Privacy-Preserving Collaborative Association Rule Mining

Justin Zhan, Stan Matwin and LiWu Chang

[1] School of Information Technology & Engineering,
University of Ottawa, Canada
zhizhan@site.uottawa.ca
[2] School of Information Technology & Engineering,
University of Ottawa, Canada
Institute for Computer Science,
Polish Academy of Sciences, Warsaw, Poland
stan@site.uottawa.ca
[3] Center for High Assurance Computer Systems,
Naval Research Laboratory, USA
lchang@itd.nrl.navy.mil

Abstract. This paper introduces a new approach to a problem of data sharing among multiple parties, without disclosing the data between the parties. Our focus is data sharing among parties involved in a data mining task. We study how to share private or confidential data in the following scenario: multiple parties, each having a private data set, want to collaboratively conduct association rule mining without disclosing their private data to each other or any other parties. To tackle this demanding problem, we develop a secure protocol for multiple parties to conduct the desired computation. The solution is distributed, i.e., there is no central, trusted party having access to all the data. Instead, we define a protocol using homomorphic encryption techniques to exchange the data while keeping it private.

Key Words: Privacy, security, association rule mining.

1 INTRODUCTION

In this paper, we address the following problem: multiple parties are cooperating on a data-rich task. Each of the parties owns data pertinent to the aspect of the task addressed by this party. More specifically, the data consists of instances, all parties have data about all the instances involved, but each party has its own view of the instances - each party works with its own attribute set. The overall performance, or even solvability, of this task depends on the ability of performing data mining using all the attributes of all the parties. The parties, however, may be unwilling to release their attribute to other parties, due to privacy or confidentiality of the data. How can we structure information sharing between the parties so that the data will be shared for the purpose of data mining, while at the same time specific attribute values will be kept confidential by the parties to whom they belong? This is the task addressed in this paper. In the privacy-oriented data mining this task is known as data mining with vertically partitioned data (also known as heterogeneous collaboration [6].) Examples of such tasks abound in business, homeland security, coalition building, medical research, etc.

The following scenarios illustrate situations in which this type of collaboration is interesting: (1) Multiple competing supermarkets, each having an extra large set of data records of its customers' buying behaviors, want to conduct data mining on their joint data set for mutual benefit. Since these companies are competitors in the market, they do not want to disclose too much about their customers' information to each other, but they know the results obtained from this collaboration could bring them an advantage over other competitors. (2) Success of homeland security aiming to counter terrorism depends on combination of strength across different mission areas, effective international collaboration and information sharing to support coalition in which different organizations and nations must share some, but not all, information. Information privacy thus becomes extremely important: all the parties of the collaboration promise to provide their private data to the collaboration, but neither of them wants each other or any other party to learn much about their private data. (3) Vidya and Clifton [6] provide the following convincing example in the area of automotive safety: Ford Explorers with Firestone tires from a specific factory had tread separation problems in certain situations. Early identification of the real problem could have avoided at least some of the 800 injuries that occurred in accidents attributed to the faulty tires. Since the tires did not have problems on other vehicles, and other tires on Ford Explorers did not pose a problem, neither side felt responsible. Both manufacturers had their own data, but only early generation of association rules based on all of the data may have enabled Ford and Firestone to collaborate in resolving this safety problem.

Without privacy concerns, all parties can send their data to a trusted central place to conduct the mining. However, in situations with privacy concerns, the parties may not trust anyone. We call this type of problem the *Privacy-preserving Collaborative Data Mining problem.* Homogeneous collaboration means that each party has the same sets of attributes [7]. As stated above, in this paper we are interested in heterogeneous collaboration where each party has different sets of attributes [6].

Data mining includes a number of different tasks, such as association rule mining, classification, and clustering. This paper studies the association rule mining problem. The goal of association rule mining is to discover meaningful association rules among the attributes of a large quantity of data. For example, let us consider the database of a medical study, with each attribute representing a characteristic of a patient. A discovered association rule pattern could be "70% of patients who suffer from medical condition C have a gene G". This information can be useful for the development of a diagnostic test, for pharmaceutical research, etc. Based on the existing association rule mining technologies, we study the *Privacy-preserving Collaborative Association Rule Mining* problem defined as follows: multiple parties want to conduct association rule mining on a data set that consists of all the parties' private data, but neither party is willing to disclose her raw data to each other or any other parties. In this paper, we develop a protocol, based on homomorphic cryptography, to tackle the problem.

