitivity* [11]. Informally, sensitivity refers to how much the output of a script can change, if one were to change a single row of the data. Unfortunately, determining the sensitivity of a researcher script may be difficult, or even impossible.

For an instructive example, again suppose that that an adversary is interested in a specific target, t, which may or may not be in the data. The adversary writes a script that outputs 0 unless t is in the data, in which case the script outputs some large $V \gg 1$. If t is not in the data, there may be no way to detect that the script is capable of outputting any value other than 0 - especially if the script is obfuscated. The adversary may thus hope to trick the data

[1] For information on US Census' Secure Research Environment, see https://www.census.gov/about/adrm/fsrdc/about/secure_rdc.html.

[2] See *Data & Security* of Social Science One, available at https://socialscience.one/data-security-privacy.

Proceedings on Privacy Enhancing Technologies 2023(3), 418–431

https://doi.org/10.56553/popets-2023-0089

holder into assuming the script has low sensitivity, leading them to add a small amount of noise.

There is also a theoretical reason to believe that determining the sensitivity of a script is difficult. In the language of the theory of computation, the property of meeting a sensitivity bound is semantic and non-trivial. Rice's theorem tells us that the problem of deciding if arbitrary code meets such properties is undecidable [27]. This suggests that a traditional strategy of adding noise based on sensitivity cannot work in all cases.

1.2 Motivating a Novel Privacy Wrapper

To overcome the challenges of untrusted code, this study introduces a novel privacy wrapper, called TAHOE. TAHOE treats the researcher script as a black box and uses subsets of data to ensure a type of sensitivity bound. It has the following properties:

(1) **Privacy:** Given any researcher script, the mechanism generated by the wrapper meets *approximate differential privacy*.
(2) **Accuracy:** The output of the wrapper is related to the researcher script in the sense that it is found by adding (predetermined) noise to the output of the script on *some* subset of data.
(3) **Flexibility:** TAHOE places no limitation on the code written by a researcher. Any script that returns a real vector, including those that may terminate without returning an output, can be used.

A disadvantage of our algorithm is that the runtime can be prohibitively large, depending on the privacy parameters chosen. As we will show in Section 6, TAHOE can be optimized to run efficiently for the special case of a finite data alphabet.

We will describe TAHOE formally in Section 4. Its design is motivated by two central ideas, which we present at an intuitive level.

1.2.1 Remove influential individuals. An individual may be called influential, roughly speaking, if removing them causes a large change in the output of a script. For example, suppose that a researcher submits a script to compute the mean age of participants in a medical trial. If a target individual happens to be very young, an adversary may be able to distinguish whether they are in the data or not by observing the mean. On the other hand, if a privacy wrapper removes the influential individual before running the script, the presence of the individual can no longer be determined in such a direct manner.

Based on this observation, we consider looking for a subset of data, such that removing further individuals does not change the output very much. This intuition will be formalized in our notion of *stable subsets*.

Definition 1. (*Informal Version of Definition* 7) Given dataset D, researcher script R, and threshold $\alpha\lambda \in \mathbb{R}$, a subset $S \subseteq D$ is *stable* if for any $A, B \subseteq S$ of some minimum size, R returns a real vector for both A and B, and $||R(A) - R(B)||_1 \leq \alpha\lambda$.

A formal definition of this property is given in Section 4. Based on our intuition, we search for algorithms that run the script on a stable subset and add noise to protect privacy.

1.2.2 Randomize number of individuals removed. A further problem arises when basing a privacy wrapper on stable subsets: the existence of a stable subset of a given size can itself reveal information about individuals in the data. Suppose that a wrapper was programmed to find a stable subset of some fixed size n, and release some output if it exists. If a stable subset of size n does not exist, the system terminates without returning a value, denoted by the non-response symbol $\perp$. A malicious researcher could exploit this wrapper design to learn whether a target individual, t, is inside the data. An example of a script that does this is given in Algorithm 1.

Algorithm 1: Exploiting fixed n through non-response

Input:
- Dataset $S \subseteq D$

1 (*Return 1*) If $|S| < n$ or $t \in S$, return 1.
2 (*Or non-response*) Else, return $\perp$.

If $t \in D$, the script will output 1 for any subset of size n that contains t. Moreover, any such subset is stable, because if any rows are removed, the size will be less than n, so the script will again return 1. On the other hand, if $t \notin D$, the script will return $\perp$ for any subset of size n, so no stable subsets of size n exist. Thus, a stable subset of size n exists if and only if $t \in D$. Therefore the adversary will know whether the target is in the data, based on whether or not the wrapper returns an output.

To defend against attacks like this, our second idea is to randomize over the size of subsets we consider, n. Intuitively, by making the number of individuals we remove random, we may conceal how many individuals are removed because they are influential.

In Section 4, we define a distribution G, used to select a subset size n. The size ranges from $N - M$ to N, where N is the size of the dataset and M is a parameter representing the maximum number of individuals removed. The specific form of G is chosen to optimize privacy parameters. It is necessary that the internal parameter n is kept secret, in order our privacy guarantees to hold.

1.3 Summary of Results

Our algorithm TAHOE (**T**rim **A**nd with**H**old **O**r **E**xecute), which is formally defined in Section 4, combines the two previous ideas. At a high level, TAHOE begins by selecting a random size n from distribution G. TAHOE then checks if any stable subsets of size n exist for a dataset D, which can be viewed as *trimming* influential individuals from the data. If no stable subset of size n exists, TAHOE *withholds* from responding, represented by non-response $\perp$. Otherwise, TAHOE selects a stable subset S of size n uniformly at random and *executes* the researcher's script R on it, with the addition of Laplace noise.

We prove in Theorem 1 that TAHOE imposes (ϵ, δ)-differential privacy. More precisely, fixing any researcher script R, the resulting behavior of TAHOE can be viewed as a function from the set of possible datasets to distributions over possible outputs. The differential privacy bound applies to this function, and it holds no matter what script R is chosen. Even if every individual is highly influential with respect to script R, differential privacy continues to hold, though the lack of stable subsets implies that TAHOE will always return $\perp$ in this case.

Without restrictions on the set of possible data entries $\mathbb{D}$, TAHOE is too computationally expensive to be practical. We provide a big-O bound in Proposition 5, and argue that (when maintaining $\delta < 1/N$), the runtime is superpolynomial in the size of the dataset.

By contrast, we consider scenarios in which the data alphabet has finite size, $|\mathbb{D}| = f$. In this case, we describe optimizations that allow TAHOE to run efficiently. Proposition 6 shows that the runtime now becomes $O\left(\tau\epsilon^{-f}\ln(\delta^{-1})^f\right)$ as $\epsilon, \delta \to 0$, where τ is a bound on the runtime of R. In Section 7 we describe the results of experiments involving finite data alphabets, in which TAHOE is able to run on a single computer.

In general, measuring the accuracy of TAHOE's output is difficult, because its behavior depends crucially on what is in the script R. Nevertheless, some guarantees can be derived for special cases. Theorem 2 shows that when R returns a statistically consistent estimate for a population parameter, and TAHOE is configured to never return $\perp$, TAHOE's output will be consistent as well.

To better understand the accuracy of TAHOE's output, we perform a set of simulation exercises in Section 7. Each exercise is based around an honest researcher that wishes to perform a stylized analysis task involving a finite data alphabet. As a benchmark for comparison, we also simulate the same tasks under the GUPT system, which is another algorithm that can fill the role of a privacy wrapper [23]. While these systems are very different from each other, we establish guidelines to make our comparison as fair as possible.

The first task we consider is generating a histogram showing the proportion of each possible data entry. In our experiments, we find that GUPT outperforms TAHOE for small datasets. However, above a certain data size, TAHOE's output becomes more accurate, reflecting favorable scaling properties. TAHOE also performs relatively well as the size of the alphabet increases, but less well as ϵ approaches 0.

The second task we consider is computing the sample entropy of the data. We once again find that GUPT outperforms TAHOE for small datasets. TAHOE does gain an advantage as the dataset grows large, but it takes a much larger dataset (above 10^6 rows) before TAHOE becomes more accurate than GUPT. We argue that this is related to the fact that entropy is a one-dimensional quantity, and to the shape of the entropy function.

2 RELATED WORKS

Differential privacy was introduced by Dwork et al. as a rigorous standard for mechanisms that compute real-valued statistics from personal data [8, 9, 11]. The authors pioneered privacy analysis based on global sensitivity, which is defined as the maximal change in a statistic resulting from a one row change to any dataset. Subsequent papers developed variants of the definition of differential privacy [3, 4, 6, 13, 22] whose analyses leverage global sensitivity. A number of studies have since developed differentially private mechanisms leveraging local sensitivity, which is often much smaller than global sensitivity [15, 24, 31]. Such approaches cannot immediately be applied to black box researcher scripts, as neither local nor global sensitivity may be estimated.

Within the differential privacy literature, three high-level strategies can be applied to black boxes. We will refer to these as subsample and aggregate, Lipschitz extension, and restricted language.

In the subsample and aggregate approach, a dataset is divided into (small) subsets and the researcher script is run on each piece. The results are then aggregated together in some way to yield a final answer. Privacy protection emerges from the fact that a single individual can only affects the output resulting from a single subset (or a small fraction). Nissim, Raskhodnikova, and Smith were the first to outline this strategy, providing an example using the median as the aggregation function with noise calibrated to smooth sensitivity [24]. GUPT is a similar system, in which the mean is used as an aggregation function [23]. A feature of both of these systems is that a bound is required on the script output. As an alternative, a script can be run on subsamples, then the median can be released using the the Propose-Test-Release framework of Dwork and Lei [10]. Such a system does not require any bounds on the script output, but there is always some probability that the script returns a non-response character $\perp$ instead of returning an output. The privacy wrapper we propose similarly requires no bounds on script output, and returns $\perp$ in some cases. In cases when the output of a script is categorical instead of metric, aggregation can be performed through noisy voting [16, 26]. Compared to this lineage, our privacy wrapper also examines subsets of data, but does not require an aggregation step. Instead, our work is focused on finding subsets of data with favorable privacy characteristics prior to computing statistics.

In the Lipschitz extension approach, a researcher script is replaced by an approximating function that has low global sensitivity but matches the output of the script on at least one reference dataset. The resulting function is known as a Lipschitz extension, and may be useful if its output matches the original script on typical datasets. Furthermore, the low global sensitivity means that a small amount of noise is sufficient to impose differential privacy. This approach was pioneered by Jha and Raskhodnikova, who developed algorithms for constructing Lipschitz extensions when each data entry takes on a finite set of values [14]. The original script is run on possible databases in a predetermined order, and the output is used to calibrate the extension.

