Understanding the Sparse Vector Technique for Differential Privacy

Min Lyu [#*], Dong Su [⋆], Ninghui Li [⋆]

[#] *University of Science and Technology of China*
lvmin05@ustc.edu.cn

[⋆] *Purdue University*
{su17, ninghui}@cs.purdue.edu

ABSTRACT

The Sparse Vector Technique (SVT) is a fundamental technique for satisfying differential privacy and has the unique quality that one can output some query answers without apparently paying any privacy cost. SVT has been used in both the interactive setting, where one tries to answer a sequence of queries that are not known ahead of the time, and in the non-interactive setting, where all queries are known. Because of the potential savings on privacy budget, many variants for SVT have been proposed and employed in privacy-preserving data mining and publishing. However, most variants of SVT are actually not private. In this paper, we analyze these errors and identify the misunderstandings that likely contribute to them. We also propose a new version of SVT that provides better utility, and introduce an effective technique to improve the performance of SVT. These enhancements can be applied to improve utility in the interactive setting. Through both analytical and experimental comparisons, we show that, in the non-interactive setting (but not the interactive setting), the SVT technique is unnecessary, as it can be replaced by the Exponential Mechanism (EM) with better accuracy.

1. INTRODUCTION

Differential privacy (DP) is increasingly being considered the privacy notion of choice for privacy-preserving data analysis and publishing in the research literature. In this paper we study the Sparse Vector Technique (SVT), a basic technique for satisfying DP, which was first proposed by Dwork et al. [7] and later refined in [18] and [12], and used in [11, 14, 20, 1, 19]. Compared with other techniques for satisfying DP, SVT has the unique quality that one can output some query answers without apparently paying any privacy cost. More specifically, in SVT one is given a sequence of queries and a certain threshold T, and outputs a vector indicating whether each query answer is above or below T; that is, the output is a vector $\{\perp, \top\}^\ell$, where ℓ is the number of queries answered, $\top$ indicates that the corresponding query answer is above the threshold, and $\perp$ indicates below. SVT works by first perturbing the threshold T and then comparing each perturbed individual query answer against the noisy threshold. When one expects that the predominant majority of queries are on one side, e.g., below the threshold, one can use SVT so that while each output of $\top$ (which we call a **positive outcome**) consumes some privacy budget, each output of $\perp$ (**negative outcome**) consumes none. That is, with a fixed privacy budget and a given level of noise added to each query answer, one can keep answering queries as long as the number of $\top$'s does not exceed a pre-defined cutoff point.

This ability to avoid using any privacy budget for queries with negative outcomes is very powerful for **the interactive setting**, where one answers a sequence of queries without knowing ahead of time what these queries are. Some well-known lower-bound results [3, 5, 6, 10] suggest that "one cannot answer a linear, in the database size, number of queries with small noise while preserving privacy" [7]. This limitation can be bypassed using SVT, as in the iterative construction approach in [11, 12, 18]. In this approach, one maintains a history of past queries and answers. For each new query, one first uses this history to derive an answer for the query, and then uses SVT to check whether the error of this derived answer is below a threshold. If it is, then one can use this derived answer for this new query without consuming any privacy budget. Only when the error of this derived answer is above the threshold would one need to spend the privacy budget accessing the database to answer the query.

With the power of SVT comes the subtlety of why it is private and the difficulty of applying it correctly. The version of SVT used in [11, 12], which was abstracted into a generic technique and described in Roth's 2011 lecture notes [17], turned out to be not differentially private as claimed. This error in [11, 12] is arguably not critical because it is possible to use a fixed version of SVT without affecting the main asymptotic results. Since 2014, several variants of SVT were developed; they were used for frequent itemset mining [14], for feature selection in private classification [20], and for publishing high-dimensional data [1]. These usages are in the **non-interactive setting**, where all the queries are known ahead of the time, and the goal is to find c queries that have large answers, e.g., finding the c most frequent itemsets. Unfortunately, these variants do not satisfy DP, as pointed out in [2]. When using a correct version of SVT in these papers, one would get significantly worse accuracy. Since these papers seek to improve the tradeoff between privacy and utility, the results in them are thus invalid.

The fact that many usages of SVT are not private, even when proofs of their privacy were given, is already known [2, 22]; however, we feel that what led to the erroneous proofs were not clearly explained, and such an explanation can help researchers to avoid similar errors in the future. One piece of evidence of the continuing confusion over SVT appears in [2], the first paper that identifies errors in some SVT variants. In [2], the SVT variants in [14, 20, 1]

*The work was partially done while the author was visiting Purdue University, West Lafayette, IN USA.

Proceedings of the VLDB Endowment, Vol. 10, No. 6

were modeled as a generalized private threshold testing algorithm (GPTT), and a proof showing that GPTT does not satisfy ϵ-DP for any finite ϵ (which we use ∞-DP to denote in this paper) was given. However, as we show in this paper, the proof in [2] was incorrect. This error was not reported in the literature. One goal of this paper is to clearly explain why correct usages of SVT are private, and what are the most likely confusions that caused the myriad of incorrect usages of SVT.

A second goal of this paper is to improve the accuracy of SVT. A version of SVT with a correct privacy proof appeared in Dwork and Roth's 2014 book [8], and was used in some recent work, e.g., [19]. In this paper, we present a version of SVT that adds less noise for the same level of privacy. In addition, we develop a novel technique that optimizes the privacy budget allocation between that for perturbing the threshold and that for perturbing the query answers, and experimentally demonstrates its effectiveness.

A third goal of this paper is to point out that usage of SVT can be replaced by the Exponential Mechanism (EM) [16] when used in the non-interactive setting. Most recent usages of SVT in [1, 14, 19, 20] are in the non-interactive setting, where the goal is to select up to c queries with the highest answers. In this setting, one could also use EM [16] c times to achieve the same objective, each time selecting the query with the highest answer. Using analysis as well as experiments, we demonstrate that EM outperforms SVT.

In summary, this paper has the following novel contributions. First, we propose a new version of SVT that provides better utility. We also introduce an effective technique to improve the performance of SVT. These enhancements achieve better utility than previous SVT algorithms and can be applied to improve utility in the interactive setting. Second, while previous papers have pointed out most of the errors in usages of SVT, we identify the misunderstandings that likely caused the different non-private versions. We also point out a previously unknown error in the proof in [2] of the non-privacy of some SVT variants. Finally, through analysis and experiments on real datasets, we have evaluated the effects of various SVT optimizations and compared them to EM. Our results show that for non-interactive settings, one should use EM instead of SVT.

The rest of the paper is organized as follows. Section 2 gives background information on DP. We analyze six variants of SVT in Section 3. In Section 4, we present our optimizations of SVT. We compare SVT with the Exponential Mechanism in Section 5. The experimental results are shown in Section 6. Related works are summarized in Section 7. Section 8 concludes our work.

2. BACKGROUND

DEFINITION 1 (ϵ-DP [4, 5]). *A randomized mechanism $\mathcal{A}$ satisfies ϵ-differential privacy (ϵ-DP) if for any pair of neighboring datasets D and D', and any $S \in Range(\mathcal{A})$,*

$$\mathsf{Pr}[\mathcal{A}(D) = S] \leq e^{\epsilon} \cdot \mathsf{Pr}\big[\mathcal{A}(D') = S\big] .$$

Typically, two datasets D and D' are considered to be neighbors when they differ by only one tuple. We use $D \simeq D'$ to denote this.

There are several primitives for satisfying ϵ-DP. The Laplacian mechanism [5] adds a random noise sampled from the Laplace distribution with the scale parameter proportional to Δ_f, the *global sensitivity* of the function f. That is, to compute f on a dataset D, one outputs

$$\begin{aligned} \mathcal{A}_f(D) &= f(D) + \mathsf{Lap}\left(\tfrac{\Delta_f}{\epsilon}\right), \\ \text{where} \quad \Delta_f &= \max_{D \simeq D'} |f(D) - f(D')|, \\ \text{and} \quad \mathsf{Pr}[\mathsf{Lap}\,(\beta) = x] &= \tfrac{1}{2\beta} e^{-|x|/\beta}. \end{aligned}$$

In the above, $\mathsf{Lap}\,(\beta)$ denotes a random variable sampled from the Laplace distribution with scale parameter β.

The *Exponential Mechanism* [16] samples the output of the data analysis mechanism according to an exponential distribution. The mechanism relies on a *quality* function $q : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$ that assigns a real valued score to one output $r \in \mathcal{R}$ when the input dataset is D, where higher scores indicate more desirable outputs. Given the quality function q, its global sensitivity Δ_q is defined as:

$$\Delta_q = \max_r \max_{D \simeq D'} |q(D, r) - q(D', r)|.$$

Outputting r using the following distribution satisfies ϵ-DP:

$$\mathsf{Pr}[r \text{ is selected}] \propto \exp\left(\frac{\epsilon}{2\,\Delta_q} q(D, r)\right).$$

In some cases, the changes of all quality values are one-directional. For example, this is the case when the quality function counts the number of tuples that satisfy a certain condition, and two datasets are considered to be neighboring when one is resulted from adding or deleting a tuple from the other. When adding one tuple, all quality values either stay unchanged or increase by one; the situation where one quality increases by 1 and another decreases by 1 cannot occur. In this case, one can make more accurate selection by choosing each possible output with probability proportional to $\exp\left(\frac{\epsilon}{\Delta_q} q(D, r)\right)$, instead of $\exp\left(\frac{\epsilon}{2\,\Delta_q} q(D, r)\right)$.

DP is sequentially composable in the sense that combining multiple mechanisms $\mathcal{A}_1, \cdots, \mathcal{A}_m$ that satisfy DP for $\epsilon_1, \cdots, \epsilon_m$ results in a mechanism that satisfies ϵ-DP for $\epsilon = \sum_i \epsilon_i$. Because of this, we refer to ϵ as the privacy budget of a privacy-preserving data analysis task. When a task involves multiple steps, each step uses a portion of ϵ so that the sum of these portions is no more than ϵ.

3. VARIANTS OF SVT

In this section, we analyze variants of SVT; six of them are listed in Figure 1. Alg. 1 is an instantiation of our proposed SVT. Alg. 2 is the version taken from [8]. Alg. 3, 4, 5, and 6 are taken from [17, 14, 20, 1] respectively.

The table in Figure 2 summarizes the differences among these algorithms. Their privacy properties are given in the last row of the table. Alg. 1 and 2 satisfy ϵ-DP, and the rest of them do not. Alg. 3, 5, 6 do not satisfy ϵ-DP for any finite ϵ, which we denote as ∞-DP.