The paper is organized as follows: The related work is discussed in Section 2. We describe the association rule mining procedure in Section 3. We then present our proposed secure protocols in Section 4. We give our conclusion in Section 5.

2 RELATED WORK

2.1 Secure Multi-Party Computation

A Secure Multi-party Computation (SMC) problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output. The SMC problem literature was introduced by Yao [13]. It has been proved that for any polynomial function, there is a secure multi-party computation solution [5]. The approach used is as follows: the function F to be computed is firstly represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets corresponding shares of the input wires and the output wires for every gate. This approach, though appealing in its generality and simplicity, is highly impractical for large datasets.

2.2 Privacy-Preserving Data Mining

In early work on privacy-preserving data mining, Lindell and Pinkas [8] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by secure multi-party computation (SMC) research. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [6], where a solution to the association rule mining problem for the case of two parties was proposed.

Randomization approaches were firstly proposed by Agrawal and Srikant in [3] to solve privacy-preserving data mining problem. In addition to perturbation, aggregation of data values [11] provides another alternative to mask the actual data values. In [1], authors studied the problem of computing the kth-ranked element. Dwork and Nissim [4] showed how to learn certain types of boolean functions from statistical databases in terms of a measure of probability difference with respect to probabilistic implication, where data are perturbed with noise for the release of statistics. In this paper, we focus on privacy-preserving among the intra-party computation.

The work most related to ours is [12], where Wright and Yang applied homomorphic encryption [10] to the Bayesian networks induction for the case of *two* parties. However, the core protocol which is called *Scalar Product Protocol* can be easily attacked. In their protocol, since Bob knows the encryption key e, when Alice sends her encrypted vector $(e(a_1), \cdots, e(a_n))$ where a_is are Alice's vector elements, Bob can easily figure out whether a_i is 1 or 0 through the following attack: Bob computes $e(1)$, and then compares it with $e(a_i)$. If $e(1) = e(a_i)$, then $a_i = 1$, otherwise $a_i = 0$. In this paper, we develop a secure *two-party* protocol and a secure *multi-party* protocol based on homomorphic encryption. Our contribution not only overcomes the attacks which exist in [12], but more importantly, a general secure protocol involving multiple parties is provided.

3 MINING ASSOCIATION RULES ON PRIVATE DATA

Since its introduction in 1993 [2], the association rule mining has received a great deal of attention. It is still one of most popular pattern-discovery methods in the field of knowledge discovery. Briefly, an association rule is an expression $X \Rightarrow Y$, where X and Y are sets of items. The meaning of such rules is as follows: Given a database D of records, $X \Rightarrow Y$ means that whenever a record R contains X then R also contains Y with certain confidence. The rule confidence is defined as the percentage of records containing both X and Y with regard to the overall number of records containing X. The fraction of records R supporting an item X with respect to database D is called the support of X.

3.1 Problem Definition

We consider the scenario where multiple parties, each having a private data set (denoted by D_1, D_2, $\cdots$ and D_n respectively), want to collaboratively conduct association rule mining on the concatenation of their data sets. Because they are concerned about their data privacy, neither party is willing to disclose its raw data set to others. Without loss of generality, we make the following assumptions about the data sets (the assumptions can be achieved by pre-processing the data sets D_1, D_2, $\cdots$ and D_n, and such a pre-processing does not require one party to send her data set to other parties): (1) all the data sets contain the same number of transactions. Let N denote the total number of transactions for each data set. (2) The identities of the ith (for $i \in [1, N]$) transaction in all the data sets are the same.

Privacy-Preserving Collaborative Association Rule Mining problem: Party 1 has a private data set D_1, party 2 has a private data set D_2, $\cdots$ and party n has a private data set D_n. The data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ forms a database, which is actually the concatenation of D_1, D_2, $\cdots$ and D_n (by putting D_1, D_2, $\cdots$ and D_n together so that the concatenation of the ith row in D_1, D_2, $\cdots$ and D_n becomes the ith row in $[D_1 \cup D_2 \cup \cdots \cup D_n]$). The n parties want to conduct association rule mining on $[D_1 \cup D_2 \cup \cdots \cup D_n]$ and to find the association rules with support and confidence being greater than the given thresholds. We say an association rule (e.g., $x_i \Rightarrow y_j$) has confidence $c\%$ in the data set $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ $c\%$ of the records which contain x_i also contain y_j (namely, $c\% = P(y_j \mid x_i)$). We say that the association rule has support $s\%$ in $[D_1 \cup D_2 \cup \cdots \cup D_n]$ if $s\%$ of the records in $[D_1 \cup D_2 \cdots \cup D_n]$ contain both x_i and y_j (namely, $s\% = P(x_i \cap y_j)$). Consequently, in order to learn association rules, one must compute the candidate itemsets, and then prune those that do not meet the preset confidence and support thresholds. In order to compute confidence and support of a given candidate itemset, we must compute, for a given itemset C, the frequency of attributes (items) belonging to C in the entire database (i.e., we must count how many attributes in C are present in all records of the database, and divide the final count by the size of the database which is N.) Note that association rule mining works on binary data, representing presence or absence of items in transactions. However, the proposed approach is not limited to the assumption about the binary character of the data in the content of association rule mining since non-binary data can be transformed to binary data via discreterization.

