Locally Differentially Private Frequent Itemset Mining

Tianhao Wang
Department of Computer Science
Purdue University
West Lafayette, IN
tianhaowang@purdue.edu

Ninghui Li
Department of Computer Science
Purdue University
West Lafayette, IN
ninghui@cs.purdue.edu

Somesh Jha
Department of Computer Science
University of Wisconsin-Madison
Madison, WI
jha@cs.wisc.edu

***Abstract*—The notion of Local Differential Privacy (LDP) enables users to respond to sensitive questions while preserving their privacy. The basic LDP frequent oracle (FO) protocol enables an aggregator to estimate the frequency of any value. But when each user has a set of values, one needs an additional padding and sampling step to find the frequent values and estimate their frequencies. In this paper, we formally define such padding and sample based frequency oracles (PSFO). We further identify the privacy amplification property in PSFO. As a result, we propose SVIM, a protocol for finding frequent items in the set-valued LDP setting. Experiments show that under the same privacy guarantee and computational cost, SVIM significantly improves over existing methods. With SVIM to find frequent items, we propose SVSM to effectively find frequent itemsets, which to our knowledge has not been done before in the LDP setting.**

I. Introduction

In recent years, differential privacy [14], [16] has been increasingly accepted as the *de facto* standard for data privacy in the research community [2], [15], [17], [24]. In the standard (or centralized) setting, a data curator collects personal data from each individual, and produces outputs based on the dataset in a way that satisfies differential privacy. In this setting, the data curator sees the raw input from all users and is trusted to handle these private data correctly.

Recently, techniques for avoiding a central trusted authority have been introduced. They use the concept of Differential Privacy in the Local setting, which we call LDP. Such techniques enable collection of statistics of users' data while preserving privacy of participants, without relying on trust in a single data curator. For example, researchers from Google developed RAPPOR [18], [20] and Prochlo [8], which are included as part of Chrome. They enable Google to collect users' answers to questions such as the default homepage of their browser, the default search engine, and so on, in order to understand the unwanted or malicious hijacking of user settings. Apple [33], [34] also uses similar methods to help with predictions of spelling and other tasks. Samsung proposed a similar system [28] which enables collection of not only categorical answers but also numerical answers (e.g., time of usage, battery volume), although it is not clear whether this has been deployed by Samsung. Firefox [1] is also planning to build a "RAPPOR-like" system that collects frequent homepages.

We assume that each user possesses an input value $v \in D$, where D is the value domain. A party wants to learn the distribution of the input values of all users. We call this party the *aggregator* instead of the data curator, because it does not see the raw data. Existing research [5], [18], [36] has developed multiple frequency oracle (FO) protocols, using which an aggregator can estimate the frequency of any chosen value $x \in D$. In [30], Qin et al. considered the setting where each user's value is a set of items $\mathbf{v} \subseteq I$, where I is the item domain. Such a set-valued setting occurs frequently in the situation where LDP is applied. For example, when Apple wants to estimate the frequencies of the emoji's typed everday by the users, each user has a set of emoji's that they typed [34]. The LDPMiner protocol in [30] aims at finding the k most frequent items and their frequencies.

This problem is challenging because the number of items each user has is different. To deal with this, a core technique in [30] is "**padding and sampling**". That is, each user first pads her set of values with dummy items to a fixed size ℓ, then randomly samples one item from the padded set, and finally uses an FO protocol to report the item. When estimating the frequency of an item, one multiples the estimation from the FO protocol by ℓ. Without padding, the probability that an item is sampled is difficult to assess, making accurate frequency estimation difficult.

In [30], the FO protocol is used in a black-box fashion. That is, in order to satisfy ϵ-LDP, the FO protocol is invoked with the same privacy parameter ϵ. We observe that, since the sampling step randomly selects an item, it has an amplification effect in terms of privacy. This effect has been observed and studied in the standard DP setting [25]. If one applies an algorithm to a dataset randomly sampled from the input with a sampling rate of $\beta < 1$, to satisfy ϵ-DP, the algorithm can use a privacy budget of $\epsilon' > \epsilon$; more specifically, the relationship between ϵ', ϵ, and β is $\frac{e^{\epsilon'}-1}{e^{\epsilon}-1} = \frac{1}{\beta}$.

Intuitively, one can apply the same observation here. Since each item is selected with probability $\beta = \frac{1}{\ell}$, to satisfy ϵ-LDP, one can invoke the FO protocol with ϵ', such that $\frac{e^{\epsilon'}-1}{e^{\epsilon}-1} = \ell$ (or, equivalently $\epsilon' = \ln(\ell \cdot (e^{\epsilon}-1)+1) \geq \epsilon$).

Surprisingly, in our study of **padding-and-sampling-based frequency oracle** (PSFO), we found that one cannot always get this privacy amplification effect. Whether this benefit is applicable or not depends on the internal structure of the FO protocol. In [36], the three best performing FO protocols are Generalized Random Response, Optimized Unary Encoding, and Optimized Local Hash. The latter two offer the same accuracy, and Optimized Local Hash has lower communication cost. It was found that Generalized Random Response offers the best accuracy when $|D| < 3e^{\epsilon} + 2$, and Optimized Local Hash offers the best accuracy when $|D| \geq 3e^{\epsilon} + 2$. We found that, the privacy amplification effect exists for Generalized Random Response, but not for Optimized Local Hash. Optimized Local Hash is able to provide better accuracy when $|D|$ is large because each perturbed output can be used to support multiple input values. However, the same feature makes Optimized Local Hash unable to benefit from sampling. The difference in the ability to benefit from sampling changes the criterion to decide which of Generalized Random Response and Optimized Local Hash to use. We thus propose to adaptively select the best FO protocol in PSFO, based on $|I|, \epsilon$ and the particular ℓ value. Essentially, when $|I| > (4\ell^2 - \ell) \cdot e^{\epsilon} + 1$, Generalized Random Response should be used. Replacing the FO protocol used in [30] with such an adaptively chosen FO protocol greatly improves the accuracy of the resulting frequent items.

We also observe that the selection of an appropriate ℓ is crucial, and it can be different depending on the goal. Essentially, each user pads her itemset to size ℓ, generating two sources of errors: When ℓ is small, one would under-estimate the frequency counts, since items in a set with more than ℓ items will be sampled with probability less than $1/\ell$. On the other hand, since ℓ is multiplied to a noisy estimate, increasing ℓ magnifies the noises. The LDPMiner protocol in [30] has two phases, the first phase selects $2k$ candidate frequent items using a quite large ℓ, and the second phase computes their frequencies using $\ell = 2k$. We observe that for the purpose of identifying candidates for the frequent items, setting $\ell = 1$ is fine. While the resulting frequency counts under-estimate the true counts, the frequencies of all items are under-estimated, and it is very unlikely that the true top k items are not among the $2k$ candidates. However, when the goal is to estimate frequency, one needs select a larger ℓ. But ℓ should not be increased to the point that there is absolutely no under-estimation, because this increases the magnitude of noises. Selecting ℓ is a trade-off between under-estimation and noise.

Following these insights, we propose Set-Value Item Mining (SVIM) protocol, which handles set values under the LDP setting and provides much better accuracy than existing protocols within the same privacy constraints. There are four steps: First, users use PSFO with a small ℓ to report; the aggregator identifies frequent items as candidates, and sends this set to users. Second, users report (using a standard FO protocol) the number of candidate items they have; the aggregator estimates the distribution of how many candidate items the users have and selects appropriate ℓ, and sends ℓ to users. Third, users use PSFO with the given ℓ to report occurrences of items in the candidate set; the aggregator estimates the frequency of these items. Fourth, the aggregator selects the top k frequent items and use the size distribution in step two to further correct undercounts. Experimental results how that SVIM significantly outperforms LDPMiner in that it identifies more frequent items as well as estimates the frequencies more accurately.

In the setting where each user's input data is a set of items, a natural problem is to find frequent itemsets. Frequent itemset mining (FIM) is a well recognized data-mining problem. The discovery of frequent itemsets can serve valuable economic and research purposes, e.g., mining association rules [4], predicting user behavior [3], and finding correlations [9]. FIM while satisfying DP in the centralized setting has been studied extensively, e.g., [7], [39], [26]. However, because of the challenges of dealing with set-valued inputs in the LDP setting, no solution for the LDP setting has been proposed. Authors of [30] consider only the identification of frequent items, and leave FIM as an open problem. Using the PSFO technique, we are able to provide the first solution to FIM in the LDP setting. We call the protocol Set-Value itemSet Mining (SVSM) protocol; experimental evaluations demonstrates its effectiveness.

To summarize, the main contributions of this paper are:

- We investigate padding-and-sample-based frequency oracles (PSFO) and discover the interesting phenomenon that some FO protocols can benefit from the sampling step, but others cannot. Based on this, we proposed to adaptively select the best-performing FO protocol in each usage of PSFO.
- We design and implement SVIM to find frequent values together with their frequencies. Experimental results on both empirical and real-world datasets demonstrate the significant improvement over previous techniques.
- We provide the first FIM protocol under the LDP setting, and empirically demonstrate its effectiveness on real-world datasets. This solves a problem left open by [30].

Roadmap. In Section II, we present background knowledge of LDP and FO. We then go over the problem definition and existing solutions in Section III. With an investigation of the sample-based frequency oracle in Section IV, we present our proposed method in Section V. Experimental results are presented in VI. Finally we discuss related work in Section VII and provide concluding remarks in in Section VIII.

II. Background

We consider a setting where there are several *users* and one *aggregator*. Each user possesses a value v from a domain D, and the aggregator wants to learn the distribution of values among all users, in a way that protects the privacy of individual users.

A. Differential Privacy in the Local Setting

In the local setting, each user perturbs the input value v using an algorithm Ψ and sends $\Psi(v)$ to the aggregator. The

formal privacy requirement is that the algorithm $\Psi(\cdot)$ satisfies the following property:

Definition 1 (ϵ Local Differential Privacy)**.** *An algorithm $\Psi(\cdot)$ satisfies ϵ-local differential privacy (ϵ-LDP), where $\epsilon \geq 0$, if and only if for any input $v_1, v_2 \in D$, we have*

$$\forall T \subseteq Range(\Psi) : \Pr[\Psi(v_1) \in T] \leq e^{\epsilon} \Pr[\Psi(v_2) \in T],$$

where $Range(\Psi)$ denotes the set of all possible outputs of the algorithm Ψ.

Similar to the centralized setting, there is sequential composition in the local setting. That is, if the user executes a set of functions, each satisfying ϵ_i-LDP, then the whole process satisfies $\sum \epsilon_i$-LDP. The value ϵ is also called the *privacy budget*.

Compared to the centralized setting, the local version of DP offers a stronger level of protection, because each user only reports the perturbed data. Each user's privacy is still protected even if the aggregator is malicious.

B. Frequency Oracles

A *frequency oracle (*FO*)* protocol enables the estimation of the frequency of any given value $x \in D$ under LDP. It is specified by a pair algorithms: $\langle \Psi, \Phi \rangle$, where Ψ is used by each user to perturb her input value, and Φ is used by the aggregator; Φ takes as input the reports from all users, and can be queried for the frequency of each value.

*1) Generalized Randomized Response (*GRR*):* This FO protocol generalizes the *randomized response* technique [38]. In the special case where the value is one bit, i.e., when $d = |D| = 2$, $\Psi_{\mathsf{GRR}(\epsilon)}(v)$ keeps the bit unchanged with probability $\frac{e^{\epsilon}}{e^{\epsilon}+1}$ and flips it with probability $\frac{1}{e^{\epsilon}+1}$. In the general case, when $d > 2$, the perturbation function is defined as

$$\forall_{y \in D} \Pr\left[\Psi_{\mathsf{GRR}(\epsilon)}(v) = y\right] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+d-1}, & \text{if } y = v \\ q = \frac{1}{e^{\epsilon}+d-1}, & \text{if } y \neq v \end{cases} \tag{1}$$

This satisfies ϵ-LDP since $\frac{p}{q} = e^{\epsilon}$. To estimate the frequency of $x \in D$, one counts how many times x is reported as $C(x)$, and then computes

$$\Phi_{\mathsf{GRR}(\epsilon)}(x) \coloneqq \frac{C(x) - nq}{p - q} \tag{2}$$

where n is the total number of users. That is, the frequency estimate is a linear transformation of the noisy count $C(x)$, in order to account for the effect of randomized response. In [36], it is shown that this is an unbiased estimation of the true count, and the variance for this estimation is

$$\mathrm{Var}[\Phi_{\mathsf{GRR}(\epsilon)}(x)] = \frac{d - 2 + e^{\epsilon}}{(e^{\epsilon} - 1)^2} \cdot n \tag{3}$$

The accuracy of this protocol deteriorates fast when the domain size d increases. This is reflected in the fact that (3) is linear in d.

