IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 25, 2019, accepted October 10, 2019, date of publication October 17, 2019, date of current version November 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948033

Secure Mining of Association Rules in Distributed Datasets

QILONG HAN, DAN LU, KEJIA ZHANG, HONGTAO SONG, AND HAITAO ZHANG
College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

Corresponding author: Kejia Zhang (kejiazhang@hrbeu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant No.61370084 and No.61872105, and the Fundamental Research Funds for the Central Universities under Grant No. 3072019CF0601 and 3072019CF0602, and the Project for Innovative Talents of Science and Technology of Harbin under Grant No.2016RAXXJ013, and the China Numerical Tank Project.

ABSTRACT The arrival of Information Age, with its rapid development of information technology, has provided a wide space for Data Analysis and Mining. Yet growth in this market could be held back by privacy concerns. This paper addresses the problem of secure association rule mining where transactions are distributed across sources. The existing solutions for distributed data(vertical partition and horizontal partition) have high complexity of encryption and incomplete definition of attributes of multiple parties. In this paper, we study how to maintain differential privacy in distributed databases for mining of association rules without revealing each party's raw transactions despite how strong background knowledge the attackers have. We use a intermediate server for data consolidation without assuming it is safe. Our methods offer enhanced privacy against various attacks model. In addition, it is simpler and is significantly more efficient in terms of communication rounds and computation overhead.

INDEX TERMS Data mining, distributed databases, association rule mining, differential privacy, privacy preserving.

I. INTRODUCTION

Data mining, at its core, is the transformation of large amounts of data into meaningful patterns and rules. As it extracting useful knowledge from large amount of data, data mining technology has been widely studied and applied in various research and commercial field [1], [2]. Association rules mining is an important technique in it, which is used to discover the implicit relationship between objects from massive data, revealing the hidden association patterns and then assisting in market operation and decision support system. Look at the transaction of a group-buying website. We may find that most of those who buy ''cate'' also buy ''entertainment''. Therefore, ''cate->entertainment'', which means ''buying cate implies buying entertainment'' is one of the candidate rule. Two metrics are defined to measure such a candidate rule: confidence and support. Here confidence means the number of transactions where both ''cate'' and ''entertainment'' are bought divided by the number of transactions where ''cate'' is bought. Support means the number of transactions where ''cate'' and ''entertainment'' are bought divided by the overall number of transactions. A candidate rule is considered a valid association rule if both its confidence and its support are sufficiently high [3].

This paper is concerned with a category of association rules mining, namely mining of association rules in distributed databases. Go back to the example above, there are two group-buying websites that hold partial transactions, i.e., user i buy ''cate'' in group-buying website A and buy ''entertainment'' in group-buying website B or user j like to buy in group-buying website A while user k like to buy in group-buying website B. Using a key such as userID and date, we can join these two websites to mining the global association rules. Traditionally, all these association rule mining algorithms have been carried out within a centralized model, with all data being gathered into a central site, and algorithms being run against that data [4].

Privacy concerns arise because there is no fully trusted third party(TTP) [5]. Third parties are often honest and curious, eager to get additional details of users and potentially vulnerable to abuse [6]–[8]. The semi-trusted third party is proposed to fix this problem. In this paper, we study the mining of association rules in distributed datasets using the semi-trusted intermediate server. Neither the server nor the participant have access to other participants' private transactions. Informally, the goal defines a problem of secure

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaojiang Du.

multi-party computation. In such problem, there are M participants that hold private transactions($(x_1, x_2, \ldots, x_M)$), and they wish to securely compute $y = f(x_1, x_2, \ldots, x_M)$ for some public function f. In this paper, function f is the association rule mining algorithm and $M = 2$ for simplicity.

To the best of our know, Clifton and his students were the first to study privacy-preserving distributed mining of association rules/frequent itemsets. In [4], Vaidya and Clifton gave a nice algebraic solution for vertically partitioned data. However, the computational overhead of protocol for the secure computation of union of private subsets is quadratic in the number of transactions. Goldreich pointed out in [9], generic secure computation protocols are highly expensive for practical purposes. Furthermore, the solution in [10] only works for three parties or more, not for two parties. Desseni study hiding sensitive association rules through suggesting a lattice like graph by declining support degree of frequent item-sets [11]. In particular, many researches have addressed the privacy issues in mining of association rules in various fields [12]–[14]. However, most of these papers are high computation complexity and ignored the background of attackers.

Differential privacy [15], [16] is one of the privacy protection concepts with strongest theoretical proof in recent years that guarantees the output of an Algorithm is indistinguishable of the change of one individual's data [17]–[20]. Differential privacy add noise to the real output to satisfy the change of one individual is insensitivity to the final noisy output(in our case, the output is the count of frequent items) [21]–[24]. The main advantage of differential privacy attracted the attention of frequent itemsets mining boffins, much researches have been done in this field [25]–[27]. Li and Gan reduce the dimensionality by projecting the data into a lower dimensional space and by truncating the transactions, respectively. Cheng proposed a differential private frequent itemset mining algorithm based on transaction splitting.

In this paper, we propose an alternative protocol for the secure mining of association rules in distributed databases. The proposed protocol improves upon that in [4], [28] in terms of simplicity and the theory of privacy, broadens the range of 1-frequent itemsets with certain probability. In tradition, the itemsets whose supports are above the minimal support are used as frequent itemsets. However, those itemsets whose support are near the minimal support have simply been abandoned even with a few missing values or abnormal data. Based upon this observation, we add Laplace noise into the count of each candidate 1-frequent item. In particular, our methods are robust against various attacks model. In this work, we use the intermediate server for data processing and integration without assuming it is safe. The proposed framework is composed of two compositions: (i)association rules discovery under Honest Cooperation(HC) and (ii)association rules discovery under Malicious Spy(MS). In the first case, all of the participants are sincere partners while there could be a malicious participant who collude with the Third-party in the second case. For the latter, the proposed algorithm injects noise in mining of 1-frequent itemsets and construct FP-tree data structure with these noisy value.