An important input parameter to any SVT algorithm is the number c, i.e., how many positive outcomes one can answer before stopping. This number can be quite large. For example, in privately finding top-c frequent itemsets [14], c ranges from 50 to 400. To understand the differences between these variants, one can view SVT as having the following four steps:

1. Generate the threshold noise ρ (Line 1 in each algorithm), which will be added to the threshold during the comparison between each query and the threshold (line 5). In all except Alg. 2, ρ scales with Δ/ϵ_1. In Alg. 2, however, ρ scales with $c\Delta/\epsilon_1$. This extra factor of c causes Alg. 2 to be much less accurate than Alg. 1.
2. For each query q_i, generate noise ν_i to be added to the query (Line 4), which should scale with $2c\Delta/\epsilon_2$. In Alg. 4 and 6, ν_i scales with Δ/ϵ_2. Removing the factor of c from the magnitude of the noise will result in better utility; however, this is done at the cost of being non-private. Alg. 5 adds no noise to q_i at all, and is also non-private.
3. Compare the perturbed query answer with the noisy threshold and output whether it is above or below the threshold (Lines 5, 6, 9). Here Alg. 3 differs in that it outputs the noisy

Input/Output shared by all SVT Algorithms

Input: A private database D, a stream of queries $Q = q_1, q_2, \cdots$ each with sensitivity no more than Δ, either a sequence of thresholds $\mathbf{T} = T_1, T_2, \cdots$ or a single threshold T (see footnote *), and c, the maximum number of queries to be answered with $\top$.

Output: A stream of answers $a_1, a_2, \cdots$, where each $a_i \in \{\top, \bot\} \cup \mathbb{R}$ and $\mathbb{R}$ denotes the set of all real numbers.

Algorithm 1 An instantiation of the SVT proposed in this paper.

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots, c$.
$\epsilon_1 = \epsilon/2$, $\rho = \mathsf{Lap}(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}(2c\Delta/\epsilon_2)$
  if $q_i(D) + \nu_i \geq T_i + \rho$ **then**
    Output $a_i = \top$
    count = count + 1, **Abort** if count $\geq c$.
  else
    Output $a_i = \bot$

Algorithm 2 SVT in Dwork and Roth 2014 [8].

Input: D, Q, Δ, T, c.
$\epsilon_1 = \epsilon/2$, $\rho = \mathsf{Lap}(c\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}(2c\Delta/\epsilon_2)$
  if $q_i(D) + \nu_i \geq T + \rho$ **then**
    Output $a_i = \top$, $\rho = \mathsf{Lap}(c\Delta/\epsilon_1)$
    count = count + 1, **Abort** if count $\geq c$.
  else
    Output $a_i = \bot$

Algorithm 3 SVT in Roth's 2011 Lecture Notes [17].

Input: D, Q, Δ, T, c.
$\epsilon_1 = \epsilon/2$, $\rho = \mathsf{Lap}(\Delta/\epsilon_1)$,
$\epsilon_2 = \epsilon - \epsilon_1$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}(c\Delta/\epsilon_2)$
  if $q_i(D) + \nu_i \geq T + \rho$ **then**
    Output $a_i = q_i(D) + \nu_i$
    count = count + 1, **Abort** if count $\geq c$.
  else
    Output $a_i = \bot$

Algorithm 4 SVT in Lee and Clifton 2014 [14].

Input: D, Q, Δ, T, c.
$\epsilon_1 = \epsilon/4$, $\rho = \mathsf{Lap}(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$, count = 0
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}(\Delta/\epsilon_2)$
  if $q_i(D) + \nu_i \geq T + \rho$ **then**
    Output $a_i = \top$
    count = count + 1, **Abort** if count $\geq c$.
  else
    Output $a_i = \bot$

Algorithm 5 SVT in Stoddard et al. 2014 [20].

Input: D, Q, Δ, T.
$\epsilon_1 = \epsilon/2$, $\rho = \mathsf{Lap}(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$
for each query $q_i \in Q$ **do**
  $\nu_i = 0$
  if $q_i(D) + \nu_i \geq T + \rho$ **then**
    Output $a_i = \top$
  else
    Output $a_i = \bot$

Algorithm 6 SVT in Chen et al. 2015 [1].

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots$.
$\epsilon_1 = \epsilon/2$, $\rho = \mathsf{Lap}(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$
for each query $q_i \in Q$ **do**
  $\nu_i = \mathsf{Lap}(\Delta/\epsilon_2)$
  if $q_i(D) + \nu_i \geq T_i + \rho$ **then**
    Output $a_i = \top$
  else
    Output $a_i = \bot$

Figure 1: A selection of SVT variants

	Alg. 1	**Alg. 2**	**Alg. 3**	**Alg. 4**	**Alg. 5**	**Alg. 6**
ϵ_1	$\epsilon/2$	$\epsilon/2$	$\epsilon/2$	$\epsilon/4$	$\epsilon/2$	$\epsilon/2$
Scale of threshold noise ρ	Δ/ϵ_1	$c\Delta/\epsilon_1$	Δ/ϵ_1	Δ/ϵ_1	Δ/ϵ_1	Δ/ϵ_1
Reset ρ after each output of $\top$ **(unnecessary)**		Yes				
Scale of query noise ν_i	$2c\Delta/\epsilon_2$	$2c\Delta/\epsilon_2$	$c\Delta/\epsilon_2$	Δ/ϵ_2	0	Δ/ϵ_2
Outputting $q_i + \nu_i$ instead of $\top$ **(not private)**			Yes			
Outputting unbounded $\top$'s **(not private)**					Yes	Yes
Privacy Property	ϵ-DP	ϵ-DP	∞-DP	$\left(\frac{1+6c}{4}\epsilon\right)$-DP	∞-DP	∞-DP

Figure 2: Differences among Algorithms $1 \sim 6$.

* Algorithms 1 and 6 use a sequence of thresholds $\mathbf{T} = T_1, T_2, \cdots$, allowing different thresholds for different queries. The other algorithms use the same threshold T for all queries. We point out that this difference is mostly syntactical. In fact, having an SVT where the threshold always equals 0 suffices. Given a sequence of queries $q_1, q_2, \cdots$, and a sequence of thresholds $\mathbf{T} = T_1, T_2, \cdots$, we can define a new sequence of queries $r_i = q_i - T_i$, and apply the SVT to r_i using 0 as the threshold to obtain the same result. In this paper, we decide to use thresholds to be consistent with the existing papers.

query answer $q_i(D) + \nu_i$, instead of an indicator $\top$. This makes it non-private.

4. Keep track of the number of $\top$'s in the output, and stop when one has outputted c $\top$'s (Line 7). This step is missed in Alg. 5 and 6. Without this limitation, one is answering a potentially unbounded number of queries with a fixed accuracy level for each query. This is not private.

3.1 Privacy Proof for Alg. 1

Before proving that Alg. 1 is private, we first give a "fake proof" of privacy for Alg. 6 to illustrate both the power of SVT and the misunderstandings behind Alg. 4, 5, and 6. While this "proof" is incorrect, the correct portion of it is used in the proof for Alg. 1.

For any output vector $\boldsymbol{a} = \{\top, \bot\}^\ell$, let a_i denote the i-th component of $\boldsymbol{a}$, $\mathbf{I}_\top = \{i : a_i = \top\}$, and $\mathbf{I}_\bot = \{i : a_i = \bot\}$. We have the probability of outputting $\boldsymbol{a}$ over D as follows:

$$\begin{aligned}\Pr[\mathcal{A}(D) = \boldsymbol{a}] = &\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\top} \Pr[q_i(D)+\nu_i \geq T_i + z] \\ &\prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D)+\nu_i < T_i + z]\ dz \qquad (1)\\ = &\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\top} \Pr[q_i(D)+\nu_i \geq T_i + z]\ dz \\ &\times \int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D)+\nu_i < T_i + z]\ dz \qquad (2)\end{aligned}$$

That is, to compute the probability $\Pr[\mathcal{A}(D) = \boldsymbol{a}]$, we integrate over all possible values for ρ, the noise added to the threshold. Conditioned on ρ taking a particular value z, the probability that $\boldsymbol{a}$ is the output is the product of the probabilities that per-query noise ν_i takes an appropriate value to cause the corresponding query q_i to result in the output ($\top$ or $\bot$) indicated by $\boldsymbol{a}$.

Of course, the step from (1) to (2), which breaks one integration into two, is incorrect. This is the main mistake made in the thinking behind Alg. 4, 5, and 6. However, accepting (2), we can then prove that Alg. 6 is private. For any neighboring D, D', we have:

$$\begin{aligned}\frac{\Pr[\mathcal{A}(D) = \boldsymbol{a}]}{\Pr[\mathcal{A}(D') = \boldsymbol{a}]} = &\frac{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\top} \Pr[q_i(D)+\nu_i \geq T_i + z]\ dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\top} \Pr[q_i(D')+\nu_i \geq T_i + z]\ dz} \qquad (3)\\ &\times \frac{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D)+\nu_i < T_i + z]\ dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D')+\nu_i < T_i + z]\ dz} \qquad (4)\end{aligned}$$

We now prove that (4) is bounded by $e^{\frac{\epsilon}{2}}$, and the same logic applies to (3). This then completes the "proof".

$$\begin{aligned}&\frac{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D)+\nu_i < T_i + z]\ dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D')+\nu_i < T_i + z]\ dz} \qquad (5)\\ =&\frac{\int_{-\infty}^{\infty} \Pr[\rho = z-\Delta] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D)+\nu_i < T_i + z - \Delta]\ dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D')+\nu_i < T_i + z]\ dz} \qquad (6)\\ \leq&\frac{\int_{-\infty}^{\infty} e^{\frac{\epsilon}{2}}\Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D)+\nu_i < T_i + z - \Delta]\ dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D')+\nu_i < T_i + z]\ dz} \qquad (7)\\ \leq&e^{\frac{\epsilon}{2}}\frac{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D')-\Delta+\nu_i < T_i + z - \Delta]\ dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} \Pr[q_i(D')+\nu_i < T_i + z]\ dz} \qquad (8)\\ =&e^{\frac{\epsilon}{2}}\end{aligned}$$

The step from (5) to (6) is by a **change of integration variable** from z to $z - \Delta$. The next step is because of $\Pr[\rho = z-\Delta] \leq e^{\frac{\epsilon}{2}}\Pr[\rho = z]$, due to $\rho = \mathsf{Lap}(2\Delta/\epsilon)$ and the property of the Laplace distribution. The step from (7) to (8) is by replacing $q_i(D)$ with $q_i(D') - \Delta$, which is $\leq q_i(D)$ because the global sensitivity of all queries is Δ. Replacing the left hand of the condition "$q_i(D) + \nu_i < T_i + z - \Delta$" with a smaller term cannot decrease the probability for the condition to hold. After (8), the integration terms on the numerator and denominator cancel out.

The power of SVT comes from the above derivation. By adding a noise to the threshold, one can compare what happens with D' and threshold noise z with what happens under input D and threshold noise $z-\Delta$, and bound the probability ratio for all $\bot$ outputs, no matter how many such outputs there are.

Observe that in the above reasoning **we have not used any property of ν_i. In fact, even if $\nu_i = 0$, as in Alg. 5, the proof will go through**. This underlies Alg. 5's design of setting $\nu_i = 0$.

Also, one can **similarly bound the term in** (3) **by using $\rho = z+\Delta$ to change the integration variable**. This "proves" the privacy of Alg. 6, assuming that the step from (1) to (2) is correct.

However, when one cannot use (2) for $\Pr[\mathcal{A}(D) = \boldsymbol{a}]$, the probabilities for $\top$ outputs and $\bot$ outputs are under one integration; one has to choose **whether to use $\rho = z-\Delta$ or $\rho = z+\Delta$ when changing the integration variable, and cannot do both.** Typically, one chooses the former to bound all $\bot$ outputs, and then rely on adding sufficient noises to each query to bound the $\top$ outputs.