More sophisticated frequency estimators have been studied before [18], [5], [36]. In [36], several such protocols are analyzed, optimized, and compared against each other, and it was found that when d is large, the Optimized Local Hashing (OLH) protocol provides the best accuracy while maintaining a low communication cost. In this paper, we use the OLH protocol as a primitive and describe it below.

*2) Optimized Local Hashing (*OLH*) [36]:* The *Optimized Local Hashing (*OLH*)* protocol deals with a large domain size d by first using a hash function to map an input value into a smaller domain of size g (typically $g \ll d$), and then applying randomized response to the hashed value in the smaller domain. In this protocol, both the hashing step and the randomization step result in information loss. The choice of the parameter g is a tradeoff between losing information during the hashing step and losing information during the randomization step. In [36], it is found that the optimal (minimal variance) choice of g is $\lceil e^{\epsilon} + 1 \rceil$.

In OLH, the reporting protocol is

$$\Psi_{\mathsf{OLH}(\epsilon)}(v) \coloneqq \langle H, \Psi_{\mathsf{GRR}(\epsilon)}(H(v)) \rangle,$$

where H is randomly chosen from a family of hash functions that hash each value in D to $\{1 \dots g\}$, and $\Psi_{\mathsf{GRR}(\epsilon)}$ is given in (1), while operating on the domain $\{1 \dots g\}$.

Let $\langle H^j, y^j \rangle$ be the report from the j'th user. For each value $x \in D$, to compute its frequency, one first computes $C(x) = |\{j \mid H^j(x) = y^j\}|$. That is, $C(x)$ is the number of reports that "supports" that the input is x. One then transforms $C(x)$ to its unbiased estimation

$$\Phi_{\mathsf{OLH}(\epsilon)}(x) \coloneqq \frac{C(x) - n/g}{p - 1/g}. \tag{4}$$

The variance of this estimation is

$$\mathrm{Var}[\Phi_{\mathsf{OLH}(\epsilon)}(x)] = \frac{4e^{\epsilon}}{(e^{\epsilon} - 1)^2} \cdot n. \tag{5}$$

Compared with (3), the factor $d - 2 + e^{\epsilon}$ is replaced by $4e^{\epsilon}$. This suggests that for smaller d (such that $d - 2 < 3e^{\epsilon}$), one is better off with GRR; but for large d, OLH is better and has a variance that does not depend on d.

III. Set Values under LDP

In [30], Qin et al. considered the problem where each user's value is a set. Such set-valued settings occur frequently in the situation where LDP is applied. For example, iOS users type many emoji's every day, and Apple wants to estimate the frequencies of the emoji's [34].

A. Problem Definition and Challenge

Specifically, the aggregator knows a set I of items. There are n users. The j'th user has a value $\mathbf{v}^j \subseteq I$. We call this a transaction. For any item $x \in I$, its frequency is defined as the number of transactions that include x, i.e., $f_x \coloneqq |\{\mathbf{v}^j \mid x \in \mathbf{v}^j\}|$. Similarly, the frequency of any itemset $\mathbf{x} \subseteq I$ is defined as the number of transactions that include $\mathbf{x}$ as a subset, i.e., $f_{\mathbf{x}} \coloneqq |\{\mathbf{v}^j \mid \mathbf{x} \subseteq \mathbf{v}^j\}|$.

With the constraint of LDP defined on each user's value $\mathbf{v}$, the goal in this setting is to find items and, more generally,

itemsets that are frequent in the population. An item (itemset) is a top-k frequent item (itemset) if its frequency is among the k highest for all items (itemsets).

This problem is quite challenging even when one just tries to find frequent items. Encoding each transaction as a single value in the domain $D = \mathscr{P}(I)$ (i.e., D is the power set of I), and using existing FO protocols does not work. While there exist protocols specifically designed for larges domains (such as [37], [6]), such techniques still doesn't scale to the case where the binary encoding of the input domain has more than a few hundred bits. We want to be able deal with hundreds or thousands of items. An FO protocol can identify only values that are very frequent in the population, because the scale of the added noises is linear to square root of the population size [11]. It is quite possible that each particular transaction appears relative infrequently, even though some items and itemsets appear very frequently. When no value in $\mathscr{P}(I)$ is frequent enough to be identified, using a direct encoding an aggregator can obtain only noises.

See Table I for an example with five transactions. While no transaction appears more than once, items a and e each appears 4 times, and the itemset $\{a, e\}$ appears 3 times. Thus the three most frequent itemsets are $\{a\}, \{e\}, \{a, e\}$.

Transaction
a, c, e
b, d, e
a, b, e
a, d, e
a, f

TABLE I
TRANSACTIONS EXAMPLE.

B. The LDPMiner

To the best of our knowledge, LDPMiner [30] is the only protocol for dealing with set values in the LDP setting. While finding frequent itemsets is a natural goal, LDPMiner finds only frequent items (i.e., singleton itemsets) and leaves the frequent itemset mining as an open problem. LDPMiner has two phases.

Phase 1: Candidate Set Identification. The goal of Phase 1 is to identify a candidate set for frequent items. The protocol requires as input a parameter L, which is the 90th percentile of transaction lengths . That is, about 90% of all transactions have length no more than L. When L is not known, it needs to be estimated. In [30], it is assumed that L is available.

In Phase 1, each user whose transaction $\mathbf{v}$ has less than L items first pads it with dummy items so that the transaction has size L. Then, the user selects at uniform random one item v from the padded transaction (which could result in a dummy item), and uses FO to report it with privacy budget $\epsilon/2$. That is, each user sends to the aggregator $\Psi_{\mathsf{FO}(\epsilon/2)}(v)$. Note that the FO can perturb the original value into any value including the dummy item.

The aggregator then computes, for each item $x \in I$, its estimated frequency as

$$\Phi_{\mathsf{FO}(\epsilon/2)}(x) \cdot L$$

The intuition behind the above estimation is that in each transaction of length L, each item x will be selected and reported with probability $\frac{1}{L}$. Hence one needs to multiply the frequency oracle's estimation by a factor of L. Since 90% of transactions will have length exactly L after padding, this estimation is reasonably accurate. From the estimates, the aggregator identifies S, the set of $2k$ items that have the highest estimated frequencies, and sends S to the users. Size of S is set to be twice that of the goal so that few candidates are missed in this step.

Phase 2: Frequency Estimation. On receiving S, each user intersects it with $\mathbf{v}$, which results in a transaction of length no more than $|S| = 2k$. She then pads her transaction $\mathbf{v} \cap S$ to be of size $2k$, selects at uniform random one item v from the padded transaction, and sends $\Psi_{\mathsf{FO}(\epsilon/2)}(v)$ to the aggregator. Since each user sends two things, each in a way that satisfies $(\epsilon/2)$-LDP, by sequential composition, the protocol satisfies ϵ-LDP.

The aggregator estimates frequency for each item $x \in S$:

$$\Phi_{\mathsf{FO}(\epsilon/2)}(x) \cdot 2k$$

Since the size of all user's transactions have size $2k$ after padding, the estimated frequencies are unbiased.

IV. PADDING-AND-SAMPLING-BASED FREQUENCY ORACLES

The LDPMiner protocol deals with the challenge of set-valued inputs by using padding and sampling before applying an FO protocol to report. We call such protocols *Padding-and-Sampling-based Frequency Oracle (*PSFO*)* protocols. They use a padding-and-sampling function, defined as follows.

Definition 2 (PS)**.** *The padding and sampling function* PS *is specified by a positive integer ℓ and takes a set $\boldsymbol{v} \subseteq I$ as input. It assumes the existence of ℓ dummy items $\perp_1, \perp_2, \ldots, \perp_\ell \notin I$. $\mathsf{PS}_\ell(\boldsymbol{v})$ does the following: If $|\boldsymbol{v}| < \ell$, it adds $\ell - |\boldsymbol{v}|$ different random dummy elements to $\boldsymbol{v}$. It then selects an element v at uniform random and outputs that element.*

A PSFO protocol then uses an FO protocol to transmit the element v. Note that the domain of the FO becomes $I \cup \{\perp_1, \perp_2, \ldots, \perp_\ell\}$. To estimate the frequency of an item x, one obtains the frequency estimation of x from the FO protocol, and then multiplies it by ℓ. More formally,

Definition 3 (PSFO)**.** *A padding-and-sample-based frequency oracle (*PSFO*) protocol is specified by three parameters: a positive integer ℓ, a frequency oracle* FO*, and the privacy budget ϵ. It is composed of a pair of algorithms: $\langle \Psi, \Phi \rangle$, defined as follows.*

$$\mathsf{PSFO}(\ell, \mathsf{FO}, \epsilon) \coloneqq \langle \Psi_{\mathsf{FO}(\epsilon)}(\mathsf{PS}_\ell(\cdot)),\ \ \Phi_{\mathsf{FO}(\epsilon)}(\cdot) \cdot \ell \rangle$$

Note that if one does not do the padding step, it is equivalent to setting $\ell = 1$. Doing so significantly under-estimates the true counts. With padding to length ℓ and then sampling, one can multiply the estimated counts by ℓ to correct the under estimation. However, items that appear in transactions longer than ℓ can still be underestimated. At the same time,

multiplying the estimation by ℓ will enlarge any error due to noise by a factor of ℓ.

Using this notation, the two phases of LDPMiner can be cast as using $\mathsf{PSFO}(L, \mathsf{FO}, \epsilon/2)$ in Phase 1 and $\mathsf{PSFO}(2k, \mathsf{FO}, \epsilon/2)$ in Phase 2.

A. *Privacy Amplification of* GRR

LDPMiner uses the FO protocol in a black-box fashion. That is, in order to satisfy ϵ-LDP, it invokes the FO protocol with the same privacy parameter ϵ. We observe that, since the sampling step randomly selects an item, it has an amplification effect for privacy. This effect has been observed and studied in the standard DP setting [25]: If one applies an algorithm to a dataset randomly sampled from the input with a sampling rate of $\beta < 1$, to satisfy ϵ-DP, the algorithm can use a privacy budget of ϵ' such that $\frac{e^{\epsilon'}-1}{e^{\epsilon}-1} = \frac{1}{\beta}$.

We observe that the same privacy amplification effect exists when using the Generalized Random Response (GRR) in PSFO.

Theorem 1 (PSFO-GRR: Privacy Amplification)**.** *$\Psi_{\mathsf{GRR}(\epsilon')}(\mathsf{PS}_\ell(\cdot))$ satisfies ϵ-LDP, such that $\epsilon' = \ln(\ell \cdot (e^\epsilon - 1) + 1)$.*

Proof. Let $d' = |I| + \ell$ be the size of the new domain ($I' = I \cup \{\perp_1, \ldots, \perp_\ell\}$), ϵ' as the privacy budget used in GRR. As defined in (1), we have $p' = \frac{e^{\epsilon'}}{e^{\epsilon'}+d'-1}$ and $q' = \frac{1}{e^{\epsilon'}+d'-1}$ as the perturbation probabilities.