The rest of this paper is organized as follows: In Section 2, we formally define the problem and give the notations and definitions used throughout this article. Section 3 introduces the overall framework, attack models and then propose corresponded mining algorithms. In Section 4, the performance of our two algorithm are evaluated on multiple experiments. Finally, Section 5 concludes our work.

II. PROBLEM DEFINITION AND PRELIMINARIES

Before describing the specific mining of association rules algorithm, we first give the symbols and definitions used in the paper and introduce a reasonable assumption, which will help us to mining association rules in vertically distributed databases with privacy control. According to the previous section, we only use the Third-party without treating it as highly trusted.

Association Rule: We adopt the following standard formulation of association rule mining presented by [11]. Assume that $I = i_1, i_2, \ldots, i_m$ is a set of elements, which are called items. The database $T = T_1, T_2, \ldots, Tn$ consists of a group of transactions, where each transaction $T_i \in T$ is a set of items($T_i \subseteq I$). An association rule is of the form $X \Rightarrow Y$ if X and Y are a sub category of I and $X \cap Y = \varnothing$. Equation 1-2 show the way of rule support and confidence calculation respectively.

$$support(X \Rightarrow Y) = \frac{|X \cup Y|}{|T|} \tag{1}$$

$$confidence(X \Rightarrow Y) = \frac{|X \cup Y|}{|X|} \tag{2}$$

In traditionally, minimum support threshold (MST) and minimum confidence threshold (MCT) are the most applicable criteria to evaluate the value of presented rules, which means $X \Rightarrow Y$ is valuable if and only if $support(X \Rightarrow Y) \geq MST$ and $confidence(X \Rightarrow Y) \geq MCT$ strictly.

Definition 1 (Distributed Datasets): We consider our problem with respect to combination distributed datasets means that each party owns some transactions that contains some item of the set I while may has some rows in T. The following example illustrates the form of data owned by parties A and B.

Differential Privacy: Two databases D_1 and D_2 are referred to as neighboring if one can be obtained by adding or removing one tuple from the other, i.e., $|(D_1 - D_2) \cup (D_2 - D_1)| = 1$. Informally, differential privacy ensures that the outcome should be insensitive to the presence or absence of any particular tuple in D which is bounded by a constant ratio.

Definition 2 (ϵ-Differential Privacy): A randomized mechanism M is ϵ-differential private if for all neighboring databases D_1 and D_2, $\forall S \subseteq Range(M)$

$$Pr[M(D_1) \in S] \leq exp(\epsilon) Pr[M(D_2) \in S] \tag{3}$$

where the privacy budget ϵ controls the amount of noise injection. A smaller ϵ enforces a stronger privacy guarantee of M. Laplace mechanism is one of the most general method for preserving ϵ-differential privacy of any function F, where the output of F is a vector of numeric data. In fact, the mechanism exploits the global sensitivity of F over any two adjacent datasets, denoted as Δf. Given Δf, the Laplace mechanism ensures ϵ-differential privacy by injecting noise η into each value in the output of $F(D)$, where η fetch its value from Laplace distribution with zero mean and scale $\Delta f/\epsilon$.

Definition 3 (Global Sensitivity): The global sensitivity of a query function f is defined as

$$\Delta f = \max_{D_1, D_2} |f(D_1) - f(D_2)| \tag{4}$$

where D_1 and D_2 are neighboring databases.

One nice property of differential privacy is that it is composable. Specifically, if the domain of input records is partitioned into disjoint sets, independent of the actual data, and the restrictions of the input data to each part are subjected to differential private analysis, the ultimate privacy guarantee depends only on the worst of the guarantees of each analysis, not the sum [29].

Assume 1: Frequent itemsets can be expanded softly.

Assuming that the frequency of X is K and the $support(X) < MST$. However, $support(X) \geq MST$ is satisfied when $K = K + 1$. In this case, the traditional rigid standards will filter out this item X though minimal difference. In reality, data loss or corruption even delay on data acquisition that can cause such error. In this paper, we soften the criteria so that some candidates with minimal difference have the chance to be frequent.

FP-tree is a compressed representation of the input data. It is constructed by reading the dataset one transaction at a time and mapping each transaction onto a path in the FP-tree structure. As different transactions can have same items, their paths may overlap. The steps of FP-tree structure is as follows:

Definition 4 (Soft-Frequent): The frequent candidate item X is Soft-Frequent if $support_{soft}(X) = \frac{|X|+\eta}{|T|} \geq MST$. Meanwhile this mechanism satisfies ϵ-differential privacy.

Proof:

$$\frac{Pr[support_{soft}(X \in D) = O]}{Pr[support_{soft}(X \in D') = O]} = \frac{\frac{\epsilon}{2\Delta f} e^{-\frac{|O-|X_D||\epsilon}{\Delta f}}}{\frac{\epsilon}{2\Delta f} e^{-\frac{|O-|X_{D'}||\epsilon}{\Delta f}}} = e^{\frac{\epsilon|X_D - X_{D'}|}{\Delta f}} \leq e^{\epsilon}$$

Algorithm FP-Tree Construction Algorithm

Input: Transaction data set D, the minimum support threshold min_sup

Output: FP-tree

1.Scan the transaction dataset D once to get the support of each of the frequent items F. All the frequent items in F are sorted in descending order according to the support expressed as L;
2.Create a root node T of the FP-tree, marked "null";
3.for each transaction $Trans \in D$ do
4. Sort all frequent items in $Trans$ in the order of L;
5. The sorted frequent item list is represented in the format of $[p|P]$, where p is the first item and P is the frequent item list after removing p;
6. Call the function $insert_tree([p|P], T)$;
7.end for

$insert_tree([p|P], root)$
1.If root has child N and $N.item - name = p.item - name$ then
2. $N.count + +$;
3.Else
4. create a new node N;
5. $N.item - name = p.item - name$;
6. $N.count + +$;
7. $p.parent = root$;
8. Point $N.node - link$ to the node in the tree with the same node name;
9.End if
10.If $P \neq \emptyset$ then
11. Assign the first item of P to p and remove it from P;
12. Recursively call the function $insert_tree([p|P], N)$;
13.End if

III. FRAMEWORK AND MINING ALGORITHMS

In this section, we present the overall framework of secure mining of association rules in distributed datasets and proposed two discovery algorithm of getting 1-frequent itemsets safely under two attack models. We use the intermediate server for data consolidation without assuming it is safe. Besides, no one in our framework is assumed to be trusted. For simplicity, at this point we only discuss two-party distributed mining, suppose that the two parties are A and B.