This approach was extended by Cummings and Durfee, who presented algorithms for infinite database domains [5]. In their system, a Lipschitz extension is constructed by first running the source script on the empty database. The output for any other database is found recursively, based on the output of its subsets. A theme that emerges from this work is that a Lipschitz extension is accurate when the output of the original script on the starting reference dataset is close to the output on the actual data. Another feature is that Lipschitz extensions take exponential time to compute, unless the researcher script is a white box with additional structure that can be exploited.

Similarly to the above work, our privacy wrapper leverages subsetting to avoid searching through the entire space of possible datasets. However, we only consider subsets above a minimum size. As long as the researcher script has low sensitivity over this local domain, our privacy wrapper will output accurate results, even if the sensitivity bound is broken on other typical datasets. Additionally, the number of relevant subsets may be far fewer than the

number considered under a Lipschitz extension approach, though it is still too large for practical computation when the data alphabet is infinite.

In the restricted language approach, researchers are permitted to design their own script, but must use a limited language that restricts their access to data. An example of such a system is PINQ, which presents programmers with a SQL-like interface with privacy guarantees [21]. Other differentially private SQL systems have been subsequently developed [15, 19, 32]. In a similar fashion, Kifer et al. propose an architecture in which access to data is mediated by a privacy layer that implements differentially private mechanisms [17].

3 PRELIMINARIES

Let $\mathbb{D}$ be the set of possible entries that represent one individual in a dataset. A *dataset* is represented as a multiset of finite size with entries in $\mathbb{D}$. Let $\mathbb{D}^\star$ be the set of all possible datasets. We say two datasets $D_1, D_2 \in \mathbb{D}^\star$ are *neighbors* if one can be obtained from the other by switching exactly one element. That is, D_1 and D_2 are neighbors if there exists $x \in D_1$ and $y \in D_2$ such that $D_2 = (D_1 \cup \{y\}) \setminus \{x\}$.

Given a nonempty set Ω, let $\Delta(\Omega)$ be the set of all probability distributions over Ω.[3] We define an *algorithm* as a function $A : \mathbb{D}^\star \to \Delta(\mathbb{R}^k \cup \{\perp\})$. Here, $\perp$ is used to represent non-response, meaning an algorithm fails to return a real value. For ease of exposition, we define a researcher's script as a deterministic function $R : \mathbb{D}^\star \to \mathbb{R}^k \cup \{\perp\}$. All results in this paper can be extended to researcher scripts that are randomized by adding a sampling step to the wrapper. Let $\mathcal{R}$ be the set of all deterministic algorithms.

The notion of differential privacy is rooted in the *indistinguishability* of probability distributions [9].

Definition 2. For any nonempty set Ω, two distributions $d_1, d_2 \in \Delta(\Omega)$ are (ϵ, δ)-*indistinguishable* if for every measurable set $E \subseteq \Omega$, for all $i, j \in \{1, 2\}$, $d_i(E) \leq \exp(\epsilon) d_j(E) + \delta$.

We will denote the (ϵ, δ)-indistinguishability of d_1 and d_2 as $d_1 \sim_{\epsilon,\delta} d_2$. If d_1, d_2, and d_3 are distributions, then the following transitive property holds.

Lemma 1. [12] *If $d_1 \sim_{\epsilon,0} d_2$, and $d_2 \sim_{\epsilon',0} d_3$, then $d_1 \sim_{\epsilon+\epsilon',0} d_3$.*

Differential privacy applies the notion of indistinguishable probability distributions to randomized algorithms on neighboring datasets.

Definition 3. (Differential Privacy [8]) An algorithm $A : \mathbb{D}^\star \to \Delta(\mathbb{R}^k \cup \{\perp\})$ is (ϵ, δ)-*differentially private* if for any neighboring datasets D_1 and D_2, $A(D_1) \sim_{\epsilon,\delta} A(D_2)$.

A *wrapper* is a function $W : \mathbb{D}^\star \times \mathcal{R} \to \Delta(\mathbb{R}^k \cup \{\perp\})$ that takes a dataset D and an algorithm R as input and outputs a noisy response based on R and D. We extend the notion of differential privacy to wrappers as follows.

Definition 4. (Imposition of Differential Privacy) A wrapper W *imposes* $(\epsilon, \delta)-$*differential privacy* if for any researcher algorithm R, the function $W(\cdot, R)$ is $(\epsilon, \delta)-$differentially private.

[3]Technically speaking, $\Delta(\Omega)$ is the set of all probability distributions over some σ-algebra of Ω.

Our privacy wrapper will leverage the k-dimensional Laplace distribution.

Definition 5. The k-dimensional Laplace distribution with mean $\mu \in \mathbb{R}^k$ and scale $\lambda > 0$, denoted as $L^k(\mu, \lambda)$, is the distribution in $\Delta(\mathbb{R}^k)$ with density function $f(z) = (2\lambda)^{-k} \exp(-||z - \mu||_1/\lambda)$.

A necessary and sufficient condition to determine whether two Laplace distributions with the same scale λ are $(\alpha, 0)$-indistinguishable is given below.

Lemma 2. *For $\lambda > 0$, $||\mu - \nu||_1 \leq \alpha\lambda \iff L^k(\mu, \lambda) \sim_{\alpha,0} L^k(\nu, \lambda)$.*

4 ALGORITHM DESCRIPTION: TAHOE

As noted in Section 1.2, our privacy wrapper TAHOE leverages *stable subsets* of *random sizes* to impose (ϵ, δ)-differential privacy. In this section, we formalize each of these notions and specify our algorithm.

4.1 Stable Subsets

Before defining stability, a preliminary concept is that of responsive datasets. Essentially, this requires that a researcher script does not return $\perp$ when applied to the dataset or any subset of a minimum size.

Definition 6. (Responsive Subset) A subset $S \subseteq \mathbb{D}^\star$ is called (R, l)-*responsive* if for every $X \subseteq S$ of size at least l, we have $R(X) \in \mathbb{R}^k$.

Note that any multiset S of size less than l is trivially responsive by definition. Our definition of *stable subsets* refines responsiveness to convey the intuition that removing further individuals does not greatly affect script output.

Definition 7. (Stability: Formal Version of Informal Definition 1) A multiset $S \in \mathbb{D}^\star$ is called $(\alpha, \lambda, l, R)-$*stable* if S is (R, l)-responsive, and for any subsets $X, Y \subseteq S$ of size at least l, $L(R(X), \lambda) \sim_{\alpha,0} L(R(Y), \lambda)$.[4]

Because α, λ, l, and R will not change in this study, we will omit them and simply refer to the set S as *stable*. Additionally, any multiset S of size less than l is trivially stable by definition.

It will be useful to rewrite stability in terms of the following definitions. Let U be the set of all vectors of the form $(u_1, u_2, ..., u_k)$ where $u_j \in \{-1, +1\}$ for all j. For $u \in U$ and $S \in \mathbb{D}^\star$ of size at least l, define $min_u(S)$ and $max_u(S)$ as follows:

$$min_u(S) = \begin{cases} \min_{A \subseteq S, |A| \geq l} u \cdot R(A) & \text{if } S \text{ is } (R, l)\text{-responsive} \\ -\infty & \text{otherwise} \end{cases}$$

and

$$max_u(S) = \begin{cases} \max_{A \subseteq S, |A| \geq l} u \cdot R(A) & \text{if } S \text{ is } (R, l)\text{-responsive} \\ \infty & \text{otherwise} \end{cases}$$

Now stability can be expressed as follows:

Proposition 1. *A multiset $S \in \mathbb{D}^\star$ is $(\alpha, \lambda, l, R)-$stable if and only if either $|S| < l$, or $|S| \geq l$ and*

$$\max_{u \in U} \left(max_u(S) - min_u(S) \right) \leq \alpha\lambda$$

[4]By Lemma 2, $L(R(X), \lambda) \sim_{\alpha,0} (R(Y), \lambda) \iff ||R(X) - R(Y)||_1 \leq \alpha\lambda$.

The proof of this proposition can be found in the appendix. From a computational perspective, min_u and max_u satisfy a recurrence relationship which will enable us to check the stability of subsets more efficiently.

Proposition 2. *For $S \in D^\star$ of size at least l, the following hold:*

(a) If $|S| = l$ and $R(S) \neq \perp$, then $min_u(S) = max_u(S) = u \cdot R(S)$ for all $u \in U$. Alternatively, if $|S| = l$ and $R(S) = \perp$ then $min_u(S) = -\infty$ and $max_u(S) = \infty$.

(b) If $|S| > l$ and $R(S) \neq \perp$, letting $C(S)$ be the set of subsets of S of size $|S| - 1$, then for all $u \in U$,

$$min_u(S) = \min\left(u \cdot R(S), \min_{c \in C(S)} \left(min_u(c)\right)\right)$$

and

$$max_u(S) = \max\left(u \cdot R(S), \max_{c \in C(S)} \left(max_u(c)\right)\right)$$

Alternatively, if $|S| > l$ and $R(S) = \perp$, then $min_u(S) = -\infty$ and $max_u(S) = \infty$.

Stable subsets obey certain algebraic properties that facilitate algorithmic analysis. As the following lemma shows, subsetting a stable subsets creates another stable subset.

Lemma 3. *If S is $(\alpha, \lambda, l, R)-$stable, and $S' \subseteq S$, then S' is also $(\alpha, \lambda, l, R)-$stable.*

Proof. Consider $S' \subseteq S$. If $|S'| < l$, then S' is trivially stable. Hence, consider the case when $|S'| \geq l$. For any $X \subseteq S'$ of size at least l, $X \subseteq S$, so $R(X) \neq \perp$ since S is (R, l)-responsive. Hence, S' is (R, l)-responsive. Take any $X, Y \subseteq S'$ of size at least l. Since $S' \subseteq S$, the $(\alpha, \lambda, l, R)-$stability of S implies $L(R(X), \lambda) \sim_{\alpha,0} L(R(Y), \lambda)$. Hence, S' is also $(\alpha, \lambda, l, R)-$stable. □

Corollary 1. *If S and $\hat{S}$ are $(\alpha, \lambda, l, R)-$stable, then $S \cap \hat{S}$ is also $(\alpha, \lambda, l, R)-$stable.*

Proof. Apply Lemma 3 with S as is and $S' = S \cap \hat{S}$. □

Corollary 2. *If $S \subseteq D$ is $(\alpha, \lambda, l, R)-$stable, and D and D' are neighbors, then $S \cap D'$ is also $(\alpha, \lambda, l, R)-$stable.*

Proof. Apply Lemma 3 with S as is and $S' = S \cap D'$. □

Given data $D \in \mathbb{D}^\star$ and researcher script R, let $m(D, R)$ be the size of the largest stable subset of D. The following lemma relates m to the idea of neighboring datasets.