It suffices to prove that for any $\epsilon \geq 0$, any $\mathbf{v}_1, \mathbf{v}_2 \subseteq I$, and any possible output $t \in I'$, $\frac{p_1}{p_2} \leq e^\epsilon$, where

$$p_1 = \mathsf{Pr}\left[\Psi_{\mathsf{GRR}(\epsilon')}(\mathsf{PS}_\ell(\mathbf{v}_1)) = t\right], \text{ and}$$
$$p_2 = \mathsf{Pr}\left[\Psi_{\mathsf{GRR}(\epsilon')}(\mathsf{PS}_\ell(\mathbf{v}_2)) = t\right].$$

We first examine p_1. When $t \in \mathbf{v}_1$,

$$\begin{aligned} p_1 =& \mathsf{Pr}[t \text{ is sampled}] \cdot p' + \mathsf{Pr}[t \text{ is not sampled}] \cdot q' \\ =& \frac{1}{\max\{|\mathbf{v}_1|, \ell\}} \cdot p' + \frac{\max\{|\mathbf{v}_1|, \ell\} - 1}{\max\{|\mathbf{v}_1|, \ell\}} \cdot q' \\ =& q' + \frac{1}{\max\{|\mathbf{v}_1|, \ell\}} \cdot (p' - q') \\ \leq& q' + \frac{1}{\ell} \cdot (p' - q') \\ =& \frac{1}{\ell} p' + \frac{\ell - 1}{\ell} q' \end{aligned}$$

When $t \notin \mathbf{v}_1$, $p_1 = q'$. Similarly, for p_2, when $t \in \mathbf{v}_2$,

$$\begin{aligned} p_2 =& \mathsf{Pr}[t \text{ is sampled}] \cdot p' + \mathsf{Pr}[t \text{ is not sampled}] \cdot q' \\ =& \mathsf{Pr}[t \text{ is sampled}] \cdot (q' + p' - q') \\ &+ \mathsf{Pr}[t \text{ is not sampled}] \cdot q' \\ =& q' + \mathsf{Pr}[t \text{ is sampled}] \cdot (p' - q') \geq q' \end{aligned}$$

ϵ \ ℓ	2	5	10	20	50	100
0.1	0.19	0.42	0.72	1.13	1.83	2.44
0.5	0.83	1.45	2.01	2.64	3.51	4.19
1.0	1.49	2.26	2.90	3.57	4.46	5.15
2.0	2.62	3.49	4.17	4.86	5.77	6.46
4.0	4.68	5.59	6.29	6.98	7.89	8.59

TABLE II
NUMERICAL VALUE OF ϵ' UNDER DIFFERENT ϵ AND ℓ.

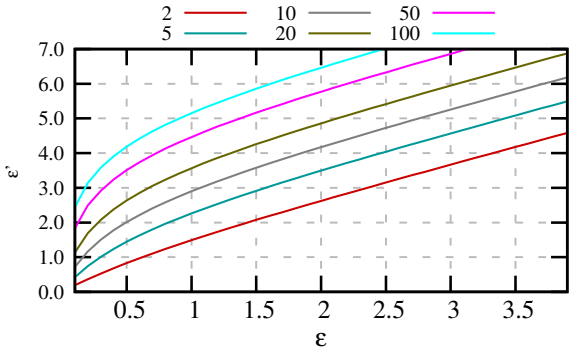


Fig. 1. Privacy amplification effect for different ℓ.

When $t \notin \mathbf{v}_2$, $p_2 = q'$. Thus $\frac{p_1}{p_2}$ is maximized when $p_1 = \frac{1}{\ell}p' + \frac{\ell-1}{\ell}q'$ and $p_2 = q'$. That is,

$$\begin{aligned} \frac{p_1}{p_2} &\leq \frac{p'/\ell + q'(\ell-1)/\ell}{q'} \\ &\leq e^{\epsilon'}\frac{1}{\ell} + \frac{\ell-1}{\ell} \\ &= \frac{(\ell \cdot (e^\epsilon - 1) + 1)}{\ell} + \frac{\ell-1}{\ell} = e^\epsilon \end{aligned}$$

□

Approximately, the privacy budget will be amplified by a factor of $\ln \ell$ (will be the same if $\ell = 1$). Table II and Figure 1 give the corresponding ϵ' value for ϵ under different ℓ.

B. *No Privacy Amplification of other* FO

Interestingly, we found that this privacy amplification effect does not exist for OLH. The reason is that, in GRR, the output domain of the perturbation is the same as the input domain; thus each reported value y can "support" a single input element $x = y$ in I. In OLH, however, the reported value takes the form of $\langle H, j \rangle$ and can support any element x in I such that $H(x) = j$. In case the chosen hash function H hashes all the user's items into the same value, no matter how we sample, the hashed result after sampling will always be the same value. Therefore, there is no privacy amplification in the sampling.

Theorem 2 (PSFO-OLH: No Privacy Amplification)**.** *$\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}_\ell(\cdot))$ does not satisfy ϵ-LDP for any $\epsilon < \epsilon'$ when the input domain I is sufficiently large.*

Proof. Let g be the output size of hash functions. Consider an input domain I such that $|I| \geq g\ell + 1$. Let H be the chosen hash function. By the pigeon hole principle, there exists a

value y such that H hashes at least ℓ items into y. Let $\mathbf{v}_1$ consists of ℓ items that are hashed to y, and $\mathbf{v}_2$ consists of items that are not hashed to y. Then

$$\begin{aligned}&\frac{\Pr\left[\Psi_{\mathsf{OLH}(\epsilon)}(\mathsf{PS}_\ell(\mathbf{v}_1))=\langle H,y\rangle\right]}{\Pr\left[\Psi_{\mathsf{OLH}(\epsilon)}(\mathsf{PS}_\ell(\mathbf{v}_2))=\langle H,y\rangle\right]}\\=&\frac{\Pr\left[\Psi_{\mathsf{GRR}(\epsilon)}(H(\mathsf{PS}_\ell(\mathbf{v}_1)))=y|H\right]}{\Pr\left[\Psi_{\mathsf{GRR}(\epsilon)}(H(\mathsf{PS}_\ell(\mathbf{v}_2)))=y|H\right]}\\=&\frac{\Pr\left[H\text{ is picked}\right]\cdot p'}{\Pr\left[H\text{ is picked}\right]\cdot q'}=\frac{p'}{q'}=e^{\epsilon'}\end{aligned}$$

Therefore, $\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}_\ell(\cdot))$ is not ϵ-LDP for any $\epsilon<\epsilon'$. □

In [36], another FO protocol, Optimal Unary Encoding (OUE), was proposed. It has similar accuracy as OLH. In OUE, the reported value is a binary vector, each bit representing one possible input value. One reported value can have multiple bits being 1, supporting multiple input values. Similar to OLH, in case the reported vector supports all the user's items, there is no privacy amplification.

Note that from each user's point of view, the hash function H is randomly chosen. Thus only when the user happens to choose a hash function H that hashes all the user's items into the same hash value, would there be no privacy amplification benefit at all. However, this can happen with only small probability. This observation suggests that (ϵ,δ)-LDP can be applied to obtain some amplification effect, as will be discussed in Appendix A.

C. Utility of PSFO

We now analyze the accuracy of PSFO. We first show that PSFO is unbiased when each user's itemset size is no more than ℓ.

Theorem 3 (PSFO Expectation)**.** $\mathsf{PSFO}(\ell,\mathsf{FO},\epsilon)$ *is unbiased when* $\ell\geq\max_{j\in[n]}|\boldsymbol{v}^j|$*. That is,*

$$\mathrm{E}\left[\,\Phi_{\mathsf{PSFO}(\ell,\mathsf{FO},\epsilon)}(x)\,\right]=n_x,$$

where n_x is the number of users who have item x.

Proof. We prove for GRR, using the aggregate function defined in (2). The proof for OLH (with aggregate function in (4)) can also be derived similarly.

$$\begin{aligned}&\mathrm{E}\left[\,\Phi_{\mathsf{PSFO}(\ell,\mathsf{GRR},\epsilon)}(x)\,\right]\\=&\mathrm{E}\left[\,\Phi_{\mathsf{GRR}(\epsilon')}(x)\cdot\ell\,\right]=\mathrm{E}\left[\,\frac{C(x)-nq'}{p'-q'}\cdot\ell\,\right]\\=&\ell\cdot\frac{n_x\frac{1}{\ell}p'+n_x\frac{\ell-1}{\ell}q'+(n-n_x)q'-nq'}{p'-q'}\\=&\ell\cdot\frac{n_x\frac{1}{\ell}(p'-q')+n_xq'+(n-n_x)q'-nq'}{p'-q'}\\=&n_x\end{aligned}$$

□

The estimation is inherently noisy. We now calculate the variance of the estimation.

Theorem 4 (PSFO Variance)**.** $\mathsf{PSFO}(\ell,\mathsf{FO},\epsilon)$ *has variance ℓ^2 times that of* FO *when* $\ell\geq\max_{j\in[n]}|\boldsymbol{v}^j|$*. That is,*

$$\mathrm{Var}[\Phi_{\mathsf{PSFO}(\ell,\mathsf{FO},\epsilon)}(x)]=\ell^2\cdot\mathrm{Var}[\Phi_{\mathsf{FO}(\epsilon')}(x)],$$

where $\epsilon'=\ln\left(\ell\cdot(e^\epsilon-1)+1\right)$ *if* FO *is* GRR*.*

Proof. We prove for GRR, and the proof for OLH can be easily derived.

$$\begin{aligned}&\mathrm{Var}[\Phi_{\mathsf{PSFO}(\ell,\mathsf{GRR},\epsilon)}(x)]=\mathrm{Var}[\Phi_{\mathsf{GRR}(\epsilon')}(x)\cdot\ell]\\=&\mathrm{Var}\left[\frac{C(x)-nq'}{p'-q'}\cdot\ell\right]=\frac{\ell^2}{(p'-q')^2}\cdot\sum_j\mathrm{Var}[C(x)]\\=&\frac{\ell^2}{(p'-q')^2}\cdot\left[n_x\left(\frac{1}{\ell}p'+\frac{\ell-1}{\ell}q'\right)\left(1-\left(\frac{1}{\ell}p'+\frac{\ell-1}{\ell}q'\right)\right)\right.\\&\left.+(n-n_x)q'(1-q')\right]\\\simeq&\frac{\ell^2}{(p'-q')^2}\cdot[nq'(1-q')]=\ell^2\cdot\mathrm{Var}[\Phi_{\mathsf{GRR}(\epsilon')}(x)]\end{aligned}$$

□

D. Adaptive FO

PSFO needs to use an FO protocol. In [36], it was shown that one should choose GRR when $d<3e^\epsilon+2$ (where $d=|D|$ is the size of the domain under consideration), and OLH otherwise. With sampling, GRR can benefit from privacy amplification, but OLH benefit less. As a result, the criterion for choosing between GRR and OLH changes. For GRR, when ϵ is used in PSFO, the effective privacy budget GRR can use becomes $\ln(\ell(e^\epsilon-1)+1)$. We use (3) (with domain size $|I'|=d+\ell$) and get:

$$\begin{aligned}&\mathrm{Var}[\Phi_{\mathsf{GRR}(\ln(\ell(\mathrm{e}^\epsilon-1)+1)}(x)\cdot\ell]\\=&n\cdot\ell^2\cdot\frac{d+l-2+\ell\cdot(e^\epsilon-1)+1}{(\ell\cdot(e^\epsilon-1)+1-1)^2}\\=&n\cdot\frac{d+l-1+\ell\cdot(e^\epsilon-1)}{(e^\epsilon-1)^2}\\=&n\cdot\frac{e^\epsilon\cdot\ell+d-1}{(e^\epsilon-1)^2}\end{aligned}\tag{6}$$

For OLH, by (5) we have variance independent on d:

$$\mathrm{Var}[\Phi_{\mathsf{OLH}(\epsilon)}(x)\cdot\ell]=n\cdot\frac{4\ell^2\cdot e^\epsilon}{(e^\epsilon-1)^2}\tag{7}$$

Comparing (6) and (7), when

$$d<\ell(4\ell-1)e^\epsilon+1,\tag{8}$$

using GRR itemset will lead to better accuracy. Note that by taking $\ell=1$, (8) is slightly different from the inequality of $d<3e^\epsilon+2$ from [36]. This is because here we consider a more general setting where some user may have no item at all, while the setting of [36] is that each user has exactly one item. We propose Adap, which becomes GRR or OLH adaptively (with new budget) based on (8). That is,

$$\mathsf{Adap}(\epsilon):=\begin{cases}\mathsf{GRR}(\ln(\ell(e^\epsilon-1)+1)\text{ if }d<e^\epsilon\ell(4\ell-1)+1,\\\mathsf{OLH}(\epsilon)\text{ otherwise.}\end{cases}$$

E. Choosing ℓ

To use PSFO, one needs to decide what value of ℓ to use. When ℓ is small, there is less variance but more bias (in the form of under estimation); when ℓ is large, there is more variance and less bias. To find the suitable ℓ, the high level idea is to find the right tradeoff between bias and variance.