Definition 5 (Attack Models): In the principles introduced earlier, no party in this system is fully trusted, so that data(even count) should be securely transfered. In this section, we present two secure mining of 1-frequent itemset algorithms for two attack models called Honest Cooperation(HC) and Malicious Spy(MS). In HC model, both A and B are honest and eager to share each other's data securely to learn more accurate association rules. However, in MS model, A and B don't trust each other and maybe a malicious spy who collaborates with the third-party to grab the private data of another. In both attack models, we assume that the attacker has the strongest background knowledge.

A. OVERALL FRAMEWORK

We present the design of the proposed association rules mining algorithm, the overview framework is shown in Figure 1 which contains two compositions: Association Rules Mining Under HC(ARHC) and Association Rules Mining Under MS(ARMS).

(a). Party A

UserId	Cate	Movie	Hotel	Entertainment
u_1	0	1	1	0
u_4	1	0	0	1
u_5	1	1	1	0

(b). Party B

UserId	Cate	Movie	Hotel	Entertainment
u_2	0	0	1	0
u_3	0	1	1	1
u_5	1	0	0	1

FIGURE 1. Sample of two parties A and B.

In ARHC model, the Third-party generates some initialized data and sends it to A, A calculate the result of its private dataset and permutes it by certain rules and then sum with the initialized data from the Third-party, then sends the result and permutation rules to B. B calculate the result of its private dataset and do the same permutation. Finally, B sends the end result to the Third-party. Since the count number of each item or pattern is real value, this mining algorithm is lossless.

In ARMS model, the Third-party do the same operations to send some initialized data to A, A adds laplace noise to real statistical results of each candidate 1-frequent item and use these noisy counts as the node value of noisy FP-tree, and then sends noisy FP-tree to B. Since B also distrust A or the Third-party, he adds noise to his count of each 1-frequent item and use A's noisy FP-tree to generate the final noisy FP-tree. The Third-party can work out all candidate patterns in this noisy tree without access each party's private dataset.

B. ASSOCIATION RULES MINING UNDER HC(ARHC)

Recall that, in the problem of mining 1-Frequent itemset under HC, $I = i_1, i_2, \ldots, i_m$ is a set of items. A has a private dataset $T_A = T_i, T_{i+1}, \ldots, TM$ and B has a private dataset $T_B = T_j, T_{j+1}, \ldots, TN$, where T_k is a transaction and there may be overlap between T_A and T_B.

1) MINING OF 1-FREQUENT ITEMSET

The steps to mining 1-Frequent itemset under HC model are as follow:

(1) The Third-party generates an m-dimension vector $C = c_1, c_2, \ldots, c_m$ randomly and each column represents the count of a item, then sends vector C to A.

(2) A counts the number of each item according to its own private dataset, and generates the m-dimension vector $A = a_1, a_2, \ldots, a_m$ firstly. Then, permutes this vector to get a new m-dimension vector $A' = a_i, a_j, \ldots, a_k$ and record the index of each item. Finally, sum vector C and A' as $\hat{A} = c_1 + a_i, c_2 + a_j, \ldots, c_m + a_k$, then send $\hat{A}$ and the permutation rule to B.

(3) B counts the number of each item according to its own private dataset and permutes this vector according to the rule received from A, to generates the disturbed m-dimension vector $B' = b_e, b_s, \ldots, b_d$. Then, sum vector $\hat{A}C$ and B' as $\hat{B} = c_1 + a_i + b_e, c_2 + a_j + b_s, \ldots, c_m + a_k + b_d$ and sends $\hat{B}$ to Third-party.

(4) The Third-party received the m-dimension vector $\hat{B}$ and minus the vector initialized C, then calculate $support(\hat{B}_i)$ where i is the ith item of $\hat{B}$. Finally, the Third-party sends the final index set $S = I_1, I_2, \ldots, I_i | support(\hat{B}_i) \geq MST$ to party A and B.

(5) A and B get the final index set of 1-Frequent item and since both A and B know the original index of each item, so 1-Frequent itemsets can be calculated.

Privacy Analysis Since the m-dimension vector that B received is the sum of vector A and vector C of the third party, the true statistical result of A is not disclosed. Similarly, the m-dimension vector that Third-party received is the sum of A and B, the true result of B is not exposed.

Secondly, A and B use the same permutation method. The Third-party can get the final statistics of each index but cannot know the real item represented by the index value.