Lemma 4. *When $D_1, D_2 \in \mathbb{D}^\star$ are neighbors, $m(D_1, R)$ and $m(D_2, R)$ differ by at most 1.*

Proof. Suppose S_1 is a maximal stable subset of D_1. Then $S_1 \cap D_2$ is a stable subset of D_2 by Corollary 2. Further, $|S_1 \cap D_2| \geq |S_1| - 1 = m(D_1, R) - 1$. Therefore $m(D_2, R) \geq m(D_1, R) - 1$. By symmetric argument, $m(D_1, R) \geq m(D_2, R) - 1$. □

4.2 Randomized Subset Sizes

Our algorithm randomly chooses how many individuals to exclude when searching for a stable subset, making it harder for an adversary to leverage the size of stable subsets to extract information from the data. Throughout this paper, we will use N to represent the size of a dataset, and M to represent the maximum number of entries that a privacy wrapper may exclude. In our algorithm, M will be set to the output of the following helper function

$$h(\epsilon, \delta, \alpha) = \lceil Q^{-1} \ln(\delta^{-1} \exp(\epsilon) Q + 1) \rceil$$

where $Q = \epsilon(\epsilon - 4\alpha)(2\epsilon - 4\alpha)^{-1}$. Note that M decreases in both ϵ and δ.

To select a size between $N - M$ and N, we define the distribution $G(\epsilon, \alpha, N, M)$ as follows: $G(\epsilon, \alpha, N, M)(n)$ equals

$$\delta' \exp\left(\min\left\{(\epsilon - 4\alpha)(n - N + M) - 2\alpha, \epsilon(N - n)\right\}\right)$$

for $n \in \{N - M, ..., N\}$ and 0 otherwise, where δ' is a constant selected so that the total probability sums to 1. As we will see in the following section, the normalization constant δ' of G is closely related to the privacy parameter δ in the (ϵ, δ)-differential privacy guarantee of TAHOE. We will normally omit the arguments to G for readability purposes. A specific example of the shape of G when $N = 100, M = 42, \epsilon = 0.1$, and $\alpha = 0.01$ is provided in Figure 1. For these values, $\delta' \approx 0.0098$.

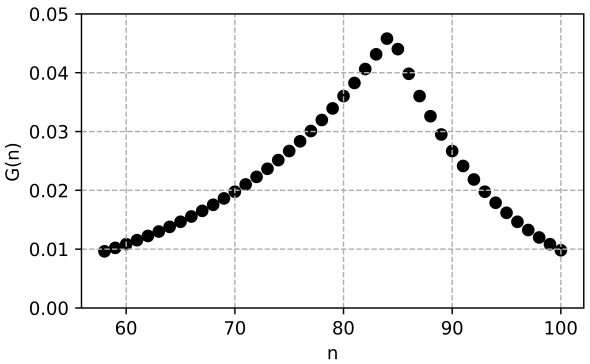


Figure 1: The distribution $G(n)$ when $N = 100, M = 42, \epsilon = 0.1$, and $\alpha = 0.01$. For these values, $\delta' \approx 0.0098$.

G is designed to rise and fall in an exponential manner. In the privacy analysis of Section 5, it will be helpful to relate the probability at n to the total probability below n and the total probability above n. This is done by the following two propositions.

Proposition 3. *For any r in the support of G,*

$$G(r) \leq (\exp(\epsilon - 4\alpha) - 1) \sum_{n<r} G(n) + \exp(-2\alpha)\delta'$$

Proposition 4. *For any r in the support of G,*

$$G(r) \leq (\exp(\epsilon) - 1) \sum_{n>r} G(n) + \delta'$$

A proof of Proposition 3 is provided in the Appendix; the proof of Proposition 4 is similar to the previous one, so we omit it.

Algorithm 2: TAHOE

Input:
(From Data Holder)

- Dataset D with size N
- Privacy parameters $\epsilon > 0$ and $0 < \delta \leq 1$
- Distinguishability parameter $\alpha < \epsilon/4$

such that $h(\epsilon, \delta, \alpha) < (N-1)/2$.
(From Researcher)

- Researcher script R
- Scale parameter λ

Output: $\omega \in \mathbb{R}^k \cup \{\perp\}$

1 (*Choose subset size*) Set $M = h(\epsilon, \delta, \alpha)$ and sample $n \sim G(\epsilon, \alpha, N, M)$.
2 (*Compute min_u and max_u*)
(a) For all subsets S of size $N - 2M - 1$, execute the researcher script R on S. Compute $min_u(S) = max_u(S)$ for all $u \in U$ according to the base case in Proposition 2(a).
(b) Set $i = N - 2M$.
(c) While $i \leq n$
(I) For each subset S of size i, execute the researcher script R on S. Compute $min_u(S)$ and $max_u(S)$ for all $u \in U$ according to Proposition 2(b).
(II) Set $i = i + 1$.
3 (*Check stability*) Initialize $\mathcal{S} = \emptyset$. For each subset S of size n, check whether S is stable according to Proposition 1. If S is stable, add it to set $\mathcal{S}$.
4 (*Non-response condition*) If $\mathcal{S} = \emptyset$, set $\omega = \perp$.
5 (*Execute condition*) Otherwise, select a stable subset S from $\mathcal{S}$ uniformly at random, and sample $\omega \sim L(R(S), \lambda)$.
6 **Return** ω

4.3 Our General Algorithm

Our specific privacy wrapper TAHOE (**T**rim **A**nd with**H**old **O**r **E**xecute) described in Algorithm 2 takes as input a researcher script $R : \mathbb{D}^\star \to \Delta(\mathbb{R}^k \cup \{\perp\})$ and a dataset $D \in \mathbb{D}^\star$ of size N. Additionally, TAHOE uses the following auxiliary parameters: $\epsilon > 0$, $\delta > 0$, $\alpha < \epsilon/4$, and $\lambda > 0$.

Configuring TAHOE is complicated by the large number of parameters, and by the different ways one might want to assess its behavior. In general, a multi-way trade-off exists among how much noise TAHOE adds, how likely TAHOE is to return $\perp$, the privacy parameters, and how much sampling variation is introduced by subsetting. The relationship among these quantities depends on what is in the researcher script, and therefore cannot be known precisely. In the following table, we provide some guidance by discussing the effects of each parameter in general terms.

Table 1: Description of Parameters

$\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$: These are the differential privacy parameters, which are chosen by the data holder. Decreasing ϵ and δ corresponds to a stronger privacy guarantee. However, setting these parameters too small presents two disadvantages. One disadvantage is that a smaller ϵ and δ results in a higher M, requiring the algorithm to consider smaller subsets of data. At least in typical cases, we expect that increasing M will make it harder to find stable subsets, resulting in a higher probability of returning $\perp$. Moreover, the requirement $M = h(\epsilon, \delta, \alpha) < (N-1)/2$ places a lower bound on ϵ and δ. A second disadvantage is that a small ϵ also requires α to be smaller, which may cause TAHOE to return $\perp$ with high probability, or require more noise.
$\boldsymbol{\lambda}$: This parameter is chosen by the researcher and represents the scale of added Laplace noise. Decreasing λ results in more accurate output; however, this also makes the definition of stable subsets stricter, which increases the probability that TAHOE halts without returning an output. We propose two heuristics for setting λ. One is to decide how much noise can be added to the result while preserving its utility. In our experiments, we found that the greatest source of error was the Laplace noise added in Step 5 of the algorithm. Further, the standard deviation of this noise is given by $\sqrt{2}\lambda$. The other heuristic we propose is to compute a value for λ such that TAHOE is guaranteed to never (or rarely) to output $\perp$. We give examples of such computations in Section 7.
$\boldsymbol{\alpha}$: This parameter is chosen by the data holder, and controls how indistinguishable $L(R(A), \lambda)$ and $L(R(B), \lambda)$ must be when A and B are subsets of a stable set. If α is too small, TAHOE will tend to return $\perp$ with high probability. If α is too close to $\epsilon/4$, M will become large, increasing the probability of returning $\perp$. Additionally, setting α too high will violate the requirement that $M = h(\epsilon, \delta, \alpha) < (N-1)/2$. In our simulation studies, we have found that $\alpha = \epsilon/5$ is a reasonable rule of thumb for all scenarios we considered.

5 PRIVACY ANALYSIS OF TAHOE

In this section, we show that our privacy wrapper TAHOE imposes (ϵ, δ)-differential privacy.

Throughout this section, we will use $\mathcal{T}_D \in \Delta(\mathbb{R}^k \cup \{\perp\})$ to denote the probability distribution induced by TAHOE on dataset D, researcher script R, and parameters ϵ, δ, α, and λ. Since D is the only input that will change in the upcoming proofs, we omit writing the other inputs for readability purposes. For any $D \in \mathbb{D}^\star$, write $\mathcal{T}_D(\cdot|n)$ to represents the conditional probability distribution of the wrapper when n has been chosen in Step 1 of Algorithm 2.

Theorem 1. *TAHOE imposes (ϵ, δ)-differential privacy.*

Before providing the proof, we sketch an outline of our argument. In order to show that TAHOE imposes (ϵ, δ) differential privacy, we first show in Lemma 5 that the value of M set in TAHOE produces $\delta' < \delta$. Since (ϵ, δ') differential privacy implies (ϵ, δ)-differential privacy, it is sufficient to show that TAHOE imposes (ϵ, δ')-differential privacy.

Next, we proceed by considering the behavior of TAHOE on two neighboring datasets D_1 and D_2. If there are no stable subsets of size $N - M$ for both of these datasets, then TAHOE returns $\perp$ with probability 1 on both datasets. Hence, the behavior of TAHOE is constant for all neighboring databases, so TAHOE trivially imposes (ϵ, δ')-differential privacy.

