IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 16, 2018, accepted May 10, 2018, date of publication May 23, 2018, date of current version June 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2839752

Frequent Itemsets Mining With Differential Privacy Over Large-Scale Data

XINYU XIONG[1], FEI CHEN [1,2], PEIZHI HUANG[1], MIAOMIAO TIAN[3], XIAOFANG HU[4], BADONG CHEN[5], AND JING QIN[2]

[1]College of Computer Science and Engineering, Shenzhen University, Shenzhen 518060, China
[2]Center of Smart Health, School of Nursing, The Hong Kong Polytechnic University, Hong Kong
[3]School of Computer Science and Technology, Anhui University, Hefei 230039, China
[4]School of Computer and Information Science, Southwest University, Chongqing 400716, China
[5]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China

Corresponding author: Fei Chen (fchen@szu.edu.cn)

This work was supported in part by the Innovation and Technology Fund of Hong Kong under Project ITS/304/16, in part by The Hong Kong Polytechnic University under Project 1-ZE8J, in part by the National Natural Science Foundation of China under Grants 61502314 and 61601376, in part by the Science and Technology Plan Projects of Shenzhen under Grants JCYJ20160307115030281 and JCYJ20170302145623566, in part by the Fundamental Science and Advanced Technology Research Foundation of Chongqing under Grant cstc2016jcyjA0547, in part by the Chongqing Postdoctoral Science Foundation Special Fund under Grant Xm2017039, and in part by the Doctoral Foundation of Southwest University under Grant SWU116005.

ABSTRACT Frequent itemsets mining with differential privacy refers to the problem of mining all frequent itemsets whose supports are above a given threshold in a given transactional dataset, with the constraint that the mined results should not break the privacy of any single transaction. Current solutions for this problem cannot well balance efficiency, privacy, and data utility over large-scale data. Toward this end, we propose an efficient, differential private frequent itemsets mining algorithm over large-scale data. Based on the ideas of sampling and transaction truncation using length constraints, our algorithm reduces the computation intensity, reduces mining sensitivity, and thus improves data utility given a fixed privacy budget. Experimental results show that our algorithm achieves better performance than prior approaches on multiple datasets.

INDEX TERMS Frequent itemsets mining, differential privacy, sampling, transaction truncation, string matching.

I. INTRODUCTION

In recent years, with the explosive growth of data and the rapid development of information technology, various industries have accumulated large amounts of data through various channels. To discover useful knowledge from large amounts of data for upper-layer applications (e.g. business decisions, potential customer analysis, etc.), data mining [1]–[9] has been developed rapidly. It has produced a positive impact in many areas such as business and medical care.

Along with the great benefits of these advances, the large amount of data also contains privacy sensitive information, which may be leaked if not well managed. For instance, smart phone applications are recording the whereabouts of users through GPS sensors and are transferring the data to their servers. Medical records are also storing potential relationships between diseases and a variety of data. Mining on user location data or medical record data both provide invaluable information; however, they may also leak user privacy. Thus mining knowledge under confident privacy guarantees is highly expected.

This paper investigates how to mine frequent itemsets with privacy guarantee for big data. We consider the following application scenario. A company (such as information consulting firm) has a large-scale dataset. The company would like to make the dataset public and therefore allow the public to execute frequent itemsets mining for getting cooperation or profits. But due to privacy considerations, the company cannot provide the original dataset directly. Therefore, privacy mechanisms are needed to process the data, which is the focus of this paper.

To ensure privacy of data mining, traditional methods are based on k-anonymity and its extended models [10]–[16]. These methods require certain assumptions; it is difficult to protect privacy when the assumptions are violated. The insufficiency of k-anonymity and its extended models is that there is no strict definition of the attack model, and that the knowledge of the attacker cannot be quantitatively defined. To pursue strict privacy analysis, Dwork [17] proposed a strong privacy protection model called differential privacy. This privacy definition features independence of background knowledge of the attacker and proves very useful.

Frequent pattern mining with privacy protection has also received extensive attention. As preliminary methods [18]–[24], these works have provided a lot of contributions in this area. But with the advance of research, these privacy methods have not been able to provide effective privacy. In order to overcome these difficulties, researches began to focus on the differential privacy protection framework [25]–[31]. Although guaranteeing privacy temporary, however, the balance between privacy and utility of frequent itemsets mining results needs to be further pursued.

In this paper, we propose a novel differential private frequent itemsets mining algorithm for big data by merging the ideas of [27] and [30], which has better performance due to the new sampling and better truncation techniques. We build our algorithm on FP-Tree for frequent itemsets mining. In order to solve the problem of building FP-Tree with large-scale data, we first use the sampling idea to obtain representative data to mine potential closed frequent itemsets, which are later used to find the final frequent items in the large-scale data. In addition, we employ the length constraint strategy to solve the problem of high global sensitivity. Specifically, we use string matching ideas to discover the most similar string in the source dataset, and implement transaction truncation for achieving the lowest information loss. We finally add the Laplace noise for frequent itemsets to ensure privacy guarantees.

A few challenges exist: First, how to design a sampling method to control the sampling error? We use the central limit theorem to calculate a reasonable sample size to control the error range. After obtaining the sample size, the dataset is randomly sampled using a data analysis toolkit. The second challenge is how to design a good string matching method to truncate the transaction without losing information as far as possible? We match the potential itemsets in the sample data to find the most similar items and then merge them with the most frequent items until the maximum length constraint is reached.

As a result, our algorithm reduces the computation intensity and addresses high sensitivity of frequent itemsets mining. The performance is also guaranteed. Through the analysis of privacy, our algorithm achieves ϵ-differential privacy. Experiment results using multiple datasets showed that our algorithm achieves better performance than prior approaches.

To summarize, we make the following contributions:

- We propose a differentially private big data frequent itemsets mining algorithm with high utility and low computational intensity. The algorithm guarantees the trade-off between data utility and privacy.
- We achieve high data utility by employing the large-scale data sampling and length constraint strategy, reducing the number of candidate sets of frequent itemsets and the global sensitivity. Experimental results demonstrated the data utility.
- We conduct formal privacy analysis. The proposed algorithm achieves ϵ-differential privacy.

The rest of this paper is organized as follows: Section II discusses related works. Section III introduces background knowledge about differential privacy and basic tools that to be used. Section IV presents the proposed algorithm to mine top k frequent itemsets with differential privacy. Section V gives the analysis. Section VI shows the performance evaluation on multiple datasets. Section VII finally concludes our work.

II. RELATED WORK

The privacy issue of frequent itemsets mining is a main focus of research efforts. We categorize relevant work based on the underlying techniques - from anonymity to differential privacy.

A. ANONYMITY APPROACHES

For distributed datasets, Kantarcioglu and Clifton [18] proposed a secure multi-party privacy-protecting association rule mining algorithm. The idea is to transform the problem into a secure multi-party computation problem under horizontal distribution. Vaidya and Clifton [19] proposed a privacy-preserving association rule algorithm that uses secure scalar calculation method to find all frequent itemsets under vertical distribution. Teng and Du [20] proposed a hybrid privacy-preserving algorithm under vertical distribution.

For centralized datasets, Wong *et al.* [21] proposed to employ 1-to-n encryption method to change original itemsets in order to protect data privacy when outsourcing frequent itemsets mining. Ling *et al.* [22] proposed an algorithm that transforms business information into very long binary vector and a series of random mapping functions based on bloom filters. Later, Tai *et al.* [23] proposed a k-support anonymity based frequent itemsets mining algorithm. All these methods above sacrifice the precision of mining result.

B. DIFFERENTIAL PRIVACY APPROACHES

Because traditional approaches are based on heuristics, a solid privacy guarantee is missing. Therefore, researchers began to investigate frequent itemsets mining with differential privacy. Bhaskar *et al.* [25] presented two mining algorithms, which are representatives of frequent itemsets mining with differential privacy. Later, in order to solve the high dimensional challenge of dataset, Li *et al.* [26] proposed the *PrivBasis* algorithm that combines θ-basis and mapping technique to achieve top-k frequent itemsets mining. Zeng *et al.* [27] proposed a greedy method of transaction truncation approach by limiting the maximum length of transactions of dataset.