Algorithm Association Rules Mining Under HC

Input: The set of items I, MST, MCT, initialized m-dimension vector C, $|U| \times |I|$ boolean matrix D, m transactions T, the ultimately count of each item $count_i$, the final FP-tree $Tree'$ and the final $|U| \times |I|$ boolean matrix D';

Output: The 1-frequent itemsets S, all association rules R satisfying MST and MCT;

1.for each $i \in I$ do:
2. if $count_i - C_i > MST$
3. $S.add(i)$;
4.for each entry in D' do:
5. OR operation with the entry in D in the same position to $\bar{D}$;
6. for any itemsets I' do:
7. if $\bar{D}_i = 1 | i \in T'$
8. $count(I'_1, I'_2 \ldots) + +$;
9.for each branch of $Tree'$
10. calculate the count of each candidate itemset I'';
11.for each candidate frequent itemset a do:
12. if $count_a > MSTandconfidence_{a_i \to a_j} > MCT$;
13. $R.add(a_i \to a_j)$;
14.return R;

2) MINING OF FREQUENT ITEMSET

Defined in the above section, A has a private dataset $T_A = T_i, T_{i+1}, \ldots, TM$ and B has a private dataset $T_B = T_j, T_{j+1}, \ldots, TN$. In this section, we extend the definitions of A and B as:

$$T_A = T_{u_i}, T_{u_j}, \ldots, Tu_k$$
$$T_B = T_{u_l}, T_{u_m}, \ldots, Tu_n \quad (5)$$

where T_{u_i} is the transaction of user u_i and there may also be overlap (same user) between T_A and T_B. Since A and B are both vertically and horizontally partitioned, mining of frequent itemset should be considered separately. First, we need to get the common user between them.

Mining of common user U: A and B use the same encryption method to encrypt the users identity in their private dataset and then send them to the Third-party, who compares and returns the same ciphertext. For the common user U, both A and B transactions should be considered when mining of association rules, and not vice versa.

The steps to mining association rules under HC model are as follow:

(1) Sort by the frequency of the 1-frequent items obtained in the previous section represented as R.

(2) The Third-party generates m transactions $T_C = T_{u_1}, T_{u_2}, \ldots, T_{u_m}$ of non-common users and for each $i \in I$, $Rank_i = R_i | i \in T_C$ where $u_i \notin U$ and $Rank_i$ is the order of item i.

(3) The Third-party generates $|U| \times |I|$ boolean matrix D_C randomly and then send T_C and D_C to A.

(4) A divides the private dataset horizontally and extracts the transaction set of non-common users as T'_A. A generates the FP-tree $Tree'_A$ according to the rank R, where $i \in T'_A \cup T_C$.

(5) A extracts the transaction set of common users and generates $|U| \times |I|$ boolean matrix D_A. Each row of the matrix corresponds to a transaction, while each column corresponds to an item. A matrix element $D(i; j)$ is 1 if the ith transaction T_i contains the jth item I_j, and it is 0 otherwise. Then transpose the rows and columns of the matrix following certain regulations and do OR operation with each of the entries in matrix D_C then get the result as D'_A. Finally, sends transposition regulations and D'_A and $Tree'_A$ to B.

(6) B divides the private dataset horizontally and extracts the transaction set of non-common users as T'_B. B generates FP-tree $Tree'_B$ on the basis of $Tree'_A$ according to the rank R, where $i \in T'_A \cup T_C \cup T_B$.

(7) B extracts the transaction set of common users and generates $|U| \times |I|$ boolean matrix D_B and do the same transposition as A, then do OR operation with each of the entries in matrix D'_A to get the result as D'_B. Finally, sends $Tree'_B$ and D'_B to the Third-party.

(8) The Third-party received the $Tree'_B$ and subtract the branch that initializes in the transaction set T_C to get the FP-tree of all non-common users' transactions.

(9) The Third-party received the D'_B and do OR operation with each entry in matrix D_C, then get the common users' $|U| \times |I|$ boolean matrix.

Privacy Analysis

Since the $|U| \times |I|$ boolean matrix D'_A that B received is the OR operation of matrix A and matrix C of the Third-party, the $Tree'_A$ that B received is the unit of transactions T'_A and T_C. The true common users' statistical result or non-common users' statistical result of A is not disclosed. Similarly, the final result that Third-party received is the sum of A and B or the OR of A and B, the true result of B is not exposed.

Secondly, A and B use the same permutation method. The Third-party can get the final statistics of each index but cannot know the real item represented by the index value same as the above section.

Complexity Analysis

Computational Overhead FP-tree algorithm needs to scan the database for only twice and avoid a mass of candidate generation, which improves the efficiency of traditional association rules mining algorithm (Apriori). Besides, the process of generating FP-tree can be distributed between A and B synchronously, which can further reduce the calculation time.

For A or B, the matrix representation of the common user can be obtained in the first scan, and the transposition operation and OR operation can be performed in linear time. The distributed system is not applicable to mining association rules of common users, and the encrypted matrix representation of A and B needs to be perform in sequential implementation.

Communication Overhead For FP-tree construction of non-common users, the Third-party sends initialize transactions set T_C, which length is $|T_C|$. Besides, there are $M - 1$ rounds of communication, where in each one of them each of the M parties sends to the next party a message. In this paper, there are two user and a Third-party, so the rounds is two and the message is a FP-tree.

For common users, there are three rounds of communication between two user and a Third-party, which length is $|U| \times |I|$.

C. ASSOCIATION RULES MINING UNDER MS(ARMS)

Recall that, in the problem of mining 1-Frequent itemset under MS, party A and B have its own private datasets T_A and T_B same as the above section and there may be overlap between T_A and T_B. In the MS model, there maybe a spy in party A and B who collude with the Third-party to usurp the private data of another party. We assume A or B distrust each other and there maybe a spy between them. where $count_i = c^i_A + Lap_A(\epsilon) + c^i_B + lap_B(\epsilon)$, thus we can figure out the expectation according to the joint probability density function. λ is the threshold of expectation value which balance the granularity of frequent patterns. In this article, we assume that A and B distrust each other and add noise to their private count value respectively. While, there maybe some smart tricks like game theory or adding noise hierarchically can reduce the noise which will considered in later paper.

1) MINING OF 1-FREQUENT ITEMSET

The steps to mining 1-Frequent itemset under MS model are as follow:

(1) The Third-party generates an m-dimension vector $C = c_1, c_2, \ldots, c_m$ randomly and each column represents the count of a item, then sends vector C to A.