On the other hand, if there is a stable subset of size $N - M$ of at least one of the datasets, we can pick any stable subset K and define a reference distribution $H = L(R(K), \lambda)$. For any value of n for which TAHOE does not return $\perp$, we then show in Lemma 6

that both conditional distributions $T_{D_1}(\cdot|n)$ and $T_{D_2}(\cdot|n)$ are $(2\alpha, 0)$-indistinguishable from H. Using this result, we finish the proof by showing that, for any measurable set $E \subseteq \mathbb{R}^k \cup \{\perp\}$, $T_{D_1}(E) \leq \exp(\epsilon)T_{D_2}(E) + \delta'$ based on two cases: when $\perp \notin E$ and when $\perp \in E$.

Proof. We will show that TAHOE imposes (ϵ, δ')-differential privacy, where δ' is the normalization constant in the definition of G. This is sufficient because of the following lemma, which is proven in the Appendix.

Lemma 5. *Given the value of M chosen by TAHOE, $\delta' < \delta$.*

Let $E \subseteq \mathbb{R}^k \cup \{\perp\}$ be measurable, and let D_1 and D_2 be neighboring datasets. We will prove the bound,

$$\frac{\mathcal{T}_{D_1}(E) - \delta'}{\mathcal{T}_{D_2}(E)} \leq \exp(\epsilon)$$

If there are no stable subsets of size $N - M$ of both D_1 and D_2, then $\mathcal{T}_{D_1}$ and $\mathcal{T}_{D_2}$ are the same distribution (giving probability 1 to $\perp$), so the bound follows immediately. If there are any stable subsets of size $N - M$ of either D_1 or D_2, choose one and call it K. Define reference distribution $H = L(R(K), \lambda)$.

Lemma 6. *H is $(2\alpha, 0)-$indistinguishable from $\mathcal{T}_{D_i}(\cdot|n)$ for any $i \in \{1, 2\}$ and for any n for which this algorithm does not return $\perp$.*

Proof. Let $\mathcal{S}_i^n$ be the set of stable subsets of size n of D_i. For $i \in \{1, 2\}$, if $\mathcal{T}_{D_i}(\cdot|n)$ does not return $\perp$, then for all measurable sets $E \subseteq \mathbb{R}^k \cup \{\perp\}$,

$$\mathcal{T}_{D_i}(\cdot|n)(E) = \frac{1}{|\mathcal{S}_i^n|} \sum_{S \in \mathcal{S}_i^n} L(R(S), \lambda)(E)$$

So it is sufficient to show that every distribution of the form $L(R(S), \lambda)$ where $S \subseteq D_i$ is a stable subset of size $n \geq N - M$ is 2α-indistinguishable from H.

For stable $S \subseteq D_i$ of size $n \geq N - M$, we first establish that $|S \cap K| \geq N - 2M - 1$. By inclusion-exclusion, $|S \cup K| = |S| + |K| - |S \cap K|$. Since $|S| \geq N - M$, $|K| = N - M$, and $S \cup K \subseteq D_1 \cup D_2$ implies $|S \cup K| \leq |D_1 \cup D_2| = N + 1$, we have $|S \cap K| \geq N - 2M - 1$. Because S and K are stable, and $S \cap K$ is a subset of size at least $N - 2M - 1$, we have $L(R(S), \lambda) \sim_{\alpha,0} L(R(S \cap K), \lambda)$ by Corollary 1. Additionally, since K is stable, and $S \cap K$ is a subset, $L(R(K), \lambda) \sim_{\alpha,0} L(R(S \cap K), \lambda)$. Thus $L(R(S), \lambda) \sim_{2\alpha,0} L(R(K), \lambda)$ by Lemma 1. □

Recall that $m(D_i, R)$ is the size of the largest stable subset in D_i. Let $r = \max(m(D_1, R), m(D_2, R))$. By Lemma 4, we know that $m(D_1, R), m(D_2, R) \in \{r - 1, r\}$. For $i \in \{1, 2\}$, TAHOE returns $\perp$ on data D_i if and only if $n > m(D_i, R)$. This means that TAHOE will return $\perp$ for any $n > r$ and will not return $\perp$ for any $n < r$.

For $i \in \{1, 2\}$ the law of total probability implies

$$\mathcal{T}_{D_i}(E) = \sum_{n<r} \mathcal{T}_{D_i}(E|n)G(n) + \mathcal{T}_{D_i}(E|r)G(r) + \sum_{n>r} \mathcal{T}_{D_i}(E|n)G(n)$$

We consider two cases:

Case $\perp \notin E$: For $n > r$, TAHOE returns $\perp$, so $\mathcal{T}_{D_i}(E|n) = 0$. Hence,

$$\begin{aligned}\mathcal{T}_{D_1}(E) &= \sum_{n<r} \mathcal{T}_{D_1}(E|n)G(n) + \mathcal{T}_{D_1}(E|r)G(r) \\ &\leq \sum_{n<r} \exp(2\alpha)H(E)G(n) + \exp(2\alpha)H(E)G(r)\end{aligned}$$

where the inequality follows from Lemma 6. Additionally,

$$\begin{aligned}\mathcal{T}_{D_2}(E) &= \sum_{n<r} \mathcal{T}_{D_2}(E|n)G(n) + \mathcal{T}_{D_2}(E|r)G(r) \\ &\geq \sum_{n<r} \mathcal{T}_{D_2}(E|n)G(n) \\ &\geq \sum_{n<r} \exp(-2\alpha)H(E)G(n)\end{aligned}$$

where the last inequality results from Lemma 6. Combining these two observations,

$$\begin{aligned}&\frac{\mathcal{T}_{D_1}(E) - \delta'}{\mathcal{T}_{D_2}(E)} \\ &\leq \frac{\sum_{n<r} \exp(2\alpha)H(E)G(n) + \exp(2\alpha)H(E)G(r) - \delta'}{\sum_{n<r} \exp(-2\alpha)H(E)G(n)} \\ &= \exp(4\alpha)\frac{\sum_{n<r} G(n) + G(r) - \exp(-2\alpha)H(E)^{-1}\delta'}{\sum_{n<r} G(n)} \\ &\leq \exp(4\alpha)\frac{\sum_{n<r} G(n) + G(r) - \exp(-2\alpha)\delta'}{\sum_{n<r} G(n)}\end{aligned}$$

where the last inequality follows since $H(E) \leq 1$. Plugging in $G(r)$ from Proposition 3 and simplifying bounds the ratio by $\exp(\epsilon)$.

Case $\perp \in E$: By Lemma 6, $\mathcal{T}_{D_i}(\cdot|n) \sim_{2\alpha,0} H(\cdot)$ when $\mathcal{T}_{D_i}$ doesn't return $\perp$.

When $n < r$, $\mathcal{T}_{D_i}$ doesn't return $\perp$, so $\mathcal{T}_{D_i}(E|n) = \mathcal{T}_{D_i}(E \setminus \{\perp\}|n)$, hence $\exp(-2\alpha)H(E \setminus \{\perp\}) \leq \mathcal{T}_{D_i}(E|n) \leq \exp(2\alpha)H(E \setminus \{\perp\})$. Additionally, $\mathcal{T}_{D_i}(\{\perp\}|n) = 1$ for all $n > r$, so $\mathcal{T}_{D_i}(E|n) = 1$ for all $n > r$. Hence,

$$\begin{aligned}\mathcal{T}_{D_1}(E|n) &= \sum_{n<r} \mathcal{T}_{D_1}(E|n)G(n) + \mathcal{T}_{D_1}(E|r)G(r) + \sum_{n>r} \mathcal{T}_{D_1}(E|n)G(n) \\ &\leq \sum_{n<r} \exp(2\alpha)H(E \setminus \{\perp\})G(n) + G(r) + \sum_{n>r} G(n)\end{aligned}$$

Additionally,

$$\begin{aligned}\mathcal{T}_{D_2}(E|n) &= \sum_{n<r} \mathcal{T}_{D_1}(E|n)G(n) + \mathcal{T}_{D_2}(E|r)G(r) + \sum_{n>r} \mathcal{T}_{D_2}(E|n)G(n) \\ &\geq \sum_{n<r} \exp(-2\alpha)H(E \setminus \{\perp\})G(n) + \mathcal{T}_{D_2}(E|r)G(r) + \sum_{n>r} G(n) \\ &\geq \sum_{n<r} \exp(-2\alpha)H(E \setminus \{\perp\})G(n) + \sum_{n>r} G(n)\end{aligned}$$

where the last inequality follows as $G(r) \geq 0$. Combining these observations,

$$\begin{aligned}&\frac{\mathcal{T}_{D_1}(E) - \delta'}{\mathcal{T}_{D_2}(E)} \\ &\leq \frac{\sum_{n<r} \exp(2\alpha)H(E \setminus \{\perp\})G(n) + G(r) + \sum_{n>r} G(n) - \delta'}{\sum_{n<r} \exp(-2\alpha)H(E \setminus \{\perp\})G(n) + \sum_{n>r} G(n)}\end{aligned}$$

Substituting for $G(r)$ in the numerator from Proposition 4 and rearranging, the right-hand side is upper-bounded by

$$\exp(4\alpha)\frac{\sum_{n<r} H(E \setminus \{\perp\})G(n) + \exp(\epsilon - 2\alpha) \sum_{n>r} G(n)}{\sum_{n<r} H(E \setminus \{\perp\})G(n) + \exp(2\alpha) \sum_{n>r} G(n)}$$

Since $\epsilon > 4\alpha$, the fraction is greater than 1, so we can subtract the first terms from the numerator and denominator and maintain the inequality:

$$\begin{aligned}\frac{\mathcal{T}_{D_1}(E) - \delta'}{\mathcal{T}_{D_2}(E)} &\leq \exp(4\alpha)\exp(\epsilon - 4\alpha)\sum_{n>r} G(n) \\ &= \exp(\epsilon)\sum_{n>r} G(n) \\ &\leq \exp(\epsilon)\end{aligned}$$

□

6 RUNTIME ANALYSIS OF TAHOE

To analyze the execution time of TAHOE, we let τ represent the worst case runtime of R.[5] Without any restrictions, TAHOE takes prohibitively long to execute. The reason is that, in the worst case, Step 2 of the algorithm requires executing the researcher script on every possible subset of data with size ranging from $N - 2M - 1$ to N. The following proposition places a bound on the execution time without any restrictions on $|\mathbb{D}|$. To attain this result, we must specify how the input parameter α changes as the other parameters change. We suppose that the data holder sets α proportional to ϵ, which is consistent with the rule of thumb ($\epsilon = 5\alpha$) we use in our simulations in Section 7.