3.2 Association Rule Mining Procedure

The following is the procedure for mining association rules on $[D_1 \cup D_2 \cdots \cup D_n]$.

```
1. $L_1$ = large 1-itemsets
2. **for** (k = 2; $L_{k-1} \neq \phi$; k++) **do begin**
3.     $C_k$ = **apriori-gen**($L_{k-1}$)
4.         **for** all candidates $c \in C_k$ **do begin**
5.             **Compute *c.count*** (*c.count divided by the total number of records is the support of a given item set. We will show how to compute it in Section 3.3.*)
6.         **end**
7.     $L_k = \{c \in C_k | c.count \geq \textit{min-sup}\}$
8. **end**
9. Return L = $\cup_k L_k$
```

The procedure **apriori-gen** is described in the following (please also see [2] for details).

apriori-gen(L_{k-1}: large (k-1)-itemsets)

```
1. insert into $C_k$
```

```
2. select $p.item_1$, $p.item_2$, $\cdots$, $p.item_{k-1}$,$q.item_{k-1}$
3. from $L_{k-1}$ p, $L_{k-1}$ q
4. where $p.item_1 = q.item_1, \cdots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$;
```

Next, in the *prune* step, we delete all itemsets $c \in C_k$
such that some (k-1)-subset of c is not in L_{k-1}:

```
1. for all itemsets $c \in C_k$ do
2.     for all (k-1)-subsets s of c do
3.         if($s \notin L_{k-1}$) then
4.             delete c from $C_k$;
```

3.3 How to compute *c.count*

In the procedure of association rule mining, the only steps accessing the actual data values are: (1) the initial step which computes large 1-itemsets, and (2) the computation of *c.count*. Other steps, particularly computing candidate itemsets, use merely attribute names. To compute large 1-itemsets, each party selects her own attributes that contribute to large 1-itemsets. As only a single attribute forms a large 1-itemset, there is no computation involving attributes of other parties. Therefore, no data disclosure across parties is necessary. However, to compute *c.count*, a computation accessing attributes belonging to different parties is necessary. How to conduct this computations across parties without compromising each party's data privacy is the challenge we address.

If all the attributes belong to the same party, then *c.count*, which refers to the frequency counts for candidates, can be computed by this party. If the attributes belong to different parties, they then construct vectors for their own attributes and apply our secure protocols, which will be discussed in Section 4, to obtain *c.count*. We use an example to illustrate how to compute *c.count* among two parties. Alice and Bob construct vectors C_{k1} and C_{k2} for their own attributes respectively. To obtain *c.count*, they need to compute $\sum_{i=1}^{N}(C_{k1}[i] \cdot C_{k2}[i])$ where N is the total number of values in each vector. For instance, if the vectors are as depicted in Fig.1, then $\sum_{i=1}^{N}(C_{k1}[i] \cdot C_{k2}[i]) = \sum_{i=1}^{5}(C_{k1}[i] \cdot C_{k2}[i]) = 3$. We provide a secure protocol in Section 4 for the two parties to compute this value without revealing their private data to each other.

4 COLLABORATIVE ASSOCIATION RULE MINING PROTOCOL

How the collaborative parties jointly compute *c.count* without revealing their raw data to each other presents a great challenge. In this section, we develop two secure protocols to compute *c.count* for the case of two parties as well as the case of multiple parties, respectively.