Algorithm Association Rules Mining Under MS

Input: The set of items I, MST, MCT, initialized m-dimension vector C, m transactions T, the ultimately count of each item $count_i$, the final FP-tree $Tree'$, threshold parameter λ;
Output: The 1-frequent itemsets S, all association rules R satisfying MST and MCT;
1.for each $i \in I$ do:
2. if $E(count_i - C_i - MST > 0) > \lambda$
3. $S.add(i)$;
4.for each branch of $Tree'$ do:
10. calculate the count of each candidate itemset I'';
11.for each candidate frequent itemset a do:
12. if $count_a > MST$ and $confidence_{a_i \to a_j} > MCT$;
13. $R.add(a_i \to a_j)$;
14.return R;

(2) A counts the number of each item according to its own private dataset, and generates the m-dimension vector $A = a_1, a_2, \ldots, a_m$ firstly. Then, add laplace noise($Lap(\Delta f/\epsilon)$) in each dimension where $\Delta f = 1$ to generate the new noisy m-dimension vector $A_n = a_1 + \Delta_1, a_2 + \Delta_2, \ldots, a_m + \Delta_m$. Finally, sum vector C and A_n as $\hat{A} = c_1 + a_1 + \Delta_1, c_2 + a_2 + \Delta_2, \ldots, c_m + a_m + \Delta_m$, then send $\hat{A}$ to B. The permutation is useless if B colluded with the Third-party.

(3) B counts the number of each item according to its own private dataset and add laplace noise($Lap(\Delta f/\epsilon)$) in each dimension where $\Delta f = 1$ to generate the new noisy m-dimension vector $B_n = b_1 + \Delta'_1, b_2 + \Delta'_2, \ldots, b_m + \Delta'_m$, and then sum vector $\hat{A}$ and B_n as:

$$\hat{B} = c_1 + a_1 + \Delta_1 + b_1 + \Delta'_1, c_2 + a_2 + \Delta_2 + b_2 + \Delta'_2, \ldots c_m + a_m + \Delta_m + b_m + \Delta'_m \quad (6)$$

and then send $\hat{B}$ to the Third-party.

(4) The Third-party received the m-dimension vector $\hat{B}$ and minus the vector initialized C. The noisy m-dimension vector of A and B is received. Given a probability threshold λ, X is a 1-frequent item if $E(support(X) > MST) > \lambda$, where $support(X) = \frac{A_X + \Delta + B_X}{T} + \Delta'$, T is the total number of transactions, Δ and Δ' obey the Laplace distribution.

Privacy And Utility Analysis Since the m-dimension vector that B and C gotten is a noisy m-dimension vector. After adding noise(($Lap(\Delta f/\epsilon)$)) to real statistics of each item, ϵ-differential privacy is satisfied.

Secondly, we use the expectation of noise as the method to calculate the support of frequent items. For the item with minimum difference, it can be output as frequent with a probability.

2) MINING OF FREQUENT ITEMSET

Since noise has been added to the statistics of 1-frequent items in private dataset of A or B in the previous section, which is the value of the node in FP-Tree and the statistics of association rules of multi-items are also noisy, the association rules of common users is not considered in this section.

Loss Function of FP-Tree: Given two FP-Trees T_A and T_B, the loss function is the quantification of the difference between these two trees. The difference between two trees is the sum of the differences of every two nodes under each branch.

$$L(A, B) = distance[T_A, T_B] \quad (7)$$

Optimal generation of FP-Tree:Given the real FP-Tree and the noisy count of each 1-frequent item, the count of nodes with the same name is same as that of frequent items after adding noise. On this basis, satisfying minimum difference as:

$$L_{min}(A, B) = \sum_{\tau_i, \tau_j \in I} \sum_{b \in \tau} \lambda \frac{|\Delta(n_i, n_j)|}{max(C_A(n_i, n_j), C_B(n_i, n_j))} \quad (8)$$

where τ is the set of all branches of A's FP-Tree, $\Delta(n_i, n_j)$ is the count difference of node i and j in branch b between T_A and T_B and λ is the hyper-parameter to measure the degree of difference. The optimal generation of noisy tree is a NP-hard problem. In this paper, greedy algorithm is used to reconstruct FP-Tree from the minimum frequent item to achieve local optimization. Figure 2 shows the example of generating the noisy FP-Tree.

The steps to mining association rules under MS model are as follow:

(1) Sort by the frequency of the 1-frequent items obtained in the previous section represented as R.

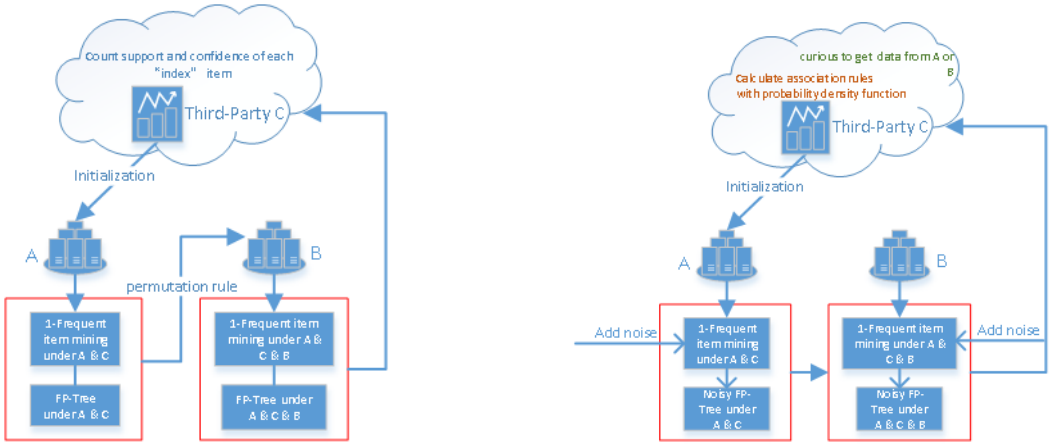
(2) The Third-party generates m transactions $T_C = T_{u1}, T_{u2}, \ldots, T_{um}$ and for each $i \in I$, $Rank_i = R_i | i \in T_C$ where $ui \notin U$ and $Rank_i$ is the order of item i. Then sends T_C to A.