THEOREM 1. *Alg. 1 is ϵ-DP.*

PROOF. Consider any $\boldsymbol{a} \in \{\bot, \top\}^\ell$. Let $\boldsymbol{a} = \langle a_1, \cdots, a_\ell\rangle$, $\mathbf{I}_\top = \{i : a_i = \top\}$, and $\mathbf{I}_\bot = \{i : a_i = \bot\}$. Let

$$f_i(D, z) = \Pr[q_i(D)+\nu_i < T_i + z] \qquad (9)$$
$$g_i(D, z) = \Pr[q_i(D)+\nu_i \geq T_i + z]\,. \qquad (10)$$

We have

$$\begin{aligned}&\frac{\Pr[\mathcal{A}(D) = \boldsymbol{a}]}{\Pr[\mathcal{A}(D') = \boldsymbol{a}]}\\ =&\frac{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D, z) \prod_{i\in\mathbf{I}_\top} g_i(D, z)\, dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D', z) \prod_{i\in\mathbf{I}_\top} g_i(D', z)\, dz} \qquad (11)\\ =&\frac{\int_{-\infty}^{\infty} \Pr[\rho = z-\Delta] \prod_{i\in\mathbf{I}_\bot} f_i(D, z-\Delta) \prod_{i\in\mathbf{I}_\top} g_i(D, z-\Delta)\, dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D', z) \prod_{i\in\mathbf{I}_\top} g_i(D', z)\, dz} \qquad (12)\\ \leq&\frac{\int_{-\infty}^{\infty} e^{\epsilon_1}\Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D', z) \prod_{i\in\mathbf{I}_\top} g_i(D, z-\Delta)\, dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D', z) \prod_{i\in\mathbf{I}_\top} g_i(D', z)\, dz} \qquad (13)\\ \leq&\frac{\int_{-\infty}^{\infty} e^{\epsilon_1}\Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D', z) \prod_{i\in\mathbf{I}_\top} e^{\frac{\epsilon_2}{c}} g_i(D', z)\, dz}{\int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i\in\mathbf{I}_\bot} f_i(D', z) \prod_{i\in\mathbf{I}_\top} g_i(D', z)\, dz} \qquad (14)\\ \leq&e^{\epsilon_1}\left(e^{\frac{\epsilon_2}{c}}\right)^c = e^{\epsilon_1+\epsilon_2} = e^\epsilon\end{aligned}$$

From (11) to (13) uses the same logic as from (5) to (8). The step to (14) is because $g_i(D, z-\Delta) \leq e^{\frac{\epsilon_2}{c}} g_i(D', z)$, proven below. This step uses the fact that noise ν_i added to each query answer. The last step is because $|\mathbf{I}_\top| \leq c$, i.e., there are at most c positive outcomes.

$$\begin{aligned}g_i(D, z-\Delta) &= \Pr[q_i(D)+\nu_i \geq T_i + z - \Delta]\\ &\leq \Pr\big[q_i(D')+\Delta+\nu_i \geq T_i + z - \Delta\big] \qquad (15)\\ &= \Pr\big[q_i(D')+\nu_i \geq T_i + z - 2\Delta\big]\\ &\leq e^{\frac{\epsilon_2}{c}}\Pr\big[q_i(D')+\nu_i \geq T_i + z\big] \qquad (16)\end{aligned}$$

$$= e^{\frac{\epsilon_2}{c}} g_i(D', z).$$

Eq. (15) is because $q_i(D) \leq \Delta + q_i(D')$, and Eq. (16) is because ν_i is sampled from the distribution $\mathsf{Lap}\left(\frac{2c\Delta}{\epsilon_2}\right)$. □

3.2 Privacy Properties of Other Variants

Alg. 2 is taken from the differential privacy book published in 2014 [8]. It satisfies ϵ-DP. It has two differences when compared with Alg. 1. First, ρ follows $\mathsf{Lap}(c\Delta/\epsilon_1)$ instead of $\mathsf{Lap}(\Delta/\epsilon_1)$. This causes Alg. 2 to have significantly worse accuracy than Alg. 1, as we show in Section 6. Second, Alg. 2 refreshes the threshold noise ρ after each output of $\top$. We note that making the threshold noise scale with c is necessary *only if* one refreshes the this noise after each output of $\top$; however, such refreshing is unnecessary.

Alg. 3 is taken from [17], which in turn was abstracted from the algorithms used in [11, 12]. It has two differences from Alg. 1. First, ν_i follows $\mathsf{Lap}(c\Delta/\epsilon_2)$ instead of $\mathsf{Lap}(2c\Delta/\epsilon_1)$; this is not enough for ϵ-DP (even though it suffices for $\frac{3\epsilon}{2}$-DP). Second, it actually outputs the noisy query answer instead of $\top$ for a query above the threshold. This latter fact causes Alg. 3 to be not ϵ'-DP for any finite ϵ'. A proof for this appeared in Appendix A of [22]; for completeness, see Appendix (Section 11.1) for the proof. The error in the proof for Alg. 3's privacy in [17] occurs in the following steps:

$$\begin{aligned}
&\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}] \\
&= \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z] \prod_{i \in \mathbf{I}_\perp} f_i(D, z) \prod_{i \in \mathbf{I}_\top} \mathsf{Pr}[q_i(D) + \nu_i \geq T + z \wedge q_i(D) + \nu_i = a_i]\, dz \\
&= \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z] \prod_{i \in \mathbf{I}_\perp} f_i(D, z) \prod_{i \in \mathbf{I}_\top} \mathsf{Pr}[q_i(D) + \nu_i = a_i]\, dz \qquad (17)
\end{aligned}$$

The error occurs when going to (17), which is implicitly done in [17]. This step removes the condition $q_i(D) + \nu_i \geq T + z$.

Outputting the positive query answers reveals information about the noisy threshold, since the noisy threshold must be $\leq$ all the outputted query answers. Once information about the noisy threshold is leaked, the ability to answer each negative query "for free" disappears. Mathematically, the integration is no longer $\int_{-\infty}^{\infty}$, but $\int_{-\infty}^{m}$, where m is the smallest outputted query answer; and any change in the integration variable also changes the integration boundary, making it impossible to cancel out matching terms.

Alg. 4, taken from [14], differs from Alg. 1 in the following ways. First, it sets ϵ_1 to be $\epsilon/4$ instead of $\epsilon/2$. This has no impact on the privacy. Second, ν_i does not scale with c. As a result, Alg. 4 is only $\left(\frac{1+6c}{4}\right)\epsilon$-DP in general. In [14], Alg. 4 is applied for finding frequent itemsets, where the queries are counting queries and are monotonic. Because of this monotonicity, the usage of Alg. 4 here is $\left(\frac{1+3c}{4}\right)\epsilon$-DP. Theorem 2 can be applied to Alg. 4 to establish this privacy property; we thus omit the proof of this.

Alg. 5 and Alg. 6 do not satisfy ϵ-DP for any finite ϵ. A proof for Alg. 6 is given in Appendix B of [22]. For completeness, we include it in Appendix 11.2. While this proof also applies to Alg. 5, here we give a simpler proof, using a counterexample with $T = 0$, $\Delta = 1$, $\mathbf{q} = \langle q_1, q_2 \rangle$ such that $\mathbf{q}(D) = \langle 0, 1 \rangle$ and $\mathbf{q}(D') = \langle 1, 0 \rangle$, and $\boldsymbol{a} = \langle \perp, \top \rangle$. From Eq. (1), we have

$$\begin{aligned}
\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}] &= \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z]\, \mathsf{Pr}[0 < z]\, \mathsf{Pr}[1 \geq z]\ dz \\
&= \int_{0}^{1} \mathsf{Pr}[\rho = z]\ dz > 0, \\
\mathsf{Pr}[\mathcal{A}(D') = \boldsymbol{a}] &= \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z']\, \mathsf{Pr}[1 < z']\, \mathsf{Pr}[0 \geq z']\ dz',
\end{aligned}$$

which is zero. So the probability ratio $\frac{\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}]}{\mathsf{Pr}[\mathcal{A}(D') = \boldsymbol{a}]} = \infty$.

Other Variants. Some usages of SVT aim at satisfying (ϵ, δ)-DP [5], instead of ϵ-DP. These often exploit the advanced composition theorem for DP [9], which states that applying k instances of ϵ-DP algorithms satisfies (ϵ', δ')-DP, where $\epsilon' = \sqrt{2k \ln(1/\delta')}\epsilon + k\epsilon(e^\epsilon - 1)$. In this paper, we limit our attention to SVT variants to those satisfying ϵ-DP, which are what have been used in the data mining community [1, 14, 19, 20].

The SVT used in [12, 18] has another difference from Alg. 3. In [12, 18], the goal of using SVT is to determine whether the error of using an answer derived from past queries/answers is below a threshold T. This check takes the form of "$\mathbf{if}\ |\tilde{q}_i - q_i(D) + \nu_i| \geq T + \rho\ \mathbf{then}$ output i," where $\tilde{q}_i$ gives the estimated answer of a query obtained using past queries/answers, and $q_i(D)$ gives the true answer. This is incorrect because the noise ν_i should be outside the absolute value sign. In the usage in [12, 18], the left hand of the comparison is always ≥ 0; thus whenever the output includes at least one $\top$, one immediately knows that the threshold noise $\rho \geq -T$. This leakage of ρ is somewhat similar to Alg. 3's leakage caused by outputting noisy query answers that are found to be above the noisy threshold. This problem can be fixed by using "$\mathbf{if}\ |\tilde{q}_i - q_i(D)| + \nu_i \geq T + \rho\ \mathbf{then}$ output i" instead. By viewing $r_i = |\tilde{q}_i - q_i(D)|$ as the query to be answered, this becomes a standard application of SVT.

3.3 Error in Privacy Analysis of GPTT

In [2], the SVT variants in [14, 20, 1] were modeled as a generalized private threshold testing algorithm (GPTT). In GPTT, the threshold T is perturbed using $\rho = \mathsf{Lap}(\Delta/\epsilon_1)$ and each query answer is perturbed using $\mathsf{Lap}(\Delta/\epsilon_2)$ and there is no cutoff; thus GPTT can be viewed as a generalization of Algorithm 6. When setting $\epsilon_1 = \epsilon_2 = \frac{\epsilon}{2}$, GPTT becomes Alg. 6.

There is a constructive proof in [2] to show that GPTT is not ϵ'-DP for any finite ϵ'. However, this proof is incorrect. This error is quite subtle. We discovered the error only after observing that the technique of the proof can be applied to show that Alg. 1 (which we have proved to be private) to be non-private. The detailed discussion of this error is quite technical, and is included in Appendix 11.3.

4. OPTIMIZING SVT

Alg. 1 can be viewed as allocating half of the privacy budget for perturbing the threshold and half for perturbing the query answers. This allocation is somewhat arbitrary, and other allocations are possible. Indeed, Alg. 4 uses a ratio of $1:3$ instead of $1:1$. In this section, we study how to improve SVT by optimizing this allocation ratio and by introducing other techniques.

