On Differentially Private Frequent Itemsets Mining

Chen Zeng
University of Wisconsin-Madison
zeng@cs.wisc.edu

Jeffrey F. Naughton
University of Wisconsin-Madison
naughton@cs.wisc.edu

Jin-Yi Cai
University of Wisconsin-Madison
jyc@cs.wisc.edu

ABSTRACT

Frequent itemsets mining finds sets of items that frequently appear together in a database. However, publishing this information might have privacy implications. Accordingly, in this paper we are considering the problem of guaranteeing differential privacy for frequent itemsets mining. We measure the utility of a frequent itemsets mining algorithm by its likelihood to produce a complete and sound result where "completeness" requires the algorithm to include the "sufficiently" frequent itemsets and "soundness" needs the algorithm to exclude the "sufficiently" infrequent ones. We prove that it is hard to simultaneously satisfy both differential privacy and a non-trivial utility requirement. However, we find that we can produce reasonably accurate results while still guaranteeing differential privacy on benchmark datasets [19] by truncating transactions to limit their cardinality.

1. INTRODUCTION

Recently, concomitant with the increasing ability to collect personal data, privacy has become a major concern. In this paper, we focus on privacy issues that arise in the context of finding frequent itemsets in "transactional" data. Frequent itemsets mining is widely used in many applications, perhaps the best known of which is market basket analysis. The goal of frequent itemsets mining in market basket analysis is to find sets of items that are frequently bought together, which is helpful in applications ranging from product placement to marketing and beyond. The problem of developing efficient algorithms for frequent itemsets mining has been widely studied by our community [13]. However, the privacy issues arising in frequent itemsets mining have received little attention.

A frequent itemsets mining algorithm takes as input a dataset consisting of the transactions by a group of individuals, and produces as output the frequent itemsets. This immediately creates a privacy concern — how can we be confident that publishing the frequent itemsets in the dataset does not reveal private information about the individuals whose data is being studied? This problem is compounded by the fact that we may not even know what data the individuals would like to protect nor what background information might be possessed by an adversary. Fortunately, a possible answer to that challenge is presented by *differential privacy* [7], which intuitively guarantees that the presence of an individual's data in a dataset does not reveal much about that individual. In this paper, we explore the possibility of developing differentially private frequent itemsets mining algorithms. Our goal is to guarantee differential privacy while still finding useful frequent itemsets. To the best of our knowledge, ours is the first paper to explore differential privacy in this context.

An obvious but important observation is that privacy is just one aspect of the problem; utility also matters. In this paper, we quantify the utility of a differentially private frequent itemsets mining by its likelihood to produce a complete and sound result. Intuitively speaking, "completeness" requires an algorithm to include the sufficiently "frequent" itemsets, and "soundness" needs an algorithm to exclude the sufficiently "infrequent" ones. We start by showing the tradeoff between privacy and utility in frequent itemsets mining. Our result indicates that no matter how sophisticated a differentially private frequent itemsets mining algorithm is, it has to suffer from a huge risk of revealing an individual's information in order to satisfy a non-trivial utility requirement.

In view of that result, we utilize the constraint on each transaction's cardinality in databases to promote the utility of a frequent itemsets mining algorithm while ensuring differential privacy. We first consider a constrained frequent itemsets mining problem — frequent 1-itemsets mining in which we are only interested in the itemsets of cardinality 1. We prove that by limiting the maximal cardinality of transactions, we can significantly promote the utility of a specific frequent 1-itemsets mining algorithm while still guaranteeing differential privacy. At the most abstract level, that algorithm works by adding "noise" in computing frequent itemsets. As we will prove, if this is done properly, that algorithm is differentially private.

Motivated by that theoretical result, we impose the cardinality constraint to a database by truncating transactions. However, that truncating approach has privacy implications since it needs to access the database. Fortunately, we can prove that as long as that truncating approach is "local", which takes as input a single transaction, and produces as an output only depending on the input, then applying any differentially private algorithm on the truncated database

also ensures differential privacy for the original database. Of course, by truncating transactions, we are adding a new kind of error by "throwing away" information that was in the original dataset. In an experimental study using three benchmark datasets [19], we find that this trade-off is worth making. In particular, we also evaluate our frequent 1-itemsets mining algorithm using an intuitive metric F-score, which measures the percentage of correct frequent 1-itemsets, and our results indicate that by truncating transactions, our algorithm gets much better F-score than the non-truncating algorithm.

Encouraged by that success, we generalize our idea of truncating transactions to frequent k-itemsets mining in which we are interested in itemsets of cardinality not exceeding k. However, as described in the body of the paper, that generalization is far from trivial since (a) the total number of itemsets is too large, and (b) the precise computation of the required "noise" to guarantee differential privacy is hard. Specifically, we first propose a naïve frequent k-itemsets mining algorithm which attacks (a) by exploiting the "a priori" property of frequent itemsets to reduce the number of computations, and (b) by computing the upper bound of the required "noise." However, we find the performance of our naïve algorithm is not satisfying in practice. To remedy that problem, we propose two heuristic methods to improve our naïve algorithm. We further extend our frequent k-itemsets mining algorithm to frequent itemsets mining by estimating the maximal cardinality of frequent itemsets in a differentially private way, and then applying our frequent k-itemsets mining algorithm by setting k to be that estimated maximal cardinality. In an experimental study using the benchmark datasets, we find that our differentially private frequent itemsets mining algorithm is able to generate reasonable accuracy on these test datasets.

The rest of the paper is organized as follows: Section 2 briefly describes the problem of frequent itemsets mining, and the notion of differential privacy. Section 3 quantifies the trade-off between utility and privacy in frequent itemsets mining. Section 4 proposes our differentially private frequent 1-itemsets mining algorithm. Section 5 generalizes the idea of truncating transactions to frequent k-itemsets mining. Section 6 extends our frequent k-itemsets mining algorithm to frequent itemsets mining. Section 7 evaluates our algorithm on benchmark datasets. Section 8 discusses related work, and Section 9 concludes our work. Most of the proofs and the pseudocode of some algorithms can be found in the long version of our paper.

2. PRELIMINARIES

In this section, we formulate the problem of frequent itemsets mining [1], and the notion of differential privacy [7].

2.1 Frequent Itemsets Mining

We model a database τ as a vector in D^m, where each entry represents the information contributed by an individual from the domain D. In our context, the database in frequent itemsets mining is called a *transaction database*.

DEFINITION 1. *(Transaction database): A* transaction database *is a vector of transactions $\tau = \langle t_1, \ldots, t_m \rangle$ where each transaction t_i is a subset of the alphabet $\mathcal{I} = \{1, \ldots, n\}$.*

The domain of a single transaction is thus the power set of the alphabet $\mathcal{I}$. In this paper, we use "database" as a shorthand for "transaction database." Each subset of the alphabet $\mathcal{I}$ is called an *itemset*. If the number of transactions containing an itemset exceeds a predefined threshold, then that itemset is called a *frequent itemset*.

DEFINITION 2. *(Frequent itemset): For any itemset X, the support of X in a database is the number of transactions containing X. If that number $X.supp$ exceeds a predefined threshold λ, then X is called a* frequent itemset *with respect to the threshold λ.*

In the rest of this paper, we assume the threshold λ is given, and use the term "frequent itemsets" as a shorthand for "frequent itemsets with respect to the threshold" when the threshold is clear from the context. For ease of presentation, we denote "the itemsets of cardinality k" by "k-itemsets", and the query that computes the support of a k-itemset by a "k-itemset query." In literature [8], a *count query* computes the number of entries in a database satisfying a given predicate, and a k-itemset query is basically a count query which computes the number of transactions containing the given itemset.

2.2 Differential Privacy

Speaking intuitively, *differential privacy* guarantees that whether or not an individual's information participates in the database has little effect on the output of an algorithm, and thus, an adversary can learn limited information about that individual from the output. In our context, the information contributed by an individual is her transaction. More precisely, for any database $\tau \in D^m$, let $nbrs(\tau)$ denote the set of *neighboring databases* of τ, each of which differs from τ by at most one *transaction*. *Differential privacy* requires that the probabilities of an algorithm to output the same result on any pair of neighboring databases are bounded by a constant ratio.

DEFINITION 3. *(ϵ-differential privacy [7]): For any input database τ, a randomized algorithm f is ϵ-differentially private iff for any $\mathcal{S} \subseteq Range(f)$, and any database $\tau' \in nbrs(\tau)$,*

$$\Pr(f(\tau) \in S) \leq e^{\epsilon} \times \Pr(f(\tau') \in \mathcal{S})$$

where Pr *is the probability taken over the coin tosses of the algorithm f.*

One way to guarantee differential privacy for a count query is to perturb the correct result. In particular, Ghosh et al. [10] propose the *geometric mechanism* to guarantee ϵ-differential privacy for a single count query, which adds noise Δ drawn from the two-sided geometric distribution $G(\epsilon)$ with the following probability distribution: for any integer σ,

$$\Pr(\Delta = \sigma) \sim e^{-\epsilon|\sigma|} \tag{1}$$

The geometric mechanism is a discrete variant of the simple and well-studied Laplacian mechanism [8] which adds random noise drawn from the Laplacian distribution. To ensure differential privacy for multiple count queries, we first compute the *sensitivity* of those queries, which is the largest difference between the outputs of those queries on any pair of neighboring databases.

DEFINITION 4. *(Sensitivity): Given d count queries,* $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$*, the* sensitivity *of* $\mathbf{q}$ *is:*

$$S_{\mathbf{q}} = \max_{\forall \tau, \tau' \in nbrs(\tau)} |\mathbf{q}(\tau) - \mathbf{q}(\tau')|_1$$

Notice that the output of $\mathbf{q}$ is a vector of dimension d, and we use $|\mathbf{x} - \mathbf{y}|_p$ to denote the L_p distance between two vectors $\mathbf{x}$ and $\mathbf{y}$. The following theorem is a straightforward extension of the Laplacian mechanism to the geometric mechanism.

THEOREM 1. *Given d count queries* $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$*, for any database* τ*, the database access mechanism:* $A_{\mathbf{q}}(\tau) = \mathbf{q}(\tau) + \langle \Delta_1, \ldots, \Delta_d \rangle$ *where* Δ_i *is drawn i.i.d from the geometric distribution* $G(\epsilon/S_{\mathbf{q}})$ (1)*, guarantees* ϵ*-differential privacy for* $\mathbf{q}$*.*

As proved in [8], a sequence of differentially private computations also ensures differential privacy. This is called the *decomposition property* of differential privacy as shown in Theorem 2.

THEOREM 2. *[8] Given a sequence of computations, denoted as* $\mathbf{f} = f_1, \ldots, f_d$*, if each computation* f_i *guarantees* ϵ_i*-differential privacy, then* $\mathbf{f}$ *is* $(\sum_{i=1}^{i=d} \epsilon_i)$*-differentially private.*

3. TRADE-OFF BETWEEN PRIVACY AND UTILITY

In this section, we present a theoretical study showing the trade-off between utility and privacy in frequent itemsets mining. First, we formally define the utility of a frequent itemsets mining algorithm.

3.1 Our Utility Model

We quantify the utility of a frequent itemsets mining algorithm in an approximate measurement similar to that in [4]. Intuitively speaking, if the support of an itemset is much larger than the threshold, then the result should include that itemset; on the other hand, if the support of an itemset is much smaller than the threshold, then that itemset should be excluded from the output. More precisely, we specify two criteria to capture that intuition as formulated in Definition 5.