Proposition 5. *For fixed R and k, and $\epsilon = g\alpha$ for constant $g > 4$, the worst-case runtime of TAHOE is*

$$O\left((\tau + N)(N+1)^{\hat{g}\epsilon^{-1}(1-\ln(\delta))+3}\right)$$

as $N \to \infty$ and $\epsilon, \delta \to 0$, where $\hat{g} = (4g - 8)(g - 4)^{-1}$.

A proof of this proposition can be found in the Appendix. For fixed privacy parameters, ϵ and δ, and fixed τ, the runtime of TAHOE is polynomial in N. However, a best practice first suggested by Dwork, et al. is to maintain $\delta < 1/N$ [11]. This causes the runtime to be superpolynomial in N.

6.1 Runtime Analysis for Finite Alphabets

While the runtime bound of Proposition 5 is impractical for all but small datasets with weak privacy parameters, there is one situation in which TAHOE can be modified to run much faster. In particular, when the set of possible data entries $\mathbb{D}$ is finite of size f, the number of unique subsets of $D \in \mathbb{D}^\star$ that TAHOE must consider is greatly reduced. Intuitively, when $N \geq f$, by the pigeonhole principle some of the rows in the dataset will necessarily hold the same value; removing any one of them results in the same subset as removing any other.

More formally, when $\mathbb{D} = \{\sigma_1, ..., \sigma_f\}$ any dataset $D \in \mathbb{D}^N$ can be written in histogram form as $D = (v_1, ..., v_f)$ where v_j denotes the number times σ_j appears in the dataset and $\sum_{j=1}^{f} v_j = N$ [12]. Any subset $S \subseteq D$ of size at least $N - i$ is represented by a histogram $(v_1 - w_1, v_2 - w_2, ..., v_f - w_f)$, where each $w_j \in \{0, ..., v_j\}$ represents the number of excluded data elements with value σ_j, and the total number of excluded elements is $\sum_{j=1}^{f} w_j = i$. The following lemma places bounds on the number of such histograms, with the proof deferred to the Appendix.

Lemma 7. *When $|\mathbb{D}| = f$, the number of subsets of a dataset $D \in \mathbb{D}^N$*

(a) of size $N - i$ for $i \geq 0$ is upper bounded by $\binom{i+f-1}{f-1}$

(b) of size at least $N - 2M - 1$ is upper bounded by $\binom{2M+1+f}{f}$.

To optimize TAHOE for this setting, we index subsets of D according to their histograms, in order to avoid duplicating computations on equivalent subsets. Thus, in Step 2(a), TAHOE must run R on every histogram of size $N - 2M - 1$. By Lemma 7(a), there are at most $\binom{N-2M-2+f}{f-1}$ such histograms. By similar analysis, in Step 2(c)(I), TAHOE runs R on at most $\binom{i+f-1}{f-1}$ such histograms of size $N - i$.

Overall, when TAHOE samples n in Step 1, in Step 2 TAHOE runs R on every histogram with size in $\{N - 2M - 1, ..., n\}$. In the worst case, $n = N$ in Step 1. By Lemma 7(b), the total number of histograms to check is upper bounded by $\binom{2M+1+f}{f}$. When $|\mathbb{D}| = f$, the runtime of TAHOE is given in the following proposition, with the proof deferred to the appendix.

Proposition 6. *For fixed R, k, and $\mathbb{D}$ with $|\mathbb{D}| = f$, and $\epsilon = g\alpha$ for constant $g > 4$, the worst-case runtime of TAHOE is $O\left(\tau\epsilon^{-f}\ln(\delta^{-1})^f\right)$ as $\epsilon, \delta \to 0$.*

Notice that the dependency on N has disappeared, and a dependency on f has been introduced. At an intuitive level, the histogram representation provides a lower dimensional representation of subsets, reducing the number of subsets that need to be examined. In upcoming Section 7, we run experiments involving data alphabet sizes from 2 to 4 and find that the algorithm can readily run on a single machine.

For finite data alphabets that are too large for TAHOE to run on a single machine, parallel computation can expand the applicability of the algorithm. In particular, the most computationally expensive steps are 2(a) and 2(c)(I). Both of these can be distributed over a parallel cluster.

7 ACCURACY ANALYSIS OF TAHOE

The performance of privacy-protecting systems is often assessed via a statement that relates the privacy level to accuracy, which may be measured in terms of mean squared error (MSE), standard error, or some other metric. For researcher scripts in general, there is no way to know what accuracy TAHOE provides for given parameter values, because that depends crucially on what is inside the script. However, we are able to provide some results related to accuracy for specific scripts, or classes of scripts.

Our first result motivates the use of TAHOE for inferential statistics. When performing inference tasks, it is common to assume that the data entries are drawn independently from a probability distribution [29, 30]. Let $\mathbb{F}$ be a distribution over the set of data values $\mathbb{D}$, described by a parameter $\theta \in \mathbb{R}^k$. Further, let $\boldsymbol{D}^N$ represent a random dataset of N rows drawn independently and identically from $\mathbb{F}$. A useful set of scripts are those that estimate θ *consistently*.

[5] In practice, to prevent side-channel attacks, we believe the data holder must impose a timeout on the researcher script. In case R breaches the timeout on subset S, TAHOE should assign $R(S)$ to be $\perp$. The length of the timeout could, in principle, depend on N and other parameters.

Definition 8. A script $R : \mathbb{D}^\star \to \mathbb{R}^k \cup \{\perp\}$ is *consistent* for parameter $\theta \in \mathbb{R}^k$ if

$$\text{plim}_{N\to\infty} R(\boldsymbol{D}^N) = \theta$$

That is, for any open set[6] $B \subseteq \mathbb{R}^k \cup \{\perp\}$ that contains θ, $\mathbb{P}(R(\boldsymbol{D}^N) \in B) \to 1$ as $N \to \infty$.

When R has bounded global sensitivity, it is possible to configure TAHOE so that it preserves consistency. For this discussion, we allow that parameters ϵ, α, M, λ may be chosen as functions of N. Three conditions are required. First, α and λ must be chosen so that TAHOE never returns $\perp$.[7] Second, the noise scale λ must approach zero as $N \to \infty$. Third, M must be fixed or grow slowly enough that $\lim_{N\to\infty} N - M = \infty$.

Theorem 2. *If R is consistent for parameter θ and TAHOE is configured to never return $\perp$, $\lim_{N\to\infty} \lambda = 0$, and $\lim_{N\to\infty} N - M = \infty$ then the random variable formed by composing TAHOE with R is consistent for parameter θ.*

A proof is given in the appendix. A similar result holds for researcher scripts that implement unbiased estimators of θ.

7.1 Accuracy Simulations

To gain insight into additional accuracy characteristics of our privacy wrapper, we consider a set of example scripts that an (honest) researcher may want to run. To make the runtime practical, we chose stylized analysis tasks involving finite data alphabets. For each script, our goal is to understand how well our privacy wrapper performs, and compare it to a benchmark.

For this exercise, the benchmark that we select is GUPT, which follows the sample-and-aggregate strategy [23]. To summarize the algorithm, GUPT partitions the dataset into $N^{0.4}$ pieces, runs the script on each piece, then releases the average output with Laplace noise. Unlike TAHOE, GUPT has the advantage of using pure differential privacy. As an additional benchmark, we consider the Laplace mechanism with global sensitivity as a white-box privacy preserving algorithm [11]. This provides a measure of the cost arising from treating a script in a black-box fashion.

A comparison of TAHOE and GUPT is complicated by fundamental differences between the algorithms. First, the parameters of each algorithm are different, and setting them requires judgement on the part of the researcher. Second, GUPT guarantees pure differential privacy while TAHOE uses approximate differential privacy. Finally, GUPT always provide an output, while TAHOE sometimes returns $\perp$, making it difficult to compare accuracy on the same scale. These differences between the systems prevent us from comparing them "straight out of the box."

To configure TAHOE and GUPT and make the comparison as fair as possible, we adopt a set of guidelines. First, we follow simple heuristics to set parameter values when possible. This is aligned with our view that a privacy wrapper should not require specialized knowledge on the part of a researcher. Second, we set parameters in a way that makes the privacy wrappers as comparable as possible.

[6]We define a topology over $\mathbb{R}^k \cup \{\perp\}$ using the Alexandroff Extension [7] of the standard topology of $\mathbb{R}^k$ with the singleton $\{\perp\}$.

[7]By Lemma 2, a necessary and sufficient condition to ensure TAHOE does not return $\perp$ is to set α and λ such that $||R(X) - R(Y)||_1 \leq \alpha\lambda$ for all sets S, X, Y such that $X, Y \subseteq S \subseteq D$, where $|S| \in \{N - M, ..., N\}$ and $|X|, |Y| \geq N - M$.

In the case of GUPT, the main parameter we must set is the bounding rectangle for the script output. For the sake of simplicity, we use the most extreme values that are mathematically possible, meaning that the output is never censored. In the case of TAHOE, we must set the noise scale λ. For each script, we compute a value of λ large enough to guarantee that the algorithm never returns $\perp$. This is done so that both systems always output real vectors, and accuracy can be measured using a common metric (standard error).

An advantage of GUPT that we do not capture in our results is that it uses pure differential privacy. We nevertheless use a common ϵ for all systems. We allow TAHOE to have the extra advantage of approximate differential privacy, but require that $\delta < 1/N$ [1, 25].

7.2 Normalized Histogram

The first script we consider is one that returns a normalized histogram representing the data. Given alphabet $\mathbb{D} = (\sigma_1, ..., \sigma_f)$ and dataset $D \in \mathbb{D}^N$, define the (un-normalized) histogram $\mathcal{H}(D) = (v_1, ..., v_f)$, where v_j denote the number of times σ_j appears in D. A script to return a normalized histogram is then specified by $\mathcal{P}(D) = \mathcal{H}(D)/|D|$.