(3) The noisy count of each item of A is c'_i. Then A generates the FP-Tree $Tree_A$ according to the rank R, where $i \in T_A$. For each node name A_i of FP-Tree $Tree_A$, compare the count of A_i and noisy value c'_i. If $count_{A_i} > c'_i$, the pruning strategy is used. Otherwise, the FP-Tree should be extended appropriately.

(4) A generates optimal FP-Tree $Tree'_A$ on the basis of $Tree_A$ satisfying L_{min}, where $i \in T_A \cup T_C$ and sends $FP - Tree'_A$ to B.

(5) B do the similar operations and generates optimal FP-Tree $FP - Tree'_B$ on the basis of $FP - Tree'_A$ according to the rank R, where $i \in T'_A \cup T_C \cup T_B$.

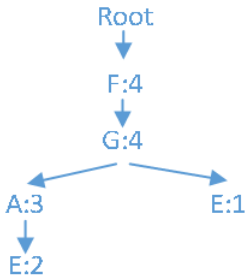
(6) The Third-party received the FP-Tree $FP - Tree'_B$ and subtract the branch that initializes in the transaction set T_C to get the FP-Tree of all users' transactions. where ''calculate upward'' means only consider the patterns between node i and its parent and ancestor nodes, and ''calculate downward'' indicates only consider the patterns between node i and its child and grandchild nodes. **Privacy Analysis** Since the count of each 1-frequent item is noisy obtained by the above section and the FP-Tree generated is based on these noisy value,



(a) Mining of Association Rules Under HC(ARHC)

(b) Mining of Association Rules Under MS(ARMS)

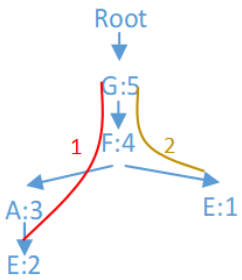
FIGURE 2. The overview framework.



(a) original FP-Tree

Node	Original	Noisy
F	4	4
G	4	5
A	3	3
E	3	2

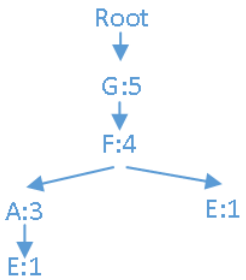
(b) count of node



(c) the tree built

Branch	Loss
1	$\sum(G,E)+(F,E)+(A,E)=\lambda(\frac{1}{2}+\frac{1}{2}+\frac{1}{2})=\frac{3}{2}\lambda$
2	$\sum(G,E)+(F,E)=\lambda(\frac{1}{1}+\frac{1}{1})=2\lambda$

(d) loss of two branches



(e) the final FP-Tree

FIGURE 3. Construction of noisy FP-Tree.

the final noisy FP-Tree is robust even if the attacker has the strongest background knowledge. **Complexity Analysis**

Computational Overhead The greedy noisy FP-Tree generation algorithm need to scan the database for only twice and generate FP-Tree in the order of R. Besides, for each 1-frequent item where the noise is not 0, the algorithm finds the minimum loss branch that contains the item which costs linear time.

As the final FP-Tree is noisy, vertical partitioning of common user attributes is not considered in this section, which can further reducing the computational overhead.

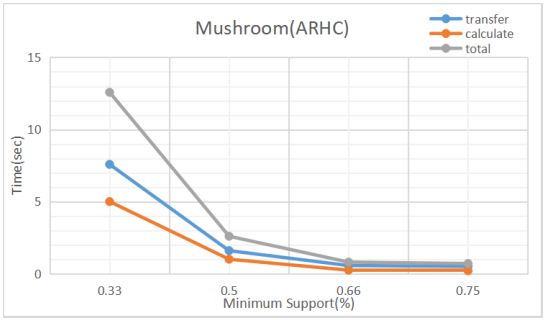
Communication Overhead For FP-Tree construction of users, the Third-party send initialize transactions set T_C, which length is $|T_C|$. Besides, there are $M-1$ rounds of communication, where in each one of them each of the M parties sends to the next party a message. In this paper, there are two user and a Third-party, so the rounds is two and the message is a FP-Tree.

TABLE 1. Dataset description.

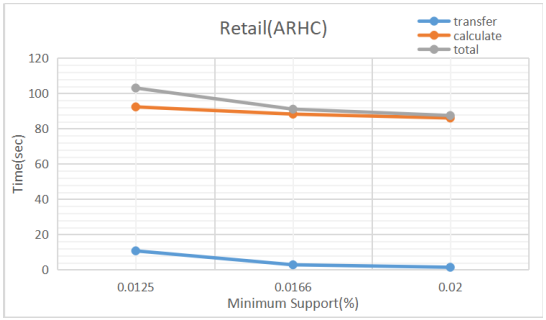
dataset	|D|	|I|	max|t|	avg|t|
mushroom(MUS)	8124	119	23	23
retail(RETL)	88162	16470	76	10.3

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed algorithm *ARHC* and *ARMS* on a variety of synthetic dataset based on the real transaction datasets. The datasets used in the experiments are described in Table 1. All experiments are conducted on an Intel(R) 2.5GHz machine with 8G of

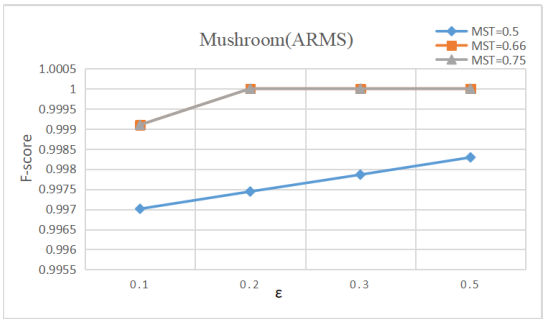


(a) Execution time of Mushroom under HC

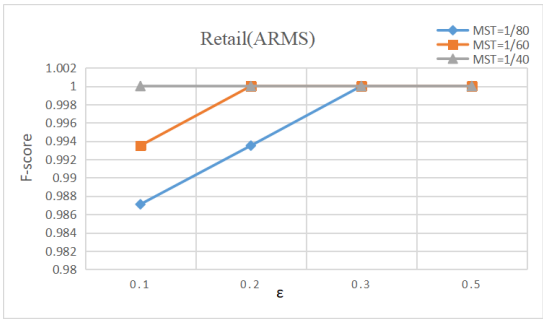


(b) Execution time of Retail under HC

FIGURE 4. Execution time.