DEFINITION 5. *(δ-approximation): Given a database* τ *and a threshold* λ*, let* $\mathcal{S}$ *be the output of a frequent itemsets mining algorithm on the database* τ*. We say* $\mathcal{S}$ *is* δ-approximate *iff the following two properties are satisfied:*

1. *(Completeness) Every itemset with support exceeding* $(1+\delta)\lambda$ *is in* $\mathcal{S}$*.*
2. *(Soundness) No itemset with support less than* $(1-\delta)\lambda$ *is in* $\mathcal{S}$*.*

We quantify the utility of a frequent itemsets mining algorithm by its likelihood to produce a good approximate result as shown in Definition 6.

DEFINITION 6. *($(\delta, \eta)-usefulness$): A frequent itemsets mining algorithm* f *is* (δ, η)-useful *iff for any database* τ*, with probability at least* $1 - \eta$*, the output of* f *on* τ *is* δ*-approximate.*

Both δ and η are within the range $(0, 1)$ by definition. Next, we quantify the trade-off between privacy and utility for frequent itemsets mining.

3.2 A Lower Bound on the Privacy Parameter

Let us first consider a constrained frequent itemsets mining problem in which we are only interested in frequent 1-itemsets. We call that problem *frequent 1-itemsets mining*. Lemma 1 shows the lower bound on the privacy parameter ϵ if a frequent 1-itemsets mining algorithm must be both (δ, η)-useful and ϵ-differentially private.

LEMMA 1. *For any frequent 1-itemsets mining algorithm that is both* ϵ*-differentially private and* (δ, η)*-useful,*

$$\epsilon \geq \frac{\ln[(2^n - 1)\eta']}{2\delta\lambda + 2}$$

where $\eta' = (1 - \eta)/\eta$*.*

Proof Sketch: Consider a database τ which contains $\lceil (1 - \delta)\lambda \rceil - 1$ transactions that are precisely the alphabet $\mathcal{I}$. Let f be a frequent 1-itemsets mining algorithm that is both (δ, η)-useful and ϵ-differentially private. It is not hard to show that $\emptyset$ is the only δ-approximate output of τ, and thus $\Pr(f(\tau) = \emptyset) \geq 1 - \eta$. We can also prove that for any non-empty set $\mathcal{S} \subseteq \mathcal{I}$, $\Pr(f(\tau) = \mathcal{S}) > 0$. Thus, there must exist a particular non-empty set $\mathcal{S}'$ such that $\Pr(f(\tau) = \mathcal{S}') \leq \eta/(2^n - 1)$, and we construct a database τ' by adding $2\lfloor \delta\lambda \rfloor + 2$ transactions which are exactly $\mathcal{S}'$ to τ. Since $\mathcal{S}'$ is the only δ-approximate output in τ', $\Pr(f(\tau') = \mathcal{S}') \geq 1 - \eta$. By differential privacy,

$$\Pr(f(\tau') = \mathcal{S}') \leq e^{(2\lfloor \delta\lambda \rfloor + 2)\epsilon} \Pr(f(\tau) = \mathcal{S}')$$

the lemma then follows. □

We observe that if an output of a frequent itemsets mining is δ-approximate, then that output is also δ-approximate for frequent 1-itemsets mining. Therefore, a (δ, η)-useful frequent itemsets mining algorithm is also a (δ, η)-useful frequent 1-itemsets mining algorithm. Thus, Lemma 1 quantifies the trade-off between privacy and utility in frequent itemsets mining. This is shown in Theorem 3.

THEOREM 3. *For any frequent itemsets mining algorithm that is both* ϵ*-differentially private and* (δ, η)*-useful,*

$$\epsilon \geq \frac{\ln[(2^n - 1)\eta']}{2\delta\lambda + 2}$$

where $\eta' = (1 - \eta)/\eta$*.*

In a typical transaction database like BMS-WebView-2 [19], $n = 3340$. Therefore, if we set $\lambda = 310$ which is 0.4% of the total number of transactions, $\delta = 0.2$ and $\eta = 0.5$, then $\epsilon \geq 18.4$. That lower bound shows that no matter how sophisticated a differentially private frequent itemsets mining algorithm is, in order to guarantee $(0.2, 0.5)$-usefulness, the privacy parameter ϵ must exceed 18.4, which suggests a huge risk of privacy breach. Therefore, instead of designing a sophisticated privacy mechanism for frequent itemsets mining, in this paper we explore another orthogonal direction — we explore certain constraint in databases to improve the utility of a differentially private frequent itemsets mining algorithm.

4. FREQUENT 1-ITEMSETS MINING

In this section, we are considering the problem of frequent 1-itemsets mining. We will show that by truncating transactions, we can significantly promote the utility of frequent 1-itemsets mining while still guaranteeing differential privacy on the benchmark datasets.

4.1 Intuition of Truncating Transactions

Recall the proof of Lemma 1: the key idea is to construct a database τ' whose only δ-approximate output is a particular set $\mathcal{S}'$, which is implemented by adding transactions that are precisely $\mathcal{S}'$. However, that construction implicitly assumes no constraint on transactions' cardinality. Otherwise, that construction no longer works by requiring the cardinality of a transaction to be at most ten. The reason is that the cardinality of the set $\mathcal{S}'$ might be greater than ten, and thus, the addition of a single transaction does not necessarily increase the support of every single item in $\mathcal{S}$ by one. As a result, the total number of transactions required to construct the database τ' is larger than $2\delta\lambda + 2$, which decreases the lower bound on the privacy parameter.

More precisely, by fixing the threshold λ, and the utility parameters δ, η to be constants, Lemma 1 suggests that the lower bound on the privacy parameter for frequent 1-itemsets mining is $\Omega\big(n\big)$. We will show, by limiting the maximal cardinality of transactions, there exists a frequent 1-itemsets mining algorithm that is both (δ, η)-useful and ϵ-differentially private provided $\epsilon \geq \Omega\big(\log n\big)$. We construct that algorithm by utilizing the geometric mechanism.

We can formulate the problem of frequent 1-itemsets mining by first computing n 1-itemset queries $\mathbf{q} = \langle q_1, \ldots, q_n \rangle$ where each q_i computes the support of the 1-itemset $\{i\}$, and then selecting those 1-itemsets whose support exceeds the threshold. We observe that as long as we cmopute $\mathbf{q}$ in a differentially private way, then the frequent 1-itemsets mining algorithm must also be differentially private since the second step relies on the results that have already been differentially private. Theorem 4 shows that the sensitivity of computing $\mathbf{q}$ is equal to the maximal cardinality of transactions.

THEOREM 4. *Let ℓ be the maximal cardinality of transactions, $\ell \leq n$. The sensitivity of computing n different 1-itemset queries is ℓ.*

For ease of presentation, we refer to the frequent 1-itemsets mining algorithm, which first adds geometric noise to the support, and then checks the threshold, as the "geometric noise algorithm." We will prove that by assuming the maximal cardinality $\ell \ll n$, the geometric noise algorithm guarantees both ϵ-differential privacy and (δ, η)-usefulness provided $\epsilon \geq \Omega\big(\log n\big)$. This is shown in Theorem 5.

THEOREM 5. *If the maximal cardinality $\ell \ll n$, then the geometric noise algorithm is both ϵ-differentially private and (δ, η)-useful provided $\epsilon \geq \Omega\big(\log n\big)$.*

We can also prove that the geometric noise algorithm is optimal as shown in Theorem 6.

THEOREM 6. *For any frequent 1-itemsets mining algorithm that is both ϵ-differentially private and (δ, η)-useful, ϵ must be $\Omega\big(\log n\big)$ provided the maximal cardinality $\ell \ll n$.*

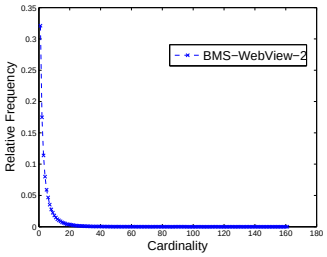


Figure 1: Frequencies of Transactions by Cardinality

Both Theorem 5 and Theorem 6 suggest that the constraint on the maximal cardinality of transactions has a significant impact on the utility of a differentially private frequent 1-itemsets mining algorithm. In this paper, we impose that constraint by truncating transactions.

4.2 Rationale in Truncating Transactions

Our idea of limiting the maximal cardinality of transactions is simple — we truncate a transaction whose cardinality violates that constraint by only keeping a subset of that transaction. Of course, that truncating approach incurs certain information loss. However, if the cardinality of transactions in a dataset follows a distribution in which most are "short" and a few are "long", then these few "long" transactions, while having little impact on which itemsets are frequent, have a major effect on the sensitivity. If this is the case, perhaps we can reduce the sensitivity while still finding a relatively accurate set of frequent 1-itemsets by truncating the long transactions. The three benchmark datasets used in KDD cup 2000 [19] do follow such a distribution. Figure 1 illustrates the correlation between frequency and the cardinality of transactions in one of those datasets, BMS-WebView-2. As we can see, the short transactions, which contain fewer than 10 items, dominate the datasets. The other two datasets, BMS-WebView-1 and BMS-POS, also have a similar distribution. Therefore, we hope that the reduction in the magnitude of noise can offset the information loss incurred by truncating, and thus, we can get accurate results for frequent 1-itemsets.

However, since we transform the database by truncating transactions which needs to access the database, that approach immediately creates a privacy concern: does it violate differential privacy? We will show that as long as that transformation is *local*, whose output only depends on the input transaction, then applying any ϵ-differentially private algorithm on the truncated database also guarantees ϵ-differential privacy for the original database. The notion of *local transformation* is formulated in Definition 7.

DEFINITION 7. *(Local Transformation): A* local transformation *is a probabilistic function r: $2^{\mathcal{I}} \rightarrow 2^{\mathcal{I}}$ such that for any $t \subseteq \mathcal{I}$*

$$\sum_{t' \subseteq \mathcal{I}} \Pr(r(t) = t') = 1$$

In the rest of this paper, without ambiguity, we use "transformation" as a shorthand for "local transformation", and $r(\tau)$ to denote the computation that applies the transformation r to every transaction in the database τ. Theorem 7

proves that applying any differentially private algorithm to a transformed database also guarantees differential privacy for the original database.

THEOREM 7. *Let r be an arbitrary local transformation, and f be an ϵ-differentially private algorithm. Then for any pair of neighboring databases τ and τ', and any $\mathcal{S} \subseteq Range(f)$,*

$$\Pr(f(r(\tau)) \in \mathcal{S}) \leq e^{\epsilon} \Pr(f(r(\tau')) \in \mathcal{S})$$

Next, we will present our differentially private frequent 1-itemsets mining algorithm.

4.3 Our Algorithm

Algorithm 1 TRUNCATEDATABASE

Input: input database τ; privacy parameter ϵ
Output: truncated database

1: $\langle z_1, \ldots, z_n \rangle$ = ESTIMATEDISTRIBUTION(τ, ϵ)
2: ℓ be the smallest integer such that $\sum_{i=1}^{\ell} z_i \geq 0.85$
3: $\tau' = \emptyset$
4: **for** each transaction $t \in \tau$ **do**
5: add t' = TRUNCATE(ℓ, t) to τ'
6: **end for**
7: **return** τ'
8: **function** ESTIMATEDISTRIBUTION(τ, ϵ)
9: Let $\mathbf{z} = \langle z_1, \ldots, z_n \rangle$ where z_i is the number of transactions with cardinality i in τ
10: $\mathbf{z}' = \mathbf{z} + \langle \Delta_1, \ldots, \Delta_d \rangle$ where Δ_i is drawn i.i.d. from the geometric distribution $G(\epsilon)$ in (1).
11: **return** $\mathbf{z}'/m$ where m is the # of transactions in τ
12: **end function**
13: **function** TRUNCATE(ℓ,t)
14: $h = \min\{\ell, |t|\}$
15: $t' = \{$Randomly pick h items from $t.\}$
16: **return** t'
17: **end function**

We determine the maximal cardinality ℓ in a heuristic way in which we set ℓ to the value such that the percentage of the transactions with cardinality no greater than ℓ is at least 85%. That heuristic approach requires us to compute the percentage of transactions by different cardinality, which also has privacy implications, and thus, we add geometric noise to that computation. More precisely, let $\mathbf{z} = \langle z_1, \ldots, z_n \rangle$ where z_i is the number of transactions of cardinality i. It is not hard to show that the sensitivity of computing $\mathbf{z}$ is 1 since the addition or deletion of a single transaction can at most increase or decrease a single z_i by 1. By Theorem 1, adding geometric noise $G(\epsilon)$ in computing $\mathbf{z}$ gurantees ϵ-differential privacy. Since that information is differentially private, it is safe to utilize that information. This is shown in the function "ESTIMATEDISTRIBUTION" in Algorithm 1. We want to emphasize that the 85% is just an experimental setting that works best for the three benchmark datasets, and one is free to set different values for different datasets.