Besides researches in the interactive framework, differentially private frequent itemsets mining is also studied in the non-interactive framework [28]–[30]. Han *et al.* [28] focused on the issue of top-k query privacy in MapReduce.

Chen *et al.* [29] proposed a method that employs a context-free classification tree and combines a top-down tree partitioning method to publish a dataset. Lee and Clifton [30] proposed a method of using the prefix tree to privately publish frequent itemsets. Su *et al.* [31] proposed a cryptographic algorithm that divides the dataset based on the high global sensitivity. Despite all these work, there are still rooms for balancing utility and privacy, which is our work here.

In addition to the above general researches, domain-specific frequent itemsets mining with differential privacy is also studied. Chen *et al.* [32] proposed the top-down prefix tree to publish the trajectory dataset. Chen *et al.* [33] also proposed a method for publishing sequence dataset based on variable-length N-gram models. Bonomi and Xiong [34] analyzed the both above algorithms and proposed a two-phase algorithm to improve performance. For solving the problem of large frequent sequence candidate sets, Xu *et al.* [35] presented to shrink and convert dataset, which reduces the number of candidate sets to improve data utility. Shen and Yu [36] focused on the issue of publishing map dataset. Xu *et al.* [37] studied mining frequent subgraphs with differential privacy in a complex large graph based on constructing directed lattices.

III. PRELIMINARIES

A. DIFFERENTIAL PRIVACY

Differential privacy as a new type of privacy definition is proposed for the privacy of statistical databases by Dwork [17]. It defines a very strict attack model, and gives a rigorous, quantitative representation and proof for the risk of privacy disclosure.

Definition 1 (ϵ-Differential Privacy): Let D and D' denote any databases which differ by at most one record, $Range(K)$ represent the range of a random function K. If a random function K satisfies ϵ-differential privacy, for any $S \subseteq Range(K)$, we have

$$\Pr\left[K\left(D\right) \in \ S\right] \leq \exp\left(\epsilon\right)\Pr\left[K\left(D'\right) \in \ S\right] \tag{1}$$

where ϵ is a real number denoting the privacy budget parameter.

The smaller the ϵ is, the higher the degree of privacy is preserved. The differential privacy protection is achieved by adding quantitative noise; the amount of required noise depends on the sensitivity. Intuitively, the sensitivity quantifies the change of the query results caused by deleting any transaction in the dataset.

Definition 2 (Sensitivity): Given any function: $D^n \rightarrow \mathbb{R}^k$, denote $\triangle f$ as the sensitivity of f; it is defined as follows: for all neighboring databases (i.e., differs only in one row) D and D'

$$\triangle f = \max_{D,D'} \|f\left(D\right) - f\left(D'\right)\|_1 \tag{2}$$

The sensitivity magnitude of the function is determined by the function itself; different functions have different sensitivities. For most query functions f, the value of $\triangle f$ is relatively small. The sensitivity is then used to control the noise level in differential privacy. When the noise is too large, it affects data utility. It is worth noting that sensitivity is independent of the dataset.

B. NOISE MECHANISM

The main technique of achieving differential privacy protection is to add noise. Dwork proposed the Laplace mechanism to achieve differential privacy. For different situations, the exponent mechanism, geometric mechanism and Gaussian mechanism were also proposed. The commonly used noisy addition mechanisms are the Laplace mechanism and the Exponential mechanism. The Laplace mechanism is usually for mining algorithms that output numeric result; the exponential mechanism is mainly applied to algorithms that output non-numerical results.

The amount of noise is affected by the sensitivity and the privacy budget. Generally, the privacy budget is set in advance, then the noise is determined by the sensitivity.

Definition 3 (Laplace Distribution): The probability density function of the Laplace distribution with scale parameter λ is defined as:

$$Pr\left(x|\lambda\right) = \frac{1}{2\lambda}e^{-|x|/\lambda} \tag{3}$$

Theorem 1 (Laplace Mechanism): Let $f : D^n \rightarrow \mathbb{R}$ be a function with image over real number values. The following mechanism K satisfies ϵ-differential privacy.

$$K\left(D\right) = f\left(D\right) + \mathrm{Lap}\left(\frac{\triangle f}{\epsilon}\right) \tag{4}$$

where $\mathrm{Lap}\left(\frac{\triangle f}{\epsilon}\right)$ *is a noise with the Laplace distribution.*

The noise size is proportional to $\triangle f$ and is inversely proportional to ϵ.

Composability Theorems: In general, a complex privacy-preserving algorithm requires multiple application of different differential privacy mechanisms. In this case, in order to ensure that the whole process satisfies ϵ-differential privacy, it is necessary to allocate the privacy budget reasonably. The composability theorems of differential privacy guarantee the overall privacy.

Theorem 2 (Sequential Composition): Given a fixed dataset, let $\{A_1, A_2, \ldots, A_n\}$ be n mechanisms where each A_i provides ϵ-differential privacy. A sequential application of each mechanism provides $\sum_{i=1}^{n} \epsilon_i$—differential privacy.

Theorem 3 (Parallel Composition): Given disjoint datasets, let $\{A_1, A_2, \ldots, A_n\}$ be n mechanisms where each A_i provides ϵ-differential privacy. A parallel application of each mechanism provides $\max\left(\epsilon_i\right)$*—differential privacy.*

C. FREQUENT ITEMSETS MINING

We now briefly introduce frequent itemsets mining. Let $T_I = \{t_1, t_2, \ldots, t_N\}$ be a transactional dataset consisting of N transactions, $I = \{i_1, i_2, \ldots, i_n\}$ be a set of different items, and X be a subset of I such that $X \subseteq I$. If X is contained in a transaction and X has k items, X is called a k-itemset. The support of an itemset is defined as the total number of transactions that contains the itemset.

The task of frequent itemsets mining is to find all itemsets that have support greater than a given threshold. Frequent itemsets is employed for finding association rules for a group of data items. Association rules show correlational relations of different items, which have numerous practical application [38], [39]. Association rule generation is usually split up into two separate steps: 1) a minimum support threshold is applied to find all frequent itemsets in a database; 2) a minimum confidence constraint is applied to these frequent itemsets in order to form rules. While the second step is straightforward, the first step needs more attention. Finding all frequent itemsets in a database is challenging because it involves searching all possible itemsets. The representative algorithms for mining frequent itemsets include the Apriori algorithm [38] and the FP-Growth algorithm [39].

We describe the basics of the FP-Growth algorithm [39] which underlies our proposed privacy-preserving algorithm. The FP-Growth algorithm features small database scanning operations: it only has two-pass database scanning. In the first pass, the algorithm counts occurrence of each item (attribute-value pairs), and stores them to a header table in descending order. It also builds the NULL FP-Tree. In the second pass, it inserts the FP-Tree with data items and stores their frequency. Items in each instance that do not meet minimum support threshold are discarded. The final core data structure ''FP-Tree'' stores all the information for frequent itemsets. Finally, all frequent itemsets can be mined from the FP-Tree.

D. CENTRAL LIMIT THEOREM

In our scheme, we use the central limit theorem for reasonable sampling.

Theorem 4: Let $\{X_1, X_2, , X_n\}$ be a sequence of independent and identically distributed random variables, whose expectation is μ and variance is σ^2, a finite value. Let $Y_n = \frac{\sum_{k=1}^{n} X_k - n\mu}{\sqrt{n}\sigma}$ be a random variable. Then, the distribution function $F_n(x)$ of Y_n satisfies the following:

$$\lim_{n\to\infty} F_n(x) = \lim_{n\to\infty} P\left\{\frac{\sum_{k=1}^{n} X_k - n\mu}{\sqrt{n}\sigma} \leq x\right\} = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \tag{5}$$

That is, when n is sufficiently large, the distribution approximately follows the normal distribution. For random sampling, with the increasing of sample size, the distribution of the sampling average also tends to be the normal distribution. In our proposed algorithm, we employ this theorem to determine our sampling strategy.

E. PROBLEM STATEMENT

Finally, we state our problem explicitly. Given a large-scale dataset, a privacy budget ϵ, and a minimum threshold σ, the task is to design a privacy-preserving algorithm that mines top k frequent itemsets whose supports are not less than the threshold σ, where k is an arbitrary given number. The algorithm should have minimum computational cost and high mining result utility, besides satisfying ϵ-differential privacy.