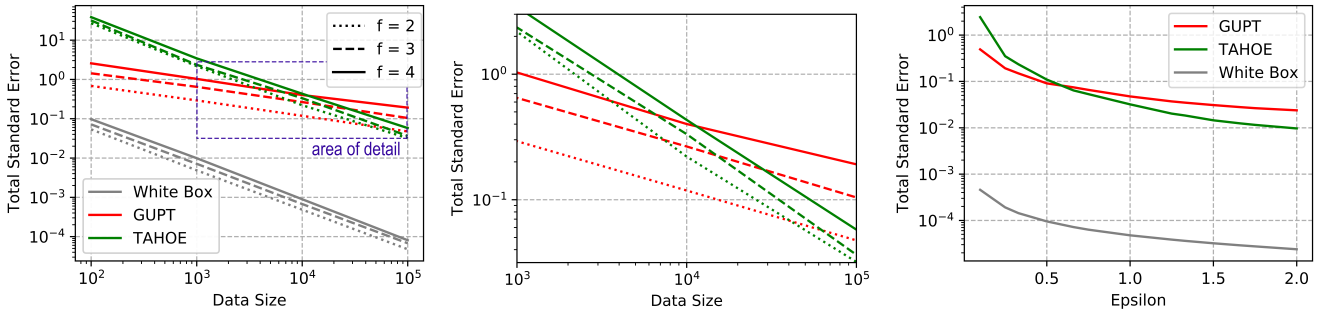
At a high level, the normalized histogram is well-matched with a subsample-and-aggregate strategy, because the proportion of each σ_j on a small subset of data tends to represent the overall proportion of σ_j well [24]. In fact, as long as the subsets have equal size, GUPT introduces no sampling variation, so the only error comes from added Laplace noise. To configure GUPT, we set the bound on script output in each dimension to be $[0, 1]$.

Unlike GUPT, TAHOE introduces both sampling variation and added noise. Several parameters must be configured to run TAHOE. We assume that the end user follows our recommendation of setting $\alpha = \epsilon/5$. We follow the practice of setting $\delta = 1/(N + 1)$.

To guarantee that TAHOE never returns $\perp$, it is necessary and sufficient to make sure it doesn't return $\perp$ when $n = N$ is chosen in Step 1. This is true whenever the entire dataset D is stable. Equivalently, for any subsets $A, B \subseteq D$, with size at least $N - 2M - 1$, we must check that $||\mathcal{P}(A) - \mathcal{P}(B)||_1 \leq \lambda\alpha$. Let vector $a = (a_1, .., a_f) = \mathcal{H}(D) - \mathcal{H}(A)$ represent how many instances of each σ_j were removed from D to form A. Likewise, let $b = (b_1, .., b_f) = \mathcal{H}(D) - \mathcal{H}(B)$. Then $||a||_1 = \sum_{j=1}^f a_j \leq 2M + 1$ and $||b||_1 = \sum_{j=1}^f b_j \leq 2M + 1$. We can then write,

$$\begin{aligned}
||\mathcal{P}(A) - \mathcal{P}(B)||_1 &= ||(\mathcal{P}(D) - \mathcal{P}(B)) - (\mathcal{P}(D) - \mathcal{P}(A))||_1 \\
&= \left\| \frac{b}{|B|} - \frac{a}{|A|} \right\|_1 \\
&\leq \frac{||b||_1}{|B|} + \frac{||a||_1}{|A|} \\
&\leq \frac{2M+1}{|B|} + \frac{2M+1}{|A|} \\
&\leq \frac{2(2M+1)}{N - 2M - 1}
\end{aligned}$$

where the last inequality follows as $|A|, |B| \geq N - 2M - 1$. Thus setting $\lambda = 2(2M + 1)/((N - 2M - 1)\alpha)$ where $M = h(\epsilon, \delta, \alpha)$ is sufficient to guarantee that TAHOE never returns $\perp$.



(a) Total error when $\epsilon = 1$. **(b) View of Figure 2(a)'s area of detail.** **(c) Total error when $f = 2$ and $N = 10^5$.**

Figure 2: The total error, including sampling error and added noise, for the normalized histogram script under different systems in log-scale.

Given these configurations, we perform experiments in which GUPT and TAHOE are each executed on a randomly generated dataset. Each row of data is an independent draw from a discrete uniform distribution. The parameter ϵ is set to values ranging from 0.25 to 2 and N is set to values in $\{10^2, 10^3, 10^4, 10^5\}$. The alphabet size is set to values in $\{2, 3, 4\}$. To estimate error, each experiment was replicated 50 times. All experiments were completed on a single computer with TAHOE and GUPT implemented in Python.

Each panel of Figure 2 displays the total standard error (root-mean-square error) for different parameter values. The error for each histogram is measured using the L1 norm, and includes both sampling error and added noise as a function of dataset size. In practice, we found that sampling variation accounted for less than 1% of the total error in typical simulations.

First, as can be seen in Figure 2 (a) and (b), the error of every system decreases as the data size grows along the horizontal axis. For small datasets, GUPT outperforms TAHOE, featuring an order of magnitude less error in some cases. In this range, our internal parameter M is large relative to N, meaning that TAHOE must consider relatively small subsets. These subsets can have significantly different proportions, so a lot noise is required to make them indistinguishable. By contrast, the number of subsets in GUPT is large as a fraction of N, reducing the sensitivity of the output to any one subset.

For larger datasets, TAHOE's advantages grow. The internal parameter M grows slowly compared to N, meaning that the subsets under consideration have more similar proportions. Beyond a certain dataset size, TAHOE features less total error than GUPT.

Next, as also seen in Figure 2 (a) and (b), increasing the alphabet size f from 2 to 3, and to 4, leads to more error for both privacy wrappers. As the dimensionality increases, TAHOE shows a relative advantage, in the sense that it takes fewer rows of data before TAHOE's error drops below that of GUPT. In the case of TAHOE, the noise scale λ does not depend on f. Since the same amount of Laplace noise is added to each dimension of the histogram, the total added noise increases linearly with f. By contrast, the noise scale used in GUPT is tied to the L1 diameter of the output space, which is proportional to f. Summing over the f histogram dimensions, the total added noise is therefore proportional to f^2.

Finally, as seen in Figure 2(c), decreasing ϵ results in greater error for every system. In the case of GUPT, the curve is approximately hyperbolic, since the added noise scale increases with $1/\epsilon$. On the other hand, TAHOE's total error increases faster as ϵ approaches zero, crossing the curve for GUPT. Examining the expression for TAHOE's noise scale λ, there is an α in the denominator, and α is set proportionally to ϵ, which would suggest a hyperbolic relationship. However, there is an extra dependency on ϵ, since M increases as ϵ decreases, driving λ higher for small values of ϵ.

7.3 Sample Entropy

We now turn our attention to statistics other than the normalized histogram. Before we begin, we note that a histogram script could be used as a building block to compute other statistics. A researcher could first submit the normalized histogram script from the previous section to TAHOE, then compute a statistic of interest directly from the privatized histogram. While this approach is general, there are situations in which a researcher can achieve more accurate results by supplying a script that directly computes the statistic of interest.

In this section, we consider a researcher who supplies a script to compute sample entropy. As a general rule, we have found that for roughly balanced datasets, a researcher can achieve greater accuracy by first computing a histogram using TAHOE, then computing the entropy of that histogram. For unbalanced datasets with low entropy, a researcher can achieve greater accuracy by submitting a script that computes entropy directly.

Again letting f represent the size of the data alphabet $\mathbb{D}$, the sample entropy of database D with normalized histogram $\mathcal{P}(D) = (p_1, ..., p_f)$ is defined as,

$$\mathcal{E}(D) = -\sum_{j=1}^{f} p_j \log_2(p_j)$$

To configure TAHOE, we set $\alpha = \epsilon/5$ and $\delta = 1/(N+1)$ as before. To make the privacy wrappers as comparable as possible, we again set λ so that TAHOE never returns $\perp$. Applying the same argument we used for the proportion script, we must ensure that for any subsets $A, B \subseteq D$, with size at least $N - 2M - 1$, $||\mathcal{E}(A) - \mathcal{E}(B)||_1 \leq \lambda\alpha$. While the left hand side can be bounded mathematically, we

performed a brute force computational search to find the smallest value of λ.

With our systems configured this way, we use simulation to measure the error of each system for randomly generated datasets. Each datapoint is drawn from a discrete uniform distribution over $\mathbb{D}$. We found experimentally that varying the size of the alphabet does not significantly alter the results, so we fix the alphabet size at 3. Parameter ϵ is set to values from $\{1, 2\}$ and N is set to values in $\{10^4, 10^5, 10^6, 10^7\}$. Each experiment was replicated 50 times and all experiments were run on a single computer.

In Figure 3, we plot the total standard error (root-mean-square error) as a function of dataset size, for different parameter values. As in the histogram example, we found that GUPT outperforms TAHOE for small datasets. TAHOE does overtake GUPT's performance for large enough datasets, but the number of datapoints required is considerably greater than we observed for the normalized histogram. Several factors can help explain this result. First, the slope of the entropy function is greatest for unbalanced datasets with entropy near zero. To set the noise scale λ, we must consider subsets containing all of one letter, for which changing $2M + 1$ entries can result in a large change in entropy. TAHOE therefore requires a relatively large amount of noise. Second, entropy is a one-dimensional quantity, so GUPT doesn't suffer a penalty for each dimension, as we saw in the previous experiment. Finally, because the entropy function has slope close to zero for relatively balanced datasets, and subsets are likely to have proportions close to D, GUPT introduces very little sampling variation.

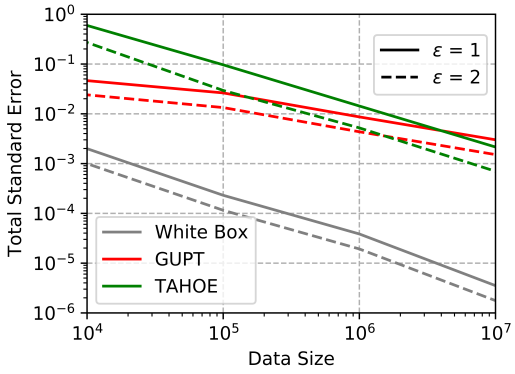


Figure 3: The total error, including sampling error and added noise, for the entropy script under different systems in log-scale.

8 DISCUSSION

This study formalizes the notion of a privacy wrapper: an algorithm that can pass data to a researcher script and observe the return values, before returning an output to the user. We believe that this is a useful framework for reasoning about untrusted code. Differential privacy extends naturally to this setting, with the standard probability bound required to hold for every possible script. Moreover, the script is treated as a black box, avoiding the limitations inherent in analyzing code.

We present our own design for a privacy wrapper, which we call TAHOE. Our algorithm operationalizes two core ideas: find subsets fulfilling a certain stability primitive, and randomize over the subset size. Due to the large number of subsets involved, TAHOE is impractically slow in most scenarios. On the other hand, we consider the special case of a finite data alphabet and describe a set of optimizations that allow TAHOE to run efficiently.