We impose the cardinality constraint by randomly truncating transactions: if the cardinality of a transaction violates that constraint, then only a subset of its items is picked at random without replacement to generate a new transaction whose cardinality is equal to the maximal cardinality. This is shown in the function "TRUNCATE" in Algorithm 1.

Algorithm 2 F1M(τ, ϵ, λ)

Input: input database τ; privacy parameter ϵ; threshold λ
Output: frequent 1-itemsets

1: $\epsilon' = \min\{0.05, \epsilon/10\}$
2: τ' = TRUNCATEDATABASE(τ, ϵ')
3: ℓ = the maximal cardinality of transactions in τ'
4: $R = \emptyset$
5: **for all** 1-itemset X in the alphabet $\mathcal{I}$ **do**
6: $X.supp' = i$'s support in $\tau' + G((\epsilon - \epsilon')/\ell)$
7: **if** $X.supp' \geq \lambda$ **then**
8: Add X to R
9: **end if**
10: **end for**
11: **return** R

Our differentially private *f*requent *1*-itemsets *m*ining algorithm F1M is shown in Algorithm 2. The code in Line 1 in Algorithm 2 is also an experimental setting that works best for our datasets. We prove that F1M guarantees ϵ-differential privacy in Theorem 8.

THEOREM 8. F1M *is ϵ-differentially private.*

In addition to the theoretical improvement of utility shown in Theorem 5, we also use a more intuitive metric F-score to evaluate our idea of truncating transactions, which measures the percentage of the correct frequent itemsets. We present those results in Section 7.

5. FROM 1-ITEMSETS TO K-ITEMSETS

In this section, we are considering a generalized problem of frequent 1-itemsets mining — the frequent k-itemsets mining in which we are interested in the frequent itemsets of cardinality not exceeding k. Our idea is to still truncate transactions as we did for frequent 1-itemsets. However, we will show this generalization is far from trivial.

5.1 Challenges

There are two challenges in frequent k-itemsets mining:

1. Complexity: It is inefficient to compute the support of all the itemsets as there are many.

2. Privacy: It is hard to precisely quantify the relationship between the maximal cardinality of transactions and the sensitivity of multiple i-itemset ($i = 2, \ldots k$) queries.

5.1.1 Complexity

Recall our frequent 1-itemsets mining algorithm in which we first compute the support of all the 1-itemsets, and then perturb their support by adding geometric noise. That algorithm is efficient since the total number of 1-itemsets is n — the size of the alphabet. However, it is not practical to extend that idea to frequent k-itemsets mining since the number of itemsets grows combinatorially. More precisely, it is not hard to show that the number of all the itemsets is $\sum_{i=1}^{k} \binom{n}{i}$. In particular, $n = 1657$ in the dataset BMS-POS,

and thus, when $k = 5$, the total number of all the itemsets is approximately 10^{15}!

In this paper, we attack the complexity problem by utilizing the well-known *a priori property* proposed in [1]. The a priori property states that a k-itemset is frequent only if all its subsets of cardinality $k-1$ are frequent. For ease of presentation, we denote the "subset of cardinality $k-1$" by "$(k-1)$-subset." Our frequent k-itemsets mining algorithm utilizes the a priori property by iteratively finding frequent itemsets in the increasing order of their cardinality: for the mining of frequent i-itemsets, we will only compute the support of the i-itemsets whose every $(i-1)$-subset is frequent in a differentially private way. In accordance with [1], we call those i-itemsets as *candidate i-itemsets*. That order of computation is well-defined since we have already presented our frequent 1-itemsets mining algorithm in Algorithm 2. The generation of candidate i-itemsets is also safe in that it relies on the noisy results of $(i-1)$-itemsets, which have already been differentially private.

5.1.2 Privacy

By the a priori property, we only need to compute the support of the candidate i-itemsets ($i = 2, \ldots, k$). Similar to frequent 1-itemsets mining, we can still model that computation by d i-itemset queries $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$ where q_j computes the support of the candidate i-itemset C_j. To utilize the geometric mechanism, we need to compute the sensitivity of $\mathbf{q}$. However, we will show that given the maximal cardinality ℓ ($2 \leq \ell \leq n$), the precise computation of $\mathbf{q}$'s sensitivity is NP-hard. First, we formulate the problem of computing the sensitivity of multiple i-itemset queries in Problem 1.

PROBLEM 1. *(i, ℓ)-sensitivity: Given a set of itemsets $\mathcal{C}$, each element of which is a subset of the alphabet $\mathcal{I}$, and is of the same cardinality i, find a set $\mathcal{T} \subseteq \mathcal{I}$ such that $|\mathcal{T}| \leq \ell$, and the number of itemsets in $\mathcal{C}$ contained by $\mathcal{T}$ is maximized.*

The number of itemsets in $\mathcal{C}$ contained in $\mathcal{T}$ is precisely the sensitivity of $\mathbf{q}$. As indicated by Theorem 4, when $i = 1$, $(1, \ell)$-sensitivity can be trivially solved in polynomial time. Therefore, we may hope that for some other integer i, (i, ℓ)-sensitivity can also be solved in polynomial time. However, we will prove this is not so. This is shown in Theorem 9.

THEOREM 9. *When $i \geq 2$, (i, ℓ)-sensitivity is NP-hard.*

In view of the hardness result for (i, ℓ)-sensitivity, in this paper we employ a *safe* method by computing the upper bound of multiple i-itemset queries' sensitivity, and use that upper bound to perturb the support of those i-itemsets. The upper bound is shown in Theorem 10.

THEOREM 10. *Given d i-itemset queries $\mathbf{q} = \langle q_1, \ldots, q_d \rangle$, let ℓ be the maximal cardinality of transactions, then the sensitivity of $\mathbf{q}$ is no greater than* $\min(\binom{\ell}{i}, d)$.

We want to emphasize that Theorem 10 only computes the upper bound of multiple i-itemset queries instead of the exact sensitivity. To show that, suppose the itemsets are {2,3}, {4,5}, and {6,7}, and $\ell = 3$. The sensitivity of those three 2-itemset queries is one instead of three since the addition or deletion of any transaction of cardinality three can only increase or decrease the support of one itemset by one simultaneously. We use Theorem 10 to perturb the support of the candidate i-itemsets. Although our approach is not optimal with respect to the utility — we add more noise than required to guarantee differential privacy — our approach is safe in that we guarantee differential privacy for multiple i-itemset queries because we use the upper bound of the sensitivity.

5.2 A Naïve Algorithm

By utilizing the a priori property and Theorem 10, we can extend our idea of truncating transactions to frequent k-itemsets mining: we set the truncated database produced for frequent 1-itemset mining as the input database, and then by the a priori property, iteratively compute frequent 2, 3, ..., k-itemsets. The pseudocode of the naïve algorithm is shown in the long version of our paper. However, we find that the performance of the naïve algorithm is not satisfying in practice. There are two reasons for that:

Unbiased Truncating: We are only interested in the frequent itemsets, and thus, it is more important to preserve those itemsets than other itemsets when truncating transactions. However, random truncating does not consider that.

Propagated Errors: By adding geometric noise to the support, the computation of frequent i-itemsets ($1 \leq i \leq k-1$) is randomized, and thus, it may commit errors, which have a side-effect on subsequent computations of frequent itemsets because of the a priori property. That is, if a frequent itemset is mistakenly labeled as infrequent, then any of its supersets is regarded infrequent without even computing the support.

To remedy those two problems, we propose two heuristic methods: *smart truncating*, which only keeps "useful" items in a transaction, and *double standards*, which sets one threshold to check whether or not an itemset is frequent, and a possibly different one to determine whether or not to use that itemset to generate candidate itemsets. We discuss those two methods in details next.

5.3 Smart Truncating

At the most abstract level, we only need to preserve the "information" related to those frequent itemsets when truncating transactions. More precisely, we will address two problems: what is "related information", and how to preserve this information?

5.3.1 Intuition

Ideally, when truncating a transaction, we only need to keep the frequent itemsets in that transaction since we are only interested in those itemsets. On the other hand, our goal is to find those frequent itemsets in a differentially private way, and thus, how could it be possible to know which itemsets are frequent in advance without violating differential privacy? That obvious paradox is resolved by providing a heuristic method to predicate whether or not a candidate itemset is frequent. We observe that in practice, if all the subsets of a candidate itemset are "sufficiently" frequent, then that itemset is very likely to be frequent. To quantify that observation, we assign each candidate i-itemset ($i \geq 2$) a *frequency score* which is the summation over all its $(i-1)$-subsets' noisy support.

DEFINITION 8. *(Frequency Score): Given a set of $(i-1)$-itemsets $\mathcal{Y} = \{Y_1, \ldots, Y_d\}$, the frequency score of an i-*

itemset X is:

$$fs(X) = \sum_{Y_j \subset X \wedge Y_j \in \mathcal{Y}} Y_j.supp' \quad (2)$$

where $Y_j.supp'$ is the noisy support of the itemset Y_j.

We use the *frequency score* to predict whether or not a candidate itemset is frequent by assuming that a candidate itemset with a higher score is more likely to be frequent than that with a lower score. When truncating transactions, we will keep the itemsets with high frequency scores. This is formulated in Problem 2.

PROBLEM 2. *Optimal (i, ℓ)-truncating: Given a set of i-itemsets $\mathcal{X} = \{X_1, \ldots, X_d\}$, the cover score of a set t' is defined as:*

$$cs(t') = \sum_{X_j \subseteq t' \wedge X_j \in \mathcal{X}} fs(X_j)$$

where fs is the frequency score defined in (2). Given a transaction t, find an ℓ-subset of t such that the cover score of that subset is maximized

Unfortunately, we can prove that there is no efficient algorithm to solve the optimal (i, ℓ)-truncating problem as shown in Theorem 11.

THEOREM 11. *Optimal (i, ℓ)-truncating is NP-hard.*

Given the hardness result, we use a greedy algorithm to truncate transactions.

5.3.2 Our Greedy Algorithm

The idea of our greedy algorithm is simple: for each transaction, we construct the truncated transaction by iteratively adding items in the candidate itemset that is both contained by the input transaction and with the highest frequency score until the truncated transaction's cardinality exceeds the maximal cardinality. However, an intuitive but important observation is that the frequency score of an itemset should not be static: it is possible that some items in an itemset have been added to the truncated transaction, and thus, the number of extra items to include that itemset is less than other itemsets. Therefore, the frequency score of an itemset should be updated with the addition of items to the truncated transaction. To avoid confusion with the static frequency score in Definition 8, we simply refer the "dynamic frequency score" by "weight."