4.1 A Generalized SVT Algorithm

We present a generalized SVT algorithm in Alg. 7, which uses ϵ_1 to perturb the threshold and ϵ_2 to perturb the query answers. Furthermore, to accommodate the situations where one wants the noisy counts for positive queries, we also use ϵ_3 to output query answers using the Laplace mechanism.

We now prove the privacy for Alg. 7; the proof requires only minor changes from the proof of Theorem 1.

THEOREM 2. *Alg. 7 is $(\epsilon_1 + \epsilon_2 + \epsilon_3)$-DP.*

PROOF. Alg. 7 can be divided into two phases, the first phase outputs a vector to mark which query is above the threshold and the second phase uses the Laplace mechanism to output noisy counts for the queries found to be above the threshold in the first phase.

Algorithm 7 Our Proposed Standard SVT

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots, c$ and ϵ_1, ϵ_2 and ϵ_3.
Output: A stream of answers $a_1, a_2, \cdots$

1: $\rho = \mathsf{Lap}\left(\frac{\Delta}{\epsilon_1}\right)$, count = 0
2: **for** Each query $q_i \in Q$ **do**
3: $\quad \nu_i = \mathsf{Lap}\left(\frac{2c\Delta}{\epsilon_2}\right)$
4: $\quad$ **if** $q_i(D) + \nu_i \geq T_i + \rho$ **then**
5: $\quad\quad$ **if** $\epsilon_3 > 0$ **then**
6: $\quad\quad\quad$ **Output** $a_i = q_i(D) + \mathsf{Lap}\left(\frac{c\Delta}{\epsilon_3}\right)$
7: $\quad\quad$ **else**
8: $\quad\quad\quad$ **Output** $a_i = \top$
9: $\quad\quad$ count = count + 1, **Abort** if count $\geq c$.
10: $\quad$ **else**
11: $\quad\quad$ **Output** $a_i = \bot$

Since the second phase is ϵ_3-DP, it suffices to show that the first phase is $(\epsilon_1 + \epsilon_2)$-DP, which can be obtained by Theorem 1. □

4.2 Optimizing Privacy Budget Allocation

In Alg. 7, one needs to decide how to divide up a total privacy budget ϵ into $\epsilon_1, \epsilon_2, \epsilon_3$. We note that $\epsilon_1 + \epsilon_2$ is used for outputting the indicator vector, and ϵ_3 is used for outputting the noisy counts for queries found to be above the threshold; thus the ratio of $(\epsilon_1 + \epsilon_2) : \epsilon_3$ is determined by the domain needs and should be an input to the algorithm.

On the other hand, the ratio of $\epsilon_1 : \epsilon_2$ affects the accuracy of SVT. Most variants use $1 : 1$, without a clear justification. To choose a ratio that can be justified, we observe that this ratio affects the accuracy of the following comparison:

$$q_i(D) + \mathsf{Lap}\left(\frac{2c\Delta}{\epsilon_2}\right) \geq T + \mathsf{Lap}\left(\frac{\Delta}{\epsilon_1}\right).$$

To make this comparison as accurate as possible, we want to minimize the variance of $\mathsf{Lap}\left(\frac{\Delta}{\epsilon_1}\right) - \mathsf{Lap}\left(\frac{2c\Delta}{\epsilon_2}\right)$, which is

$$2\left(\frac{\Delta}{\epsilon_1}\right)^2 + 2\left(\frac{2c\Delta}{\epsilon_2}\right)^2,$$

when $\epsilon_1 + \epsilon_2$ is fixed. This is minimized when

$$\epsilon_1 : \epsilon_2 = 1 : (2c)^{2/3}. \tag{18}$$

We will evaluate the improvement resulted from this optimization in Section 6.

4.3 SVT for Monotonic Queries

In some usages of SVT, the queries are monotonic. That is, when changing from D to D', all queries whose answers are different change in the same direction, i.e., there do not exist q_i, q_j such that $(q_i(D) > q_i(D')) \wedge (q_j(D) < q_j(D'))$. That is, we have either $\forall_i\, q_i(D) \geq q_i(D')$, or $\forall_i\, q_i(D') \geq q_i(D)$. This is the case when using SVT for frequent itemset mining in [14] with neighboring datasets defined as adding or removing one tuple. For monotonic queries, adding $\mathsf{Lap}\left(\frac{c\Delta}{\epsilon_2}\right)$ instead of $\mathsf{Lap}\left(\frac{2c\Delta}{\epsilon_2}\right)$ suffices for privacy.

THEOREM 3. *Alg. 7 with $\nu_i = \mathsf{Lap}\left(\frac{c\Delta}{\epsilon_2}\right)$ in line 3 satisfies $(\epsilon_1 + \epsilon_2 + \epsilon_3)$-DP when all queries are monotonic.*

PROOF. Because the second phase of Alg. 7 is still ϵ_3-DP, we just need to show that for any output vector $\boldsymbol{a} \in \{\top, \bot\}^\ell$,

$$\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}] = \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z] \prod_{i \in \mathbf{I}_\bot} f_i(D, z) \prod_{i \in \mathbf{I}_\top} g_i(D, z)\, dz$$
$$\leq e^{\epsilon_1 + \epsilon_2} \mathsf{Pr}\left[\mathcal{A}(D') = \boldsymbol{a}\right].$$

First consider the case that $\forall_i\, q_i(D) \geq q_i(D')$. In this case, we always have $\frac{f_i(D,z)}{f_i(D',z)} \leq 1$ and do not need to change the integration variable to bound the ratio of the f_i terms. Because of this, having $\nu_i = \mathsf{Lap}\left(\frac{c\Delta}{\epsilon_2}\right)$ suffices to bound the ratio of the g_i terms. That is,

$$f_i(D, z) = \mathsf{Pr}[q_i(D) + \nu_i < T_i + z] =$$
$$\leq \mathsf{Pr}\left[q_i(D') + \nu_i < T_i + z\right] = f_i(D', z),$$
$$g_i(D, z) = \mathsf{Pr}[q_i(D) + \nu_i \geq T_i + z] \leq \mathsf{Pr}\left[q_i(D') + \nu_i + \Delta \geq T_i + z\right]$$
$$\leq e^{\frac{\epsilon_2}{c}} \mathsf{Pr}\left[q_i(D') + \nu_i \geq T_i + z\right] = e^{\frac{\epsilon_2}{c}} g_i(D', z).$$

Thus, noting that $|\mathbf{I}_\top| \leq c$,

$$\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}] \leq \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z] \prod_{i \in \mathbf{I}_\bot} f_i(D', z) \prod_{i \in \mathbf{I}_\top} e^{\frac{\epsilon_2}{c}} g_i(D', z)\, dz$$
$$\leq \left(e^{\frac{\epsilon_2}{c}}\right)^c \mathsf{Pr}\left[\mathcal{A}(D') = \boldsymbol{a}\right] < e^{\epsilon_1 + \epsilon_2} \mathsf{Pr}\left[\mathcal{A}(D') = \boldsymbol{a}\right].$$

Then consider the case in which $\forall_i\, q_i(D) \leq q_i(D')$. Noting that $q_i(D) \geq q_i(D') - \Delta$, we can have

$$f_i(D, z - \Delta) = \mathsf{Pr}[q_i(D) + \nu_i < T_i + z - \Delta]$$
$$\leq \mathsf{Pr}\left[q_i(D') - \Delta + \nu_i < T_i + z - \Delta\right] = f_i(D', z),$$

and $\mathsf{Pr}[\rho = z - \Delta] \leq e^{\epsilon_1} \mathsf{Pr}[\rho = z]$.

With the constraint $q_i(D) \leq q_i(D')$, we have

$$g_i(D, z - \Delta) = \mathsf{Pr}[q_i(D) + \nu_i \geq T_i + z - \Delta]$$
$$\leq \mathsf{Pr}\left[q_i(D') + \nu_i \geq T_i + z - \Delta\right]$$
$$\leq e^{\frac{\epsilon_2}{c}} \mathsf{Pr}\left[q_i(D') + \nu_i \geq T_i + z\right] = e^{\frac{\epsilon_2}{c}} g_i(D', z).$$

Thus, with a change of integration variable from z to $z - \Delta$,

$$\mathsf{Pr}[\mathcal{A}(D) = \boldsymbol{a}]$$
$$= \int_{-\infty}^{\infty} \mathsf{Pr}[\rho = z - \Delta] \prod_{i \in \mathbf{I}_\bot} f_i(D, z - \Delta) \prod_{i \in \mathbf{I}_\top} g_i(D, z - \Delta)\, dz$$
$$\leq \int_{-\infty}^{\infty} e^{\epsilon_1} \mathsf{Pr}[\rho = z] \prod_{i \in \mathbf{I}_\bot} f_i(D', z) \prod_{i \in \mathbf{I}_\top} e^{\frac{\epsilon_2}{c}} g_i(D', z)\, dz$$
$$\leq e^{\epsilon_1 + \epsilon_2} \mathsf{Pr}\left[\mathcal{A}(D') = \boldsymbol{a}\right].$$

□

For monotonic queries, the optimization of privacy budget allocation (18) becomes $\epsilon_1 : \epsilon_2 = 1 : c^{2/3}$.

5. SVT VERSUS EM

We now discuss the application of SVT in the non-interactive setting, where all the queries are known ahead of time. We note that most recent usages of SVT, e.g., [1, 14, 19, 20, 21], are in the non-interactive setting. Furthermore, these applications of SVT aim at selecting up to c queries with the highest answers. In [14], SVT is applied to find the c most frequent itemsets, where the queries are the supports for the itemsets. In [1], the goal of using SVT is to determine the structure of a Bayesian network that preserves as much information of the dataset as possible. To this end, they select attribute groups that are highly correlated and create edges for such groups in the network. While the algorithm in [1] takes the form of selecting attribute groups with a score above a certain threshold, the real goal is to select the groups with the highest scores. In [19], SVT is used to select parameters to be shared when trying to learn neural-network models in a private fashion. Once selected, noises

are added to these parameters before they are shared. The selection step aims at selecting the parameters with the highest scores.

EM or SVT. In a non-interactive setting, one can also use the Exponential Mechanism (EM) [16] to achieve the same objective of selecting the top c queries. More specifically, one runs EM c times, each round with privacy budget $\frac{\epsilon}{c}$. The quality for each query is its answer; thus each query is selected with probability proportion to $\exp\left(\frac{\epsilon}{2c\Delta}\right)$ in the general case and to $\exp\left(\frac{\epsilon}{c\Delta}\right)$ in the monotonic case. After one query is selected, it is removed from the pool of candidate queries for the remaining rounds.

An intriguing question is which of SVT and EM offers higher accuracy. Theorem 3.24 in [8] regarding the utility of SVT with $c = \Delta = 1$ states: For any sequence of k queries $f_1, \ldots, f_k$ such that $|\{i < k : f_i(D) \geq T - \alpha\}| = 0$ (i.e. the only query close to being above threshold is possibly the last one), SVT is (α, β) accurate (meaning that with probability at least $1 - \beta$, all queries with answers below $T - \alpha$ result in $\perp$ and all queries with answers above $T - \alpha$ result in $\top$) for: $\alpha_{\mathrm{SVT}} = 8(\log k + \log(2/\beta))/\epsilon$.