IV. PROPOSED ALGORITHM

A. A STRAWMAN APPROACH

In order to better understand the challenges posed by differential privacy, we first discuss a basic approach. That is, first generate all the candidate itemsets, then add noise to the support of all candidate itemsets directly, and finally select the top k frequent itemsets above a given threshold.

We discuss the privacy of the above basic approach. Assume that L_f is the maximum length of the frequent itemsets, C_n^i is the number of all i-itemset, and n is the alphabet size. Then the sensitivity of the i-itemset's support is C_n^i. Assuming the privacy budget is distributed evenly, the privacy budget for each i-itemset's support is ϵ/L_f. Then, by adding noise Lap$\left(\frac{C_n^i \times L_f}{\epsilon}\right)$, the basic approach satisfies ϵ/L_f-differential privacy for each i-itemset, $1 \leq i \leq L_f$. Combining the sequential composition properties, the basic approach satisfies ϵ-differential privacy.

While achieving ϵ-differential privacy, the drawback is that the utility of the basic approach is very low. This is because the noise Lap$\left(\frac{C_n^i \times L_f}{\epsilon}\right)$ is significantly large that it makes the mining results far from accurate.

B. OVERVIEW

We now describe our newly proposed algorithm, called *DP-FIM*, which merges the ideas of [27] and [30], but employs a different(better) truncation scheme and boosts computation efficiency using both sampling and truncation. Compared with previous work using random truncation, our new string-similarity-matching-based truncation mechanism has better performance than previous work [27], [30], which is because string-similarity-matching-based truncation preserves more useful frequent itemset candidates. The experimental results in Section VI-B also confirms the better performance. The algorithm is differentially private; it takes a threshold value σ and outputs the frequent itemsets with support at least σ. The basic idea is as follows: first, compute a noisy support for the threshold $\tilde{\sigma} = \sigma + \text{Lap}(\cdot)$, then truncate the original database noisily, finally construct a noisy FP-Tree for mining frequent itemsets.

Algorithm 1 describes the high-level process of our proposed algorithm. It consists of three phases: the preprocessing phase, the mining phase, and the perturbing phase.

1) PREPROCESSING PHASE

Given the large-scale dataset, we first sample the dataset and then compute the closed frequent itemsets in the smaller sample using a traditional frequent itemsets mining algorithm. We later estimate the length distribution of the sampled dataset and obtain the maximum length constraint, which is later used to shrink the dataset. Some elements out of the

```
Algorithm 1 DP-FIM
Input: : database $D$, threshold $\sigma$, privacy budget $\epsilon = \epsilon_1 + \epsilon_2$,
    item universe $I$
Output: frequent item sets $\tilde{F}$ and their noisy frequencies
sample a smaller database
    $D_1 \leftarrow$ TransformDatabase($D$)
compute the closed frequent itemsets $\mathcal{L}$ and a maximal
    length constraint
    $l_{max} \leftarrow$ FindFrequentItemSets($D_1, I, \sigma, \eta$)
    using a parameter $\eta$
shrink the original database
    $D^S \leftarrow$ TruncateDatabase($D$, $\mathcal{L}$, $l_{max}$)
construct a noisy FP-Tree
    $T \leftarrow$ BuildNoisyFPTree($D_S, \epsilon_1, l_{max}$)
compute the frequent itemsets $\tilde{F}$ and their noisy frequen-
    cies using $\epsilon_2$ by perturbation
return $\tilde{F}$ and their noisy frequencies
```

closed frequent itemsets are removed from the source dataset if their supports are below the support threshold. We then employ string matching ideas to cut off the transactions in the dataset; in this step, the purpose of converting the dataset is to shrink the data size and simultaneously retain the potential frequent items.

2) MINING PHASE

We then build a noisy FP-Tree over the shrunken dataset. We distribute the privacy evenly; we also add noise to the real count results.

3) PERTURBING PHASE

Add Laplace noise in the candidate frequent itemsets and output them.

We explain some intuitions behind the proposed algorithm. To improve mined result utility, it is necessary to reduce the amount of noise added. The amount of noise depends on the privacy budget and the sensitivity of the underlying components of the mining function. Given that the privacy budget is set in advance, it is key to reduce the sensitivity. According to the definition of sensitivity, the sensitivity of k-itemset's support relates to $\left|C_k^l\right|$, where C_k^l is the set of all k-itemsets in all transactions with l-length. Thus, we can reduce the sensitivity by constraining the length of each transaction. Specifically, we use the string matching and the longest common subsequence idea to perform transaction truncation. That is, we find the most similar potential itemsets in the source dataset, and at the same time achieve the lowest loss of information, which improves data utility.

In this paper, the privacy budget is mainly allocated to the mining phase and the perturbing phase. Let $\epsilon = \mu\epsilon + (1-\mu)\,\epsilon = \epsilon_1 + \epsilon_2$. The value of μ affects the performance of our proposed algorithm. Different privacy budget assignment strategy may affect the accuracy of the algorithm results.

C. PREPROCESSING PHASE

At this phase, we first sample the dataset to have a rough estimation of the dataset using the central limit theorem. We first compute the sample size and then use SAS data analysis software for random sampling. The samples can reduce the computational intensity of the constructed FP-Tree and find the potential frequent itemsets of the source dataset. Similar to [27], we obtain a maximum length constraint l_{max} to shrink the transactions in the dataset.

We deduce the sample size now. Fix an item modeled as a *binomial distribution* with occurring probability p. Let $q = 1-p$, n be the sample size, and f_n be the occurrences of the item. The normal practice is to make the absolute error $|\frac{f_n}{n} - p|$ not more than a small positive δ with its confidence not less than an α value ($0 < \alpha < 1$). Then in order to achieve reliable sampling, the value of n should satisfy that $\Pr[|\frac{f_n}{n} - p| \leq \delta] \geq \alpha$. We compute the probability as

$$\begin{aligned}
&\Pr\left[|\frac{f_n}{n} - p| \leq \delta\right] \\
&\quad = \Pr\left[-\sqrt{\frac{n}{pq}}\delta \leq \frac{f_n - np}{\sqrt{npq}} \leq \sqrt{\frac{n}{pq}}\delta\right] \\
&\quad \approx 2\Phi\left(\sqrt{\frac{n}{pq}}\delta\right) - 1 \geq a \\
&\quad \Rightarrow \Phi\left(\sqrt{\frac{n}{pq}}\delta\right) \geq \frac{a+1}{2}
\end{aligned} \tag{6}$$

Let Z_α be the value such that

$$\Phi\left(Z_\alpha\right) \geq \frac{\alpha+1}{2} \tag{7}$$

where Z_α can be directly found on any normal distribution table. From Equations 6 and 7, we have that n should satisfy $\sqrt{\frac{n}{pq}}\delta \geq Z_a$. Therefore, we have $n \geq \frac{{Z_a}^2}{4\delta^2}$.

In practice, the common confidence level is 95% and 99%, corresponding to the Z_α value being 1.96 and 2.58. Therefore, in the maximum tolerance error of 1% case with confidence of 95% or 99%, the corresponding sample size calculated is 9604 and 16641. That is, for large-scale datasets, we only need to deal with the sample, which can achieve the same accuracy. In the experiments, we confirm the effectiveness of the above sampling approach. The works [40], [41] also demonstrated the effectiveness of the sampling approach in association rule mining.

Using the sampled dataset, we then employ a classical frequent itemsets mining algorithm (here using the Apriori algorithm) to obtain the set of closed frequent itemsets $\mathcal{L}$, which is a maximal set in the sense that no super set satisfies the support threshold requirement. We list the details in Algorithm 2.