Our greedy algorithm "SMARTTRUNCATING" accepts three input parameters: the original transaction t, the maximal cardinality ℓ, and the set of candidate i-itemsets $\mathcal{C} = \{C_1, \ldots, C_d\}$. For each candidate itemset C_j, its weight $C_j.weight$ is initialized by C_j's frequency score $fs(C_j)$ defined in (2). The algorithm "SMARTTRUNCATING" works as follows: we first find the candidate i-itemsets contained by the input transaction t, and denote the set of those itemsets by $\mathcal{C}'$. Then, starting from an empty transaction t', we first pick the itemset C_j in $\mathcal{C}'$ with the highest weight, add the items in C_j to t', and delete C_j from $\mathcal{C}'$. Next, we update the weights of the remaining itemsets in $\mathcal{C}'$: for each remaining itemset C_h, we compute the average weight of a single item in C_h by $\alpha_h = fs(C_h)/i$. Suppose the number of items in C_h that has already been added to the truncated transaction t' is β_h. Then the weight of C_h is updated by $C_h.weight = C_h.weight + \alpha_h * \beta_h$. After updating the weight for every remaining itemset, we repeat those steps until the cardinality of t' exceeds ℓ. The pseudocode of SMARTTRUNCATING is shown in the long version of our paper. We show a running example of our greedy algorithm in Example 1.

EXAMPLE 1. *Given a transaction t = {1,2,3,4,5}, and three 2-itemsets {1,2}, {2,3} and {4,5} with weights 10, 8, and 9, respectively. Suppose the maximal cardinality of transactions is 3. The truncated transaction t' is initialized to be empty. First, we pick the itemset {1,2} which has the largest weight, and then t' is updated to {1,2}. Next, we update the weight of the remaining itemsets as follows: since the single item "2" has been added to t', the new weight of the itemset {2,3} is 8 + 1*8/2 = 12, and that for {4,5} remains the same. Hence, we add the itemset {2,3} to t', which is then updated to {1,2,3}. Since the cardinality of t' meets the constraint, the algorithm "*SMARTTRUNCATING*" stops, and the truncated transaction is t' = {1,2,3}.*

It is not hard to see that our smart truncating method is a *local transformation* which only relies on the noisy support, which have already been differentially private.

5.4 Double Standards

As discussed in Section 5.2, by the a priori property, if a frequent itemset is mistakenly labeled as infrequent, then any of its supersets is regarded infrequent without even computing the support, which shows how the errors are propagated to subsequent computations. To alleviate that problem, we observe that a frequent itemset indeed serves two different roles — the result of frequent itemsets mining and to generate candidate itemsets. The errors are propagated due to the second role. Therefore, instead of setting the same threshold for both roles for every itemset, we customize the thresholds for each itemset by setting a threshold to determine whether or not that itemset is frequent, and a possibly different one to decide whether or not to use that itemset to generate candidate itemsets. Speaking intuitively, the previous threshold is computed by measuring the "average" information loss in truncating while the latter one by the "maximal" information loss. By that approach, it is possible that some itemsets are labeled infrequent but are used to generate candidate itemsets, which helps to reduce the propagated errors by the a priori property.

5.4.1 Quantifying Information Loss in Truncating

So far, we have only considered the benefit of truncating transactions, which reduces the sensitivity in computing frequent itemsets. On the other hand, there is information loss when truncating transactions — the support of some itemsets decrease. For example, given a transaction {1,2,3}, by truncating that transaction to be {2,3}, the support of the itemset {1,2} changes from 1 to 0. To quantify that information loss, we formulate our problem as follows: given the noisy support θ' of an i-itemset X in the truncated database, estimate the support θ of X in the original database in a differentially private way. The major difficulty in solving that problem comes from the privacy requirement that whenever we need to utilize some information from the database, those information have privacy implications. Therefore, we must guarantee that our algorithm is either differentially private or only depends on differentially private information. In this

paper, we take the latter approach, and our algorithm takes two steps:

1. Given the noisy support of an itemset X, compute its support in the truncated database in a differentially private way.
2. Given the support of the itemset X in the truncated database, estimate its support in the original database in a differentially private way.

In the rest of this section, for ease of presentation, we use "original support" as a shorthand for "the support in the original database", and "truncated support" for "the support in the truncated database." Unless otherwise specified, we assume the i-itemset X is given, and use θ, $\hat{\theta}$ and θ' to denote X's original support, truncated support and noisy truncated support, respectively.

The first step is relatively straightforward where we use the Bayesian rule to compute the probability distribution of the truncated support as shown in Theorem 12

THEOREM 12.

$$\Pr(\hat{\theta}|\theta') \sim e^{-\epsilon|\theta'-\hat{\theta}|} \tag{3}$$

The second step quantifies the information loss in truncating. To give an intuition of our result, we start from an inverse problem: given the original support θ, what is the truncated support $\hat{\theta}$? In this paper, we address that problem by using the analysis in random truncating to approximate the itemset X's truncated support. We want to emphasize that our smart truncating method is by no means equivalent to random truncating, and we *only* utilize the analysis in random truncating as a heuristic method to approximately quantify the information loss by our smart truncating method. We will discuss the rationale of that approximation at the end of this section.

We assume a uniform distribution among transactions with different cardinality containing the itemset X. More precisely, let z_j be the relative frequency of the transactions with cardinality j in the original database. Given an i-itemset X, we assume that the number of transactions with cardinality j containing X is $\theta * z_j / \sum_{h=i}^{n} z_h$. [1] Recall that we have already computed z_j $(j = 1 \dots n)$ in Algorithm 2 for frequent 1-itemsets mining in a differentially private way, and thus, it is safe to utilize those information. For each transaction t, let t' be the truncated transaction of t. We observe that if a transaction does not contain X, then the truncated transaction does not contain X, either. Thus, it suffices to only consider the transactions containing X to compute X's truncated support. We define a binary random variable M_t to quantify the effect of truncating a transaction on the support of the itemset X:

$$M_t = \begin{cases} 1 & \text{if } X \subseteq t' \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

(5) quantifies the probability that X remains in the truncated transaction.

$$\Pr(M_t = 1) = \frac{\binom{|t|-i}{\ell-i}}{\binom{|t|}{\ell}} \tag{5}$$

[1] Strictly speaking, it must be rounded to an integer.

We also observe that the random variable M_t is identical for the transactions of the same cardinality, and thus, we denote M_t by M_h where $h = |t|$. Let f_h be the number of transactions with cardinality j containing the itemset X, and by our uniform assumption, $f_h = \theta * z_h / \sum_{j=i}^{n} z_j$. Thus, the truncated support of X is also a random variable M where:

$$M = \sum_{h=i}^{n} \sum_{j=1}^{f_h} M_h \tag{6}$$

The expectation of M is:

$$\begin{aligned} E(M) &= \sum_{h=i}^{n} f_i \Pr(M_h = 1) \\ &= \theta(\sum_{h=i}^{n} \frac{z_h}{\sum_{j=i}^{n} z_j} \frac{\binom{h-k}{\ell-k}}{\binom{h}{\ell}}) \end{aligned} \tag{7}$$

which quantifies the "average" truncated support of the itemset X. Next, we will compute the minimal truncated support, which quantifies the maximal information loss. It is not hard to see that a trivial lower bound for M is 0. However, by using 0 as the lower bound, it hardly provides us a way to estimate the original support given the truncated support. Therefore, we quantify that lower bound in a probabilistic way such that the truncated support is very unlikely to be smaller than that lower bound. Definition 9 formalizes that idea.

DEFINITION 9. *(ρ-lower bound): An integer σ is called a* ρ-lower bound *for the truncated support iff:*

$$\Pr(M \leq \sigma) \leq \rho$$

where M is the random variable defined in (6).

We compute the ρ-lower bound by using the Chernoff bound [2]. Let $\mu = E(M)$, and for any $\gamma \geq 0$, by the multiplicative form of Chernoff bound,

$$\Pr(M \leq (1-\gamma)\mu) \leq exp(\frac{-\gamma^2\mu}{2})$$

Therefore, it suffices to solve the inequality $exp(-\gamma^2\mu/2) \leq \rho$, and the resulting $\lfloor(1-\gamma)\mu\rfloor$ serves as the ρ-lower bound. Thus, given the truncated support $\hat{\theta}$, let

$$exp\_ratio(i) = \sum_{h=i}^{n} \frac{z_h}{\sum_{j=i}^{n} z_j} \frac{\binom{h-k}{\ell-k}}{\binom{h}{\ell}}$$

By (7), the average original support is computed by:

$$avg\_os(\hat{\theta}) = \frac{\hat{\theta}}{exp\_ratio(i)} \tag{8}$$

When computing the maximal original support, let μ be the expectation of the truncated support given the maximal original support. We compute μ by treating the truncated support $\hat{\theta}$ as the ρ-lower bound, which leads to solving the following two inequalities:

$$(1-\gamma)\mu \leq \hat{\theta} \leq \mu, \quad exp(-\frac{\gamma^2\mu}{2}) = \rho$$

It is not hard to show that

$$\mu \leq \mu^* = \hat{\theta} - \ln\rho + \sqrt{\ln^2\rho - 2\hat{\theta}\ln\rho} \tag{9}$$

provided $\ln \rho \leq 2\hat{\theta}$. Thus, the maximal original support is computed by

$$max\_os(\hat{\theta}) = \frac{\mu^*}{exp\_ratio(i)} \quad (10)$$

where μ^* is defined in (9), and if $\ln \rho > 2\hat{\theta}$, then we will set $max\_os(\hat{\theta})$ to be $avg\_os(\hat{\theta})$.

By combining our result of the first step, which estimates the truncated support given the noisy support, and our second step, which infers the original support given the truncated support, given θ', the noisy truncated support of the itemset X, we can compute the average original support by:

$$avg\_supp(\theta') = \sum_{j=1}^{n} \Pr(j|\theta') avg\_os(j) \quad (11)$$

and the maximal original support by:

$$max\_supp(\theta') = \sum_{j=1}^{n} \Pr(j|\theta') max\_os(j) \quad (12)$$

where $\Pr(j|\theta')$, $avg\_os(i)$ and $max\_os(i)$ are defined in (3), (8) and (10), respectively.

We use the average original support of an itemset to check whether or not it is frequent, and the maximal original support to determine whether or not to use it to generate candidate itemsets. Equivalently speaking, given the noisy truncated support θ', we have changed the threshold to determine whether or not X is frequent from λ to $\lambda - avg\_supp(\theta') + \theta'$, and that for whether or not to use X to generate itemsets to $\lambda - max\_supp(\theta') + \theta'$. In that way, we have actually relaxed both thresholds.

Algorithm 3 FREQUENT_K_ITEMSETS_MINING

Input: input database τ; itemsets' cardinality k; privacy parameter ϵ; threshold λ
Output: frequent itemsets of cardinality not exceeding k

1: $\epsilon' = \epsilon/k$
2: $\epsilon'' = \min\{0.05, \epsilon'/10\}$
3: $\tau' =$ TRUNCATEDATABASE(τ, ϵ'')
4: $\ell =$ the maximal cardinality of transactions in τ'
5: $\mathcal{S}_1 =$ I_ITEMSETS_MINING$(\tau', \ell, \epsilon' - \epsilon'', \lambda, \emptyset)$
6: **for** i = 2 to k **do**
7: $\quad \ell = \psi(\lambda, i)$
8: $\quad \mathcal{S}_i =$ I_ITEMSETS_MINING$(\tau, \ell, \epsilon', \lambda, \mathcal{S}_{i-1})$
9: **end for**
10: $R = \emptyset$
11: **for all** itemset X_j in $\cup_{i=1}^{k} \mathcal{S}$ **do**
12: $\quad$ **if** $avg\_supp(X_j.supp') \geq \lambda$ **then**
13: $\quad\quad$ Add X_j to R
14: $\quad$ **end if**
15: **end for**
16: **return** R

5.4.2 Discussion

We want to emphasize that our approach of estimating the original support of an itemset is solely a heuristic method. As discussed, the major difficulty of estimating an itemset's original support comes from the privacy requirement, and thus, we must be quite careful of utilizing any information from the database by ensuring that those information must be differentially private. The reason why we use random truncating to approximate the information loss of smart truncating is because it relies on little information from the database. The exploration of other approaches to quantify the information loss by smart truncating is an interesting direction for future work.