When **identifying** candidate items, the goal is find the most frequent items (but not accurate frequencies for them), we propose to use a small ℓ. The intuition is that, while the bias is large when ℓ is small, the bias tends to be the same direction (namely under estimation) for all items. While the absolute values of the counts are very inaccurate, the relative order remain mostly unchanged. Note that it is possible the order is reversed after sampling (if one item appears more often in smaller transactions, and another item appears more often in larger transactions). To reduce this risk, we identify $2k$ candidate items (the optimal size of the candidate set is dependent on the data distribution; we tried different values and $2k$ appears to be a reasonable choice).

When **estimating** the actual frequency, one should use a larger ℓ to reduce bias. We propose to use the 90th percentile L of the length of the input itemsets. While under estimation can still occur, the degree is limited. Furthermore, when given the distribution of the lengths of input itemsets, we propose to correct this under estimation by multiplying the estimation by the factor:

$$u(L) = \frac{N}{N - \sum_{\ell=L+1}^{d} n^{\ell}(\ell - L)}. \tag{9}$$

Here N denotes the total number of items, n^{ℓ} denotes the number of users with itemset size ℓ, and $\sum_{\ell=L+1}^{d} N^{\ell}(\ell - L)$ gives the total number of missed items.

V. Proposed Method

In this section, we propose solution for the frequent item and itemset mining. We first present Set-Value Item Mining (SVIM) protocol to find frequent items in the set-value setting. Based on the result from SVIM, we build Set-Value itemSet Mining (SVSM) protocol to find frequent itemsets. The high level protocol structure is given in Figure 2.

A. Frequent Item Mining

At a high level, SVIM works as follows: A set of candidate items are identified first. Then these items are estimated and updated. The users are partitioned into three groups, each participating in a task. Given that each task requires privacy budget of ϵ, each user is protected by ϵ-LDP.

Step 1: Prune the Domain. When the domain is big (e.g., tens of thousands), the aggregator has to first narrow down the focus to a small candidate set. Specifically, in Step 1, each user reports with a randomly selected value from her private set with length limit set to 1:

$$\Psi_{\mathsf{PSFO}(1,\mathsf{Adap},\epsilon)}(\mathbf{v}).$$

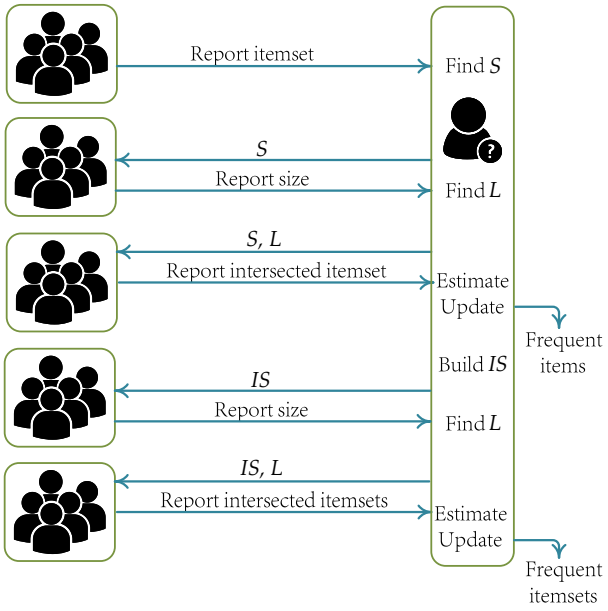


Fig. 2. Illustration of SVIM and SVSM. The users to the left are partitioned into five groups. The aggregator to the right first runs SVIM with the first three groups, and find the frequent items. Then the aggregator interacts with the following two groups to find frequent itemsets.

The advantages of setting $\ell = 1$ are first, every user will report an item, making the signal strong; second, there is no extra cost of obtaining the exact L value.

The aggregator then estimates the frequency of the domain by

$$\Phi_{\mathsf{PSFO}(1,\mathsf{Adap},\epsilon)}(x),$$

and obtains the set S of the $2k$ most frequent items. S is then sent to users who participate in Step 2. Note that this phase is unnecessary when the original domain size close to or less than $2k$.

Step 2: Size Estimation. Having narrowed down the domain from I to S, the aggregator now estimates frequencies of items in S. As suggested by the analysis of PSFO (Section IV-E), the aggregator first finds the 90-th percentile L (in this step) and then uses it as the limit to estimate frequencies of S (next step).

To find L, each user in this task reports the size of the private set intersected with the candidate S, i.e.,

$$\Psi_{\mathsf{OLH}(\epsilon)}(|\mathbf{v} \cap S|).$$

There is no sampling involved in this step, because each user has only one value. Here OLH is as FO by default.

The aggregator in this step estimates the length distribution by calculating

$$\Phi_{\mathsf{OLH}(\epsilon)}(\ell)$$

for all $\ell \in [1, 2, \ldots, 2k]$, and finds the 90 percentile L. That is, the aggregator then finds the smallest L such that $\frac{\sum_{\ell=1}^{L} \Phi_{\mathsf{OLH}(\epsilon)}(\ell)}{\sum_{\ell=1}^{2k} \Phi_{\mathsf{OLH}(\epsilon)}(\ell)} > 0.9$. Information of S and L are then sent to the users for the next task.

Note that some of the estimates may be overwhelmed by noise, making it useless. For this reason, we use the significance threshold $T = F^{-1}\left(1 - \frac{0.05}{2k}\right)\sqrt{\mathrm{Var}}$ to filter the estimates, where F^{-1} is the inverse of standard normal CDF, and Var is specified by (5). Specifically, the aggregator keeps estimates that are greater than T, and replaces all the others with zeros.

Step 3: Candidates Estimation. On receiving S and L, each of the rest of the users reports a value sampled from the intersection of his private set $\mathbf{v}$ and the candidate set S, padded to L, i.e.,

$$\Psi_{\mathsf{PSFO}(L,\mathsf{Adap},\epsilon)}(\mathbf{v} \cap S).$$

The aggregator can estimates the candidates by running

$$\Phi_{\mathsf{PSFO}(L,\mathsf{Adap},\epsilon)}(x),$$

for all $x \in S$. Since the 90-th percentile L is used as limit, the estimates are slightly under-estimate the truth. Therefore, the estimates are updated in the next step.

Step 4: Estimation Update. The update assumes that the missed count follow similar distribution as the reported ones. Given that L is the 90 percentile, the difference will not be significant. Thus the estimate for each item x is multiplied with a fixed update factor (the noisy version of (9))

$$u'(L) \coloneqq \frac{\sum_{\ell=1}^{2k} \Phi_{\mathsf{OLH}(\epsilon)}(\ell)}{\sum_{\ell=1}^{2k} \Phi_{\mathsf{OLH}(\epsilon)}(\ell) - \sum_{\ell=L+1}^{2k} \Phi_{\mathsf{OLH}(\epsilon)}(\ell)\,(\ell - L)} \tag{10}$$

Note that there is no privacy concern in this step because no user is involved. The information is obtained from Step 2 and 3.

Difference from LDPMiner. The major differences between SVIM and LDPMiner are many. (1) In Phase 1 of SVIM, the limit is set to one, instead of the 90-th percentile of lengths of full transactions. (2) In Phase 2 of SVIM, the limit is reduced from $|S|$ to the 90-th percentile L of the length of transactions when limited to items in S. (3) Phase 1 of LDPMiner uses the 90-th percentile; it was assumed that this is provided as input. In SVIM, the 90-th percentile of length is obtained in a way that satisfies LDP. (4) SVIM uses Adap instead of black box FO. (5) SVIM has an update step at the end, which uses the length distribution information to further reduces the bias. (6) In SVIM, users are partitioned into groups, each answering one separate question, instead of answering multiple questions each with part of ϵ. It is proved in [36] that this will make the overall result more accurate. (7) SVIM uses OLH, a more accurate FO introduced in [36]. Since improvements (6) and (7) are not introduced in this paper, in the experiments, for a fair comparison, we evaluate on an improved version of LDPMiner. Specifically, OLH is used as the FO, and users are partitioned into groups. That is, the evaluation shows only differences due to (1), (2), (4), (5). Difference (3) means that SVIM is end-to-end private, and LDPMiner needs a data-dependent input.

B. Frequent Itemset Mining

The problem of mining frequent itemsets is similar to mining frequent items. The desired result becomes a set of itemsets instead of items. These frequent itemsets can be used, for example, by websites, to mine association rules and make recommendations. However, the task is much more challenging, because there are exponentially more itemsets to consider, and each user also has many more potential itemsets.

In this section, we introduce SVSM for finding frequent itemsets effectively. In the high level, the aggregator first obtains the frequent items by executing SVIM. The aggregator then constructs a candidate set of itemsets IS. Finally the set IS is estimated in a fashion similar to the latter part of SVIM.

Constructing Candidate Set. The challenging part of frequent itemset mining is to construct IS. There are exponentially many possible itemsets that can be frequent. If one can reduce it to a manageable range (thousands), one can cast the problem to the item mining problem and run SVIM. Moreover, if size of IS is close to k, only the estimation of IS (latter part of SVIM) suffices.

Let S' be the k most frequent items returned by SVIM. To effectively further reduce the candidate size, we use information of the estimates of S'. Specifically, for an itemset $\mathbf{x}$, we first guess its frequency, denoted by $\tilde{f}_{\mathbf{x}}$, as the product of the estimates for all its items, i.e., $\tilde{f}_{\mathbf{x}} = \prod_{x \in \mathbf{x}} \Phi'(x)$, where $\Phi'(x) = \frac{0.9 \cdot \Phi(x)}{\max_{x \in S'} \Phi(x)}$ is the normalized estimate. The 0.9 factor of $\Phi'(x)$ serves to lower the normalized estimates for the most frequent item, because otherwise, the guessed frequency of any set without the most frequent item equals that of the set plus the most frequent item, which is unlikely to be true. Then $2k$ itemsets with highest guessing frequencies are selected to construct IS. The intuition is that, it is very unlikely that a frequent itemset is composed of several infrequent items (while it is theoretically possible). The guessing frequency is thus an effective measurement of the likelihood each itemset is among the frequent ones.

Formally, in SVSM, the domain IS is constructed as

$$IS \coloneqq \{\mathbf{x} : \mathbf{x} \subseteq S', 1 < |\mathbf{x}| < \log_2 k, \prod_{x \in \mathbf{x}} \Phi'(x) > t\},$$

where t is choosen so that $|IS| = 2k$.

Mining Frequent Itemset. After the domain IS is defined, the following protocol works similar to SVIM for frequent item mining. Note that step 1 is not necessary since IS is already small. For each user with value $\mathbf{v}$, a set of values from the domain IS is obtained first:

$$\mathbf{vs} = \{\mathbf{x} : \mathbf{x} \in IS, \mathbf{x} \subseteq \mathbf{v}\}$$

such that each itemset $\mathbf{x} \in \mathbf{vs}$ is a value in IS.

Then a group of users report the size of their $\mathbf{vs}$'s with FO:

$$\Psi_{\mathsf{OLH}(\epsilon)}(|\mathbf{vs}|).$$

After the aggregator evaluates the number of users that has ℓ itemsets for each $\ell \in [1, 2, \ldots, 2k]$, the aggregator finds the

90 percentile L and send it to the users in the final group, who then reports **vs** by

$$\Psi_{\mathsf{PSFO}(L,\mathsf{Adap},\epsilon)}(\mathbf{vs}).$$

The aggregator obtains the estimates by evaluating

$$\Phi_{\mathsf{PSFO}(L,\mathsf{Adap},\epsilon)}(\mathbf{x}) \cdot u'(L)$$

for any itemset $\mathbf{x} \in IS$, where $u'(L)$ is the update factor used for correcting bias (same format as (10)), and get results for the heavy itemsets and their estimates.

VI. Evaluation

Now we discuss experiments that evaluate different protocols. Basically, we want to answer the following questions: First, how many frequent items and itemsets can be effectively identified. Second, how much do our proposed protocols improve over existing ones.

As a highlight, in the POS dataset, our protocols can correctly identify around 45 frequent items (while existing ones can identify around 12), with much more accurate estimates (error is 3 orders of magnitudes less).

A. Experimental Setup

Environment. All algorithms are implemented in Python 2.7 and all the experiments are conducted on an Intel Core i7-4790 3.60GHz PC with 16GB memory. Each experiment is run 10 times, with mean and standard deviation reported.