Algorithm 2 FindFrequentItemSets(D_1, I, σ, η)

```
Input: sampled data set $D_1$, threshold $\sigma$, item universe $I$,
    truncation percentage variable $\eta$
Output: closed frequent item sets $\mathcal{L}$, and a maximal length
    constraint $l_{max}$
let $\mathcal{L} = \varnothing$
invoke the Apriori algorithm: for all 1-item set $L_1$ in the $D_1$ do
if $L_1.support \geq \sigma$ then
    $k$=2, $C_1 = L_1$
    $C_k$ = all candidate $k$-item sets from $C_{k-1}$
    for each transaction $t$ in $D_1$ do
        $C_t$ = all subsets of $C_k$ contained in the transaction $t$
        for each candidate $C$ in $C_t$ do
            $C.support++$
        end for
    end for
    if $C.support \geq \sigma$ then
        add $C$ to $L_k$
        $\mathcal{L} += L_k$
    end if
    $k++$
end if
estimate distribution of $D_1$, getting the distribution
    $\{z_1, \ldots, z_i, \ldots, z_n\}$, where $z_i$ is the number of transactions with length $i$ in $D_1$
$l_{max}$ = the smallest integer such that $(\sum_{i=1}^{l} z_i)/|D_1| \geq \eta$
obtain all closed frequent item sets $\mathcal{L}$ and the maximal
    length constraint $l_{max}$.
return $\mathcal{L}$, $l_{max}$
```

In Algorithm 2, we use the sampled dataset to find the potential itemsets in the source dataset, without paying attention to the support count. Again, note that the closed frequent item sets is that if there exists superset of itemset X with support at least the threshold, then the itemset X is not a closed frequent itemset.

For the value of maximum length constraint l_{max} in Algorithm 2, we refer to [27] to estimate the distribution of transaction length $\{z_1, \ldots, z_i, \ldots, z_n\}$ in a sampled dataset using a heuristic method. That is, starting from the itemset with length 1, incrementally calculate z_i and summarize until the formula $(\sum_{i=1}^{l} z_i)/|D_1| \geq \eta$ is satisfied. We get the smallest i value as a maximum length constraint l_{max}. We note that the maximum length constraint l_{max} value affects the performance of the proposed algorithm. Thus the value of η is also very important. In the experimental evaluation section, we will discuss the effect of η on our algorithm.

Algorithm 3 TruncateDatabase(D, $\mathcal{L}$, l_{max})

```
Input: database $D$, closed frequent item sets $\mathcal{L}$, maximal
    length constraint $l_{max}$
Output: shrunken database $D^S$
for all 1-item set $L_1$ with frequencies $\geq \sigma$ in the alphabet $I$ do
    sort $L_1$ in decreasing order according to $D$
end for
for each potentially item set $X \in \mathcal{L}$ do
    generate the set of contained items $S$
    $\mathcal{L}' +=$ decreasing order $(X, L_1)$
end for
let $D^S = \varnothing$
for each transaction $t \in D$ do
    add $t' =$ TruncateTransaction($l_{max}, t$) to $D^S$
end for
return shrunken database $D^S$

function TruncateTransaction($l_{max}, t$)
$t' = t \cap S$
if $|t'| > l_{max}$ then
    truncate each transaction
    return $t' =$ StringMatching($l_{max}, t'$)
else
    return $t'$
end if
end function

function StringMatching($l_{max}, t'$)
given $t'$, find the most similar item set $L_k$ from $\mathcal{L}$
select $t'' = L_k \cap t'$
if $|t''| = l_{max}$ then
    return $t' = t''$
else
    add the most frequent ($l_{max} - |t''|$)-item set to $t''$
    return $t' = t''$
end if
end function
```

Algorithm 3 describes the detailed process by which we get the shrunken dataset. Given the closed frequent itemsets $\mathcal{L}$ and the length constraint l_{max}, we compute all 1-itemsets L_1 with the support $\geq \sigma$, and sort the itemset in descending order by support. For each transaction in the original dataset, each transaction is sorted and transformed according to the length constraint l_{max}. The unnecessary elements are eliminated, finally obtaining the shrunken dataset. Given a transaction $t = \{i_1, i_2, \ldots, i_n\}$ that violates the length constraint, we employ the closed frequent itemsets set to find *the most similar* string to execute truncation using the idea of string matching. We keep the frequent itemsets as much as possible in the truncated transactions. Specifically, we use the longest common subsequence(LCS) algorithm [42] to find the most similar truncated transaction. The intuition is that it can preserve maximal useful information.

Based on the truncated dataset obtained in the preprocessing phase, the size is greatly reduced. We note that the use of length constraint to cut off the dataset leads to the loss of some information. We try to lose the lowest information. In the experimental section, we further discuss it.

D. MINING PHASE

Algorithm 3 describes the detailed algorithm. After the preprocessing phase, we get the shrinking dataset which has smaller number of transactions and smaller dimension to build a noisy FP-Tree. Because computing support directly destroys the privacy, we initialize the FP-Tree with count Lap$\left(\frac{l_{max}}{\epsilon_1}\right)$ on each node, and then iteratively update the count. In the process of building the tree, the privacy budget adopts the average allocation strategy, and the privacy budget ϵ_1 is based on the depth h of the FP-Tree (i.e. the optimal item set length). Each level is allocated the budget of ϵ_1/l_{max}, adding Laplace noise Lap$\left(\frac{\Delta f}{\epsilon_1/l_{max}}\right)$. For the noisy FP-Tree, each item of the transaction corresponds to a node. Thus, when a transaction (single path) in a tree is removed or added, only one path is changed, so the sensitivity at this phase is

$$\triangle f = \max_{D,D'} \|f(D) - f\left(D'\right)\|_1 = 1 \tag{8}$$

Algorithm 4 BuildNoisyFPTree (D^S, ϵ_1, l_{max})

```
Input: shrunken database $D^S$, privacy budget $\epsilon_1$, maximal
    length constraint $l_{max}$
Output: noisy FP-Tree $T$ and $\mathcal{F}$
scan the transaction dataset $D^S$; get the set of frequent
    items $V$ and its support for each item; sort all the frequent
    items in $V$ in descending order which is denoted as $L$
insert a virtual root $R(T)$ to FP-Tree $T$
let $\bar{\epsilon} = \frac{\epsilon_1}{l_{max}}$
for each transaction $t$ in dataset $D^S$ do
    for each item $u$ in $t$ sorted using the order of $L$ do
        initialize the count of each node with Lap($\bar{\epsilon}$)
        create a possible new node $v$ as $u$'s child
        iteratively update the count for $v$ with $\tilde{c}(v) =$
            $support(v) +$ Lap($\bar{\epsilon}$)
        if $\tilde{c}(v) \geq \tilde{\sigma}$ then
            add $v$ to $T$ as $u$'s child
        end if
    end for
end for
obtain the noisy FP-Tree $T$
generate all top $k$ frequent itemsets $\mathcal{F}$ by FP-Growth
    algorithm
return $T$ and $\mathcal{F}$
```

E. PERTURBING PHASE

This phase is simple, relatively. Based on the noisy FP-Tree, mine all the frequent itemsets that satisfy the threshold $\tilde{\sigma}$, select all the top k frequent itemsets, add the noise Lap$\left(\frac{|\mathcal{C}|}{\epsilon_2 n}\right)$ where $\mathcal{C}$ contains the final candidates sets and n represents the size of the dataset. Finally, output the result.

V. ANALYSIS

The algorithm consists of three phases: the preprocessing phase, the mining phase, and the perturbing phase. For the preprocessing phase, it is irrelevant with privacy disclosure to calculate the support in the source dataset. We give the results of the privacy analysis in the latter two phases. The privacy guarantee of the entire algorithm is given by the sequential composition property of differential privacy.

Theorem 5: The proposed scheme achieves ϵ-differential privacy.

Proof: In the mining phase, the noisy FP-tree is constructed by the truncated dataset. In order to reduce the amount of noise added in the tree contruction process, the length constraint is used to obtain the truncated dataset. Since an itemset with a length of l_{max} can get $C_{l_{max}}^k$ different k-itemsets, it is crucial to achieve sensitivity 1 when increasing or deleting a transaction. This is achieved in the noisy FP-Tree. When removing or adding a transaction (single path) in the tree, it only changes 1 path for the overall value, then the sensitivity at this stage is 1.

Therefore, it satisfies ϵ_{11}-differential privacy to add Lap$\left(\frac{\Delta f}{\epsilon_{11}}\right)$ to the support of each node of the noisy FP-Tree, where $\epsilon_{11} = \epsilon_1/l_{max}$. Because the above process repeats l_{max} times, the whole process then satisfies $\epsilon_1/l_{max} * l_{max} = \epsilon_1$-differential privacy.