In the case where the last query is at least $T + \alpha$, being (α, β)-correct ensures that with probability at least $1-\beta$, the correct selection is made. For the same setting, we say that *EM is* (α, β)*-correct* if given $k - 1$ queries with answer $\leq T - \alpha$ and one query with answer $\geq T + \alpha$, the correct selection is made with probability at least $1 - \beta$. The probability of selecting the query with answer $\geq T + \alpha$ is at least $\frac{e^{\epsilon(T+\alpha)/2}}{(k-1)e^{\epsilon(T-\alpha)/2}+e^{\epsilon(T+\alpha)/2}}$ by the definition of EM. To ensure this probability is at least $1 - \beta$,

$$\alpha_{\mathrm{EM}} = (\log(k-1) + \log((1-\beta)/\beta))/\epsilon,$$

which is less than $1/8$ of the α_{SVT}, which suggests that EM is more accurate than SVT.

The above analysis relies on assuming that the first $k-1$ queries are no more than $T - \alpha$. When that is not assumed, it is difficult to analyze the utility of either SVT or EM. Therefore, we will use experimental methods to compare SVT with EM.

SVT with Retraversal. We want to find the most optimized version of SVT to compare with EM, and note that another interesting parameter that one can tune when applying SVT is that of the threshold T. When T is high, the algorithm may select fewer than c queries after traversing all queries. Since roughly each selected query consumes $\frac{1}{c}$'th of the privacy budget, outputting fewer than c queries "wastes" the remaining privacy budget. When T is low, however, the algorithm may have selected c queries before encountering later queries. No matter how large some of these later query answers are, they cannot be selected.

We observe that in the non-interactive setting, there is a way to deal with this challenge. One can use a higher threshold T, and when the algorithm runs out of queries before finding c above-threshold queries, one can retraverse the list of queries that have not been selected so far, until c queries are selected. However, it is unclear how to select the optimal threshold. In our experiments, we consider SVT-ReTr, which increases the threshold T by different multiples of the scale factor of the Laplace noise injected to each query, and applies the retraversal technique.

6. EVALUATION

In this section, we experimentally compare the different versions of the SVT algorithm, including our proposed SVT algorithm with different privacy budget allocation methods. We also compare the SVT variants applicable in the non-interactive setting with EM.

We use three metrics to compare the different algorithms, and now explain the rationale for using these metrics. For an algorithm $\mathcal{A}$ and input dataset D, let $U_{\mathcal{A}(D)}$ denote an output from running $\mathcal{A}$ on D. $U_{\mathcal{A}(D)}$ is an unranked set of queries. When $\mathcal{A}$ is EM, $U_{\mathcal{A}(D)}$ has exactly c queries. When $\mathcal{A}$ is an SVT algorithm, $U_{\mathcal{A}(D)}$ may include $< c$ queries. The ground truth is given by the set of all queries and their true answers. Algorithm $\mathcal{A}$ provides better utility if $U_{\mathcal{A}(D)}$ better matches the ground truth.

F-measure . Let U_T denote the c queries with the highest answers. When using U_T as the ground truth, we want to compute the utility of a computed unordered set $U_{\mathcal{A}(D)}$ against a ground truth unordered set U_T. We use the widely used *F-measure* [15] which is the harmonic mean of precision and recall, i.e.,

$$F = \frac{2PR}{P+R},$$
$$\text{where} \quad P = \frac{|U_{\mathcal{A}(D)} \cap U_T|}{|U_{\mathcal{A}(D)}|}, \quad R = \frac{|U_{\mathcal{A}(D)} \cap U_T|}{|U_T|}.$$

We note that when $|U_{\mathcal{A}(D)}| = |U_T|$, the precision equals the recall, and the F-measure equals the precision, and can also be interpreted as 1 minus the false negative rate.

Normalized Cumulative Gain (NCG) . The F-measure uses only the unordered set U_T as the ground truth. As a result, missing the query with the highest answer is penalized the same as missing the c'th one. To address this limitation, we can assign a relevance score $\mathsf{rel}(q)$ to each query q, and use the Normalized Cumulative Gain (NCG) metric [13]:

$$\mathsf{NCG}(U_{\mathcal{A}}(D)) = \frac{\sum_{q \in U_{\mathcal{A}}(D)} \mathsf{rel}(q)}{c}.$$

We point out that since the SVT and EM algorithms output an unorder set of queries, we cannot use discounted cumulative gain or normalized discounted cumulative gain [13], which discount the gain of a query based on which position it is outputted. Below, we use two instantiations of NCG, by choosing different relevance score functions.

Normalized Cumulative Rank (NCR) . We first define $\mathsf{rel}(q)$ to be q's rank score as follows: the highest query has a score of c, the next one has score $c-1$, and so on. Thus, the c'th query has a score of 1. All other queries have a score of 0. To normalize this into a value between 0 and 1, we divide the sum of relevance scores by the maximum possible score, $\frac{c(c+1)}{2}$. This gives rise to what we call the Normalized Cumulative Rank (NCR); this metric uses the true rank information of the top c queries.

Normalized Cumulative Support (NCS) . Using NCR still misses some information. Selecting a query with a very low answer will be penalized the same as selecting the $(c+1)$'th query, whose answer may be quite close to the c'th query. We thus define $\mathsf{rel}(q)$ to be the true answer of q, which we call the support of q. To normalize this into a value between 0 and 1, we divide the cumulative support by the maximum possible support, that is,

$$\mathsf{NCS}(U_{\mathcal{A}(D)}) = \frac{\sum_{q \in U_{\mathcal{A}(D)}} q(D)}{\sum_{q \in U_T} q(D)}.$$

This measures the ratio of cumulative supports by selecting $U_{\mathcal{A}(D)}$ to the total supports of queries in U_T. This metric uses the actual query answer values.

All the above three metrics, F-measure, NCR and NCS are in the range $[0.0, 1.0]$, where higher values indicate better accuracy. We present results under these metrics and observe that the correlation among them is quite stable.

Table 1: Dataset characteristics

Dataset	Number of Records	Number of Queries
BMS-POS	515,597	1,657
Kosarak	990,002	41,270
AOL	647,377	2,290,685
Zipf	10,000,000	10,000

Table 2: Summary of algorithms

Settings	Methods	Description
Interactive	SVT-DPBook	DPBook SVT (Alg. 2).
	SVT-S	Standard SVT (Alg. 7).
Non-interactive	SVT-ReTr	Standard SVT with Retraversal.
	EM	Exponential Mechanism.

Datasets. We use the item frequencies in three real datasets: BMS-POS, Kosarak and AOL as representative distributions of query scores. That is, each item is viewed as a query, and the answer to the query is the support of the item. In addition, we also use the distribution inspired by the Zipf's law, which states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Similar phenomenon occurs in many other rankings unrelated to language, such as the population ranks of cities in various countries, corporation sizes, income rankings, ranks of number of people watching the same TV channel, and so on. In this distribution, the i'th query has a score proportional to $\frac{1}{i}$. The characteristics of these datasets are summarized in Table 1.

Evaluation Setup. We consider the following algorithms. SVT-DPBook is from the book [8] (Alg. 2). SVT-S is our proposed standard SVT, i.e., Alg. 7 without numerical outputs ($\epsilon_3 = 0$); and since the count query is monotonic, we use the version for monotonic queries in Section 4.3. We consider four privacy budget allocations, 1:1, 1:3, 1:c and 1:$c^{2/3}$, where the last is what our analysis suggests for the monotonic case. These algorithms can be applied in the interactive as well as the non-interactive setting.

For the non-interactive setting, we consider EM and SVT-ReTr, which increases the threshold and retraverses through the queries until c of them are selected. We use the 1:$c^{2/3}$ privacy budget allocation and consider 3 variants: 1D, 3D, and 5D, where 1D means adding one standard deviation of the added noises to the threshold.

We vary c from 50 to 300, and uses the average score for the c'th query and the $c+1$'th query as the threshold. We show results for privacy budget varied from $\epsilon = 0.1$ to $\epsilon = 2.0$. We run each experiment 100 times, each time randomizing the order of queries to be examined. We report the average and standard deviation of the three metrics, F-measure, NCR and NCS. All algorithms are implemented in Python 2.7 and all the experiments are conducted on an Intel Core i7-3770 3.40GHz PC with 16GB memory.

Results in the Interactive Setting. Figure 3 reports the results for the algorithms that can be applied in the interactive setting for $\epsilon = 0.25$. Each row is for one dataset, and each column is for one metric. While it is clear that in some settings (such as when $c = 50$) all methods are quite accurate, and in some other settings (such as when $c \geq 250$) all methods are very inaccurate, in the settings between the two extremes, the differences among these methods are quite large. SVT-DPBook performs the worst, followed by SVT-S-1:1, then by SVT-S-1:3, and finally by SVT-S-1:c and SVT-S-1:c23. The differences among these algorithms can be quite pronounced. For example, from Figure 3(f), when $c = 100$, SVT-DPBook's NCS is 0.238, which means that the average support of selected queries is only around 24% of that for the true top-100 queries, which we interpret to mean that the output is meaningless. In contrast, all four SVT-S algorithms have NCS greater than 0.85, suggesting high accuracy in the selection. SVT-DPBook's poor performance is due to the fact that the threshold is perturbed by a noise with scale as large as $c\Delta/\epsilon$.

Among the four budget allocation approaches, it appears that the performance of $1 : c$ and $1 : c^{\frac{2}{3}}$ are clearly better than the others; and their advantages over the standard $1 : 1$ allocation is quite pronounced. Using $1 : c^{\frac{2}{3}}$ is clearly the best, although its advantage over using $1 : c$ is small.

Results in the Non-interactive Setting. Figure 4 reports the results for the algorithms that can be applied in the non-interactive setting. We observe that EM clearly performs better than SVT-ReTr-1:c23-1D, which performs noticeably better than SVT-S-1:c23, which is the best algorithm for the interactive setting.

It is interesting to see that increasing the threshold can significantly improve the accuracy of SVT with retraversal. However, the best threshold increment value depends on the number of queries to be selected. For example, 5D works well when c is large, but works not as well when c is small. Since it is unclear how to select the best threshold increment value, and even with the best threshold increment, SVT-ReTr performs no better than EM, our experiments suggest that usage of SVT should be replaced by EM in the non-interactive setting.

Varying the privacy budget. Figures 5, 6, and 7, compare eight methods while varying both the privacy budget ϵ from 0.1 ro 2.0 and c from 50 to 300. We use heatmap and plot 1.0 minus the metric values, so that blue and green denote metric values close to 1.0. In each figure, the first row are interactive algorithms and the second row are non-interactive algorithms. The observations we made above continue to hold. Also note that for any dataset and any ϵ, the accuracy levels that SVT-DPBook achieves for finding top-50 queries are about the same as what SVT-S-1:c23 achieves for finding top-150 queries, and what EM achieves for finding top-250 or 300 queries.