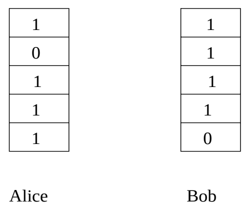

Fig. 1. Raw Data For Alice and Bob

4.1 Introducing Homomorphic Encryption

In our secure protocols, we use homomorphic encryption [10] keys to encrypt the parties' private data. In particular, we utilize the following characterizer of the homomorphic encryption functions: $e(a_1) \times e(a_2) = e(a_1 + a_2)$ where e is an encryption function; a_1 and a_2 are the data to be encrypted. Because of the property of associativity, $e(a_1 + a_2 + .. + a_n)$ can be computed as $e(a_1) \times e(a_2) \times \cdots \times e(a_n)$ where $e(a_i) \neq 0$. That is

$$e(a_1 + a_2 + \cdots + a_n) = e(a_1) \times e(a_2) \times \cdots \times e(a_n) \tag{1}$$

4.2 Secure Two-Party Protocol

Let us firstly consider the case of two parties ($n = 2$). Alice has a vector A_1 and Bob has a vector A_2. Both vectors have N elements. We use A_{1i} to denote the *ith* element in vector A_1, and A_{2i} to denote the *ith* element in vector A_2. In order to compute the *c.count* of an itemset containing A_1 and A_2, Alice and Bob need to compute the scalar product between A_1 and A_2.

Firstly, one of parties is randomly chosen as a key generator. For simplicity, let's assume Alice is selected as the key generator. Alice generates an encryption key (e) and a decryption key (d). She applies the encryption key to the addition of each value of A_1 and $R_i * X$ (e.g., $e(A_{1i} + R_i * X)$), where R_i is a random integer and X is an integer which is greater than N. She then sends $e(A_{1i} + R_i * X)$s to Bob. Bob computes the multiplication $\prod_{j=1}^{n}[e(A_{1j} + R_i * X) \times A_{2j}]$ when $A_{2j} = 1$ (since when $A_{2j} = 0$, the result of multiplication doesn't contribute to the *c.count*). He sends the multiplication results to Alice who computes $[d(e(A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X)])modX = (A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X)modX$ and obtains the *c.count*. In more detail, Alice and Bob apply the following protocol:

Protocol 1 *(Secure Two-Party Protocol)*

1. Alice performs the following:
   (a) Alice generates a cryptographic key pair (d, e) of a homomorphic encryption scheme. Let's use $e(.)$ denote encryption and $d(.)$ denote decryption. Let X be an integer number which is chosen by Alice and greater than N (i.e., the number of transactions).

   (b) Alice randomly generates a set of integer numbers R_1, R_2, $\cdots$, R_N and sends $e(A_{11} + R_1 * X)$, $e(A_{12} + R_2 * X)$, $\cdots$, and $e(A_{1N} + R_N * X)$ to Bob.
2. Bob performs the following:
   (a) Bob computes $E_1 = e(A_{11} + R_1 * X) * A_{21}$, $E_2 = e(A_{12} + R_2 * X) * A_{22}$, $\cdots$ and $E_N = e(A_{1N} + R_N * X) * A_{2N}$. Since A_{2i} is either 1 or 0, $e(A_{1i} + R_i * X) * A_{2i}$ is either $e(A_{1i} + R_i * X)$ or 0. Note that R_1, R_2, $\cdots$, and R_N are unrelated random numbers.
   (b) Bob multiplies all the E_is for those A_{2i}s that are not equal to 0. In other words, Bob computes the multiplication of all non-zero E_is, e.g., $E = \prod E_i$ where $E_i \neq 0$. Without loss of generality, let's assume only the first j elements are not equal to 0s. Bob then computes $E = E_1 * E_2 * \cdots * E_j = [e(A_{11}+R_1*X)\times A_{21}]\times[e(A_{12}+R_2*X)\times A_{22}]\times\cdots\times[e(A_{1j}+R_j*X)\times A_{2j}] = [e(A_{11} + R_1 * X) \times 1] \times [e(A_{12} + R_2 * X) \times 1] \times \cdots \times [e(A_{1j} + R_j * X) \times 1] = e(A_{11} + R_1 * X) \times e(A_{12} + R_2 * X) \times \cdots \times e(A_{1j} + R_j * X) = e(A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X)$ according to Eq. 1.
   (c) Bob sends E to Alice.
3. Alice computes $d(E) mod X$ which is equal to $c.count$.