Datasets. We run experiments on the following datasets:

- POS: A dataset containing merchant transactions of half a million users and 1657 categories.
- Kosarak: A dataset of click streams on a Hungarian website that contains around one million users and 42 thousand categories.
- Online: Similar to POS dataset, this is a dataset that contains merchant transactions of half a million users and 2603 categories.
- Synthesize: The dataset is generated by the IBM Synthetic Data Generation Code for Associations and Sequential Patterns 1.8 million transactions was generated, with 1000 categories. The average transaction size is 5.

For brevity, we only plot results for the one dataset (POS). The detailed results for other datasets are deferred to the appendix.

Metrics. To measure utility, we use the following metrics. Define x_i as the i-th most frequent value (x_i is an item in the task of item mining and an itemset in itemset mining). Let the ground truth for top k values as $\mathbf{x}_t = \{x_1, x_2, \ldots, x_k\}$. Denote the k values identified by the protocol using $\mathbf{x}_r$. Then $\mathbf{x}_t \cap \mathbf{x}_r$ is the set of real top-k values that are identified by the protocol.

1. Normalized Cumulative Rank (NCR). For each value x, we assign a quality function $q(\cdot)$ to each value, and use the Normalized Cumulative Gain (NCG) metric [22]:

$$\mathsf{NCG} = \frac{\sum_{x \in \mathbf{x}_r} q(x)}{\sum_{x \in \mathbf{x}_t} q(x)}.$$

We instantiate the quality function using x's rank as follows: the highest ranked value has a score of k (i.e., $q(x_1) = k$), the next one has score $k-1$, and so on; the k-th value has a score of 1, and all other values have scores of 0. To normalize this into a value between 0 and 1, we divide the sum of scores by the maximum possible score, i.e., $\frac{k(k+1)}{2}$. This gives rise to what we call the Normalized Cumulative Rank (NCR); this metric uses the true rank information of the top-k values.

2. Squared Error (Var): We measure the estimation accuracy by squared errors. That is,

$$\mathsf{Var} = \frac{1}{|\mathbf{x}_t \cap \mathbf{x}_r|} \sum_{x \in \mathbf{x}_t \cap \mathbf{x}_r} \left(f_x \cdot n - \Phi(x)\right)^2,$$

Note that we only account heavy hitters that are successfully identified by the protocol, i.e., $x \in \mathbf{x}_t \cap \mathbf{x}_r$.

B. Evaluation of Item Mining

For the item mining problem, our main focus is to compare the performance of our proposed method SVIM, and the existing method, LDPMiner. We implemented them as follows:

LDPMiner is almost implemented as described in [30]. For a fair comparison, we made two modifications. First, we partition the users into two groups. The first group focus on finding S, while the second focus on estimating S. Users in each group use the full privacy budget ϵ to report. It is proven [36] that by this way, the overall utility is better, compared to keeping asking all the users multiple questions, with splited privacy budget. Second, to get the 90th percentile L, an additional group of users are assigned to report the size of their private set. As a result, there will be three groups, 10% of users report size in advance, 40% report in the first phase, and 50% report in the second phase.

For SVIM, we do the similar thing. Half of the users report based on the original itemsets to find the candidate set S, and the other half report after seeing the candidate set to estimate S. The difference is, the 90th percentile L is used when estimating S. Therefore, 10% of all users are allocated to estimate L from the second half. That is, 50% report in the first phase, 10% of users report size of the their itemsets intersected with S, and 40% report one actual item.

To demonstrate the precise effect of each design detail, we also line up several intermediate protocols between LDPMiner and SVIM. We present them with synonyms (that specify the FO and ℓ used in both tasks) to highlight the difference as follows:

- $(\mathsf{BLH}, L), (\mathsf{SUE}, 2k)$: LDPMiner. LDPMiner uses two FO's BLH [5] and SUE [18]. It is proven in [36] that the two performs not as good as OLH.
- $(\mathsf{OLH}, L), (\mathsf{OLH}, 2k)$: The frequency oracles are replaced with OLH.
- $(\mathsf{OLH}, 1), (\mathsf{OLH}, 2k)$: The first phase uses $\ell = 1$. Note that L is no longer needed, so there are two groups each consists of half of the users.
- $(\mathsf{OLH}, 1), (\mathsf{Adap}, 2k)$: The second phase uses adaptive frequency oracle.

- $(\mathsf{OLH}, 1), (\mathsf{Adap}, L)$: The second phase uses L. An extra group of 10% of users are assigned to estimate that.
- $(\mathsf{OLH}, 1), (\mathsf{Adap}, L)(c)$: The final results are updated based on the length distribution. This is the SVIM.

Note that the allocation of 10% of users for length distribution is not fully justified. This is because the optimal allocation depends heavily on the dataset, and 10% seems a reasonable choice.

Detailed Results. In Figure 3, we evaluate the above six protocols on POS dataset, and plot the NCR and Var scores. Overall, the identification accuracy (indicated by NCR) increases with ϵ, and decreases with k. Similarly, the estimation accuracy becomes better (as the indicator Var decreases) when ϵ is larger, and worse (Var increases) if k is larger. Now we analyze performance of each competitor in more detail.

1. $(\mathsf{BLH}, L), (\mathsf{SUE}, 2k) \rightarrow (\mathsf{OLH}, L), (\mathsf{OLH}, 2k)$: First of all, we observe the identification accuracy improves when the FO in the first phase is changed from BLH to OLH. This happens because, by using OLH, a more accurate S will be returned, and by using OLH in the second phase, one can better identify the top k items. Note that the estimation accuracy actually does not improve significantly, because better FO does a better job at reducing the noise for the lower ranked values (thus NCR is higher). The estimation improvement is nearly unnoticeable in the log based figures.

2. $(\mathsf{OLH}, L), (\mathsf{OLH}, 2k) \rightarrow (\mathsf{OLH}, 1), (\mathsf{OLH}, 2k)$: One major NCR improvement happens when the length limit is changed from the 90th percentile L to 1. To this point, the top $2k$ items returned by the first phase contains most of the top k items. The NCR bottle neck lies on the second phase, which cannot effectively identify the top k from the $2k$ items. Note that the estimation accuracy does not improve because the same FO is used in the second phase.

3. $(\mathsf{OLH}, 1), (\mathsf{OLH}, 2k) \rightarrow (\mathsf{OLH}, 1), (\mathsf{Adap}, 2k)$: The most significant improvement happens when changing from OLH to Adap in the second phase. Both identification and estimation accuracy significantly (NCR almost doubled, and Var reduced by two magnitudes). This is because Adap significantly reduces the variance (from a factor of $(2k)^2$ to $2k$).

4. $(\mathsf{OLH}, 1), (\mathsf{Adap}, L)$ and $(\mathsf{OLH}, 1), (\mathsf{Adap}, L)(c)$: By reducing $2k$ to the 90th percentile L in the second phase, the results are further improved. Note that the improvement is not that significant but still meaningful. This is partly because an additional 10% of users are assigned to estimate the size distribution (to find L and update the results).

Remark. Because of the noisy nature (noise is in the order of $\sqrt{n}$) of the local setting of DP, in order to get meaningful information, one has to increase ϵ or n (or both). When the number of users is not sufficiently large, as in our experiment, the improvement is not significant in the small ϵ range, as being used by experiments of centralized DP (e.g., 0.1). However, in the case of deployed LDP protocol (Google uses $\epsilon > 4$ [18], and Apple uses $\epsilon = 1$ or 2 [32]), the advantage of the proposed protocol is profound.

C. Evaluation of Itemset Mining

We evaluate the effectiveness of SVSM. We want to answer the questions how many itemsets can be identified, and the effectiveness of using SVIM in SVSM.

We implement SVSM as follows, half of the users are allocated to find frequent items first. Then the set IS is constructed and estimated, by taking each of the element of it as an independent item. To compare the effect of SVIM over LDPMiner, we also instantiate SVSM using LDPMiner. Specifically, half of the users are allocated to find frequent items using LDPMiner; then IS is constructed similarly; finally, Phase 2 of LDPMiner is executed to estimate frequency of IS and output the most frequent k itemsets. Note that the $50\% - 50\%$ allocation is used since mining singletons and itemsets are two goals. One can allocate more users to singletons if singleton mining is more important.

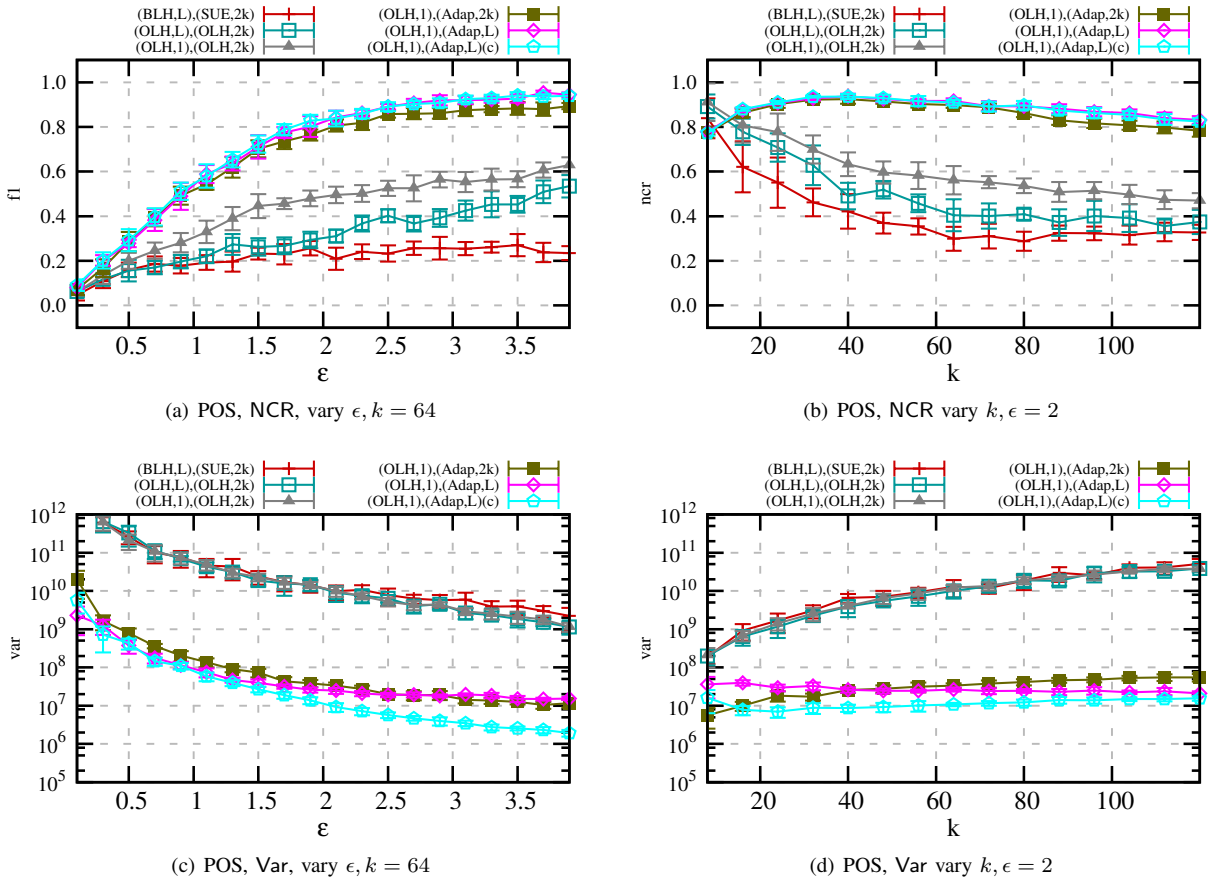
Detailed Results. Figure 4 shows the results of mining frequent itemsets. As we can see from the upper two subfigures, when fixing $k = 64$, the proposed SVSM protocol (instantiated with SVIM, as default) can achieve the NCR score of 0.7 at $\epsilon = 1$ and 0.9 when $\epsilon = 2$. As to when LDPMiner is used to instantiate SVSM, the utility drops to around 0.2. When ϵ is fixed at 2, the improvement of SVIM over LDPMiner is also significant, especially when k is greater than 64 (SVSM-SVIM keeps NCR greater than 0.8, while NCR for SVSM-LDPMiner drops to below 0.2). This suggests that SVSM with LDPMiner can effectively find only around 10 most frequent itemsets, while SVSM with SVIM can find around 70, demonstrating a $7\times$ improvement.