(a) F score of Mushroom under MS



(b) F score of Retail under MS

FIGURE 5. F score by varying ϵ.

Algorithm Greedy Noisy FP-Tree Generation Algorithm

Input: noisy count of each 1-frequent item c_i', the FP-Tree of A $Tree_A$, the sorted frequent item list R;
Output: noisy FP-Tree $Tree_A'$;
1.Sort R in descending order, named as R';
2.for each i in 1-frequent items do:
3. compare the noisy count of each 1-frequent item c' and real value c;
4. if($c_i' > c_i$)
5. increasing the value of the optimal node satisfying $L_{min}(-i)$ that calculate upward;
6. else
7. pruning the optimal node satisfying $L_{min}(-i)$ that calculate downward;
8.return noisy FP-Tree $Tree_A'$

physical memory. To obtain an average performance estimate, each experiment was run 10 times. To compare the performance of the algorithms, we employ F score and Relative error(RE) as measures of utility. DEFINITION 5 (F Score). Let F and $\hat{F}$ be the set of correct and published association rules, respectively. The F score is defined as follows:

$$Fscore = 2 \times \frac{precision * recall}{precision + recall} \tag{9}$$

where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$.

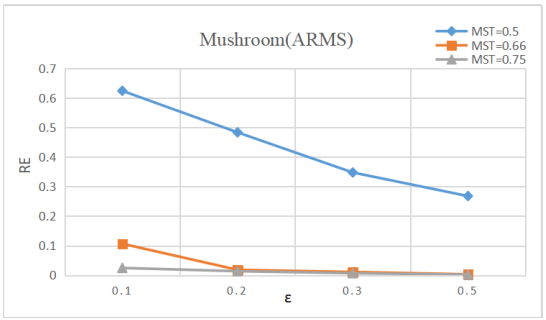
Definition 6 (Relative Error): The relative error of published frequent pattern set $\hat{F}$ is defined as

$$RE = Avg_{X \in \hat{F}}(\frac{|\hat{\sigma}(X) - \sigma(X)|}{\sigma(X)}) \tag{10}$$

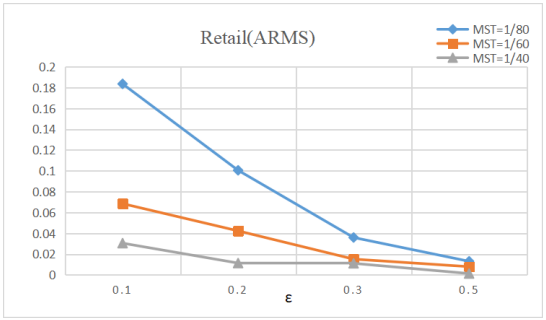
where $\sigma(X)$ is the real frequency of pattern X and the $\hat{\sigma}(X)$ is the noisy frequency of the pattern X [25].

Figure 4 shows the execution times for the two datasets by increasing values of minimum support under HC. The minimum confidence was set to 0.6 and we run the experiments with data transmission time and computing time respectively. These results are indicated as ''transfer'' and ''calculate'' and ''total'' in the figure. The execution times increase as the minimum support threshold(MST) is reduced because the total number of large and candidate itemsets increase. Also, as the average length of transactions increase, the number of large and candidate itemsets also increase.

Figure 5 shows the F scores for the two datasets by increasing values of ϵ under MS. The minimum confidence was set to 0.6 and we run the experiments setting different minimum support threshold(MST). Observe that the proposed algorithm shows stable performance for different privacy budget and MST. The F scores rapidly increase at lower privacy levels(larger ϵ). As the value of ϵ is increased, the F scores started to level off because the influence of noise has waned. Also, as the minimum support threshold(MST) increased, the higher F scores is imposed.



(a) RE of Mushroom under MS



(b) RE of Retail under MS

FIGURE 6. Relative error by varying ϵ.

Figure 6 describes the change of RE on different values of ϵ. The minimum confidence was also set to 0.6 and we run the experiments setting different minimum support threshold(MST). The RE decrease as ϵ is increased because of the less noise. Also, as the minimum support threshold(MST) increased, the lower RE is imposed because the count of each candidate itemsets is increased and the influence of noise has waned.

V. CONCLUSION

we have proposed two algorithms for mining association rules under *HC* and *MS* attack models. The mining algorithm under *HC* transfer the statistics of more than one participants and the mining algorithm under *MS* is in a differentially private way. Each party enables the algorithm to use the remaining privacy budget to efficiently and effectively built a differentially private FP-tree with the noisy counts of 1-frequent itemset. Then the Third-party and other participants can calculate the supports of all frequent patterns without access the private dataset.

In this paper, we assume participant have no idea of each party's background and generate the noisy FP-tree in a differentially private way. We believe that taking advantage of the theory of game or generating the noisy FP-tree hierarchically are likely to enhance the performance.

REFERENCES

[1] Q. Han, S. Liang, and H. Zhang, "Mobile cloud sensing, big data, and 5G networks make an intelligent and smart world," *IEEE Netw.*, vol. 29, no. 2, pp. 40–45, Mar./Apr. 2015. doi: 10.1109/MNET.2015.7064901.

[2] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart Internet of Things systems: A consideration from a privacy perspective," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 55–61, Sep. 2018. doi: 10.1109/MCOM.2018.1701245.