Recommendations. In summary, our recommendations regarding SVT, based on analysis and experiments, are:

1. **In interactive settings, use our proposed standard SVT (Alg. 1) and choose $\frac{\epsilon_1}{\epsilon_2} = \frac{1}{(2c)^{2/3}}$ for the general case, and $\frac{\epsilon_1}{\epsilon_2} = \frac{1}{(c)^{2/3}}$ for the monotonic case.**
2. **In non-interactive settings, do not use SVT and use EM instead. If one gets better performance using SVT than using EM, then it is likely that one's usage of SVT is non-private.**

7. RELATED WORK

SVT was introduced by Dwork et al. [7], and improved by Roth and Roughgarden [18] and by Hardt and Rothblum [12]. These usages are in an interactive setting. An early description of SVT as a stand-alone technique appeared in Roth's 2011 lecture notes [17], which is Alg. 3 in this paper, and is in fact ∞-DP. The algorithms in [18, 12] also have another difference, as discussed in Section 3.2. Another version of SVT appeared in the 2014 book [8], which is Alg. 2. This version is used in some papers, e.g., [19]. We show that it is possible to add less noise and obtain higher accuracy for the same privacy parameter.

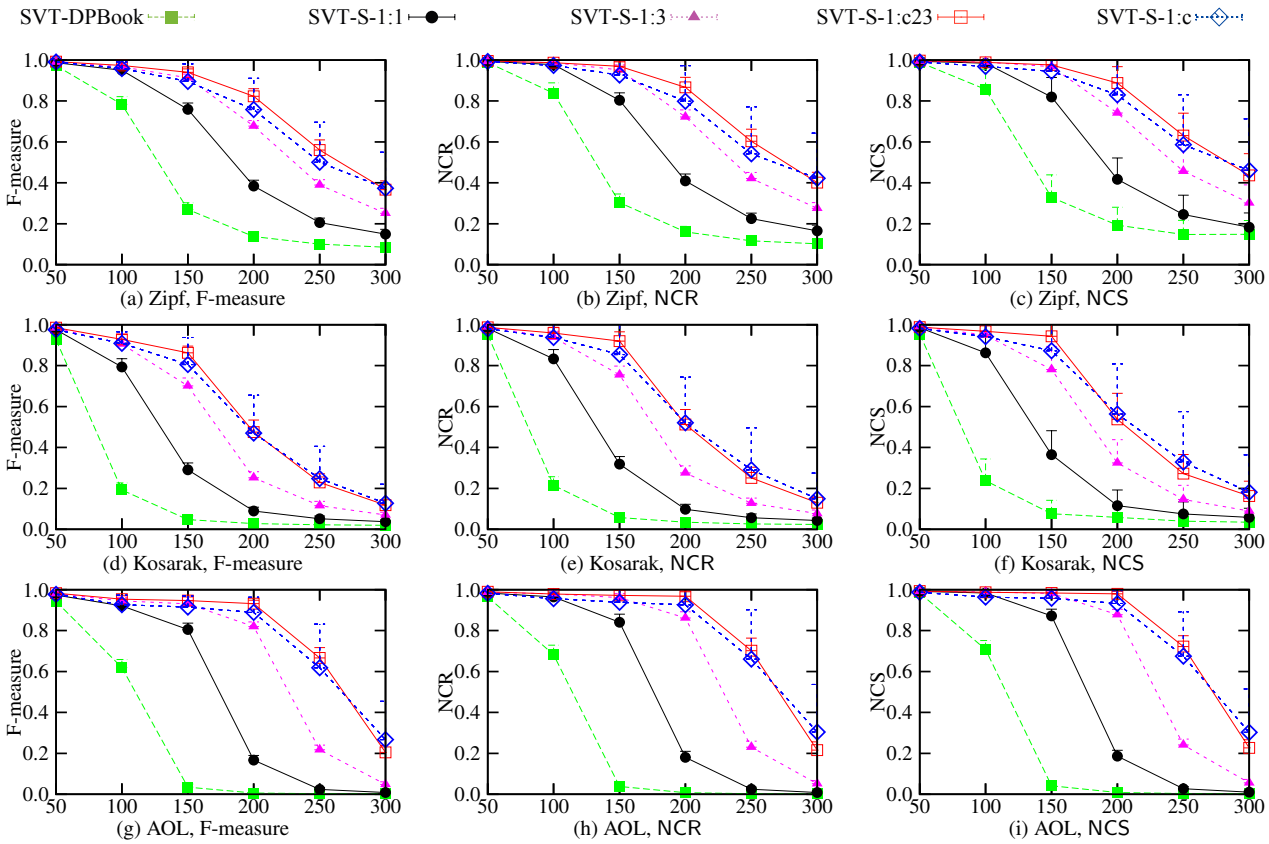


Figure 3: Comparison of interactive approaches. Privacy budget $\epsilon = 0.25$. x-axis: c

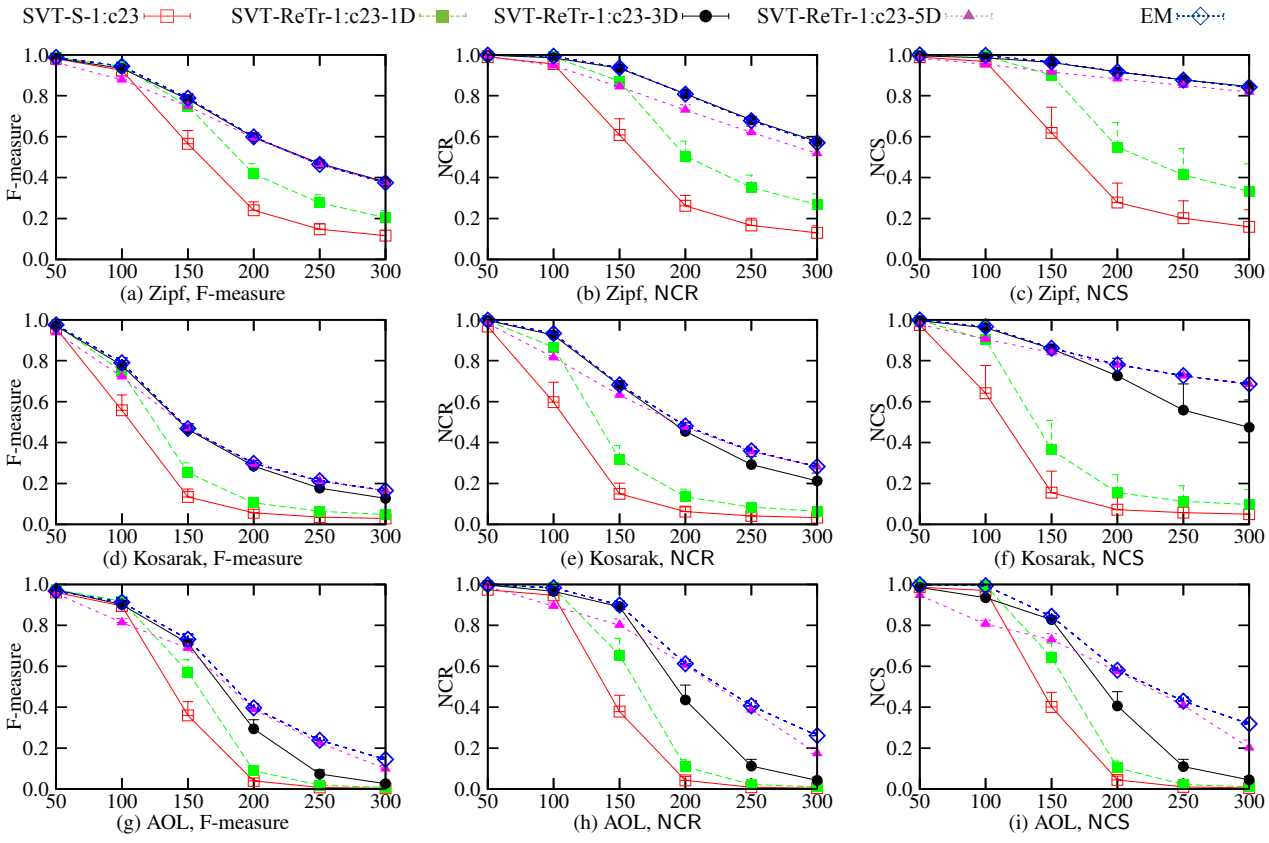
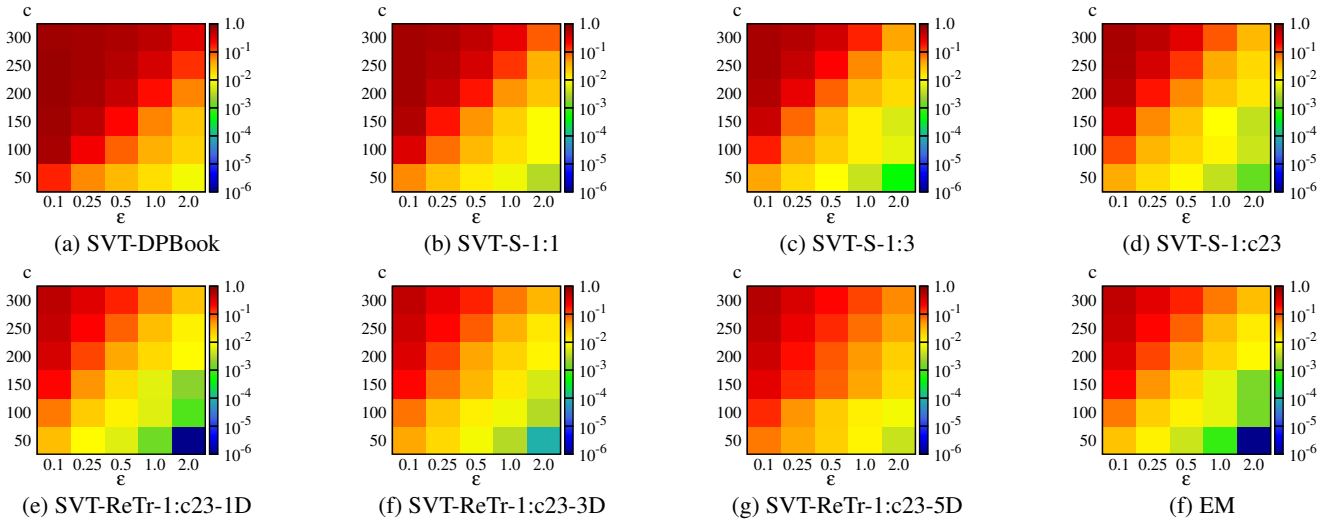


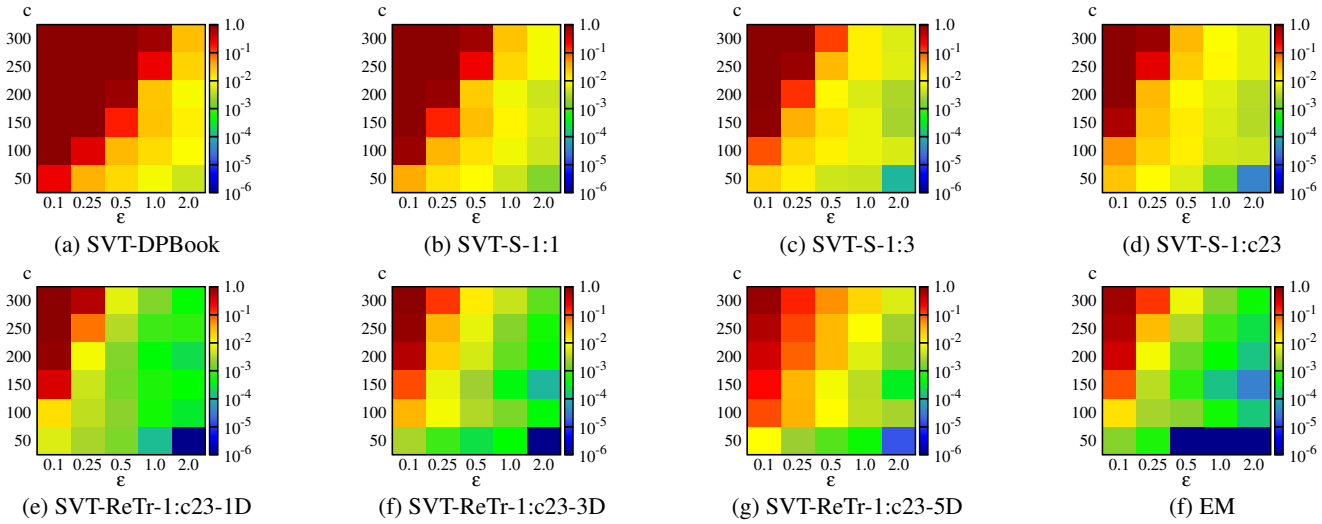
Figure 4: Comparison of non-interactive approaches. Privacy budget $\epsilon = 0.1$. x-axis: c.