For the estimation accuracy shown by the bottom two subfigures, we can see that the estimation error drops with ϵ, and increases with k. When using LDPMiner in SVSM the error is two magnitudes greater than using SVIM. This effect is more significant when k is greater than 64. This is because Var for LDPMiner is heavily dependent on k, while SVIM not.

VII. Related Work

Differential privacy has been the *de facto* notion protecting privacy. In the centralized settings, many DP algorithms have been proposed (see [17], [35] for theoretical treatments and [24] in a more practical perspective). Recently, Uber has deployed a system enforcing DP during SQL queries [23], Google also proposed several works that combine DP with machine learning, e.g., [29]. In the local setting, we have also seen real world deployment: Google deployed RAP [18] as an extension within Chrome, and Apple [34], [33] also uses similar methods to help with predictions of spelling and other things.

Of all the problems, one basic mechanism in LDP is to estimate frequencies of values. Wang et al. compare different mechanisms using estimation variance [36]. They conclude that when the domain size is small, the Generalized Random Response provides best utility, and Optimal Local Hash (OLH)/Optimal Unary Encoding (OUE) [36] perform better when the domain is large. There also exist other mechanisms with higher variance: Binary Local Hash (BLH) by Bassily



(a) POS, NCR, vary $\epsilon, k = 64$

(b) POS, NCR vary $k, \epsilon = 2$

(c) POS, Var, vary $\epsilon, k = 64$

(d) POS, Var vary $k, \epsilon = 2$

Fig. 3. Singleton identification.

and Smith [5] can be viewed as OLH with hash range always equal to 2 and RAPPOR (SUE) by Erlingsson et al. [18], whose simple version can be viewed as OUE with suboptimal parameters. These protocols use ideas from earlier work [27], [13].

The frequent itemset mining problem is to identify the frequent set of items that appear simultaneously where each user has a set of items. There exist protocols to handle this in the classic DP setting [39], [26]. In the local setting, Evfimievski et al. [19] considered an easier setting where each user has a fixed amount of items. The protocol cannot be applied for the general itemset problem. Qin et al. proposed LDPMiner [30] that finds only the frequent singletons. In this paper, we propose and optimize PSFO, and thus be able to identify both singletons and itemsets effectively.

Besides the frequent itemset mining problem, there are other problems in the LDP setting that rely on mechanisms for frequency estimation. The problem of finding heavy hitters in a very large domain was exhaustively investigated [21], [27], [20], [5], [33], [6], [37], [10].

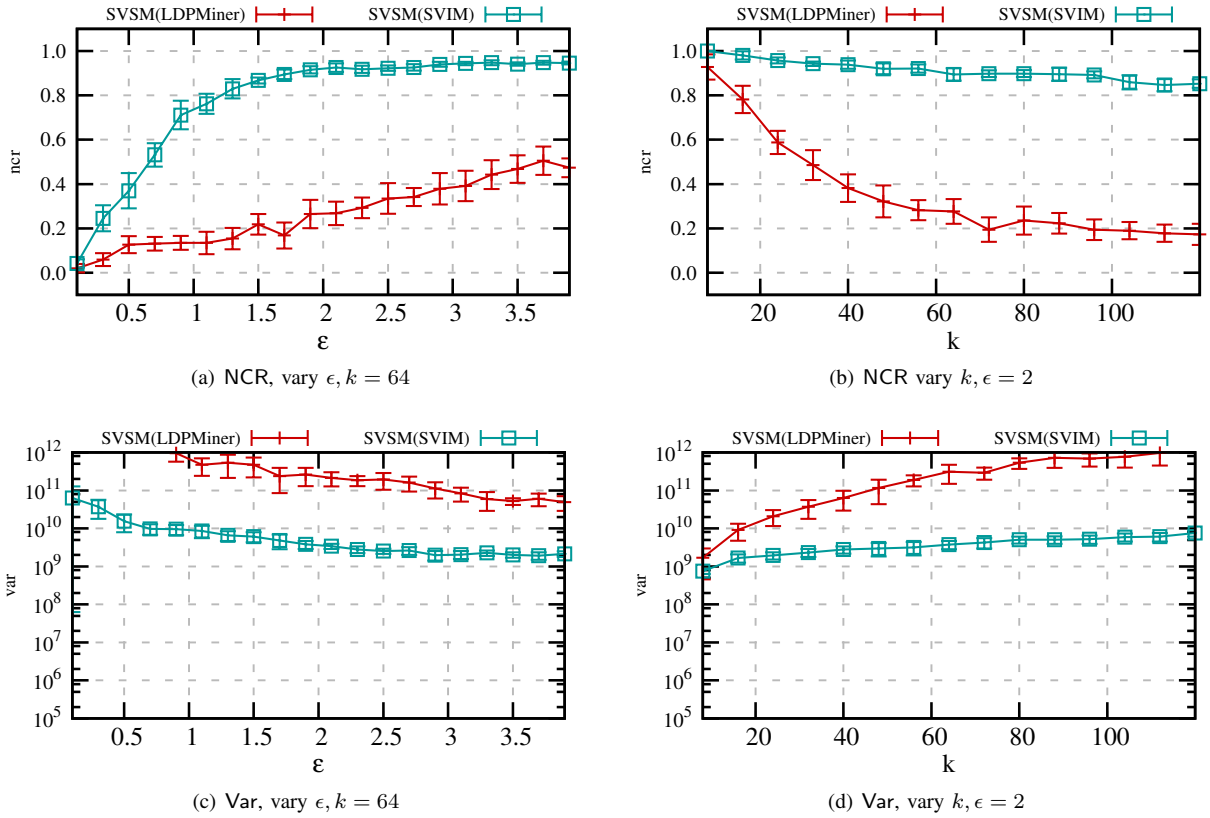
Nguyên et al. [28] studied the problem of empirical risk minimization. Smith et al. [31] also propose a protocol for the same problem but without interaction. Ding [12] and Nguyên et al. [28] studied mean estimation in the setting where the private values are continuous.

VIII. Conclusions

In this paper, we investigate the LDP protocols in a setting where each user has a set of items. We introduce PSFO, that enables users to sample one item from the set to report. The utility of PSFO is thoroughly analyzed and optimized, resulting two key observations: First, we identify the additional privacy gain provided by the sampling step, which we call privacy amplification effect; Second, we observe that the padding size ℓ should be small when domain size is large, and ℓ should be large when domain is small. Based on the analysis, we propose SVIM that significantly outperforms the existing protocol LDPMiner. Then we propose SVSM to find frequent itemset, which is an open problem in [30], for the first time. We demonstrate the effectiveness of SVIM and SVSM using empirical experiment on both synthetic and real-world datasets.

Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments. The work is supported in part by a Purdue Research Foundation award, and the NSF grant No. 1640374.



(a) NCR, vary $\epsilon, k = 64$

(b) NCR vary $k, \epsilon = 2$

(c) Var, vary $\epsilon, k = 64$

(d) Var, vary $k, \epsilon = 2$

Fig. 4. POS Itemset Mining Results.

REFERENCES

[1] Mozilla governance: Usage of differential privacy & rappor. https://groups.google.com/forum/#!topic/mozilla.governance/81gMQeMEL0w.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[3] E. Adar, D. S. Weld, B. N. Bershad, and S. S. Gribble. Why we search: visualizing and predicting user behavior. In *Proceedings of the 16th international conference on World Wide Web*, pages 161–170. ACM, 2007.

[4] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[5] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 127–135. ACM, 2015.

[6] R. Bassily, U. Stemmer, A. G. Thakurta, et al. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2285–2293, 2017.

[7] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, pages 503–512, 2010.

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

[9] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Acm Sigmod Record*, volume 26, pages 265–276. ACM, 1997.

[10] M. Bun, J. Nelson, and U. Stemmer. Heavy hitters and the structure of local privacy. *arXiv preprint arXiv:1711.04740*, 2017.

[11] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies*, volume 7384, pages 140–159. Springer, 2012.

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

[13] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438, 2013.

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

[15] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

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

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

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

[19] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222. ACM, 2003.

[20] G. Fanti, V. Pihur, and Ú. Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies (PoPETS)*, issue 3, 2016, 2016.

[21] J. Hsu, S. Khanna, and A. Roth. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2012.

[22] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[23] N. Johnson, J. P. Near, and D. Song. Practical differential privacy for sql queries using elastic sensitivity. *arXiv preprint arXiv:1706.09479*, 2017.
[24] N. Li, M. Lyu, D. Su, and W. Yang. *Differential Privacy: From Theory to Practice*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan Claypool, 2016.
[25] N. Li, W. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *ASIACCS*, 2012.
[26] N. Li, W. H. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. *VLDB*, 5(11):1340–1351, 2012.
[27] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 143–152. ACM, 2006.
[28] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053*, 2016.
[29] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with pate. 2018.
[30] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*, 2016.
[31] A. Smith, A. Thakurta, and J. Upadhyay. Is interaction necessary for distributed private learning? In *IEEE Symposium on Security and Privacy*, 2017.
[32] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753*, 2017.
[33] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudiger, V. R. Sridhar, and D. Davidson. Learning new words, Mar. 14 2017. US Patent 9,594,741.
[34] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudinger, V. V. Prakash, A. Legendre, and S. Duplinsky. Emoji frequency detection and deep link frequency, July 11 2017. US Patent 9,705,908.
[35] S. Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.
[36] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *USENIX'17: Proceedings of 26th USENIX Security Symposium on USENIX Security Symposium*. USENIX Association, 2017.
[37] T. Wang, N. Li, and S. Jha. Locally differentially private heavy hitter identification. *arXiv preprint arXiv:1708.06674*, 2017.
[38] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
[39] C. Zeng, J. F. Naughton, and J.-Y. Cai. On differentially private frequent itemset mining. *Proceedings of the VLDB Endowment*, 6(1):25–36, 2012.

APPENDIX

A. (ϵ, δ)-LDP and Limited Amplification Effect

In (ϵ, δ)-LDP, the value δ (which is typically very small) has an intuitive interpretation of "failure" probability. That is, with probability $1-\delta$, Ψ is ϵ-LDP. When $\delta = 0$, $(\epsilon, 0)$-LDP becomes ϵ-LDP.

Definition 4 ((ϵ, δ) Local Differential Privacy)**.** *An algorithm Ψ satisfies (ϵ, δ)-local differential privacy ((ϵ, δ)-LDP), where $\epsilon \geq 0$, and $0 \leq \delta < 1$ if and only if for any input $\boldsymbol{v}_1, \boldsymbol{v}_2 \subseteq I$, we have*

$$\forall T \subseteq Range(\Psi): \ \mathsf{Pr}\left[\Psi(\boldsymbol{v}_1) \in T\right] \leq e^{\epsilon} \mathsf{Pr}\left[\Psi(\boldsymbol{v}_2) \in T\right] + \delta,$$

where $Range(\Psi)$ denotes the set of all possible outputs of the algorithm Ψ.

To apply the privacy amplification, one uses δ to measure the probability that failure (multiple values are hashed to the same value) happens, and derive the corresponding ϵ' that OLH can use. For example, when $\ell = 2$, the probability both the user's items are hashed into the same value by the chosen hash function is $\delta = \frac{1}{g}$, where $g = \lceil e^{\epsilon'} + 1 \rceil$ is the range of the hash function. Under the condition the user's items are hashed to at least two results, OLH can be used with $\epsilon' = \ln(2e^{\epsilon} - 1)$.

Theorem 5 ((ϵ, δ)-LDP by $\mathsf{OLH}(\epsilon')$)**.** *$\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}_{\ell}(\cdot))$ satisfies (ϵ, δ)-LDP, where $\epsilon' = \ln\left(\frac{\ell}{\ell'} \cdot (e^{\epsilon} - 1) + 1\right)$, and ℓ' is an integer such that*

$$\binom{\ell}{\ell'+1} \cdot \frac{1}{\lceil \frac{\ell}{\ell'} \cdot (e^{\epsilon} - 1) + 2 \rceil^{\ell'}} \leq \delta.$$

That is, for any $\epsilon \geq 0$, any input $\boldsymbol{v}_1, \boldsymbol{v}_2 \subseteq I$, and any set of possible output $T \subseteq Range(\Psi_{\mathsf{PSFO}(\ell, \mathsf{GRR}, \epsilon')})$,

$$\begin{aligned} &\mathsf{Pr}\left[\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}(\boldsymbol{v}_1, \ell)) \in T\right] \\ &\leq e^{\epsilon} \cdot \mathsf{Pr}\left[\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}(\boldsymbol{v}_2, \ell)) \in T\right] + \delta. \end{aligned} \tag{11}$$

Proof. To prove (11), it is equivalent to first prove that a "failure" event, where more than ℓ' items in $\mathbf{v}_1$ are hashed to the same value, happens with probability less than δ, and then prove that under the condition the "failure" event does not happen, $\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}_{\ell}(\cdot))$ satisfies ϵ-LDP.