5.5 Our Algorithm

Our algorithm for frequent k-itemset mining improves over the naïve algorithm in Algorithm 6 by using smart truncating to truncate transactions instead of random truncating, and setting different thresholds for itemset being frequent and to generate candidate itemsets. This is shown in Algorithm 3. In particular, Algorithm 3 differs from Algorithm 6 in two places. First, instead of randomly truncating the transactions only once, we apply the method "SMARTTRUNCATING" to the original database for the mining of i-itemsets ($i = 2, \ldots, k$) by utilizing the results of the noisy $(i-1)$-itemsets as shown in line 10 in Algorithm 4. We initialize a candidate itemset's weight by its frequency score as shown in the code from line 7 to line 9 in Algorithm 4. Second, an itemset is used to generate candidate itemsets iff its estimated maximal original support exceeds the threshold. This is shown in line 18 in Algorithm 4. However, whether an itemset is frequent or not is determined by its estimated average original support. This is shown in line 12 in Algorithm 3.

The function ψ in line 7 of Algorithm 3 is a function tailored for each dataset, whose output is the maximal cardinality of transactions given the threshold and the current candidate itemsets' cardinality. We argue that since the data curator has full access to the raw data, she can try to tune the ψ function to produce a reasonable result. The idea of tuning certain parameters in a differentially private algorithm by the data curator was first proposed by [6]. We consider how to automatically determine the maximal cardinality of transactions as an interesting direction for future work. We prove Algorithm 3 is differentially private in Theorem 13.

THEOREM 13. *Algorithm 3 is ϵ-differentially private.*

6. FREQUENT ITEMSETS MINING

We observe that Algorithm 3 is also a differentially private frequent itemsets mining algorithm by setting k to be the maximal cardinality of frequent itemsets. However, such knowledge depends on the database, which has privacy implications, and thus, we also need to guarantee differential privacy when computing the maximal cardinality of frequent itemsets. This problem is formulated in Problem 3.

PROBLEM 3. *Let $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ where y_i is the maximal support of the i-itemsets. Given a threshold λ, find the index i^* such that y_{i^*} is the smallest integer that exceeds λ in a differentially private way.*

A naïve idea to solve Problem 3 is to first add geometric noise to each y_i, and then find such index i^*. However, we can prove that the sensitivity of computing $\mathbf{y}$ is n as shown in Theorem 14.

THEOREM 14. *The sensitivity of computing $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ is n, where y_i is the maximal support of the i-itemsets.*

Algorithm 4 I_ITEMSETS_MINING

Input: database τ; maximal cardinality ℓ; privacy parameter ϵ; threshold λ; noisy $(i-1)$-itemsets $\mathcal{S}$
Output: the i-itemsets whose estimated maximal original support exceeds the threshold

if i is 1 **then**
  $\mathcal{C} = \mathcal{I}$
else
  $\mathcal{C}$ = Generate candidate i-itemsets from $\mathcal{S}$
end if
if i is not 1 **then**
  for all itemset $C_j \in \mathcal{C}$ **do**
    $C_j.weight = fs(C_j)$
  end for
  $\tau' \leftarrow$ apply SMARTTRUNCATING to τ's transactions
else
  $\tau' = \tau$
end if
$\kappa = \min\{\binom{\ell}{i}, |\mathcal{C}|\}$
$R = \emptyset$
for all itemset C_j in $\mathcal{C}$ **do**
  $C_j.supp' = C_j$'s support in $\tau' + G(\epsilon/\kappa)$
  if $max\_supp(C_j.supp') \geq \lambda$ **then**
    Add C_j to R
  end if
end for
return R

By Theorem 14, in order to guarantee ϵ-differential privacy for Problem 3, we need to add the geometric noise $G(\epsilon/n)$ to each y_i. We will show that by the same idea of binary search, we can reduce the required noise to $G(\epsilon/\lceil \log n \rceil)$. In fact, we observe that for any integer i $(1 \leq i \leq n-1)$, by the a priori property, $y_i \geq y_{i+1}$, and thus, the sequence $y_1, \ldots, y_n$ is non-increasing. Let y_i' be the noisy support of y_i. If $y_i' < \lambda$ (or $y_i' \geq \lambda$), then we only need to search such index i^* between the range 1 to $i-1$ (or i to n), and there is no need to do so for the other half. We implement that idea in the algorithm "ESTIMATEMAXCARDINALITY" which accepts as input: the maximal support of itemsets by cardinality $\mathbf{y}$, the privacy parameter ϵ, the threshold θ and the size of the alphabet n, and produces as output the maximal cardinality of frequent itemsets. The pseudocode of that algorithm is shown in the long version of our paper.

However, it is not practical to precisely compute $\mathbf{y}$, the maximal support of itemsets by cardinality. The reason is that we can neither compute the support of all the itemsets, whose number is too large, nor utilize the a priori property to reduce the complexity since the i-itemset with the maximal support might not be a superset of the $(i-1)$-itemset with the maximal support. Therefore, we approximate $\mathbf{y}$ by setting the threshold to be $\lambda/20$, and run the original Apriori algorithm [1] to compute the frequent itemsets. Suppose the maximal cardinality of the resulting frequent itemsets is j, then for i from 1 to j, we can precisely compute the maximal support of i-itemsets y_i. For any i from $j+1$ to n, we set $y_i = y_j$. Algorithm 9 still guarantees differential privacy by using our approximation since the sensitivity of computing each y_i is still one.

In practice, a user might have certain prior knowledge on the maximal cardinality of frequent itemsets, which might stem from other information sources, previous interactions with the database, introspection, or common sense. In that way, she can estimate an upper bound on that maximal cardinality, and provide it as an input parameter. We want to emphasize that prior knowledge does not weaken our privacy criteria; We still use the standard definition of differential privacy, and only use that prior knowledge as an input parameter to estimate frequent itemsets' maximal cardinality.

By combining Algorithm 3 and Algorithm 9, we show our differentially private frequent itemsets mining algorithm in Algorithm 5. Theorem 15 proves that Algorithm 5 is differentially private.

Algorithm 5 DIFFFIM

Input: input database τ; privacy parameter ϵ; threshold λ; upper bound of frequent itemsets' maximal cardinality θ
Output: frequent itemsets

$\epsilon' = \max(\epsilon/20, 0.05)$
$\mathbf{y} = \langle y_1, \ldots, y_j, \ldots, y_j \rangle$ where j is the maximal cardinality of frequent itemsets with support $\lambda/20$ and y_i is the maximal support of i-itemsets.
k = ESTIMATEMAXCARDINALITY($\mathbf{y}$, ϵ', λ, θ)
if k is 0 **then**
  return $\emptyset$
else
  return FREQUENT_K_ITEMSETS_MINING(τ, k, ϵ-ϵ', λ)
end if

dataset	$\lvert D \rvert$	$\lvert \mathcal{I} \rvert$	max $\lvert t \rvert$	avg $\lvert t \rvert$
BMS-POS (POS)	515,597	1,657	164	6.5
BMS-WebView-1 (WV1)	59,602	497	267	2.5
BMS-WebView-2 (WV2)	77,512	3,340	161	5.0

Table 1: Dataset characteristics

Theorem 15. *Algorithm 5 guarantees ϵ-differential privacy.*

7. EXPERIMENTS

In this section, we experimentally evaluate our techniques on three benchmark datasets described in [19]. A summary of those datasets is shown in Table 1. Instead of using the (δ, η)-usefulness, we employ a more intuitive metric F-score as shown in Definition 10. We implemented our algorithms in C++, and ran the experiments on an Intel Core 2 Duo 2.33GHZ machine with 1 GB RAM running Linux. For each randomized algorithm, we ran it ten times to obtain its average performance. We use the relative threshold in our experiments, and the threshold can be derived easily by multiply-

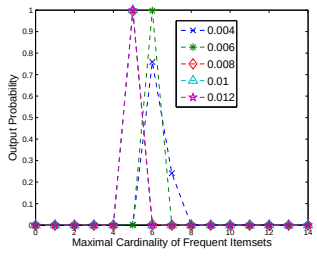
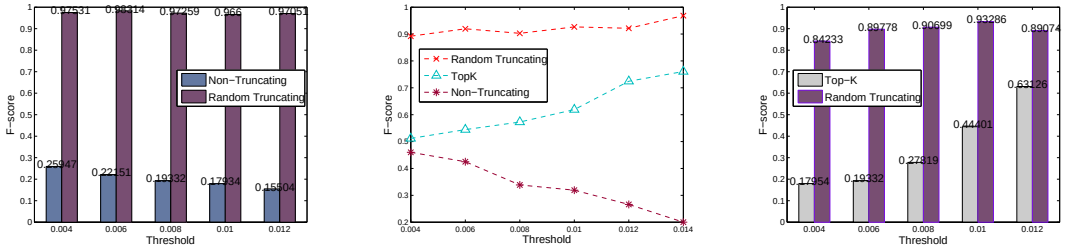
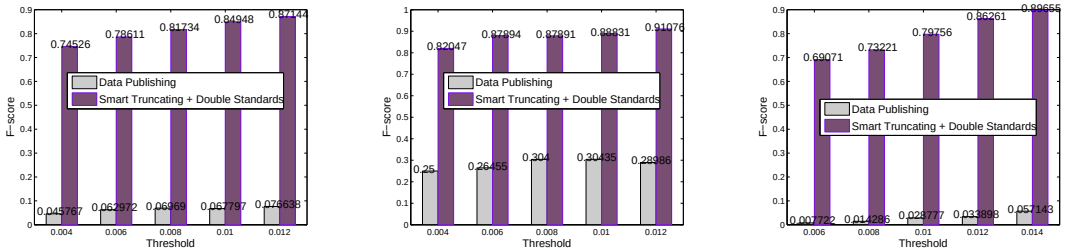


Figure 2: Frequent Itemsets' Maximal Cardinality



(a) POS (b) WV1 (c) WV2

Figure 3: F-score of 1-itemsets



(a) POS (b) WV1 (c) WV2

Figure 4: F-score of frequent itemsets

ing the relative threshold with the number of transactions in a database. In this section, we also use "threshold" as a shorthand for "relative threshold" unless otherwise specified.

DEFINITION 10. *(F-score): Let U_p be the set of frequent itemsets generated by a differentially **p**rivate frequent itemset mining algorithm, and U_c be the set of **c**orrect frequent itemsets, then*

$$precision = \frac{|U_p \cap U_c|}{|U_p|}, \quad recall = \frac{|U_p \cap U_c|}{|U_c|}$$

and the F-score is the harmonic mean of precision and recall:

$$F\text{-}score = 2 * \frac{precision * recall}{precision + recall}$$

7.1 Maximal Cardinality of Frequent Itemsets

First, we evaluate the accuracy of our algorithm in estimating the maximal cardinality of frequent itemsets in a differentially private way. We set the privacy parameter to be 0.05, and we believe that the maximal cardinality of frequent itemsets when the threshold exceeds 0.004 can not be greater than 32. Figure 2 shows the result in the dataset POS. Each curve shows the probability of outputting an integer given the threshold. For each threshold, the most likely output is the correct result. In particular, when the threshold exceeds 0.06, the probability of outputting a wrong answer is less than 0.01, which shows that our algorithm is able to accurately estimate the maximal cardinality of frequent itemsets in a differentially private way. We also have similar results on the other two datasets. Due to the space limitation, we do not include those two figures.