(a) SVT-DPBook (b) SVT-S-1:1 (c) SVT-S-1:3 (d) SVT-S-1:c23

(e) SVT-ReTr-1:c23-1D (f) SVT-ReTr-1:c23-3D (g) SVT-ReTr-1:c23-5D (f) EM

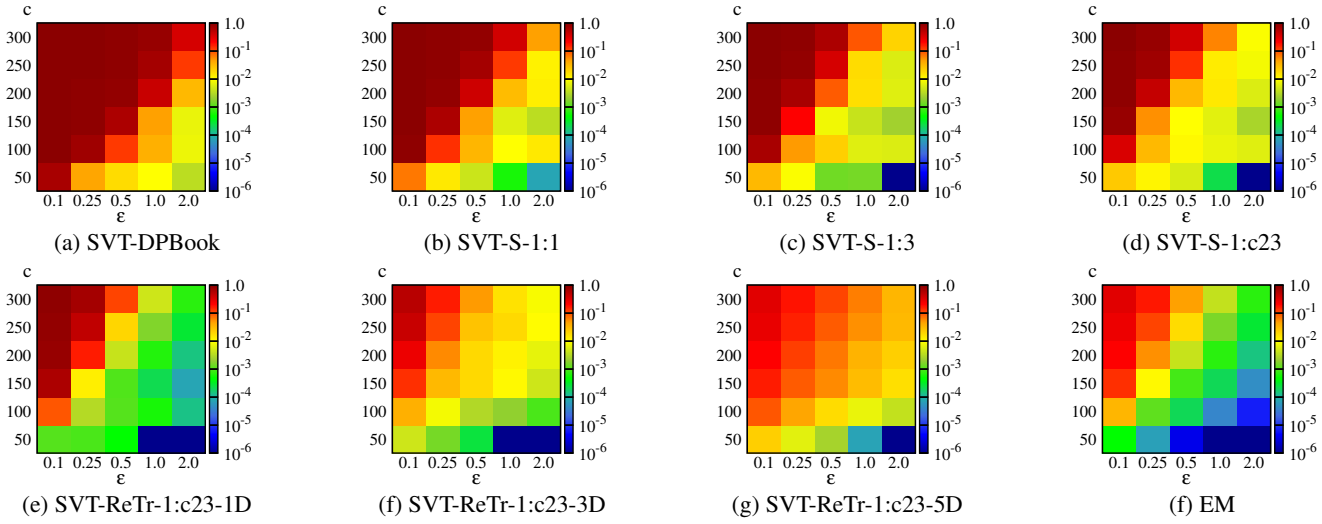
Figure 5: The heatmap by varying ϵ and c on BMS-POS dataset, measured by $1.0 -$ F-measure.



(a) SVT-DPBook (b) SVT-S-1:1 (c) SVT-S-1:3 (d) SVT-S-1:c23

(e) SVT-ReTr-1:c23-1D (f) SVT-ReTr-1:c23-3D (g) SVT-ReTr-1:c23-5D (f) EM

Figure 6: The heatmap by varying ϵ and c on AOL dataset, measured by $1.0 -$ NCR.



(a) SVT-DPBook (b) SVT-S-1:1 (c) SVT-S-1:3 (d) SVT-S-1:c23

(e) SVT-ReTr-1:c23-1D (f) SVT-ReTr-1:c23-3D (g) SVT-ReTr-1:c23-5D (f) EM

Figure 7: The heatmap by varying ϵ and c on Kosarak dataset, measured by $1.0 -$ NCS.

Lee and Clifton [14] used a variant of SVT (see Algorithm 4) to find itemsets whose support is above the threshold. Stoddard et al. [20] proposed another variant (see Algorithm 5) for private feature selection for classification to pick out the set of features with scores greater than the perturbed threshold. Chen et al. [1] employed yet another variant of SVT (see Algorithm 6) to return attribute pairs with mutual information greater than the corresponding noisy threshold. These usages are not private. Some of these errors were pointed in [2], in which a generalized private threshold testing algorithm (GPTT) that attempts to model the SVT variants in [14, 20, 1] was introduced. The authors showed that GPTT did not satisfy ϵ'-DP for any finite ϵ'. But there is an error in the proof, as shown in Section 3.3. Independent from our work, Zhang et al. [22] presented two proofs that the variant of SVT violates DP without discussing the cause of the errors. Also presented in [22] is a special case of our proposed Alg. 1 for counting queries. To our knowledge, the general version of our improved SVT (Alg. 1 and Alg. 7), the techniques of optimizing budget allocation, the technique of using re-traversal to improve SVT, and the comparison of SVT and EM are new in our work.

8. CONCLUSION

We have introduced a new version of SVT that provides better utility. We also introduce an effective technique to improve the performance of SVT by optimizing the distribution of privacy budget. These enhancements achieve better utility than the state of the art SVT and can be applied to improve utility in the interactive setting. We have also explained the misunderstandings and errors in a number of papers that use or analyze SVT; and believe that these will help clarify the misunderstandings regarding SVT and help avoid similar errors in the future. We have also shown that in the non-interactive setting, EM should be preferred over SVT.

9. ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments. This paper is based upon work supported by Key Laboratory on High Performance Computing, Anhui Province, NSFC (61672486, 61672480, 11671376), Key Program of NSFC (71631006), OATF,USTC and the United States National Science Foundation under Grant No. 1116991 and 1640374.

10. REFERENCES

[1] R. Chen, Q. Xiao, Y. Zhang, and J. Xu. Differentially private high-dimensional data publication via sampling-based inference. In *KDD*, pages 129–138, 2015.

[2] Y. Chen and A. Machanavajjhala. On the privacy properties of variants on the sparse vector technique. *CoRR*, abs/1508.07306, 2015.

[3] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[4] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[5] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[6] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94, 2007.

[7] C. Dwork, M. Naor, O. Reingold, G. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. *STOC*, pages 381–390, 2009.

[8] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013.

[9] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. FOCS '10, pages 51–60, 2010.

[10] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. CRYPTO'08, pages 469–480, 2008.

[11] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *TCC*, pages 339–356, 2012.

[12] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

[13] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.

[14] J. Lee and C. W. Clifton. Top-k frequent itemsets via differentially private fp-trees. In *KDD '14*, pages 931–940, 2014.

[15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[16] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[17] A. Roth. The sparse vector technique, 2011. Lecture notes for " The Algorithmic Foundations of Data Privacy".

[18] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.

[19] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *CCS*, pages 1310–1321, 2015.

[20] B. Stoddard, Y. Chen, and A. Machanavajjhala. Differentially private algorithms for empirical machine learning. *CoRR*, abs/1411.5428, 2014.

[21] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *SIGMOD '14*, pages 1423–1434, 2014.

[22] J. Zhang, X. Xiao, and X. Xie. Privtree: A differentially private algorithm for hierarchical decompositions. SIGMOD '16, pages 155–170, 2016.

11. APPENDIX

11.1 Proof that Alg. 3 is non-private

THEOREM 4. *Alg. 3 is not ϵ'-DP for any finite ϵ'.*

PROOF. Set $c = 1$ for simplicity. Given any finite $\epsilon' > 0$, we construct an example to show that Alg. 3 is not ϵ'-DP. Consider an example with $T = 0$, and $m + 1$ queries $\mathbf{q}$ with sensitivity Δ such that $\mathbf{q}(D) = 0^m\Delta$ and $\mathbf{q}(D') = \Delta^m 0$, and the output vector $\boldsymbol{a} = \perp^m 0$, that is, only the last query answer is a numeric value 0. Let $\mathcal{A}$ be Alg. 3. We show that $\frac{\Pr[\mathcal{A}(D)=\boldsymbol{a}]}{\Pr[\mathcal{A}(D')=\boldsymbol{a}]} \geq e^{\epsilon'}$ for any $\epsilon' > 0$ when m is large enough.

We denote the cumulative distribution function of $\mathsf{Lap}\left(\frac{2\Delta}{\epsilon}\right)$ by $F(x)$. By Eq. (1), We have

$$\begin{aligned}
&\Pr[\mathcal{A}(D) = \boldsymbol{a}] \\
&= \int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i \in \mathbf{I}_\perp} f_i(D, z) \Pr[\Delta + \nu_{m+1} \geq z \wedge \Delta + \nu_{m+1} = 0]\ dz \\
&= \int_{-\infty}^{\infty} \Pr[\rho = z] \prod_{i \in \mathbf{I}_\perp} f_i(D, z) \Pr[0 \geq z] \Pr[\nu_{m+1} = -\Delta]\ dz
\end{aligned}$$

$$=\frac{\epsilon}{4\Delta}e^{-\frac{\epsilon}{2}}\int_{-\infty}^{\infty}\mathsf{Pr}[\rho{=}z]\prod_{i\in\mathbf{I}_\perp}f_i(D,z)\mathsf{Pr}[0\geq z]\;dz$$

$$=\frac{\epsilon}{4\Delta}e^{-\frac{\epsilon}{2}}\int_{-\infty}^{0}\mathsf{Pr}[\rho{=}z]\prod_{i\in\mathbf{I}_\perp}f_i(D,z)\;dz$$

$$=\frac{\epsilon}{4\Delta}e^{-\frac{\epsilon}{2}}\int_{-\infty}^{0}\mathsf{Pr}[\rho=z]\prod_{i=1}^{m}\mathsf{Pr}[\nu_i<z]\;dz$$

$$=\frac{\epsilon}{4\Delta}e^{-\frac{\epsilon}{2}}\int_{-\infty}^{0}\mathsf{Pr}[\rho=z]\,(F(z))^m\,dz, \tag{19}$$

and similarly

$$\mathsf{Pr}\big[\mathcal{A}(D')=\boldsymbol{a}\big]=\frac{\epsilon}{4\Delta}\int_{-\infty}^{0}\mathsf{Pr}\big[\rho=z'\big]\,(F(z'-\Delta))^m\,dz'. \tag{20}$$

The fact that 0 is given as an output reveals the information that the noisy threshold is at most 0, forcing the range of integration to be from $-\infty$ to 0, instead of from $-\infty$ to ∞. This prevents the use of changing z in (19) to $z'-\Delta$ to bound the ratio of (19) to (20).