Given that the hash function is chosen randomly, and the hash family is random, bounding the "failure" probability is equivalent to bounding the probability of throwing ℓ balls randomly into g bins, and the max load is more than ℓ'. The probability can be calculated as follows:

Let $E_{i,a}$ be the event that bin i contains more than a balls, then

$$\mathsf{Pr}\left[E_{i,a}\right] = \binom{\ell}{a+1} \frac{1}{g^{a+1}}$$

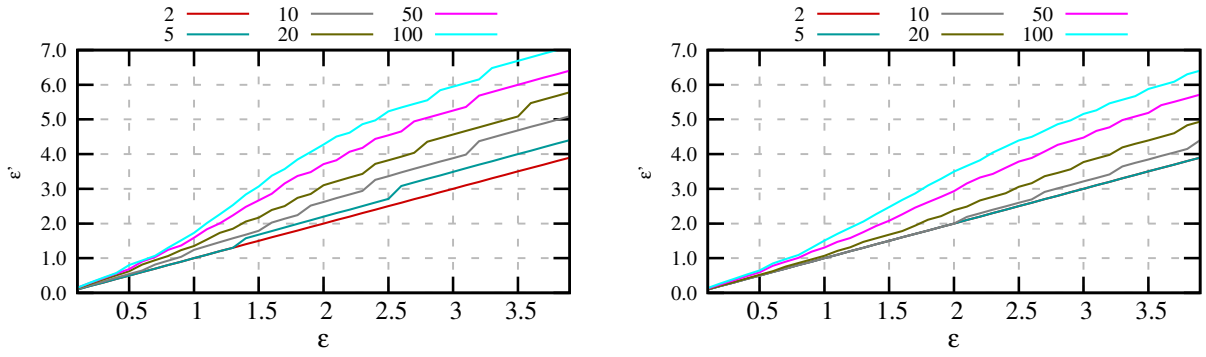
By union bound, we know that

$$\delta = \mathsf{Pr}\left[\bigcup_{i \in [g]} E_{i,\ell'}\right] \leq \sum_{i} \mathsf{Pr}\left[E_{i,\ell'}\right] = \binom{\ell}{\ell'+1} \frac{1}{g^{\ell'}}$$

where $g = \lceil e^{\epsilon'} + 1 \rceil = \lceil \frac{\ell}{\ell'} \cdot (e^{\epsilon} - 1) + 2 \rceil$.

Now it suffices to prove that for any $\epsilon \geq 0$, any $\mathbf{v}_1, \mathbf{v}_2 \subseteq I$, any possible hash function H (such that at most ℓ' items are hashed into the same value), and any $t \in [g]$, $\frac{p_1}{p_2} \leq e^{\epsilon}$, where

$$\begin{aligned} p_1 &= \mathsf{Pr}\left[\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}_{\ell}(\mathbf{v}_1)) = \langle H, t \rangle\right], \text{ and} \\ p_2 &= \mathsf{Pr}\left[\Psi_{\mathsf{OLH}(\epsilon')}(\mathsf{PS}_{\ell}(\mathbf{v}_2)) = \langle H, t \rangle\right]. \end{aligned}$$



(a) Amplification of OLH with $\delta = 10^{-3}$

(b) Amplification of OLH with $\delta = 10^{-9}$

Fig. 5. Privacy amplification effect for different ℓ.

We first upper bound p_1,

$$\begin{aligned}
p_1 =& \Pr[H \text{ is picked}] \cdot \Pr\left[\Psi_{\mathsf{GRR}(\epsilon)}(H(\mathsf{PS}_\ell(\mathbf{v}_1))) = t | H\right] \\
=& \Pr[H \text{ is picked}] \cdot \Big(\Pr[v \text{ is sampled } \wedge H(v) = t]\, p' \\
&+ \Pr[v \text{ is sampled } \wedge H(v) \neq t]\, q'\Big) \\
\leq& \Pr[H \text{ is picked}] \cdot \Big(\frac{\ell'}{\max\{|\mathbf{v}_1|, \ell\}} \cdot p' \\
&+ \frac{\max\{|\mathbf{v}_1|, \ell\} - \ell'}{\max\{|\mathbf{v}_1|, \ell\}} \cdot q'\Big)
\end{aligned}$$

The equality holds when $H(v) = t$ for all $\mathbf{v}_1$. Similarly, we lower bound p_2,

$$\begin{aligned}
p_2 =& \Pr[H \text{ is picked}] \cdot \Pr\left[\Psi_{\mathsf{GRR}(\epsilon)}(H(\mathsf{PS}_\ell(\mathbf{v}_2))) = t | H\right] \\
=& \Pr[H \text{ is picked}] \cdot \Big(\Pr[v \text{ is sampled } \wedge H(v) = t]\, p' \\
&+ \Pr[v \text{ is sampled } \wedge H(v) \neq t]\, q'\Big) \\
\geq& \Pr[H \text{ is picked}] \cdot \Big(\frac{0}{\max\{|\mathbf{v}_1|, \ell\}} \cdot p' \\
&+ \frac{\max\{|\mathbf{v}_1|, \ell\}}{\max\{|\mathbf{v}_1|, \ell\}} \cdot q'\Big) = \Pr[H \text{ is picked}] \cdot q'
\end{aligned}$$

The equality holds when none of the items from $\mathbf{v}_2$ are hashed to t by H. Thus, we now bound $\frac{p_1}{p_2}$:

$$\begin{aligned}
\frac{p_1}{p_2} &\leq \frac{p'}{q'} \cdot \frac{\ell'}{\max\{|\mathbf{v}_1|, \ell\}} + \frac{\max\{|\mathbf{v}_1|, \ell\} - \ell'}{\max\{|\mathbf{v}_1|, \ell\}} \\
&= 1 + \frac{\ell'}{\max\{|\mathbf{v}_1|, \ell\}} \cdot \left(\frac{p'}{q'} - 1\right) \\
&\leq 1 + \frac{\ell'}{\ell} \cdot \left(e^{\epsilon'} - 1\right) \\
&= 1 + \frac{\ell'}{\ell} \cdot \left(\frac{\ell}{\ell'} \cdot (e^\epsilon - 1) + 1 - 1\right) = e^\epsilon.
\end{aligned}$$

The equality is achieved when $H(v) = t$ for all $\mathbf{v}_1$ while $H(v) \neq t$ for all $\mathbf{v}_2$.

□

ϵ \ ℓ	2	5	10	20	50	100
0.1	0.10	0.10	0.11	0.13	0.15	0.15
0.5	0.50	0.50	0.54	0.62	0.68	0.80
1.0	1.00	1.00	1.24	1.35	1.59	1.73
2.0	2.00	2.20	2.62	3.10	3.71	4.28
4.0	4.00	4.50	5.19	5.88	6.80	7.20
0.1	0.10	0.10	0.10	0.10	0.12	0.14
0.5	0.50	0.50	0.50	0.52	0.59	0.65
1.0	1.00	1.00	1.00	1.07	1.30	1.51
2.0	2.00	2.00	2.00	2.38	2.93	3.49
4.0	4.00	4.00	4.50	5.04	5.82	6.51

TABLE III
NUMERICAL VALUE OF ϵ' UNDER DIFFERENT ϵ AND ℓ. THE UPPER PART IS FOR $\delta = 10^{-3}$, AND THE LOWER PART IS FOR $\delta = 10^{-9}$.

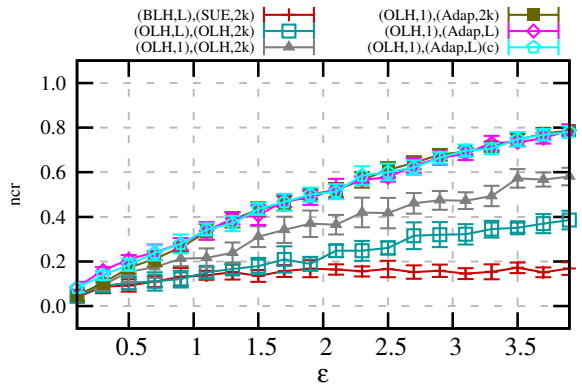
The theorem above gives us the formula to calculate δ and ϵ' for any ℓ'. Therefore, if δ is specified, we are able to come up with the highest ϵ'. Table III and Figure 5 give results of ϵ' given ϵ and ℓ, under the condition δ equals 10^{-3} and 10^{-9}, respectively. We can see $\epsilon' \geq \epsilon$, the difference becomes more significant when ϵ or ℓ is large. However, the increased amount is less than that for GRR, as shown in Table II and Figure 1.

Note that however, the (ϵ, δ)-LDP notion is strictly weaker (less secure) than ϵ-LDP and thus not directly comparable here.

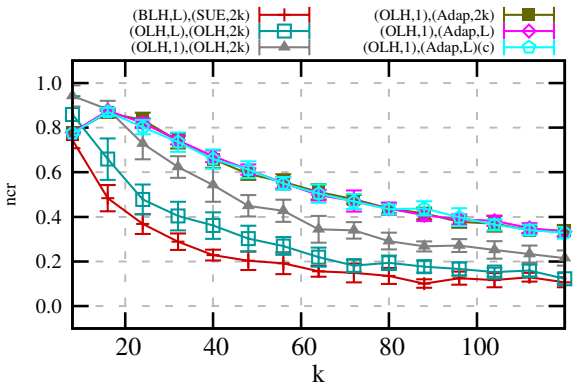
B. Additional Results

Item Mining. We report experimental results of item mining for the datasets of Kosarak, Online and Synthesize in Figures 6 and 7. We can see similar trends as that of Figure 3. Note that performance on different dataset is slightly different, because of different size, distribution, etc. Specifically, NCR and Var are worse in the Kosarak dataset, than that on the others, because the original domain is big (42 thousand, while the others are 1 to 3 thousand). Overall, the proposed method SVIM works persistently better than its competitors.

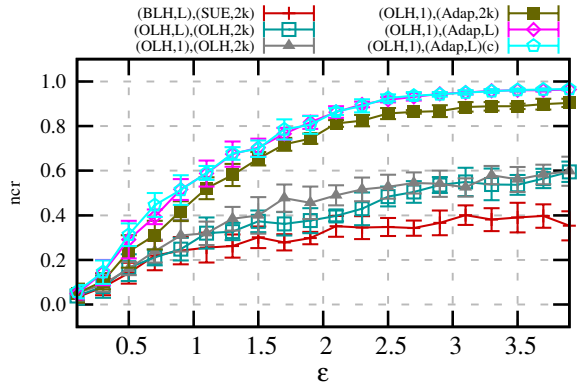
Itemset Mining. We also plot results for itemset mining in Figure 8. Results for the synthetic dataset is not included because there is no frequent itemset (the items from the generator are independent). For the others, we can still see similar trends and that our proposed solution works persistently better.