Performance simulations show that TAHOE's performance is comparable to GUPT, a benchmark algorithm from the subsample-and-aggregate lineage, with TAHOE displaying better accuracy for some scenarios and parameter values. TAHOE performs relatively well when the dataset is large, when the dimensionality of the output is high, and when ϵ is not too small.

ACKNOWLEDGMENTS

We are grateful for funding from the UC Berkeley Center for Long-Term Cybersecurity. This work was greatly improved by comments from our anonymous reviewers, Adam Smith, and participants in the workshop, Data Privacy: Foundations and Applications Reunion at the Simons Institute for the Theory of Computing.

REFERENCES

[1] John M Abowd, Robert Ashmead, Ryan Cumings-Menon, Simson Garfinkel, Micah Heineck, Christine Heiss, Robert Johns, Daniel Kifer, Philip Leclerc, Ashwin Machanavajjhala, et al. 2022. The 2020 Census Disclosure Avoidance System TopDown Algorithm. *arXiv preprint arXiv:2204.08986* (2022).
[2] Arini Balakrishnan and Chloe Schulze. 2005. Code obfuscation literature survey. *CS701 Construction of compilers* 19 (2005).
[3] Raef Bassily, Adam Groce, Jonathan Katz, and Adam Smith. 2013. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 439–448.
[4] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.
[5] Rachel Cummings and David Durfee. 2020. Individual sensitivity preprocessing for data privacy. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 528–547.
[6] Damien Desfontaines and Balázs Pejó. 2020. Sok: differential privacies. *Proceedings on privacy enhancing technologies* 2020, 2 (2020), 288–313.
[7] Richard M Dudley. 2018. *Real analysis and probability*. CRC Press.
[8] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.
[9] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 486–503.
[10] Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 371–380.
[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.
[12] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.
[13] Cynthia Dwork and Guy N Rothblum. 2016. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887* (2016).
[14] Madhav Jha and Sofya Raskhodnikova. 2013. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM J. Comput.* 42, 2 (2013), 700–731.
[15] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.
[16] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*.
[17] Daniel Kifer, Solomon Messing, Aaron Roth, Abhradeep Thakurta, and Danfeng Zhang. 2020. Guidelines for implementing and auditing differentially private

systems. *arXiv preprint arXiv:2002.04049* (2020).
[18] Gary King and Nathaniel Persily. 2020. Unprecedented facebook urls dataset now available for academic research through social science one. *Social Science One* 13 (2020).
[19] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1371–1384.
[20] Joshua Alexander Kroll. 2015. *Accountable algorithms.* Ph. D. Dissertation. Princeton University.
[21] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data.* 19–30.
[22] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF).* IEEE, 263–275.
[23] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. 2012. GUPT: privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data.* 349–360.
[24] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing.* 75–84.
[25] Hector Page, Charlie Cabot, and Kobbi Nissim. 2018. Differential privacy an introduction for statistical agencies. *NSQR. Government Statistical Service* (2018), 1–53.
[26] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755* (2016).
[27] Henry Gordon Rice. 1953. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical society* 74, 2 (1953), 358–366.
[28] Kenneth H Rosen. 1999. *Discrete mathematics & applications.* McGraw-Hill.
[29] Adam Smith. 2008. Efficient, differentially private point estimators. *arXiv preprint arXiv:0809.4794* (2008).
[30] Adam Smith. 2011. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the forty-third annual ACM symposium on Theory of computing.* 813–822.
[31] Salil Vadhan. 2017. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography.* Springer, 347–450.
[32] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2019. Differentially private sql with bounded user contribution. *arXiv preprint arXiv:1909.01917* (2019).

APPENDIX

Proof of Proposition 1

Proof. To ease notation, let $2_l^S = \{A \subseteq S : |A| \geq l\}$. We will use the following lemma in the proof:

Lemma 8. *If multiset $S \in \mathbb{D}^\star$ is $(R,l)-$responsive with $|S| \geq l$, then*

$$\max_{X,Y \in 2_l^S} ||R(X) - R(Y)||_1 = \max_{u \in U} \Big(max_u(S) - min_u(S) \Big)$$

Proof. Take any $X, Y \in 2_l^S$, and consider $||R(X)-R(Y)||_1 = \sum_{j=1}^k |R(X)_j - R(Y)_j|$. For all $j \in \{1, ..., k\}$, define $u_j^{(X,Y)}$ as 1 if $R(X)_j \geq R(Y)_j$, and -1 otherwise. Then, $|R(X)_j - R(Y)_j| = u_j^{(X,Y)}(R(X)_j - R(Y)_j)$ for all j. Even more so, for all j we have $u_j^{(X,Y)}(R(X)_j - R(Y)_j) = \max_{u_j \in \{-1,+1\}} u_j(R(X)_j - R(Y)_j)$ by construction. Hence,

$$\begin{aligned}
||R(X) - R(Y)||_1 &= \sum_{j=1}^{k} \max_{u_j \in \{-1,+1\}} u_j(R(X)_j - R(Y)_j) \\
&= \max_{u \in U} \sum_{j=1}^{k} u_j(R(X)_j - R(Y)_j) \\
&= \max_{u \in U} \Big(u \cdot R(X) - u \cdot R(Y) \Big)
\end{aligned}$$

where the second equality follows because maximizing a sum of functions, each with a different independent variable u_j, is equivalent to maximizing each function in the sum individually. It further follows that

$$\begin{aligned}
\max_{X,Y \in 2_l^S} ||R(X) - R(Y)||_1 &= \max_{X,Y \in 2_l^S} \max_{u \in U} \Big(u \cdot R(X) - u \cdot R(Y) \Big) \\
&= \max_{u \in U} \Big(\max_{X,Y \in 2_l^S} \Big(u \cdot R(X) - u \cdot R(Y) \Big) \Big) \\
&= \max_{u \in U} \Big(\max_{A \in 2_l^S} u \cdot R(A) - \min_{A \in 2_l^S} u \cdot R(A) \Big) \\
&= \max_{u \in U} \Big(max_u(S) - min_u(S) \Big)
\end{aligned}$$

where the second to last equality follows because maximizing a difference of two functions with different independent variables is equivalent to maximizing the first function and minimizing the second one, and the last equality follows by applying the definition of $max_u(S)$ and $min_u(S)$. □

Now, we prove both directions of Proposition 1.

($\implies$) Suppose S is stable. If $|S| < l$ the result follows immediately. Otherwise, $|S| \geq l$. Then by the definition of stability, S is responsive and for any $X, Y \in 2_l^S$, $L^k(R(X), \lambda) \sim_{\alpha,0} L^k(R(Y), \lambda)$. By Lemma 2, the last relation can be written as,

$$\max_{X,Y \in 2_l^S} ||R(X) - R(Y)||_1 \leq \alpha\lambda$$

Since S is responsive of minimum size l, by Lemma 8,

$$\max_{X,Y \in 2_l^S} ||R(X) - R(Y)||_1 = \max_{u \in U} \Big(max_u(S) - min_u(S) \Big)$$

Combining these results yields the desired inequality of

$$\max_{u \in U} \Big(max_u(S) - min_u(S) \Big) \leq \alpha\lambda$$

($\impliedby$) Suppose $|S| < l$. Then S is stable by definition. Suppose $|S| \geq l$ and

$$\max_{u \in U} \Big(max_u(S) - min_u(S) \Big) \leq \alpha\lambda$$

We know S is responsive, because otherwise, there must be some $X \in 2_l^S$ with $R(X) = \perp$. But then $min_u(X) = -\infty$ and $max_u(X) = \infty$ for all $u \in U$. This would imply that

$$\max_{u \in U} \Big(max_u(S) - min_u(S) \Big) = \infty > \alpha\lambda$$

giving a contradiction. Finally, by Lemma 8 we have

$$\max_{X,Y \in 2_l^S} ||R(X) - R(Y)||_1 = \max_{u \in U} \Big(max_u(S) - min_u(S) \Big) \leq \alpha\lambda$$

completing the proof.

□

Proof of Proposition 2

Proof. To prove (a), observe that when $|S| = l$, the set S itself is the only subset of size at least l. Hence, when $R(S) \neq \perp$, $min_u(S) = \min_{A\subseteq S,|A|\geq l} u \cdot R(A) = u \cdot R(S) = \max_{A\subseteq S,|A|\geq l} u \cdot R(A) = max_u(S)$. In the case where $R(S) = \perp$, then S is not (R,l)-responsive, hence $min_u(S) = -\infty$ and $max_u(S) = \infty$ by definition.

To prove (b), we first note that for any $A \subseteq S$ with $|A| \geq l$, either $A = S$ or $A \subseteq c$ for some $c \in C(S)$. Hence, we can write

$$\{A \subseteq S : |A| \geq l\} = \{S\} \cup \bigcup_{c\in C(S)} \{A \subseteq c : |A| \geq l\}$$

Then, when $R(S) \neq \perp$,

$$\begin{aligned} min_u(S) &= \min_{A\subseteq S,|A|\geq l} u \cdot R(A) \\ &= \min\left(u \cdot R(S), \min_{c\in C(S)} \min_{A\subseteq c,|A|\geq l} u \cdot R(A)\right) \\ &= \min\left(u \cdot R(S), \min_{c\in C(S)} \left(min_u(c)\right)\right) \end{aligned}$$

In the case where $R(S) = \perp$, then S is not (R,l)-responsive, hence $min_u(S) = -\infty$ by definition. The proof for $max_u(S)$ follows by a similar argument. □

Proof of Proposition 3

Proof. Note the recurrence: for any $N-M \leq n \leq N$, $G(n+1) \leq \exp(\epsilon - 4\alpha)G(n)$. Using this, we see that

$$\begin{aligned} \exp(\epsilon - 4\alpha) \sum_{n=N-M}^{r-1} G(n) &= \sum_{n=N-M}^{r-1} \exp(\epsilon - 4\alpha)G(n) \\ &\geq \sum_{n=N-M}^{r-1} G(n+1) \\ &= \sum_{n=N-M+1}^{r} G(n) \end{aligned}$$

Subtracting $\sum_{N-M}^{r-1} G(n)$ from both sides, we have,

$$\begin{aligned} (\exp(\epsilon - 4\alpha) - 1) \sum_{n=N-M}^{r-1} G(n) &= G(r) - G(N-M) \\ &\geq G(r) - \exp(-2\alpha)\delta' \end{aligned}$$