In the perturbing phase, the noise is added to the true support of the selected frequent itemsets to achieve the purpose of privacy protection. In this case, the sensitivity of the support of $\left|\mathcal{F}'\right|$ is $\Delta f = \left|\mathcal{F}'\right|$. When increasing or deleting a transaction, it affects most of the support of $\left|\mathcal{F}'\right|$ frequent itemsets by 1. Therefore, it satisfies ϵ_2-differential privacy when adding Lap$\left(\frac{|\mathcal{F}'|}{\epsilon_2}\right)$ to the support of $\left|\mathcal{F}'\right|$.

According to the sequential composition theorem 2, the *DP-FIM* algorithm satisfies ϵ-differential privacy where $\epsilon = \epsilon_1 + \epsilon_2$. □

The DP-FIM algorithm also preserves high utility. Both sampling and truncation do not hurt the frequent items. We show the utility in the following experimental evaluations.

VI. EXPERIMENTS

In this section, we evaluate the performance of our algorithm. To illustrate the effectiveness of the our algorithm, we also compare it with two state-of-art algorithms *SmartTruncation(ST)* [27] and *PrivBasis(PB)* [26] in the same conditions. One algorithm is the basis of our algorithm while the other is totally different; this arrangement is to have a broader comparison.

A. EXPERIMENT SETUP

1) IMPLEMENTATION

We implement our algorithm using C++ on a PC with CPU Intel Core i7-4790k, processor base frequency 4.00GHz, RAM 8G. In the experiments, we specifically take $\epsilon = 0.1, 0.25, 0.5, 0.75, 1.0$, $\eta = 0.75, 0.8, 0.85, 0.9, 0.95$, and $k = 50, 100$ to parameterize our experiments. We run our algorithm 10 times and report the average values as stable performance indicators.

2) DATASETS

We use five public datasets in this experiment: Accidents(ACC) [43], Kosarak(KOS), Mushroom(MUS) [44], BMS-POS (POS) [44], and Retail [43]. The characteristics of the datasets are summarized in Table 1, where max$|t|$ is the maximum record size and avg$|t|$ is the average record size.

TABLE 1. Dataset characteristics.

Dataset	$\|D\|$	$\|I\|$	max$\|t\|$	avg$\|t\|$
Kosarak(KOS)	990002	41270	2498	8.1
BMS-POS(POS)	515597	1657	164	6.5
Accidents(ACC)	340183	468	51	33.8
Retail(RET)	88162	16470	76	10.3
mushroom(MUS)	8124	119	23	23

The length of all datasets is shown in Fig. 1. These transaction datasets have different data patterns, e.g. Accidents dataset is a dense dataset while BMS-POS and Retail are sparse datasets. It can be seen that the Kosarak dataset is mainly a set of short transactions while the Accidents dataset is the opposite. For the Mushroom dataset, all transactions have the same length as in Table 1; thus we do not draw its length distribution.

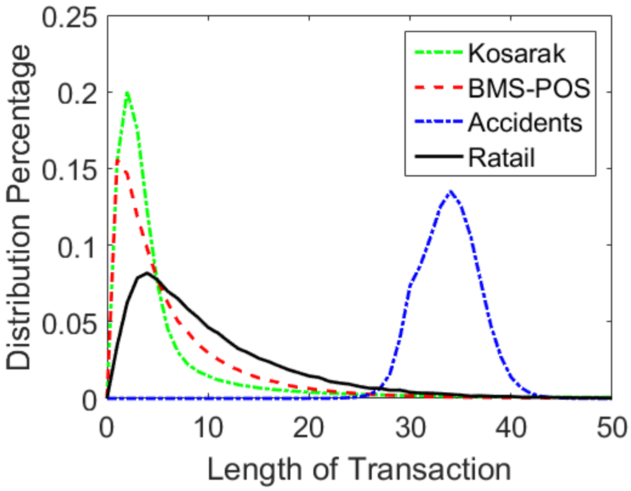


FIGURE 1. Transaction length distribution.

3) PERFORMANCE METRICS

In the experiment, we focus on the following two commonly used data utility performance metrics [26], [27] to measure the performance of our algorithm. The first is F-Score which measures accuracy; the second is RE which measures error.

Definition 4 (F-Score): Let $\mathcal{F}$ and $\mathcal{F}'$ be the set of actual and published frequent itemsets, respectively. The F-Score *is defined as follows*

$$\text{F-Score} = 2 \times \frac{precision * recall}{precision + recall} \quad (9)$$

where precision $= \frac{|\mathcal{F}' \cap \mathcal{F}|}{\mathcal{F}'}$ *and recall* $= \frac{|\mathcal{F}' \cap \mathcal{F}|}{\mathcal{F}}$.

Definition 5 (Relative Error): The relative error of published frequent item sets $\mathcal{F}'$ is defined as

$$\text{RE} = \text{median}_{X \in \mathcal{F}'} \frac{sup'_X - sup_X}{sup_X} \quad (10)$$

where $sup'_X(sup_X)$ is the noisy(actual) support of itemset X.

For the above two utility measures, the larger F-Score is, the closer the frequent itemsets to the real itemsets; it indicates that the utility of the algorithm is higher. The smaller the RE is, the smaller the error is; it also indicates that the utility of the algorithm is higher.

TABLE 2. Experiment parameters.

Parameters	Description	Default values
ϵ	privacy budget	1.0
k	top-k values	50
η	maximum length constraint parameter	0.85
μ	allocated privacy budget parameter	1/3

TABLE 3. F-Score and RE on varying ϵ's in different datasets. (a) F-Score vs. ϵ in different datasets. (b) RE vs. ϵ in different datasets.

(a)

privacy budget ϵ	Mushroom	Retail
0.1	0.84	0.58
0.25	0.94	0.68
0.5	0.94	0.72
0.75	0.96	0.74
1.0	0.98	0.76

(b)

privacy budget ϵ	Mushroom	Retail
0.1	0.0428	0.176
0.25	0.0323	0.152
0.5	0.015	0.146
0.75	0.01	0.138
1.0	0.004	0.111