4.3 Analysis of Two-Party Protocol

Correctness Analysis Let us assume that both parties follow the protocol. When Bob receives each encrypted element $e(A_{1i} + R_i * X)$, he computes $e(A_{1i} + R_i) * A_{2i}$. If $A_{2i} = 0$, then *c.count* does not change. Hence, Bob computes the product of those elements whose A_{2i}s are 1s and obtains $\prod e(A_{1j} + R_j) = e(A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X)$ (note that the first j terms are used for simplicity in explanation), then sends it to Alice. After Alice decrypts it, she obtains $[d(e(A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X))] mod X = (A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X) mod X$ which is equal to the desired *c.count*. The reasons are as follows: when $A_{2i} = 1$ and $A_{1i} = 0$, *c.count* does not change; only if both A_{1i} and A_{2i} are 1s, *c.count* changes. Since $(A_{11} + A_{12} + \cdots + A_{1j}) \leq N < X$, $(A_{11} + A_{12} + \cdots + A_{1j} + (R_1 + R_2 + \cdots + R_j) * X) mod X = (A_{11} + A_{12} + \cdots + A_{1j})$. In addition, when $A_{2i} = 1$, $(A_{11} + A_{12} + \cdots + A_{1j})$ gives the total number of times that both A_{1i} and A_{2i} are 1s. Therefore, *c.count* is computed correctly.

Complexity Analysis The bit-wise communication cost of this protocol is $\alpha(N + 1)$ where α is the number of bits for each encrypted element. The cost is approximately α times of the *optimal* cost of a two-party scalar product. The optimal cost of a scalar product is defined as the cost of conducting the product of A_1 and A_2 without privacy constraints, namely one party simply sends its data in plaintext to the other party.

The computational cost is caused by the following: (1) the generation of a cryptographic key pair; (2) the total number of N encryptions, e.g., $e(A_{1i} + R_i * X)$ where $i \in [1, N]$; (3)at most 3N-1 multiplications; (4) one decryption; (5) one modulo operation; (6) N additions.

Privacy Analysis All the information that Bob obtains from Alice is $e(A_{11} + R_1 * X)$, $e(A_{12} + R_2 * X)$, $\cdots$ and $e(A_{1N} + R_N * X)$. Bob does not know the encryption key e, R_is, and X. Assuming the homomorphic encryption is secure, he cannot know Alice's original element values. The information that Alice obtains from Bob is $\prod[e(A_{1i} + R_i * X) * A_{2i}]$ for those is that $A_{2i} = 1$. After Alice computes $[d(\prod e(A_{1i} + R_i * X) * A_{2i})]modX$ for those is that $A_{2i} = 1$, she only obtains *c.count*, and can't exactly know Bob's original element values. Note that the trouble with binary data presented in [6] does not exist for our protocol. More importantly, [6] only deals with the case of two parties; however, our protocol can cope with the case of two parties as well as the case of multiple parties.

4.4 Secure Multi-Party Protocol

We have discussed our secure protocol for two parties. In this section, we develop a protocol to deal with the case where more than two parties are involved. Without loss of generality, assuming Party 1 has a private vector A_1, Party 2 has a private vector A_2, $\cdots$ and Party n has a private vector A_n. For simplicity, we use P_i to denote Party i.

In our protocol, P_1, P_2, $\cdots$ and P_{n-1} share a cryptographic key pair (d, e) of a homomorphic encryption scheme and a large integer X which is greater than N. P_1 modifies every element of its private vectors with $R_{1i} * X$, where R_{1i} is a random integer number, then encypts and sends them to P_n. Like P_1, all other parties send their encrypted values to P_n too. P_n will multiply received values with her own element, e.g., $E_i = e(A_{1i} + R_{1i} * X) * e(A_{2i} + R_{2i} * X) * \cdots * e(A_{(n-1)i} + R_{(n-1)i} * X) * A_{ni}$. P_n randomly permutes E_is and divides those non-zero E_is into n-1 parts with each part having approximately equal number of elements, and sends them to n-1 other parties who compute $[d(E_i)]modX = [d(e(A_{1i} + R_{1i} * X) * e(A_{2i} + R_{2i} * X) * \cdots * e(A_{(n-1)i} + R_{(n-1)i} * X))]modX = (A_{1i} + A_{2i} + \cdots + A_{(n-1)i} + (R_{1i} + R_{2i} + \cdots + R_{(n-1)i}) * X)modX = (A_{1i} + A_{2i} + \cdots + A_{(n-1)i})$. Suppose P_1 gets the above $[d(E_i)mod]X$. P_1 then compares whether $(A_{1i} + A_{2i} + \cdots + A_{(n-1)i}) = n - 1$. If it is true, then $c.count_1$ increases by 1. Consequently, P_1 gets $c.count_1$. Similarly, P_2 gets $c.count_2$, $\cdots$ and P_{n-1} gets $c.count_{n-1}$.