7.2 Frequent 1-itemsets

In this experiment, we set the privacy parameter to be 0.25. We start by showing the benefit of truncating transactions. The non-truncating algorithm is generated by setting the maximal cardinality ℓ to be $|\mathcal{I}|$ in Algorithm 2. Figure 3a shows the comparison in the dataset POS. As we can see, for any threshold, the F-score of Algorithm 2 is above 0.95 while that for the non-truncating algorithm is below 0.25. This result shows that our algorithm significantly promotes the quality of frequent 1-itemsets over the non-truncating one while still guaranteeing differential privacy. Figure 3b shows a similar result in the dataset WV1. We observe an interesting phenomena in Figure 3b that the F-score of the non-truncating algorithm decreases as the threshold increases. This is because the sensitivity of the non-truncating algorithm is 497, which is comparable to the threshold, and thus, the magnitude of the geometric noise is so large that the non-truncating algorithm is like flipping an unbiased coin to determine whether or not each 1-itemset is frequent. Thus, with the increase in the threshold, the total number of noisy frequent 1-itemsets remains almost the same while the number of correct frequent 1-itemsets decreases, which decreases the precision, which lowers the F-score.

We also compare the F-score of our algorithm with the top-k frequent i-itemsets mining algorithm proposed in [4], whose output is k i-itemsets with the largest support, and we refer to that algorithm as the "top-k algorithm" when i is clear from the context. We adapt the top-k algorithm to frequent 1-itemsets mining by setting k to be the number of frequent 1-itemsets. We want to emphasize that setting might come with certain privacy implications. However, even with that relaxation, both Figure 3b and Figure 3c show that our truncating approach significantly outperforms the top-k algorithm in the F-score of frequent 1-itemsets.

7.3 Frequent Itemsets Mining

In this set of experiments, we evaluate Algorithm 5. We set the privacy parameter to be 1 and ρ to be 0.01 in the double standards method. We compare our results with the differentially private set-valued data publishing algorithm [6] on which we run the original Apriori algorithm over the anonymized data. We find that our algorithm outperforms the data publishing algorithm on all three datasets as shown in Figure 4. In particular, Algorithm 5 increases the F-score of frequent itemsets by an order of magnitude in both POS and WV2 as shown in Figure 4a and Figure 4c, respectively. The major reason why the data publishing algorithm fails to generate accurate frequent itemsets is because the total number of transactions is greatly reduced after anonymization: for the dataset WV2, the average number of transactions after anonymizing is 6734.5, which is less than 10% of the original transactions. Thus, the anonymization incurs significant information loss, which has a huge penalty on the recall, which greatly hurts the F-score.

We also want to know how the two heuristic methods — smart truncating and double standards — affect our results. We show the results in Figure 5. As we can see, the most significant improvement is the double standards approach. Although the smart truncating method improves the quality of frequent itemsets over the random truncating one in both POS and WV1, we find it does not work for WV2. This indicates that whether or not the frequency score is a good indicator of an itemset's frequency depends on the dataset. Whether there exists a truncating approach that outperforms the random truncating approach on any dataset is an interesting direction for future research.

To better understand why our heuristics work on the dataset POS, we show the results of 2-itemsets in Figure 6. We observe that the double standards approach significantly improves the recall of frequent 2-itemsets while decreases the precision a little as shown in Figure 6c and Figure 6b, respectively. That observation coincides with the intuition of the double standards approach: by setting a lower threshold for generating candidate itemsets, the number of candidate itemsets increases, which reduces the errors by the a priori property. Furthermore, we also observe that by utilizing the a priori property and truncating transactions, our algorithm increases the F-score of the top-k algorithm by orders of magnitude as shown in Figure 6a.

8. RELATED WORK

The notion of differential privacy was proposed by Dwork et al. in [7]. The same authors also propose the addition of Laplacian noise to guarantee differential privacy [8], and [10] propose to add geometric noise to achieve the same goal. The problem of frequent itemsets mining has been extensively studied in literature [1, 11]. The data mining community has focused on hiding sensitive rules generated from transactional databases [3, 17]. In [3], the authors address this problem by altering the database to hide a given set of sensitive rules. However, how to define sensitive rules is unclear, and their approach does not satisfy differential privacy.

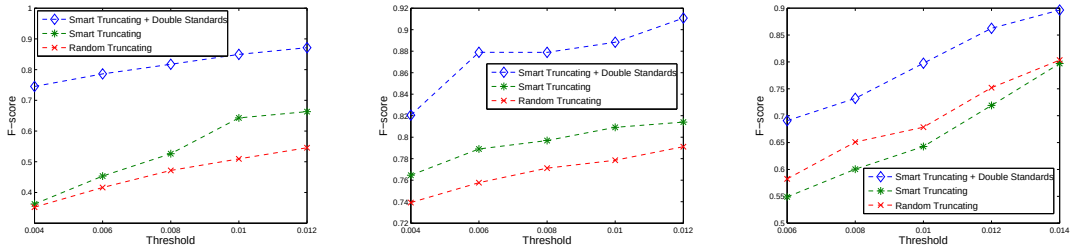
To the best of our knowledge, no previous work has addressed the problem of guaranteeing differential privacy in a frequent itemsets mining algorithm. Although Evfimievski et al. have developed a privacy preserving frequent itemsets mining algorithm in [9], their approach does not guarantee differential privacy. The most related work to ours is [4], which considers the problem of discovering top-k frequent i-itemsets in a differentially private way. Our results indicate that by adapting their algorithm to frequent i-itemsets even with a sacrifice in privacy, our approach still greatly increases F-score of the results, and on some datasets, the improvement is by orders of magnitude.

Another way to develop a differentially private frequent itemsets mining is to use a basic, non-anonymizing frequent itemsets mining algorithm, but to apply it to an anonymized version of the transactional data. This is an intriguing approach and it warrants exploration. Early work appears on anonymizing other types of data sets [18, 14] has shown a great success in this direction. In other related work, [12, 16, 5] propose ad hoc privacy criteria to resist certain kinds of attacks on publishing transaction databases. However, those ad hoc privacy criteria do not guarantee differential privacy. The latest effort to anonymize transactional data in a differentially private way is proposed by [6]. However, they only considers the workload of top-k frequent itemsets, and our experimental results indicate that our algorithm improves the F-score of frequent itemsets over [6] by an order of magnitude on two benchmark datasets.

9. CONCLUSION

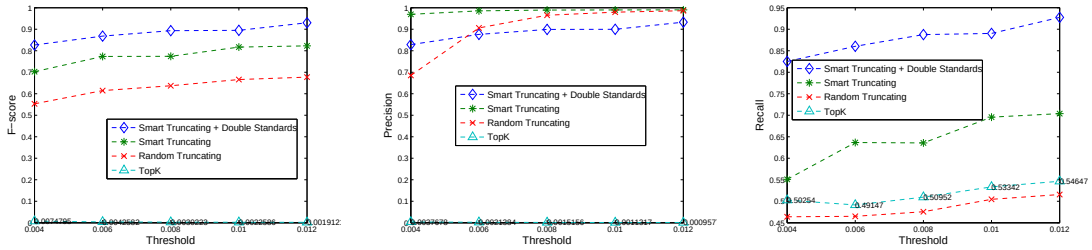
In this paper, we have proposed a differentially private frequent itemsets mining algorithm. To the best of our knowledge, our work is the first effort to consider frequent itemsets mining in the setting of differential privacy. We have precisely quantified the trade-off between privacy and utility in frequent itemsets mining, and our result indicates that in order to satisfy a non-trivial utility requirement, a frequent itemsets mining algorithm incurs a huge risk of privacy breach. However, we find that we can greatly promote the utility of a differentially private frequent itemsets mining algorithm by limiting the maximal cardinality of transactions. Motivated by that result, our naïve algorithm randomly truncates transactions in a database to satisfy the constraint on the maximal cardinality, iteratively generates candidate itemsets by the a priori property, and perturbs the support of those candidate itemsets in a differentially private way to discover frequent itemsets. We have also proposed two heuristic methods to improve our naïve algorithm: the first one predicts the frequency of an itemset, and preserves those "predictably frequent" itemsets when truncating transactions; the second one sets one threshold to check whether or not an itemset is frequent, and a possibly different one to determine whether or not to use that itemset to generate candidate itemsets by the a priori property. Our results on benchmark datasets indicate that in comparison to the latest algorithm on publishing transactional data in a differentially private way [6], our algorithm improves the F-score of frequent itemsets by more than 200% in one dataset, and by an order of magnitude on the other two datasets.

There are many potential opportunities for future work. One such direction would be to explore other sophisticated truncating algorithm. Another direction would be to explore better methods to quantify the information loss by truncating than our double standards approach. Furthermore, the success of our algorithm on three benchmark datasets relies on the assumption that the "short" transactions dominate the datasets. When that assumption fails, how to improve



(a) POS (b) WV1 (c) WV2

Figure 5: Improvements of our heuristics



(a) F-score (b) Precision (c) Recall

Figure 6: Frequent 2-itemsets in POS

the quality of our algorithm is left open.

10. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.

[2] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *STOC*, 1977.

[3] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Anonymity preserving pattern discovery. *The VLDB Journal*, 2008.

[4] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, 2010.

[5] J. Cao, P. Karras, C. Raissi, and K.-L. Tan. ρ-uncertainty: Inference proof transaction anonymization. In *VLDB*, 2010.

[6] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *VLDB*, 2011.

[7] C. Dwork. Differential privacy. In *ICALP*, 2006.

[8] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCS*, 2006.

[9] A. Evfimievski, R. Srikant, R. Agarwal, and J. Gehrke. Privacy preserving mining of association rules. 2004.

[10] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, 2009.

[11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, 2000.

[12] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *VLDB*, 2009.

[13] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining: a general survey and comparison. *SIGKDD Explor. Newsl.*, 2000.

[14] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.

[15] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 1994.

[16] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *VLDB*, 2008.

[17] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *TKDE*.

[18] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2009.

[19] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *KDD '01*, 2001.

APPENDIX

A. PSEUDOCODES

A.1 The naïve algorithm

This is shown in Algorithm 6. The method "NaïveFkIM" first randomly truncates the transactions in the same way as Algorithm 2, and then iteratively calls the method "NaïveFiIM" shown in Algorithm 7 to compute frequent $1, 2 \ldots, k$-itemsets. The computation of frequent 1-itemsets is identical to that in Algorithm 2. For the mining of frequent i-itemsets ($i \geq 2$), the function "NaïveFiIM" generates candidate i-itemsets

from the frequent $(i-1)$-itemsets by the a priori property, and then perturbs the support of those candidate i-itemsets. If the noisy support of an itemset exceeds the threshold, then that itemset is labeled as frequent. Theorem 16 proves that Algorithm 6 guarantees differential privacy.