Noting that $\frac{F(z)}{F(z-\Delta)}=e^{\frac{\epsilon}{2}}$ for any $z\leq 0$, we thus have

$$\begin{aligned}\frac{\mathsf{Pr}[\mathcal{A}(D)=\boldsymbol{a}]}{\mathsf{Pr}[\mathcal{A}(D')=\boldsymbol{a}]}&=e^{-\frac{\epsilon}{2}}\frac{\int_{-\infty}^{0}\mathsf{Pr}[\rho=z]\,(F(z))^m\,dz}{\int_{-\infty}^{0}\mathsf{Pr}[\rho=z']\,(F(z'-\Delta))^m\,dz'}\\&=e^{-\frac{\epsilon}{2}}\frac{\int_{-\infty}^{0}\mathsf{Pr}[\rho=z]\,(e^{\frac{\epsilon}{2}}F(z-\Delta))^m\,dz}{\int_{-\infty}^{0}\mathsf{Pr}[\rho=z']\,(F(z'-\Delta))^m\,dz'}\\&=e^{(m-1)\frac{\epsilon}{2}},\end{aligned}$$

and thus when $m>\lceil\frac{2\epsilon'}{\epsilon}\rceil+1$, we have $\frac{\mathsf{Pr}[\mathcal{A}(D)=\boldsymbol{a}]}{\mathsf{Pr}[\mathcal{A}(D')=\boldsymbol{a}]}>e^{\epsilon'}$. □

11.2 Proof that Alg. 6 is non-private

THEOREM 5. *Alg. 6 is not ϵ'-DP for any finite ϵ'.*

PROOF. We construct a counterexample with $\Delta=1$, $T=0$, and $2m$ queries such that $\mathbf{q}(D)=0^{2m}$, and $\mathbf{q}(D')=1^m(-1)^m$. Consider the output vector $\boldsymbol{a}=\perp^m\top^m$. Denote the cumulative distribution function of ν_i by $F(x)$. From Eq. (1), we have

$$\begin{aligned}&\mathsf{Pr}[\mathcal{A}(D)=\boldsymbol{a}]\\&=\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\prod_{i=1}^{m}\mathsf{Pr}[0{+}\nu_i<z]\prod_{i=m+1}^{2m}\mathsf{Pr}[0{+}\nu_i\geq z]\;dz\\&=\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\,(F(z)(1-F(z)))^m\,dz,\end{aligned}$$

and

$$\begin{aligned}&\mathsf{Pr}\big[\mathcal{A}(D')=\boldsymbol{a}\big]\\&=\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\prod_{i=1}^{m}\mathsf{Pr}[1{+}\nu_i<z]\prod_{i=m+1}^{2m}\mathsf{Pr}[-1{+}\nu_i\geq z]\;dz\\&=\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\,(F(z-1)(1-F(z{+}1)))^m\,dz.\end{aligned}$$

We now show that $\frac{\mathsf{Pr}[\mathcal{A}(D)=\boldsymbol{a}]}{\mathsf{Pr}[\mathcal{A}(D')=\boldsymbol{a}]}$ is unbounded as m increases, proving this theorem. Compare $F(z)(1-F(z))$ with $F(z-1)(1-F(z+1))$. Note that $F(z)$ is monotonically increasing. When $z\leq 0$,

$$\frac{F(z)(1-F(z))}{F(z-1)(1-F(z{+}1))}\geq\frac{F(z)}{F(z-1)}=\frac{\frac{1}{2}e^{\frac{\epsilon}{2}z}}{\frac{1}{2}e^{\frac{\epsilon}{2}(z-1)}}=e^{\frac{\epsilon}{2}}.$$

When $z>0$, we also have

$$\frac{F(z)(1-F(z))}{F(z-1)(1-F(z{+}1))}\geq\frac{1-F(z)}{1-F(z+1)}=\frac{\frac{1}{2}e^{-\frac{\epsilon}{2}z}}{\frac{1}{2}e^{-\frac{\epsilon}{2}(z+1)}}=e^{\frac{\epsilon}{2}}.$$

So, $\frac{\mathsf{Pr}[\mathcal{A}(D)=\boldsymbol{a}]}{\mathsf{Pr}[\mathcal{A}(D')=\boldsymbol{a}]}\geq e^{\frac{m\epsilon}{2}}$, which is greater than $e^{\epsilon'}$ when $m>\lceil\frac{2\epsilon'}{\epsilon}\rceil$ for any finite ϵ'. □

11.3 Error of non-privacy proof in [2]

The proof in [2] that GPTT is non-private considers the counterexample with $\Delta=1$, $T=0$, a sequence $\mathbf{q}$ of $2t$ queries such that $\mathbf{q}(D)=0^t1^t$ and $\mathbf{q}(D')=1^t0^t$, and the output vector $\boldsymbol{a}=\perp^t\top^t$. Then

$$\frac{\mathsf{Pr}[\mathsf{GPTT}(D)=\boldsymbol{a}]}{\mathsf{Pr}[\mathsf{GPTT}(D')=\boldsymbol{a}]}=\frac{\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\left(F_{\epsilon_2}(z)-F_{\epsilon_2}(z)F_{\epsilon_2}(z-1)\right)^t dz}{\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\left(F_{\epsilon_2}(z-1)-F_{\epsilon_2}(z)F_{\epsilon_2}(z-1)\right)^t dz}$$

where $F_\epsilon(x)$ is the cumulative distribution function of $\mathsf{Lap}\,(1/\epsilon)$.

The goal of the proof is to show that the above is unbounded as t increases. A key observation is that the ratio of the integrands of the two integrals is always larger than 1, i.e.,

$$\kappa(z)=\frac{F_{\epsilon_2}(z)-F_{\epsilon_2}(z)F_{\epsilon_2}(z-1)}{F_{\epsilon_2}(z-1)-F_{\epsilon_2}(z)F_{\epsilon_2}(z-1)}>1$$

For example, since $F_\epsilon(x)$ is the cumulative distribution function of $\mathsf{Lap}\,(1/\epsilon)$, we have $F_{\epsilon_2}(0)=1/2$ and $F_{\epsilon_2}(-1)<1/2$; and thus $\kappa(0)=\frac{1-F_{\epsilon_2}(-1)}{F_{\epsilon_2}(-1)}>1$. However, when $|z|$ goes to ∞, $\kappa(z)$ goes to 1. Thus the proof tries to limit the integrals to be a finite interval so that there is a lower-bound for $\kappa(z)$ that is greater than 1. It denotes $\alpha=\mathsf{Pr}[\mathsf{GPTT}(D')=\boldsymbol{a}]$. Then choose parameter $\delta=|F^{-1}_{\epsilon_1}(\frac{\alpha}{4})|$ to use $[-\delta,\delta]$ as the finite interval, and thus

$$\alpha\leq 2\int_{-\delta}^{\delta}\mathsf{Pr}[\rho=z]\left(F_{\epsilon_2}(z-1)-F_{\epsilon_2}(z)F_{\epsilon_2}(z-1)\right)^t dz.$$

Denote the minimum of $\kappa(z)$ in the closed interval $[-\delta,\delta]$ by κ. Then we have $\frac{\mathsf{Pr}[\mathsf{GPTT}(D)=\boldsymbol{a}]}{\mathsf{Pr}[\mathsf{GPTT}(D')=\boldsymbol{a}]}>\frac{\kappa^t}{2}$. The proof claims that for any $\epsilon'>1$ there exists a t to make the above ratio larger than $e^{\epsilon'}$.

The proof is incorrect because of dependency in the parameters. First, α is a function of t; and when t increases, α decreases because the integrand above is positive and decreasing. Second, δ depends on α, and when α decreases, δ increases. Thus when t increases, δ increases. We write δ as $\delta(t)$ to make the dependency on t explicit. Third, κ, the minimum value of $\kappa(z)$ over the interval $[-\delta(t),\delta(t)]$, decreases when t increases. That is, κ is also dependent on t, denoted by $\kappa(t)$, and decreases while t increases. It is not sure that there exists such a t that $\frac{\kappa(t)^t}{2}>e^{\epsilon'}$ for any $\epsilon'>1$.

To demonstrate the error in the proof cannot be easily fixed, we point out that following the logic of that proof, one can prove that Alg. 1 is not ϵ'-DP for any finite ϵ'. We now show such a "proof" that contradicts Theorem 1. Let $\mathcal{A}$ be Alg. 1 with $c=1$. Consider an example with $\Delta=1$, $T=0$, a sequence $\mathbf{q}$ of t queries such that $\mathbf{q}(D)=0^t$ and $\mathbf{q}(D')=1^t$, and output vector $\boldsymbol{a}=\perp^t$. Let

$$\beta=\mathsf{Pr}\Big[\mathcal{A}(D)=\perp^\ell\Big]=\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\left(F_{\frac{\epsilon}{4}}(z)\right)^t dz$$

$$\alpha=\mathsf{Pr}\Big[\mathcal{A}(D')=\perp^\ell\Big]=\int_{-\infty}^{\infty}\mathsf{Pr}[\rho=z]\left(F_{\frac{\epsilon}{4}}(z-1)\right)^t dz,$$

where $F_{\frac{\epsilon}{4}}(x)$ is the cumulative distribution function of $\mathsf{Lap}\,(4/\epsilon)$.

Find a parameter δ such that $\int_{-\delta}^{\delta}\mathsf{Pr}[\rho=z]\;dz\geq 1-\frac{\alpha}{2}$. Then $\int_{-\delta}^{\delta}\mathsf{Pr}[\rho=z]\left(F_{\frac{\epsilon}{4}}(z-1)\right)^t dz\geq\frac{\alpha}{2}$. Let κ be the minimum value of $\frac{F_{\frac{\epsilon}{4}}(z)}{F_{\frac{\epsilon}{4}}(z-1)}$ in $[-\delta,\delta]$; it must be that $\kappa>1$. Then

$$\begin{aligned}\beta&>\int_{-\delta}^{\delta}\mathsf{Pr}[\rho=z]\left(F_{\frac{\epsilon}{4}}(z)\right)^t dz\geq\int_{-\delta}^{\delta}\mathsf{Pr}[\rho=z]\left(\kappa F_{\frac{\epsilon}{4}}(z-1)\right)^t dz\\&=\kappa^t\int_{-\delta}^{\delta}\mathsf{Pr}[\rho=z]\left(F_{\frac{\epsilon}{4}}(z-1)\right)^t dz\geq\frac{\kappa^t}{2}\alpha.\end{aligned}$$

Since $\kappa>1$, one can choose a large enough t to make $\frac{\beta}{\alpha}=\frac{\kappa^t}{2}$ to be as large as needed. We note that this contradicts Theorem 1. The contradiction shows that the proof logic used in [2] is incorrect.