To avoid P_i knowing $c.count_j$, where $i \neq j$, we perform the following steps: P_n generates another cryptographic key pair (e_1, d_1) of a homomorphic encryption scheme and sends the encryption key e_1 to P_1, P_2, $\cdots$ and P_{n-1} who compute $e_1(c.count_1)$, $e_1(c.count_2)$, $\cdots$ and $e_1(c.count_{n-1})$ respectively. One of those n-1 parties (e.g., P_j) is randomly chosen. All other parties P_ks where $k \neq j$ send $e_1(c.count_k)$s to P_j. P_j multiplies all the encrypted counts and obtains the encrypted *c.count*. That is $e_1(c.count_1) * e_1(c.count_2) * \cdots * e_1(c.count_{n-1}) = e_1(c.count_1 + c.count_2 + \cdots + c.count_{n-1}) = e_1(c.count)$. P_j sends $e_1(c.count)$ to P_n who computes $d_1(e_1(c.count))$ and gets *c.count*.

Protocol 2 *(Secure Multi-Party Protocol)*

1. P_1, P_2, $\cdots$, and P_{n-1} perform the following:
   (a) P_1, P_2, $\cdots$ and P_{n-1} jointly generate a cryptographic key pair (d, e) of a homomorphic encryption scheme. Let's use $e(.)$ denote encryption and $d(.)$ denote decryption. They also generate the number, X, where X is an integer which is greater than N.
   (b) P_1 generates a set of random integers R_{11}, R_{12}, $\cdots$, R_{1N} and sends $e(A_{11} + R_{11} * X)$, $e(A_{12} + R_{12} * X)$, $\cdots$, and $e(A_{1N} + R_{1N} * X)$ to P_n; P_2 generates a set of random integers R_{21}, R_{22}, $\cdots$, R_{2N} and sends $e(A_{21} + R_{21} * X)$, $e(A_{22} + R_{22} * X)$, $\cdots$, and $e(A_{2N} + R_{2N} * X)$ to P_n, $\cdots$, P_{n-1} generates a set of random integers $R_{(n-1)1}$, $R_{(n-1)2}$, $\cdots$, $R_{(n-1)N}$ and sends $e(A_{(n-1)1} + R_{(n-1)1} * X)$, $e(A_{(n-1)2} + R_{(n-1)2} * X)$, $\cdots$, $e(A_{(n-1)N} + R_{(n-1)N} * X)$ to P_n.
2. P_n performs the following:
   (a) P_n computes $E_1 = e(A_{11}+R_{11}*X) * e(A_{21}+R_{21}*X) * \cdots * e(A_{(n-1)1}+R_{(n-1)1}) * A_{n1} = e(A_{11}+A_{21}+\cdots+A_{(n-1)1}+(R_{11}+R_{21}+\cdots+R_{(n-1)1})*X) * A_{n1}$,
   $E_2 = e(A_{12}+R_{12}*X) * e(A_{22}+R_{22}*X) * \cdots * e(A_{(n-1)2}+R_{(n-1)2}*X) * A_{n2} = e(A_{12}+A_{22}+\cdots+A_{(n-1)2}+(R_{12}+R_{22}+\cdots+R_{(n-1)2})*X)*A_{n2}$,
   $E_3 = e(A_{13}+R_{13}*X) * e(A_{23}+R_{23}*X) * \cdots * e(A_{(n-1)3}+R_{(n-1)3}*X) * A_{n3} = e(A_{13}+A_{23}+\cdots+A_{(n-1)3}+(R_{13}+R_{23}+\cdots+R_{(n-1)3})*X)*A_{n3}$,
   $\cdots$, and
   $E_N = e(A_{1N} + R_{1N} * X) * e(A_{2N} + R_{2N} * X) * \cdots * e(A_{(n-1)N} + R_{(n-1)N} * X) * A_{nN} = e(A_{1N} + A_{2N} + \cdots + A_{(n-1)N} + (R_{1N} + R_{2N} + \cdots + R_{(n-1)N}) * X) * A_{nN}$.
   Since A_{ni} is either 1 or 0, E_1 is either $e(A_{11} + A_{21} + \cdots + A_{(n-1)1} + (R_{11} + R_{21} + \cdots + R_{(n-1)1}) * X)$ or 0; E_2 is either $e(A_{12} + A_{22} + \cdots + A_{(n-1)2} + (R_{12} + R_{22} + \cdots + R_{(n-1)2}) * X)$ or 0; $\cdots$; and E_N is either $e(A_{1N} + A_{2N} + \cdots + A_{(n-1)N} + (R_{1N} + R_{2N} + \cdots + R_{(n-1)N}) * X)$ or 0.
   (b) P_n randomly permutes [9] the E_1, E_2, $\cdots$ and E_N, then obtains the permuted sequence D_1, D_2, $\cdots$ and D_N.
   (c) From computational balance point of view, we want each party among P_1, P_2, $\cdots$ and P_{n-1} to decrypt some of non-zero D_is. [4] Consequently, in our protocol P_n divides those non-zero elements from D_1, D_2, $\cdots$ and D_N into $n-1$ parts with each part having approximately equal number of elements.
   (d) P_n sends the $n-1$ parts to P_1, P_2, $\cdots$ and P_{n-1} respectively, so that P_1 gets the first part, P_2 gets the second part, $\cdots$ and P_{n-1} gets the $(n-1)th$ part.
3. Compute *c.count*