Algorithm 6 NaïveFkIM

Input: database τ; itemsets' cardinality k; privacy parameter ϵ; threshold λ
Output: frequent itemsets of cardinality not exceeding k

$\epsilon' = \epsilon/k$
$\epsilon'' = \min\{0.05, \epsilon'/10\}$
$\tau' = \text{TruncateDatabase}(\tau, \epsilon'')$
$\ell =$ the maximal cardinality of transactions in τ'
$\mathcal{S}_1 = \text{NaïveFiIM}(\tau', \ell, \epsilon' - \epsilon'', \lambda, \emptyset)$
for i = 2 to k **do**
    $\mathcal{S}_i = \text{NaïveFiIM}(\tau', \ell, \epsilon', \lambda, \mathcal{S}_{i-1})$
end for
return $\cup_{i=1}^{k} S_i$

Algorithm 7 NaïveFiIM

Input: truncated database τ'; maximal cardinality ℓ; privacy parameter ϵ; threshold λ; frequent $(i-1)$-itemsets $\mathcal{S}$
Output: frequent i-itemsets

if i is 1 **then**
    $\mathcal{C} = \mathcal{I}$
else
    $\mathcal{C} =$ Generate candidate i-itemsets from $\mathcal{S}$
end if
$R = \emptyset$
$\kappa = \min\{\binom{\ell}{i}, |\mathcal{C}|\}$
for all itemset C_j in $\mathcal{C}$ **do**
    $C_j.supp' = C_j$'s support in $\tau' + G(\epsilon/\kappa)$
    if $C_j.supp' \geq \lambda$ **then**
        Add C_j to R
    end if
end for
return R

THEOREM 16. *Algorithm 6 is ϵ-differentially private.*

A.2 Pseudocodes of SmartTruncating

A.3 Pseudocodes of EstimateMaxCardinality

The pseudocode of that algorithm is shown in Algorithm 9, and Theorem 17 proves that Algorithm 9 is differentially private.

THEOREM 17. *Algorithm 9 is ϵ-differentially private.*

B. PROOFS

B.1 Proof of Theorem 1

PROOF. Since the addition (deletion) of a row can at most increase (decrease) the result of a count query by 1, by Definition 4, the sensitivity of $\mathbf{q}$ is precisely the maximal number of queries that can be increased (decreased) simultaneously with the addition (deletion) of a row. For any pair of neighboring databases τ, τ', without loss of generality, let the first

Algorithm 8 SmartTruncating

Input: transaction t; maximal cardinality ℓ; candidate i-itemsets $\mathcal{C}$
Output: truncated transaction

$t' = \emptyset$
$\mathcal{C}' =$ {the candidate i-itemsets in $\mathcal{C}$ contained by t}
$\mathcal{A} = \{\alpha_j | \alpha_j = C_j.weight/i, C_j \in \mathcal{C}'\}$
while $|t'| \leq \ell$ **do**
    $C_j =$ the itemset with the highest weight in $\mathcal{C}'$
    $\hat{C}_j = C_j - t'$
    if $|\hat{C}_j| + |t'| > \ell$ **then**
        $\Delta t =$ {Randomly select ℓ - $|t'|$ items from $\hat{C}_j$}
    else
        $\Delta t = \hat{C}_j$
    end if
    Delete C_j from $\mathcal{C}'$
    $t' = t' \cup \Delta t$
    $\mathcal{C}' = \text{UpdateWeight}(\Delta t, \mathcal{C}', \mathcal{A})$
end while
return t'
function UpdateWeight($\Delta t, \mathcal{C}', \mathcal{A}$)
    for all itemset $C_h \in \mathcal{C}'$ **do**
        $\beta_h = |C_h \cap \Delta t|$
        $C_h.weight = C_h.weight + \alpha_h * \beta_h$
    end for
    return $\mathcal{C}'$
end function

Algorithm 9 EstimateMaxCardinality

Input: the maximal support of itemsets by cardinality $\mathbf{y}$; privacy parameter ϵ; threshold λ; the size of the alphabet n
Output: the maximal cardinality of frequent itemsets

$\epsilon' = \epsilon/\lceil \log_2 n \rceil$
return BinarySearch($\mathbf{y}$, 1, n, ϵ', λ)
function BinarySearch($\mathbf{y}$, *low*, *high*, ϵ, λ)
    if *low* == *high* **then**
        if $y_{low} + G(\epsilon) \geq \lambda$ **then**
            return *low*
        else
            return *low* − 1
        end if
    end if
    pivot = (*low* + *high*)/2
    if $y_{pivot} + G(\epsilon) \geq \lambda$ **then**
        return BinarySearch($\mathbf{y}$, *pivot*+1, *high*, ϵ, λ)
    else
        return BinarySearch($\mathbf{y}$, *low*, *pivot*-1, ϵ, λ)
    end if
end function

d' queries be the set of queries with different results in τ and τ', and thus, $d' \leq S_{\mathbf{q}}$. Let $\mathbf{x} = \langle x_1, \ldots, x_d \rangle$ be a vector of dimension d with each element in $\{0, \ldots, m\}$, then,

$$\begin{aligned}
&\Pr(A_{\mathbf{q}}(\tau) = \mathbf{x}) \\
&= \prod_{i=1}^{d} \Pr(q_i(\tau) + \Delta_i = x_i) \\
&\leq \prod_{i=1}^{d'} (e^{\frac{\epsilon}{S_{\mathbf{q}}}} \Pr(q_i(\tau') + \Delta_i = x_i)) \prod_{i=d'+1}^{d} \Pr(q_i(\tau') + \Delta_i = x_i) \\
&\leq e^{\epsilon} \prod_{i=1}^{d} \Pr(q_i(\tau') + \Delta_i = x_i) \\
&= e^{\epsilon} \Pr(A_{\mathbf{q}}(\tau') = \mathbf{x})
\end{aligned}$$

Similarly, we can also prove:

$$\Pr(A_{\mathbf{q}}(\tau') = \mathbf{x}) \leq e^{\epsilon} \Pr(A_{\mathbf{q}}(\tau) = \mathbf{x})$$

The theorem then follows. □

B.2 Proof of Lemma 1

PROOF. Consider a database τ where the first $\lceil (1-\delta)\lambda \rceil - 1$ transactions are precisely the alphabet $\mathcal{I}$, and the rest of them are empty. Let f be a frequent 1-itemset mining algorithm that is both (δ, η)-useful and ϵ-differentially private. Therefore, the range of f is the power set of the alphabet $\mathcal{I}$ whose cardinality is 2^n.

We claim that for any non empty set $\mathcal{S} \subseteq \mathcal{I}$, $\Pr(f(\tau) = \mathcal{S}) > 0$. We prove that claim by contradiction: assume that there exists a non empty set $\mathcal{S}'$ which is a subset of $\mathcal{I}$, and $\Pr(f(\tau) = \mathcal{S}') = 0$. We construct a database τ' by adding $2\lfloor \delta\lambda \rfloor + 2$ transactions which are exactly $\mathcal{S}'$ to τ. Therefore, for any $i \in \mathcal{S}'$, the support of i in τ' is $\lfloor (1+\delta)\lambda \rfloor + 1$, which exceeds $(1+\delta)\lambda$. For any j that is not in $\mathcal{S}'$, the support is $\lceil (1-\delta)\lambda \rceil$ - 1, which is less than $(1-\delta)\lambda$. By Definition 5, the only δ-approximate output in τ' is $\mathcal{S}'$. Hence, $\Pr(f(\tau') = \mathcal{S}') \geq 1 - \eta > 0$. By differential privacy,

$$0 < \Pr(f(\tau') = \mathcal{S}') \leq e^{(2\lfloor \delta\lambda \rfloor + 2)\epsilon} \Pr(f(\tau) = \mathcal{S}') = 0$$

a contradiction. We have thus proved our claim.

Furthermore, it is not hard to show that $\emptyset$ is the only δ-approximate output of τ, and thus, $\Pr(f(\tau) = \emptyset) \geq 1 - \eta$. By our claim, since the cardinality of the output range of f is 2^n, there must exists a non empty set $\mathcal{S}'$ such that $\Pr(f(\tau) = \mathcal{S}') \leq \eta/(2^n - 1)$. We construct a database τ' by adding $2\lfloor \delta\lambda \rfloor + 2$ transactions which are exactly $\mathcal{S}'$ to τ. By the proof of the claim, $\mathcal{S}'$ is the only δ-approximate output in τ'. Since f is (δ, η)-useful, $\Pr(f(\tau') = \mathcal{S}') \geq 1 - \eta$. By differential privacy,

$$\Pr(f(\tau') = \mathcal{S}') \leq e^{(2\lfloor \delta\lambda \rfloor + 2)\epsilon} \Pr(f(\tau) = \mathcal{S}')$$

Therefore,

$$1 - \eta \leq \frac{e^{(2\delta\lambda+2)\epsilon}\eta}{2^n - 1}$$

The lemma then follows. □

B.3 Proof of Theorem 4

PROOF. Since the cardinality of a transaction is at most ℓ, the addition (deletion) of a transaction to (from) a database can at most increase (decrease) the support of ℓ 1-itemsets simultaneously. The theorem then follows. □

B.4 Proof of Theorem 5

PROOF. Suppose that the maximal cardinality of transactions is ℓ, and $\ell = \mathrm{O}(1)$. By Theorem 4, the sensitivity of computing the support of all 1-itemsets is ℓ. Let Δ be a random variable drawn from the geometric noise $G(\epsilon/\ell)$ (1), and $\alpha = e^{\frac{\epsilon}{\ell}}$. It is not hard to show that for any integer σ,

$$\Pr(\Delta = \sigma) = \frac{\alpha - 1}{\alpha + 1} \alpha^{-|\sigma|}$$

Given a threshold λ, for any support $x \geq (1+\delta)\lambda$, we can show that:

$$\begin{aligned}
\Pr(x + \Delta \geq \lambda) &= \Pr(\Delta \geq \lambda - x) \\
&= \sum_{i=\lambda-x}^{+\infty} \frac{\alpha - 1}{\alpha + 1} \alpha^{-|i|} \\
&= 1 - \sum_{i=-\infty}^{i=\lambda-x-1} \frac{\alpha - 1}{\alpha + 1} \alpha^{-|i|} \\
&= 1 - \frac{\alpha^{\lambda - x}}{\alpha + 1} \\
&\geq 1 - \frac{\alpha^{-\delta\lambda}}{\alpha + 1} \\
&> 1 - \frac{\alpha^{-\delta\lambda+1}}{\alpha + 1} \qquad (13)
\end{aligned}$$

Similarly, for any support $y \leq (1-\delta)\lambda$, we can prove that

$$\Pr(y + \Delta < \lambda) \geq 1 - \frac{\alpha^{-\delta\lambda+1}}{\alpha + 1}$$

For any database τ, without loss of generality, let $\{1, \ldots, i\}$ be the set of items whose support exceeds $(1+\delta)\lambda$, and $\{j, \ldots, n\}$ be the set of items whose support is less than $(1-\delta)\lambda$ $(j > i)$. Therefore, the probability of the geometric noise algorithm f to output a δ-approximate result is:

$$\begin{aligned}
&\Pr(f(\tau) \text{ is } \delta\text{-approximate}) \\
&= \prod_{k=1}^{i} \Pr(\{k\}_{supp} + \Delta \geq \lambda) \prod_{k=j}^{n} \Pr(\{k\}_{supp} + \Delta < \lambda) \\
&\geq (1 - \frac{\alpha^{-\delta\lambda+1}}{\alpha + 1})^{i+n-j+1} \\
&\geq (1 - \alpha^{-\delta\lambda})^n
\end{aligned}$$

Thus, it suffices to prove that

$$(1 - e^{\frac{-\delta\lambda\epsilon}{\ell}})^n \geq 1 - \eta \qquad (14)$$

provided $\epsilon \geq \Omega(\log n)$. Let $\beta = 1 - \eta$. It is not hard to show that if

$$\epsilon \geq \frac{-\ell \ln(1 - \beta^{\frac{1}{n}})}{\delta\lambda}$$

then (14) holds. Since $0 \leq \beta \leq 1$, by Taylor expansion of $1 - \beta^k$ at $k = 0$, we have

$$1 - \beta^{\frac{1}{n}} = -\frac{\ln \beta}{n} + o(\frac{1}{n})$$

Hence,

$$-\ln(1 - \beta^{\frac{1}{n}}) \geq \Omega(\log n)$$

The theorem then follows. □

B.5 Proof of Theorem 6

PROOF. We construct a database τ as follows: the first $\lceil(1-\delta)\lambda\rceil$-1 transactions are precisely $\{1,\ldots,\ell\}$, and then the next $\lceil(1-\delta)\lambda\rceil$-1 transactions are $\{\ell+1,\ldots,2\ell\}$, and so hence so forth. In such a way, the support of each single item in τ is exactly $\lceil(1-\delta)\lambda\rceil$-1.