[3] Z. Sheng, "Privacy-preserving algorithms for distributed mining of frequent itemsets," *Inf. Sci.*, vol. 177, no. 2, pp. 490–503, 2007.

[4] P. Kaghazgaran 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.

[5] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 4, pp. 577–590, Jul./Aug. 2016. doi: 10.1109/TDSC.2016.2613521.

[6] K. Zhang, Q. Han, and Z. Cai, "RiPPAS: A ring-based privacy-preserving aggregation scheme in wireless sensor networks," *Sensors*, vol. 17, no. 2, p. 300, 2017. doi: 10.3390/s17020300.

[7] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1–10.

[8] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Netw. Mag.*, vol. 32, no. 4, pp. 8–14, Jul./Aug. 2018. doi: 10.1109/MNET.2018.1700349.

[9] W. A. O. Goldreich, "Secure multi-party computation," in *Information Security and Communications Privacy*, 2014.

[10] 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.

[11] M. Atallah, A. Elmagarmid, M. Ibrahim, E. Bertino, and V. Verykios, "Disclosure limitation of sensitive rules," in *Knowledge and Data Engineering Exchange*, 1999.

[12] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *J. Cryptol.*, vol. 9, no. 8, pp. 616–621, 2008.

[13] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 92–106, Jan. 2006.

[14] S. Agrawal, J. R. Haritsa, and B. A. Prakash, "FRAPP: A framework for high-accuracy privacy-preserving mining," *Data Mining Knowl. Discovery*, vol. 18, no. 1, pp. 101–139, 2009.

[15] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages and Programming*, 2006.

[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography* (Lecture Notes in Computer Science), vol. 3876, no. 8, 2012, pp. 265–284.

[17] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Trans. Ind. Informat.*, to be published.

[18] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Trans. Netw. Sci. Eng.*, to be published.

[19] Q. Han, B. Shao, L. Li, Z. Ma, H. Zhang, and X. Du, "Publishing histograms with outliers under data differential privacy," *Secur. Commun. Netw.*, vol. 9, no. 14, pp. 2313–2322, 2016. doi: 10.1002/sec.1493.

[20] D. Lu, Q. Han, K. Zhang, H. Zhang, and B. Gull, "A novel method for location privacy protection in LBS applications," *Secur. Commun. Netw.*, vol. 2019, pp. 1914038:1–1914038:11, Jul. 2019. doi: 10.1155/2019/1914038.

[21] Q. Han, Q. Chen, L. Zhang, and K. Zhang, "HRR: A data cleaning approach preserving local differential privacy," *Int. J. Distrib. Sensor Netw.*, vol. 14, no. 12, 2018. doi: 10.1177/1550147718819938.

[22] Q. Han, Z. Xiong, and K. Zhang, "Research on trajectory data releasing method via differential privacy based on spatial partition," *Secur. Commun. Netw.*, vol. 2018, pp. 4248092:1–4248092:14, Nov. 2018. doi: 10.1155/2018/4248092.

[23] Q. Han, D. Lu, K. Zhang, X. Du, and M. Guizani, "Lclean: A plausible approach to individual trajectory data sanitization," *IEEE Access*, vol. 6, pp. 30110–30116, 2018. doi: 10.1109/ACCESS.2018.2833163.

[24] X. Zheng, Z. Cai, J. Li, and H. Gao, "Location-privacy-aware review publication mechanism for local business service systems," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9. doi: 10.1109/INFOCOM.2017.8056976.

[25] 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.
[26] W. Y. Gan, W. U. Ying-Jie, L. Sun, and Y. L. Wang, "Frequent pattern mining with differential privacy based on transaction truncation," *J. Chin. Comput. Syst.*, 2015.
[27] C. Xiang, S. Su, S. Xu, and Z. Li, "DP-Apriori: A differentially private frequent itemset mining algorithm based on transaction splitting," *Comput. Secur.*, vol. 50, pp. 74–90, May 2015.
[28] Y.-Q. Zhu, Y. Tang, and G. Chen, "A privacy preserving algorithm for mining distributed association rules," in *Proc. Int. Conf. Comput. Manage.*, 2011, pp. 1–4.
[29] F. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," *Commun. ACM*, vol. 53, no. 9, pp. 89–97, Sep. 2010.

QILONG HAN is currently a Professor and the Deputy Dean of the College of Computer Science and Technology, Harbin Engineering University. He has more than 60 publications as edited books and proceedings, invited book chapters, and technical articles in refereed journals and conferences. His research interests include data security and privacy, mobile computing, and distributed and networked systems. He is a Senior Member of CCF, and the Chair of CCF YOCSEF Harbin. He has served as a Programme Committee Members and the Co-Chair of a number of international conferences/workshops for areas, including web intelligence, e-commerce, data mining, and intelligent systems.

DAN LU received the B.Eng. degree from Harbin Engineering University, China, in 2015, where she is currently pursuing the Ph.D. degree. Her research interests include privacy protection, data mining, and machine learning.

KEJIA ZHANG received the B.S. degree in information management from Beijing Normal University, China, in 2004, and the M.S. and Ph.D. degrees in computer science and technology from the Harbin Institute of Technology, China, in 2007 and 2012, respectively. He is currently an Associate Professor with Harbin Engineering University, China. His research interests include wireless sensor networks, the Internet of Things, and privacy protection.

HONGTAO SONG received the Ph.D. degree in automation from Harbin Engineering University, China, in 2009. He is currently an Assistant Professor with the College of Computer Science and Technology, Harbin Engineering University. His research interests include sensor networks, machine learning, and the IoT.

HAITAO ZHANG was born in 1972. She was the Ph.D. Associate Professor and a Master Tutor. She has publication of over 20 articles. She holds three NFSC and two patents. Her research interests include network and information security, intelligent information processing, and big data. She received the first prize and the third prize in the Science and Technology, Heilongjiang.