[4] We assume that the number of non-zero elements of D_is (Let's denote the number by ND) is $\geq$ n-1. If not, we randomly select the number of ND parties from P_1, P_2, $\cdots$ and P_{n-1}, and send each non-zero element to each of the selected parties. Moreover, in practice $N \gg n$.

(a) P_1, P_2, $\cdots$ and P_{n-1} decrypt the encrypted terms received from P_n, then modulo X. Due to the properties of homomorphic encryption, this gives them the correct value of *c.count* for a candidate itemset consisting of attributes A_1, A_2, $\cdots$ and A_n. Note that if a decrypted term is equal to n-1 *mod* X, it means the values of P_1, P_2, $\cdots$, P_{n-1} and P_n are all 1s[5]. For example, if P_i obtains E_i, she then computes $d(E_i)$ *mod* X $= (A_{1i} + A_{2i} + \cdots + A_{(n-1)i} + (R_{1i} + R_{2i} + \cdots + R_{(n-1)i}) * X)$ *mod* X $= A_{1i} + A_{2i} + \cdots + A_{(n-1)i}$. Consequently, P_1, P_2, $\cdots$ and P_{n-1} compare whether each decrypted term is equal to $n-1$ *mod*X. If yes, then each P_i (i = 1, 2, $\cdots$ and n-1) increases her *c.count*$_i$ by 1.

(b) What remains is the computation of *c.count* by adding the *c.count*$_i$s. Since we do not want a party P_i to know the *count*$_j$ for $j \neq i$, we use the following cryptographic scheme avoiding this disclosure: P_n generates another cryptographic key pair (d_1, e_1) of a homomorphic encryption scheme[6]. She then sends e_1 to P_1, P_2, $\cdots$ and P_{n-1}. P_i $(i = 1, 2, \cdots, n-1)$ encrypts *c.count*$_i$ by using e_1. In other words, P_1 computes $e_1(c.count_1)$, P_2 computes $e_1(c.count_2)$, $\cdots$ and P_{n-1} computes $e_1(c.count_{n-1})$.

(c) One of parties among P_1, P_2, $\cdots$ and P_{n-1} (e.g., P_j) is randomly selected. Other parties P_ks among P_1, P_2, $\cdots$ and P_{n-1} $(k \neq j)$ send their encrypted *c.count*$_k$ to P_j, who then multiplies all the encrypted counts including her own $e_1(c.count_j)$ and obtains the encrypted *c.count*. That is, $e_1(c.count) = e_1(c.count_1) * e_1(c.count_2) * e_1(c.count_3) * \cdots * e_1(c.count_{n-1}) = e_1(c.count_1 + c.count_2 + \cdots + c.count_{n-1})$.

(d) P_j sends $e_1(c.count)$ to P_n.

(e) P_n computes $d_1(e_1(c.count)) = c.count$. Finally, P_n obtains *c.count* and shares with P_1, P_2, $\cdots$ and P_{n-1}.