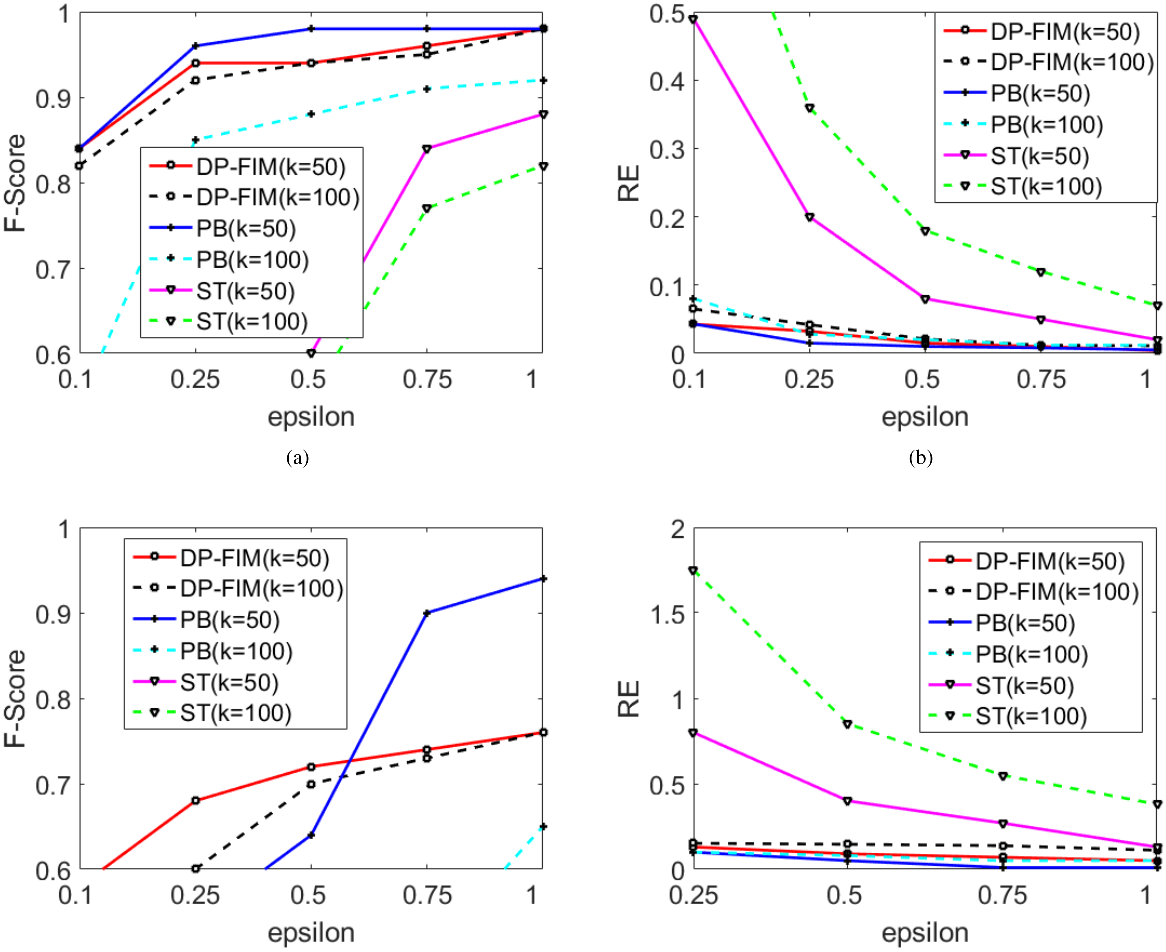
B. PERFORMANCE COMPARISON

We compare the performance of our algorithm with state-of-art algorithms using F-Score and Relative Error. We first analyze the impact of different privacy budgets over the performance indicators. Then we discuss the mining result errors with different thresholds. For performance, we note again that the larger the F-Score is, the higher the data utility is, and that the smaller the relative error is, the higher the data utility is.

Table 2 gives the description of the parameters and the default values in the experiment. Unless otherwise specified, the parameters in the experiment are taken according to Table 2.

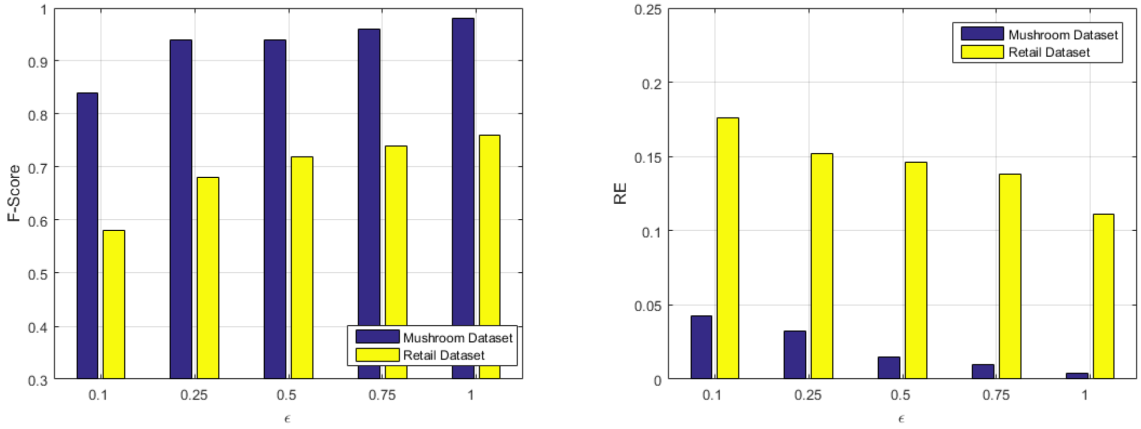
1) EFFECT OF PRIVACY BUDGET

In Figures 2 and 3, we show the performance comparison of *DP-FIM*, *PB* and *ST* algorithms under different privacy budget. We allocate the total privacy budget ϵ as follows:



(a) (b) (c) (d)

FIGURE 2. Performance vs. privacy budget. (a) Mushroom: F-Score. (b) Mushroom: RE. (c) Retail: F-Score. (d) Retail: RE.



(a) (b)

FIGURE 3. F-Score and RE on varying ϵ's in different datasets. (a) F-Score vs. ϵ in different datasets. (b) RE vs. ϵ in different datasets

$\epsilon_1 = 1/3\epsilon$ and $\epsilon_2 = 2/3\epsilon$. In this section, we present the experimental results when ϵ varies. The value of frequent itemset size k is set to be 50 and 100; the maximal length constraint parameter η is set to be 0.85, which is based on the best value that can be obtained in the experiment. The detailed results are also listed in Table 3.

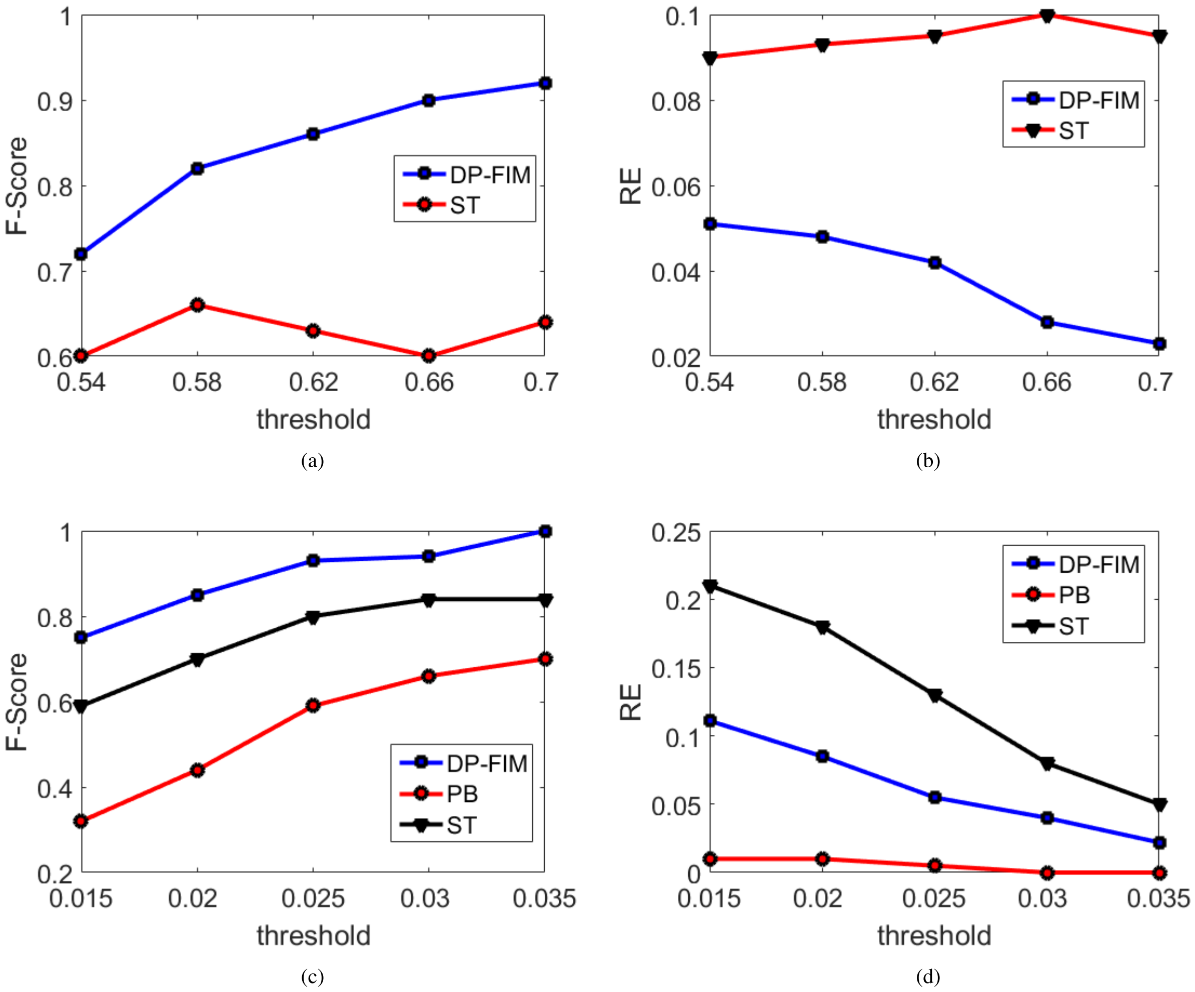


FIGURE 4. Performance vs. threshold. (a) Accidents: F-Score (b) Accidents: RE (c) Retail: F-Score (d) Retail: RE.