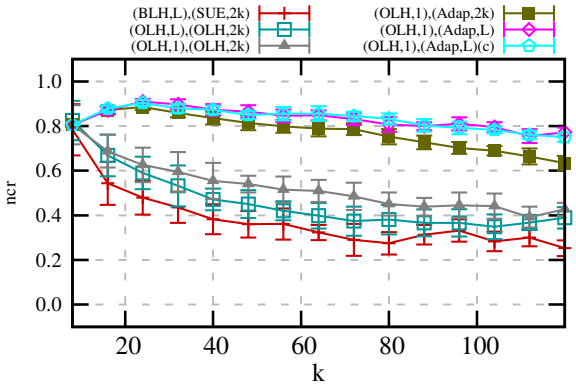
(a) Kosarak, NCR, vary $\epsilon, k = 64$



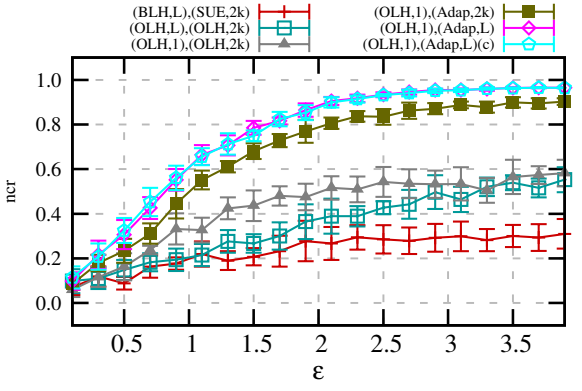
(b) Kosarak, NCR vary $k, \epsilon = 2$



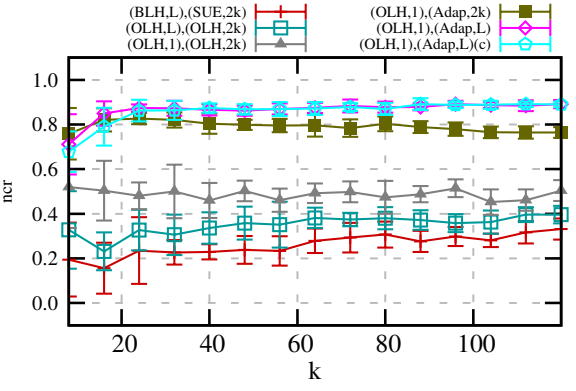
(c) Online, NCR, vary $\epsilon, k = 64$



(d) Online, NCR vary $k, \epsilon = 2$

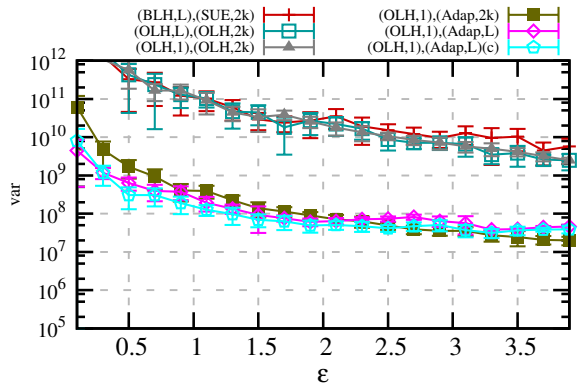


(e) Synthetic, NCR, vary $\epsilon, k = 64$

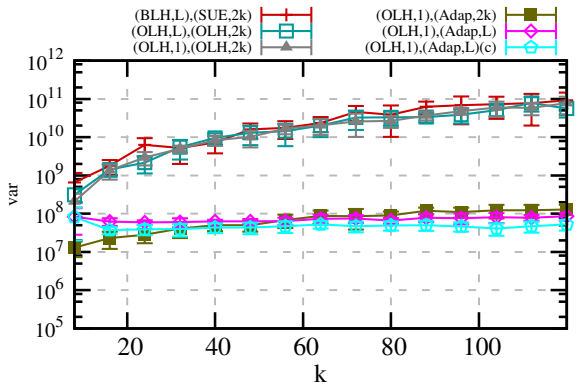


(f) Synthetic, NCR vary $k, \epsilon = 2$

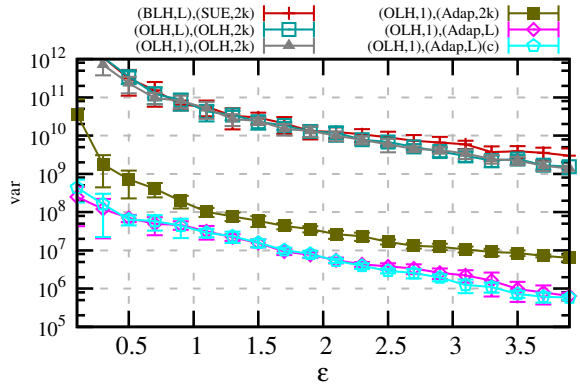
Fig. 6. More results on singleton identification.



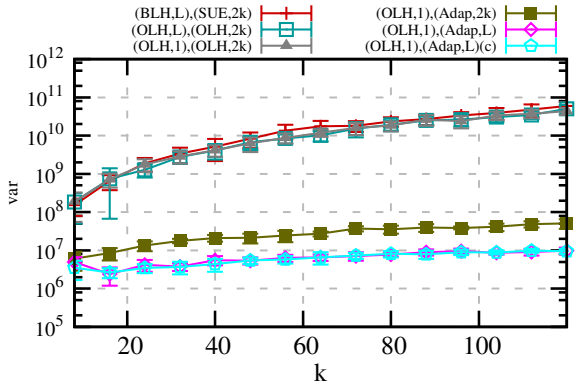
(a) Kosarak, Var, vary $\epsilon, k = 64$



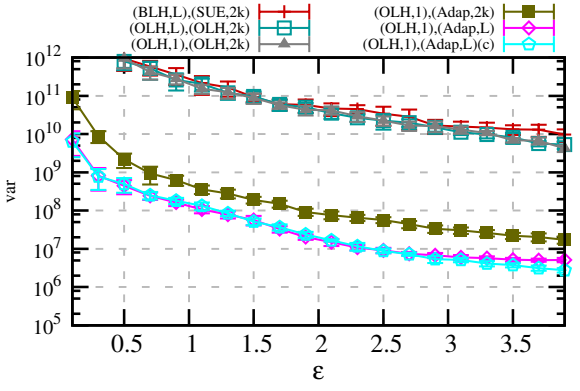
(b) Kosarak, Var vary $k, \epsilon = 2$



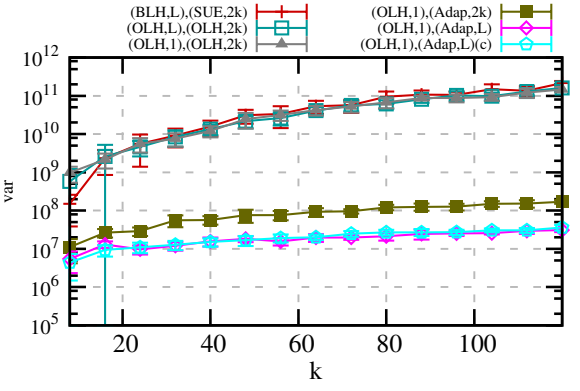
(c) Online, Var, vary $\epsilon, k = 64$



(d) Online, Var vary $k, \epsilon = 2$

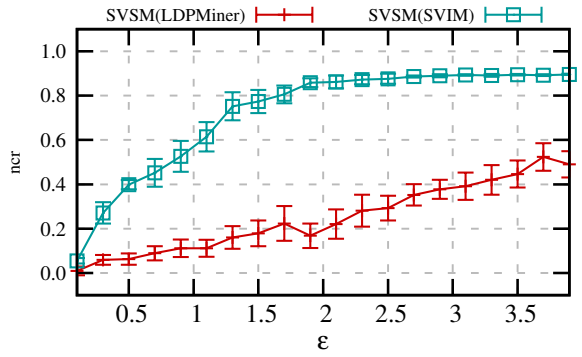


(e) Synthetic, Var, vary $\epsilon, k = 64$

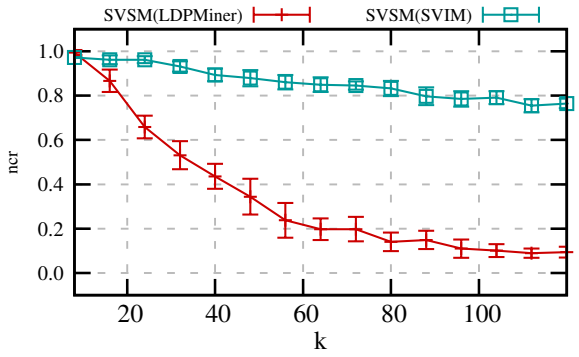


(f) Synthetic, Var vary $k, \epsilon = 2$

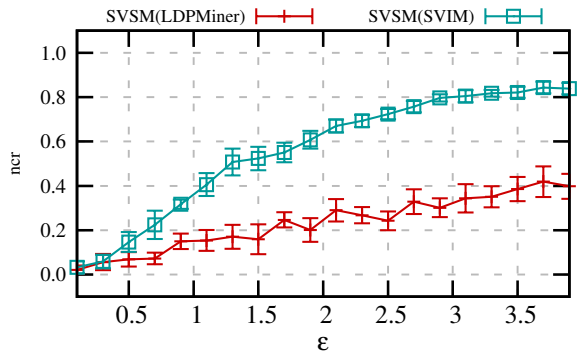
Fig. 7. More results on singleton estimation.



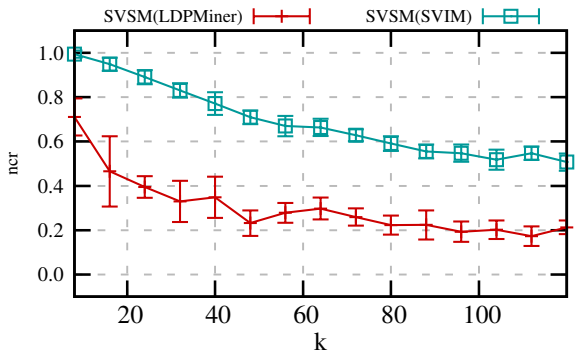
(a) Kosarak NCR, vary $\epsilon, k = 64$



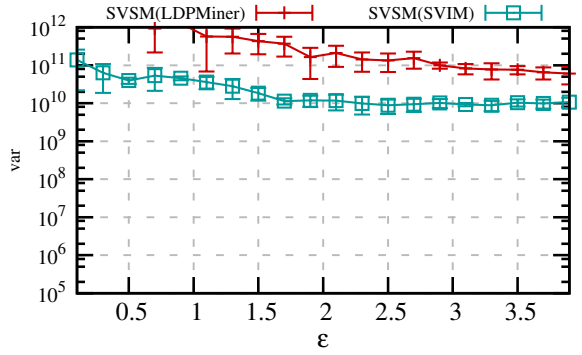
(b) Kosarak NCR vary $k, \epsilon = 2$



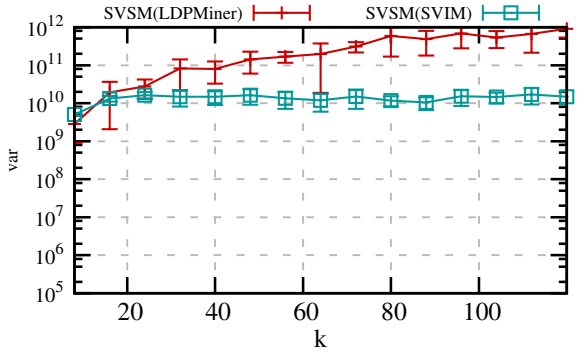
(c) Online NCR, vary $\epsilon, k = 64$



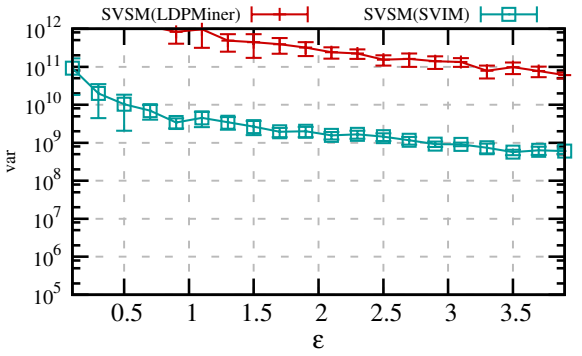
(d) Online NCR vary $k, \epsilon = 2$



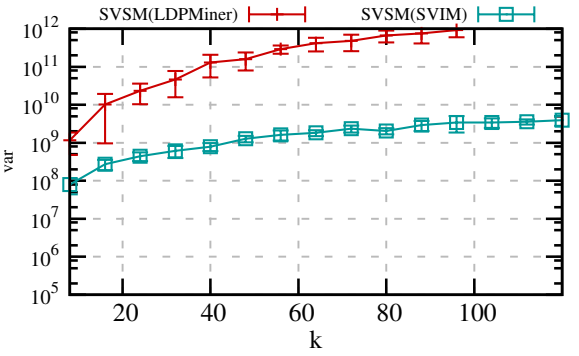
(e) Kosarak Var, vary $\epsilon, k = 64$



(f) Kosarak Var, vary $k, \epsilon = 2$



(g) Online Var, vary $\epsilon, k = 64$



(h) Online Var, vary $k, \epsilon = 2$

Fig. 8. More results on itemset mining results for Kosarak dataset.