4.5 Analysis of Multi-Party Protocol

Correctness Analysis Assuming all of the parties follow the protocol, to show the *c.count* is correct, we need to consider:

– If the element of P_n is 1 (e.g., $A_{ni} = 1$), and $A_{1i} + A_{2i} + \cdots + A_{(n-1)i} = n-1$, then *c.count* increases by 1. Since $[d(e(A_{1i} + R_{1i} * X) * e(A_{2i} + R_{2i} * X) * \cdots * e(A_{(n-1)i} + R_{(n-1)i} * X))]$ *mod* X $= [d(e(A_{1i} + A_{2i} + \cdots + A_{(n-1)i} + (R_{1i} + R_{2i} + \cdots + R_{(n-1)i}) * X))]$ *mod* X $= A_{1i} + A_{2i} + \cdots + A_{(n-1)i}$, if $A_{ni} = 1$ and $A_{1i} + A_{2i} + \cdots + A_{(n-1)i} = n-1$, that means A_{1i}, A_{2i}, $\cdots$, $A_{(n-1)i}$ and A_{ni} are all 1s, then *c.count* should increase by 1. For other scenarios, either $A_{ni} = 0$ or $A_{1i} + A_{2i} + \cdots + A_{(n-1)i} \neq n-1$ or both, *c.count* doesn't change.

– In the protocol, P_n permutes E_is before sending them to P_1, P_2, $\cdots$ and P_{n-1}. Permutation does not affect *c.count*. We evaluate whether each element contributes to *c.count*, we then sum those that contribute. Summation is not affected by a permutation. Therefore, the final *c.count* is correct.

[5] The value of P_n must be 1 because P_n doesn't send the D_is to those $n-1$ parties if $D_i = 0$.

[6] (d_1, e_1) is independent from (d, e).

Complexity Analysis The bit-wise communication cost of this protocol is at most 2αnN where α is the number of bits for each encrypted element. The following contributes to the computational cost: (1) the generation of two cryptographic key pairs; (2) the total number of nN + (n-1) encryptions; (3) the total number of $n(N+1)-1$ multiplications; (4) the generation of permutation function; (5) the total number of N permutations; (6) at most N decryptions; (7) at most N modulo operations; (8) (n-1)N additions.

Privacy Analysis P_n obtains all the encrypted terms from other parties. Since P_n does not know the encryption key, R_{ij}, and X, she cannot know the original values of other parties' elements. Each party of P_1, P_2, $\cdots$ and P_{n-1} obtains some of D_is. Since D_is are in permuted form and those n-1 parties don't know the permutation function, they cannot know the P_n's original values either.

In our protocol, those $n-1$ parties' *c.count*s are also preserved because of the encryption. What P_j receives from other $n-2$ parties is the encrypted counts. Since P_j doesn't know the encryption key e_1, P_j cannot know other $n-2$ parties' counts. What P_n receives from P_j is the multiplication of all $c.count_i$s. Therefore, she doesn't know each individual P_i's count (i = 1, 2, $\cdots$, n-1).

We also emphasis that Step (2b) are required, the goal is to prevent other parties from knowing P_n's values. Step (2c) is for the consideration of computational balance among P_1, P_2, $\cdots$, and P_{n-1}. Step (3b) to (3e) is to further prevent parties from knowing $c.count_i$s each other. If the collaborative parties allow sharing $c.count_i$s each other, some of steps can be removed and communication cost is saved.

5 CONCLUDING REMARKS

In this paper, we consider the problem of privacy-preserving collaborative association rule mining. In particular, we study how multiple parties can collaboratively conduct association rule mining on their joint private data. We develop a secure collaborative association rule mining protocol based on homomorphic encryption scheme. In our protocol, the parties do not send all their data to a central, trusted party. Instead, we use the homomorphic encryption techniques to conduct the computations across the parties without compromising their data privacy. Privacy analysis is provided. Correctness of our protocols is shown and complexity of the protocols is addressed as well. As future work, we will develop a privacy measure to quantitatively measure the privacy level achieved by our proposed secure protocols. We will also apply our technique to other data mining computations, such as secure collaborative clustering.

6 ACKNOWLEDGEMENT

The first two authors acknowledge generous support of the Natural Sciences and Engineering Research Council of Canada, and the Communications and Information Technology Ontario for their research.

References

1. G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k th-ranked element. In *EUROCRYPT pp 40-55*, 2004.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.
3. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
4. C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases.
5. O. Goldreich. Secure multi-party computation (working draft). http://www.wisdom.weizmann.ac.il /home/oded/public_html/foc.html, 1998.
6. J.Vaidya and C.W.Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*.
7. M. Kantarcioglu and C. Clifton. Privacy preserving data mining of association rules on horizontally partitioned data. In *Transactions on Knowledge and Data Engineering, IEEE Computer Society Press, Los Alamitos, CA, to appear.*
8. Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science*, volume 1880, 2000.
9. M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, January 1996.
10. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptography - EUROCRYPT '99, pp 223-238, Prague, Czech Republic*, May 1999.
11. L. Sweeney. k-anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10 (5), pp 557–570.*
12. R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.
13. A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.