Figures 2 and 3 shows how F-Score and RE changes with the increasing of ϵ for the *DP-FIM*, the *PB*, and the *ST* algorithms. In general, when other parameter values are kept at their default values, the curve of F-Score rises with the privacy parameters' increases; the RE curve on the contrary goes downward. This indicates that when the privacy budget increases, the added amount of noise becomes smaller, and the quality of the frequent itemsets becomes higher.

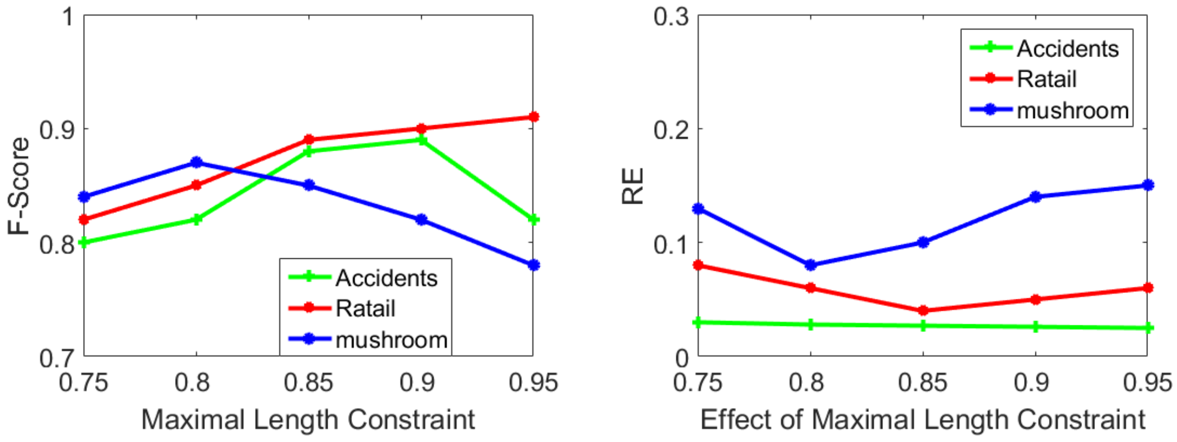
It can be found that our algorithm obtains high F-Score and low relative error; thus, our algorithm has high utility. We also observe that the more the number of output frequent itemsets have, the poorer the performance becomes. This means that the more itemsets are outputted, the more budgets are used for each itemset as well with the more added noise. Thus the utility of the dataset deteriorates as the output itemsets increases.

In addition, our algorithm is basically stable; that is, the curve was in a flat-state shape. This is in line with our expectation. But when ϵ is higher than 0.25, the precisions for Mushroom datasets have reached 90%; while in the Retail dataset, when ϵ takes 1.0, the value of the F-Score does not reach 100%. This shows that our algorithm performance is different for different datasets. Perhaps this is because the Mushroom dataset has the same length, which reduces the amount of information loss in the case of truncated transactions compared to the Retail dataset.

Comparing with state-of-art algorithms, we find that our algorithm obtains better performance (higher F-Score and lower relative error) than *ST* under the same privacy budget as in Fig. 2. For *PB*, our algorithm performance better in F-Score and relative error when more privacy is needed. In other cases, the two algorithms are roughly comparable in F-Score and relative error.

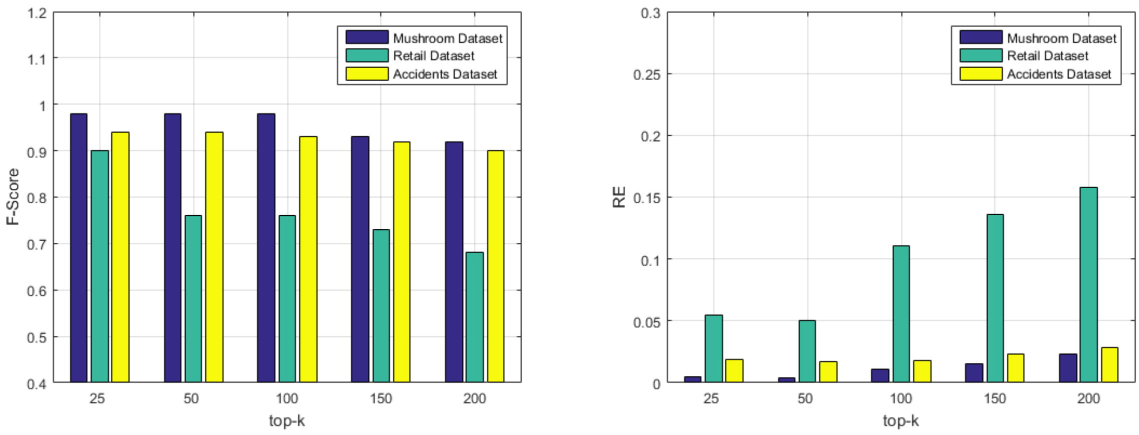
2) EFFECT OF THRESHOLD

We also compare the performance of *DP-FIM*, *PB* and *ST* under different threshold settings using the Accidents and Retail datasets as examples. The results are shown in Fig. 4. The threshold in the experiment is set relative to the size of the entire dataset. We set the privacy budget ϵ as 1.0, the maximum length constraint parameter as $\eta = 0.85$.



(a) (b)

FIGURE 5. Performance vs. Maximal Length Constraint. (a) F-Score. (b) RE.



(a) (b)

FIGURE 6. F-Score and RE on varying *k* in different datasets. (a) F-Score vs. κ in different datasets. (b) RE vs. κ in different datasets.

It can be seen from the experimental results that with the threshold increasing, the charts present a trend of growth; that is, increased threshold results in higher utility. On the Accidents and Retail datasets, compared with *ST*, it has better performance than the *ST* on the F-Score and RE metrics; this is because our transaction truncation processing method can effectively retain the potential frequent information when truncating the long transactions. Thus the performance is generally better than *ST*.

For the *PB* case, our algorithm again performs better in F-Score. However, it is also worth noting that in some datasets (i.e. Accidents) our algorithm is slightly less accurate with regard to RE. This may be because that RE is related to support. Each frequent item is included in the θ-basis in the *PB* algorithm and the amount of noise added is greatly resisted. Thus *PB* achieves better results in RE. In addition, for the trade-off between privacy and data utility, our algorithm and the *ST* algorithm first truncate the dataset and then add noise. Taking further account of privacy requirements, it loses information inevitably. However, our algorithm still remains potential frequent itemsets as far as possible; the corresponding F-Score and RE are also reasonable good for practical uses.

C. MORE DETAILED PERFORMANCE RESULTS

We further report more performance results when varying other parameters of our algorithm.

1) EFFECT OF MAXIMAL LENGTH CONSTRAINT

Figure 5 shows the performance over the maximum length constraint. We vary parameter η from 0.75 to 0.95 and keep other parameters at their default values, i.e., $\epsilon = 1$ and $k = 50$. Figure 5 displays how F-Score and RE vary under different η values with other parameters at their default values. We can observe that with the increasing of η, the F-Score and RE do not increase or decrease monotonically. Initially, the RE decrease when η increases; this is because the increment of η allows to retain more information from the database. However, after a certain threshold, the RE becomes larger with the increasing of η. This is because when η gets

TABLE 4. F-Score and RE on varying *k* in different datasets. (a) F-Score vs. *k* in different datasets. (b) RE vs. *k* in different datasets.

(a)

k	Mushroom	Retail	Accidents
25	0.98	0.90	0.94
50	0.98	0.76	0.94
100	0.98	0.76	0.93
150	0.93	0.73	0.92
200	0.92	0.68	0.90

(b)

k	Mushroom	Retail	Accidents
25	0.005	0.055	0.019
50	0.004	0.05	0.017
100	0.011	0.111	0.018
150	0.015	0.136	0.023
200	0.023	0.158	0.028

larger, the noise added to each level grows quickly. The same analysis also applies to F-Score.