Rearranging gives the desired result. □

Proof of Lemma 5

Proof. The privacy parameter δ' in the distribution G can be computed precisely, as $(\delta')^{-1}$ equals

$$\sum_{n=N-M}^{N} \exp\left(\min\left((\epsilon - 4\alpha)(n - N + M) - 2\alpha, \epsilon(N-n)\right)\right)$$

By standard analysis arguments, we can lower bound the sum with the following integral.

$$\int_{N-M-1}^{N} \exp\left(\min\left\{(\epsilon - 4\alpha)(n - N + M) - 2\alpha, \epsilon(N-n-1)\right\}\right) dn$$

To simplify computations, we replace 2α with ϵ, which makes the integral smaller. Define $\hat{\delta}^{-1}$ as

$$\int_{N-M-1}^{N} \exp\left(\min\left\{(\epsilon - 4\alpha)(n - N + M) - \epsilon, \epsilon(N-n-1)\right\}\right) dn$$

Then $\hat{\delta}^{-1} < (\delta')^{-1}$. We set the two arguments to the min function equal to one another to find the intersection point $n_\star = N - M(\epsilon - 4\alpha)(2\epsilon - 4\alpha)^{-1}$. Then splitting the integral into two integrals over $(N-M-1, n_\star)$ and $(n_\star, N)$ and evaluating them yields

$$\begin{aligned} \exp(\epsilon)\hat{\delta}^{-1} &= \frac{\exp(MQ)}{Q} - \left(\frac{\exp(-(\epsilon - 4\alpha))}{\epsilon - 4\alpha} - \frac{1}{\epsilon}\right) \\ &> \frac{\exp(MQ)}{Q} - \left(\frac{1}{\epsilon - 4\alpha} - \frac{1}{\epsilon}\right) \\ &= \frac{\exp(MQ) - 1}{Q} \end{aligned}$$

where the first inequality follows from $\epsilon > 4\alpha$. By construction in TAHOE, $M \geq Q^{-1}\ln(\delta^{-1}\exp(\epsilon)Q + 1)$. Hence,

$$\begin{aligned} \hat{\delta} &= \frac{\exp(\epsilon)Q}{\exp(MQ) - 1} \\ &\leq \frac{\exp(\epsilon)Q}{\exp(Q^{-1}\ln(\delta^{-1}\exp(\epsilon)Q + 1)Q) - 1} \\ &= \delta \end{aligned}$$

We have shown $\delta' < \hat{\delta} \leq \delta$, as required. □

Proof of Proposition 5

Proof. To start, we bound the growth rate of M as $\epsilon, \delta \to 0$. When $\epsilon = g\alpha$ for constant $g > 4$, $Q = \epsilon(g-4)(2g-4)^{-1}$. Denote $\hat{g} = (4g-8)(g-4)^{-1}$, so $Q = 2\epsilon/\hat{g}$. Then,

$$\begin{aligned} M &= \left\lceil \frac{\hat{g}}{2\epsilon} \ln\left(\delta^{-1}\exp(\epsilon)\frac{2\hat{\epsilon}}{\hat{g}} + 1\right)\right\rceil \\ &\leq \frac{\hat{g}}{2\epsilon} \ln\left(\delta^{-1}\exp(\epsilon)\frac{2\epsilon}{\hat{g}} + 1\right) + 1 \end{aligned}$$

Because we are interested in the limit as $\epsilon \to 0$, we restrict attention to $\epsilon < e^{-1}$. Then $2\exp(\epsilon)\epsilon/\hat{g} < 2\exp(e^{-1})e^{-1}/\hat{g} = 2\exp(e^{-1}-1)/\hat{g} < 2/\hat{g} < 1$. Since $\ln(\cdot)$ is an increasing function, it follows that

$$M \leq \frac{\hat{g}}{2\epsilon} \ln\left(\delta^{-1} + 1\right) + 1$$

Using standard analysis, $\ln(A+1) \leq \ln(A) + 1$ whenever $A \geq 1$. Since $\delta \leq 1, \delta^{-1} \geq 1$, so

$$M \leq \frac{\hat{g}}{2\epsilon} \ln\left(\delta^{-1}\right) + \frac{\hat{g}}{2\epsilon} + 1 = \frac{\hat{g}}{2\epsilon}\left(1 - \ln(\delta)\right) + 1$$

In the worst-case, TAHOE samples $n = N$ during Step 1. TAHOE is then required to check stability for all subsets of D of sizes between $N - 2M - 1$ and N. The researcher's script R must run on $\sum_{j=0}^{2M+1} \binom{N}{j}$ subsets in Step 2. For each subset S in Step 2, we compute $min_u(S)$ and $max_u(S)$ for each $u \in U$ where $|U| = 2^k$. Computing each of these quantities requires taking a minimum and a maximum

over $C(S)$ from Proposition 2(b), which is bounded by N. Hence, the big-O runtime of Step 2 is $\sum_{j=0}^{2M+1} \binom{N}{j}(\tau + N2^k)$.

Since

$$\sum_{j=0}^{2M+1} \binom{N}{j} \leq (N+1)^{2M+1}$$

and k is fixed by the proposition, the runtime is

$$O\big((\tau + N)(N+1)^{2M+1}\big)$$

In Step 3, TAHOE computes on at most $\binom{N}{N-2M-1}$ subsets and on 2^k vectors $u \in U$, which is less than $O\big((\tau + N)(N+1)^{2M+1}\big)$. It can be checked that Steps 1, 4, and 5 require fewer operations than Step 2, and so they do not alter the big-O. Plugging in the upper bound for M, the runtime is bounded by

$$O\left((\tau + N)(N+1)^{\hat{g}\epsilon^{-1}(1-\ln(\delta))+3}\right)$$

□

Proof of Lemma 7

Proof. We start by proving (a). The number of histograms of size $N-i$ is equal to the number of integer solutions to $\sum_{j=1}^{f} w_j = i$ where each $0 \leq w_j \leq v_j$. This is never exceeds the number of integer solutions to $\sum_{j=1}^{f} w_j = i$ where each $0 \leq w_j$, dropping the upper bound on each w_j. By "stars and bars" [28] there are $\binom{i+f-1}{f-1}$ solutions.

Next, we prove (b). The number of histograms of size at least $N-2M-1$ is equal to the number of integer solutions to $\sum_{j=1}^{f} w_j \leq 2M+1$ where each $0 \leq w_j \leq v_j$. This is never exceeds the number of integer solutions to $\sum_{j=1}^{f} w_j \leq 2M+1$ where each $0 \leq w_j$, dropping the upper bound on each w_j. Introduce a new variable $w_{f+1} \geq 0$. Then the number of non-negative integer solutions to $\sum_{j=1}^{f} w_j \leq 2M+1$ is the same as the number of non-negative integer solutions to $\sum_{j=1}^{f+1} w_j = 2M+1$. By "stars and bars" there are $\binom{2M+1+f}{f}$ solutions. □

Proof of Proposition 6

Proof. By the proof of Proposition 5

$$M \leq \frac{\hat{g}}{2\epsilon}\ln(\delta^{-1}) + \frac{\hat{g}}{2\epsilon} + 1$$

where $\hat{g} = (4g-8)(g-4)^{-1}$. Next, we examine the runtime of TAHOE as a function of M. In the worst-case when $|\mathbb{D}| = f$, TAHOE is required to run R on at most $\binom{2M+1+f}{f}$ subsets during Step 2. For each subset S in Step 2, we compute $min_u(S)$ and $max_u(S)$ for each $u \in U$ where $|U| = 2^k$. Computing each of these quantities requires taking a minimum and a maximum over $C(S)$ from Proposition 2(b), which is bounded by f. Hence, a big-O bound on Step 2 is

$$\begin{aligned}\binom{2M+1+f}{f}(\tau + f2^k) &= \frac{(2M+1+f)\ldots(2M+2)}{f!}(\tau + f2^k) \\ &\leq \frac{(2M+1+f)\ldots(2M+2)}{(f-1)!}(\tau + 2^k) \\ &\leq (2M+1+f)^f(\tau + 2^k)\end{aligned}$$

Since k and f are fixed by the proposition, the bound is $O(\tau M^f)$.

In Step 3, TAHOE computes on at most $\binom{2M+1+f}{f}$ subsets and on 2^k vectors $u \in U$, which is less than $O(\tau M^f)$. It can be checked that Steps 1, 4, and 5 require fewer operations than Step 2, and so they do not alter the big-O. Plugging in the upper bound for M, the runtime is bounded by

$$O\left(\tau\Big(\frac{\hat{g}}{2\epsilon}\ln(\delta^{-1}) + \frac{\hat{g}}{2\epsilon} + 1\Big)^f\right) \subseteq O\big(\tau\epsilon^{-f}\ln(\delta^{-1})^f\big)$$

.

□

Proof of Theorem 2

Proof. Let random variable $\boldsymbol{S}$ be the subset chosen by TAHOE, and let random variable $L \sim L^k(0,\lambda)$ be the Laplace noise added by TAHOE. Since TAHOE is configured to never halt, the output of TAHOE is then the random variable $R(\boldsymbol{S}) + L$.

Let f_n represent the distribution of $R(\boldsymbol{D}^n)$. Since the data is drawn independently from $\mathbb{D}$ with distribution $\mathbb{F}$, conditional on n, S has the distribution $\mathbb{F}^n$, and so $R(\boldsymbol{S})$ has the distribution f_n.

Therefore, the distribution of $R(\boldsymbol{S})$ is $\sum_{n=N-M}^{N} G(n)f_n$. Given an open set $B \subseteq \mathbb{R}^k \cup \{\perp\}$ containing θ, and probability $p < 1$, by consistency of R, there exists m such that for any $n > m$, $f_n(B) > p$. Since $\lim_{N\to\infty} N - M = \infty$, we can choose N such that $N - M > m$, so

$$\sum_{n=N-M}^{N} G(n)f_n(B) > \sum_{n=N-M}^{N} G(n)p = p\sum_{n=N-M}^{N} G(n) = p$$

Therefore $R(\boldsymbol{S})$ converges in probability to θ. Since $\lambda \to 0$ as $N \to \infty$, L converges in probability to 0. And since addition is continuous, by the continuous mapping theorem,

$$\text{plim}_{N\to\infty}\Big(R(\boldsymbol{S}) + L\Big) = \text{plim}_{N\to\infty} R(\boldsymbol{S}) + \text{plim}_{N\to\infty} L = \theta$$

□