Suppose f is a frequent 1-itemset mining algorithm that is both ϵ-differentially private and (δ,η)-useful. By the proof in Lemma 1, for any non-empty set $\mathcal{S}\subseteq\mathcal{I}$, $\Pr(f(\tau)=\mathcal{S})>0$. Hence, there exists a set $\mathcal{S}_k\subseteq\mathcal{I}$ and $|\mathcal{S}_k|=k(1\le k\le n)$ such that

$$\Pr(f(\tau)=\mathcal{S}_k)\le\frac{1}{\binom{n}{k}}$$

Then we transform τ to τ' as follows [2]: add $2\lfloor\delta\lambda\rfloor+2$ transactions which are precisely the first ℓ items in $\mathcal{S}_k$, then add $2\lfloor\delta\lambda\rfloor+2$ transactions which are the next ℓ items in $\mathcal{S}_k$ and so hence so forth until for any item in $\mathcal{S}_k$, the support is $\lfloor(1+\delta)\lambda\rfloor+1$, and for any item not in $\mathcal{S}_k$, the support is $\lceil(1-\delta)\lambda\rceil-1$. Therefore, $\mathcal{S}_k$ is the only δ-approximate output in τ', and thus,

$$\Pr(f(\tau')=\mathcal{S}_k)\ge\eta$$

By differential privacy,

$$\Pr(f(\tau')=\mathcal{S}_k)\le e^{(2\lfloor\delta\lambda\rfloor+2)\lceil\frac{k}{l}\rceil\epsilon}\Pr(f(\tau)=\mathcal{S}_k)$$

It follows that

$$\epsilon\ge\frac{\ln[\binom{n}{k}(1-\eta)]}{(2\delta\lambda+2)\lceil\frac{k}{l}\rceil}$$

Note that $\binom{n}{k}\ge\Omega\big(n^{k-1}\big)$. The theorem then follows. □

B.6 Proof of Theorem 7

PROOF. Without loss of generality, we shall denote the transaction in τ' but not τ as t'. For any $s\in Range(f)$,

$$\begin{aligned}\Pr(f(r(\tau))=s)&=\sum_{\hat\tau\in\mathcal{I}^m}\Pr(f(r(\tau))=s|r(\tau)=\hat\tau)\Pr(r(\tau)=\hat\tau)\\&=\sum_{\hat\tau\in\mathcal{I}^m}\Pr(f(\hat\tau)=s)\Pr(r(\tau)=\hat\tau)\qquad(15)\end{aligned}$$

By Definition 7,

$$\sum_{\hat t\subseteq\mathcal{I}}\Pr(r(t')=\hat t)=1$$

It follows that:

$$\begin{aligned}\Pr(r(\tau)=\hat\tau)&=\sum_{\hat t\subseteq\mathcal{I}}\Pr(r(t')=\hat t)\Pr(r(\tau)=\hat\tau)\\&=\sum_{\hat t\subseteq\mathcal{I}}\Pr(r(\tau)=\hat\tau,r(t')=\hat t)\\&=\sum_{\hat t\subseteq\mathcal{I}}\Pr(r(\langle t',\tau\rangle)=\langle\hat t,\hat\tau\rangle)\\&=\sum_{\hat t\subseteq\mathcal{I}}\Pr(r(\tau')=\langle\hat t,\hat\tau\rangle)\qquad(16)\end{aligned}$$

Since f is ϵ-differentially private,

$$\Pr(f(\hat\tau)=s)\le e^{\epsilon}\Pr(f(\langle\hat t,\hat\tau\rangle)=s)\qquad(17)$$

[2]Here, we assume the number of transactions m is sufficiently large.

Plugging (17) and (16) into (15), let $\hat\tau'=\langle\hat t,\hat\tau\rangle$. It follows that:

$$\begin{aligned}&\Pr(f(r(\tau))=s)\\&\le\sum_{\hat\tau\in\mathcal{I}^m,\hat t\subseteq\mathcal{I}}e^{\epsilon}\Pr(f(\hat\tau')=s)\Pr(r(\tau')=\hat\tau')\\&=\sum_{\hat\tau\in\mathcal{I}^m,\hat t\subseteq\mathcal{I}}e^{\epsilon}\Pr(f(r(\tau'))=s|r(\tau')=\hat\tau')\Pr(r(\tau')=\hat\tau')\\&=e^{\epsilon}\Pr(f(r(\tau'))=s)\end{aligned}$$

The last equation comes from following: $r(\tau')$ applies the transformation r to each transaction in τ'; Since $\hat\tau$ is the transformed result of all transactions in τ' except t' and $\hat t$ is the transformed result of t', $\hat\tau'=\langle\hat t,\hat\tau\rangle$ is the transformed result of τ'. The theorem then follows. □

B.7 Proof of Theorem 8

PROOF. First, We prove "EstimateDistribution" is ϵ'-differentially private: since the addition/deletion of a single transaction t can only increase/decrease the frequency of the transactions with cardinality $|t|$ by 1, the sensitivity of computing the frequencies of the transactions with all cardinalities is 1, and thus, by Theorem 1, the function "EstimateDistribution" guarantees ϵ'-differential privacy. By Theorem 4 and Theorem 7, it is not hard to show that the code in Line 3 to Line 9 in Algorithm 2 guarantees $(\epsilon-\epsilon')$-differential privacy. Since other steps in Algorithm 2 depend on the data that have already been differentially private, by Theorem 2, we conclude that Algorithm 2 is ϵ-differentially private. □

B.8 Proof of Theorem 9

First, we prove the following lemma.

LEMMA 2. *$(2,\ell)$-sensitivity is NP-hard.*

PROOF. We prove Lemma 2 by a reduction from a known NP-hard problem "densest ℓ-subgraph" ($D\ell S$) [15]. The problem $D\ell S$ is defined as:

Instance: An undirected graph $G=(V,E)$ and a positive integer $\ell\le|V|$.

Question: Choose a subset $V'\subseteq V$ of cardinality ℓ such that the induced subgraph on V' has the maximal number of edges.

Consider an arbitrary instance of $D\ell S$ defined by a graph $G=(V,E)$ and an integer ℓ. Let $|E|=d$. We construct an instance $P=(\mathcal{I},\mathcal{C})$ of $(2,\ell)$-sensitivity as follows: for each vertex v_i in $V=\{v_1,\ldots,v_n\}$, v_i is in one-to-one correspondence to the item i, and thus, the alphabet $\mathcal{I}=\{1,\ldots,n\}$. Then, we construct the itemsets $\mathcal{C}$: $\forall v_i,v_j$, if $(v_i,v_j)\in E$, then $\{i,j\}\in\mathcal{C}$. Clearly, this construction can be carried out in polynomial time. By our construction, for any set of vertices $V'\subseteq V$, let $\mathcal{I}'$ be the set of the correspondent items, and then the number of edges in the induced graph of V' is precisely the number of itemsets contained in $\mathcal{I}'$. Thus, for a solution $\mathcal{T}$ of $(2,\ell)$-sensitivity on the instance $P=(\mathcal{I},\mathcal{C})$, the corresponding vertices of $\mathcal{T}$ are clearly the $D\ell S$ on G. It is not hard to show the reverse also holds. Since the $D\ell S$ problem is NP-hard, and we can encode that problem into an instance of $(2,\ell)$-sensitivity, $(2,\ell)$-sensitivity is also NP-hard. □

We prove Theorem 9 by a reduction from $(2, \ell)$-sensitivity to an instance of (k, ℓ)-sensitivity: Consider an arbitrary instance of $(2, \ell)$-sensitivity, $P = (\mathcal{I}, \mathcal{C})$. Let $|\mathcal{I}| = n$, and we construct an instance $P' = (\mathcal{I}', \mathcal{C}')$ of $(k, \ell+k-2)$-sensitivity $(k > 2)$ by adding $k-2$ items to the alphabet $\mathcal{I}$ and each itemset in $\mathcal{S}$. More precisely, let $\Delta = \{a_1, \ldots, a_{k-2}\}$ which is disjoint with $\mathcal{I}$, and $\mathcal{I}' = \mathcal{I} \cup \Delta$, and $\mathcal{C}' = \{C_i' | C_i' = C_i \cup \Delta \wedge C_i \in \mathcal{C}\}$. Clearly, that construction can be carried out in polynomial time. By our construction, it is not hard to show that a solution of $(k, \ell+k-2)$-sensitivity corresponds to a solution of $(2, \ell)$-sensitivity. Since $(2, \ell)$-sensitivity is NP-hard, and we can encode that problem into an instance of (k, ℓ)-sensitivity $(k > 2)$, Theorem 9 then follows.

B.9 Proof of Theorem 10

PROOF. The number of i-itemsets contained by a transaction of cardinality ℓ is $\binom{\ell}{i}$. Thus, the addition (deletion) of such a transaction can at most increase (decrease) the support of $\binom{\ell}{i}$ i-itemsets by one simultaneously. The theorem then follows. □

B.10 Proof of Theorem 16

PROOF. As shown in the proof of Theorem 8, the method TRUNCATEDATABASE is ϵ''-differentially private, and the computation of frequent 1-itemsets guarantees $(\epsilon' - \epsilon'')$-differential privacy. By Theorem 1 and Theorem 10, it is not hard to see the method "NAÏVEFKIM" is ϵ'-differentially private, which is called $k-1$ times in the loop. By Theorem 2, since $\epsilon'' + \epsilon' - \epsilon'' + \epsilon' * (k-1) = \epsilon$, the theorem then follows. □

B.11 Proof of Theorem 11

PROOF. The proof mirrors that in Theorem 9 by a reduction from the heaviest ℓ-subgraph [15]. □

B.12 Proof of Theorem 12

PROOF. By Bayesian rules,

$$\Pr(\hat{\theta}|\theta') = \frac{\Pr(\theta'|\hat{\theta}) \Pr(\hat{\theta})}{\Pr(\theta')}$$

By assuming a uniform prior on $\Pr(\hat{\theta})$, the theorem then follows from the geometric distribution in (1). □

B.13 Proof of Theorem 13

PROOF. For the computation of frequent i-itemsets $(i = 2, \ldots, k)$, the function "SMARTTRUNCATING" relies on the noisy results of $(i-1)$-itemsets, which have already been differentially private. Thus, it is safe to call the function "SMARTTRUCATING." Furthermore, the computations of both "avg_supp" and "max_supp" utilize the frequencies of transactions by different cardinalities, which have been computed in the method "TRUNCATEDATABASE", which have already been differentially private. Therefore, it is also safe to compute "avg_supp" and "max_supp." By a similar proof of Theorem 16, the theorem follows. □

B.14 Proof of Theorem 15

PROOF. By combining Theorem 13, Theorem 17 and the decomposition property of differential privacy, the theorem then follows. □

B.15 Proof of Theorem 17

PROOF. By the analysis of binary search, the code in Line 12 in Algorithm 9 is called $\lceil \log n \rceil - 1$ times, each of which guarantees ϵ'-differential privacy, and the code in Line 5 is called once, which also guarantees ϵ'-differential privacy. Since $\lceil \log_2 n \rceil * \epsilon' = \epsilon$, by Theorem 2, the theorem then follows. □