We observe that when the constraint parameter η is small, the quality of frequent itemsets is relatively poor, as expected. This is because the smaller the η is, the more the lost information is. Even in the worst case, RE is still below 0.15, which is acceptable. For larger η, the performance behaves differently on different datasets. For example, on the Accidents dataset, when η breaks through 0.9, the F-Score is becoming smaller. This is in line with our expectation because achieving differential privacy needs to add noise. The larger the length of a record in the database is, the more noise it is to be added to the output, which leads to poor performance. From the experimentation, we empirically find that a good choice for the parameter η is 0.85.

2) EFFECT OF THE TOTAL NUMBER OF OUTPUT FREQUENT ITEMSETS

Figure 6 and Table 4 presents the performance of our scheme when varying the number of output frequent itemsets. For all datasets, the worst F-Score achieves 0.7. In most cases, the F-Score is higher than 0.9. For relative error, the worst error is 0.158 while most errors are below 3%. These high F-Score and low relative error indicate that our algorithm has high utility.

VII. CONCLUSIONS

In this paper, we propose a novel differentially private algorithm for frequent itemsets mining. The algorithm features better data utility and better computation efficiency. Various experimental evaluations validate that the proposed algorithm has high F-Score and low relative error. A lesson learned is that fine tuned parameters lead to better differentially private frequent itemsets mining algorithms with regard to data utility.

REFERENCES

[1] Z. Q. J. Lu, "The elements of statistical learning: Data mining, inference, and prediction," *J. Roy. Stat. Soc., A, (Statist. Soc.)*, vol. 173, no. 3, pp. 693–694, 2010.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Mag.*, vol. 17, no. 3, p. 37, 1999.

[3] H. Yang, K. Huang, I. King, and M. R. Lyu, "Localized support vector regression for time series prediction," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2659–2669, 2009.

[4] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 6, pp. 601–618, Nov. 2010.

[5] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, The Netherlands: Elsevier, 2011.

[6] X. Fang, Y. Xu, X. Li, Z. Lai, and W. K. Wong, "Robust semi-supervised subspace clustering via non-negative low-rank representation," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1828–1838, Aug. 2016.

[7] M. Peña, F. Biscarri, J. I. Guerrero, I. Monedero, and C. León, "Rule-based system to detect energy efficiency anomalies in smart buildings, a data mining approach," *Expert Syst. Appl.*, vol. 56, pp. 242–255, Sep. 2016.

[8] Y. Guo, F. Wang, B. Chen, and J. Xin, "Robust echo state networks based on correntropy induced loss function," *Neurocomputing*, vol. 267, pp. 295–303, Dec. 2017.

[9] H. Lim and H.-J. Kim, "Item recommendation using tag emotion in social cataloging services," *Expert Syst. Appl.*, vol. 89, pp. 179–187, Dec. 2017.

[10] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang, "(α, k)-anonymity: An enhanced k-anonymity model for privacy preserving data publishing," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 754–759.

[11] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[12] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 571–588, 2002.

[13] A. Meyerson and R. Williams, "On the complexity of optimal K-anonymity," in *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2004, pp. 223–228.

[14] Y. Zhang *et al.*, "Embedding cryptographic features in compressive sensing," *Neurocomputing*, vol. 205, pp. 472–480, Sep. 2016.

[15] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, 2007, Art. no. 3.

[16] N. Li, T. Li, and S. Venkatasubramanian, "T-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 106–115.

[17] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*. New York, NY, USA: Springer, 2011, pp. 338–340. [Online]. Available: https://www.springer.com/gp/book/9781441959058

[18] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1026–1037, Sep. 2004.

[19] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 639–644.

[20] Z. Teng and W. Du, "A hybrid multi-group approach for privacy-preserving data mining," *Knowl. Inf. Syst.*, vol. 19, no. 2, pp. 133–157, 2009.

[21] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, "Security in outsourcing of association rule mining," in *Proc. 33rd Int. Conf. Very Large Databases*, Vienna, Austria, 2007, pp. 111–122.

[22] L. Qiu, Y. Li, and X. Wu, "An approach to outsourcing data mining tasks while protecting business intelligence and customer privacy," in *Proc. 6th IEEE Int. Conf. Data Mining Workshops*, Dec. 2006, pp. 551–558.

[23] C.-H. Tai, P. S. Yu, and M.-S. Chen, "k-Support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 473–482.

[24] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," *Inf. Syst.*, vol. 29, no. 4, pp. 343–364, 2004.

[25] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, "Discovering frequent patterns in sensitive data," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 503–512.

[26] N. Li, W. Qardaji, D. Su, and J. Cao, "PrivBasis: Frequent itemset mining with differential privacy," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1340–1351, 2012.

[27] C. Zeng, J. F. Naughton, and J.-Y. Cai, "On differentially private frequent itemset mining," *Proc. VLDB Endowment*, vol. 6, no. 1, pp. 25–36, 2012.

[28] X. Han, M. Wang, X. Zhang, and X. Meng, "Differentially private top-k query over MapReduce," in *Proc. 4th Int. Workshop Cloud Data Manage.*, 2012, pp. 25–32.

[29] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong, "Publishing set-valued data via differential privacy," *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 1087–1098, 2011.

[30] J. Lee and C. W. Clifton, "Top-k frequent itemsets via differentially private fp-trees," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 931–940.

[31] S. Su, S. Xu, X. Cheng, Z. Li, and F. Yang, "Differentially private frequent itemset mining via transaction splitting," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1875–1891, Jul. 2015.

[32] R. Chen, B. C. M. Fung, and B. C. Desai. (2011). "Differentially private trajectory data publication." [Online]. Available: https://arxiv.org/abs/1112.2020

[33] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 638–649.

[34] L. Bonomi and L. Xiong, "A two-phase algorithm for mining sequential patterns with differential privacy," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 269–278.

[35] S. Xu, S. Su, X. Cheng, Z. Li, and L. Xiong, "Differentially private frequent sequence mining via sampling-based candidate pruning," in *Proc. IEEE 31st Int. Conf. Data Eng. (ICDE)*, Apr. 2015, pp. 1035–1046.

[36] E. Shen and T. Yu, "Mining frequent graph patterns with differential privacy," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 545–553.

[37] S. Xu, S. Su, L. Xiong, X. Cheng, and K. Xiao, "Differentially private frequent subgraph mining," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 229–240.

[38] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1994, pp. 487–499.

[39] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 1–12, 2000.

[40] M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara, "Evaluation of sampling for data mining of association rules," in *Proc. 7th Int. Workshop Res. Issues Data Eng.*, Apr. 1997, pp. 42–50.

[41] S. D. Lee, D. W. Cheung, and B. Kao, "Is sampling useful in data mining? A case in the maintenance of discovered association rules," *Data Mining Knowl. Discovery*, vol. 2, no. 3, pp. 233–262, 1998.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2001, pp. 350–355.

[43] Z. Zheng, R. Kohavi, and L. Mason, "Real world performance of association rule algorithms," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 401–406.

[44] Workshop on Frequent Itemset Mining Implementations. (2004). *Frequent Itemset Mining Implementations Repository*. [Online]. Available: http://fimi.ua.ac.be/data

XINYU XIONG is currently pursuing the master's degree with the College of Computer Science and Engineering, Shenzhen University, China. Her research interest is privacy-preserving data mining.

FEI CHEN is currently a Lecturer with the College of Computer Science and Engineering, Shenzhen University, Shenzhen, China. His research interest is data privacy and protection.

PEIZHI HUANG is currently a Professor with the College of Computer Science and Engineering, Shenzhen University, China. His research interest is geographic information systems.

MIAOMIAO TIAN is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. His research interest is information security.

XIAOFANG HU is currently an Associate Professor with the School of Computer and Information Science, Southwest University, China. Her research interest is nonlinear systems.

BADONG CHEN is currently a Professor with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China. His research interest is artificial intelligence.

JING QIN is currently an Assistant Professor with the Centre for Smart Health, School of Nursing, The Hong Kong Polytechnic University. His research interests include medical image processing, virtual/augmented reality for healthcare and medicine training, deep learning, visualization and human–computer interaction, and health informatics.