Private Mining of Association Rules

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
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 two parties involved in a data mining task. We study how to share private or confidential data in the following scenario: two 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 two 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. All the parties are treated symmetrically: they all participate in the encryption and in the computation involved in learning the association rules.

Key Words: Privacy, security, association rule mining.

1 INTRODUCTION

In this paper, we address the following problem: two 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, and 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 two parties, however, may be unwilling to release their attributes to other parties that are not involved in collaboration, 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 [8].) Examples of such tasks abound in business, homeland security, coalition building, medical research, etc.

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.* As stated above, in this paper we are interested in heterogeneous collaboration where each party has different sets of attributes [8].

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 *Private Mining of Association Rules* problem defined as follows: two parties want to conduct association rule mining on a data set that consists 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 [7]. 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 [9] 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 [4, 8].

Random perturbation-based 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 [5] showed how to learn certain types of boolean functions from statistical databases in the context of probabilistic implication for the disclosure of statistics.

Homomorphic encryption [10], which transforms multiplication of encrypted plaintexts into the encryption of the sum of the plaintexts, has recently been used in secure multi-party computation. For instance, Freedmen, Nissim and Pinkas [6] applied it to set intersection. The work most related to ours is [12], where Wright and Yang applied homomorphic encryption 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 based on homomorphic encryption. Our contribution not only overcomes the attacks which exist in [12], but applies our secure protocol to tackle collaborative association rule mining problems.

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 transactions, $X \Rightarrow Y$ means that whenever a transaction R contains X then R also contains Y with certain confidence. The rule confidence is defined as the percentage of transactions containing both X and Y with regard to the overall number of transactions containing X. The fraction of transactions 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 two parties, each having a private data set (denoted by D_1 and D_2 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 the other. 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 and D_2, and such a pre-processing does not require one party to send her data set to other party): (1) D_1 and D_2 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 D_1 and D_2 are the same.

Private Mining of Association Rule problem: Party 1 has a private data set D_1, party 2 has a private data set D_2. The data set $[D_1 \cup D_2]$ forms a database, which is actually the concatenation of D_1 and D_2 (by putting D_1 and D_2 together so that the concatenation of the ith row in D_1 and D_2 becomes the ith row in $[D_1 \cup D_2]$). The two parties want to conduct association rule mining on $[D_1 \cup D_2]$ 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 $[D_1 \cup D_2]$ if in $[D_1 \cup D_2]$ $c\%$ of the transactions 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 $[D_1 \cup D_2]$ if $s\%$ of the transactions in $[D_1 \cup D_2]$ 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 transactions 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.

3.2 Association Rule Mining Procedure

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

```
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 [4]
6.       end
7.    $L_k = \{c \in C_k | c.count \geq min\text{-}sup\}$
8. end
9. Return L = $\cup_k L_k$
```

[4] *c.count* divided by the total number of transactions is the support of a given item set. We will show how to compute it in Section 3.3.

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 the other party. 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 protocol, which will be discussed in Section 4, to obtain *c.count*. We use an example to illustrate how to compute *c.count*. 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 a secure protocol to compute *c.count* between two parties.

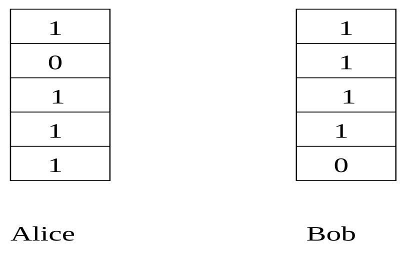

Fig. 1. Raw Data For Alice and Bob

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's assume that 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 follow the following protocol:

Protocol 1 *(Secure Two-Party Protocol)*

1. 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).

2. Alice randomly generates an integer number R_1 and sends $e(A_{11} + R_1 * X)$ to Bob.
3. Bob computes $e(A_{11} + R_1 * X) * A_{21}$.
4. Repeat Step 2 - 3 until Bob gets $E_1 = e(A_{11} + R_1 * X) * A_{21}$, $E_2 = e(A_{12} + R_2 * X) * A_{22}$, $\cdots$ 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.
5. 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.
6. Bob sends E to Alice.
7. 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 that $A_{2i} = 1$. After Alice computes $[d(\prod e(A_{1i} + R_i * X) * A_{2i})] mod X$ for those that $A_{2i} = 1$, she only obtains *c.count*, and can't exactly know Bob's original element values. From symmetric point of view, we could let Alice and Bob be the key generator in turn. When computing the first half of their vector product, Alice is selected as the key generator; when computing the second half vector product, Bob is selected as the key generator.

5 CONCLUSION

In this paper, we consider the problem of private mining of association rules. In particular, we study how two 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 need to 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.

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. W. Du and Z. Zhan. Building decision tree classifier on private data. In *Workshop on Privacy, Security, and Data Mining at The 2002 IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan, December 9 2002.
5. C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases.
6. M. Freedman, K. Nissim, and B. Pinkas. Effiicent private matching and set intersection. In *EUROCRYPT pp 1-19*, 2004.

7. O. Goldreich. Secure multi-party computation (working draft). http://www.wisdom.weizmann.ac.il /home/oded/public_html/foc.html, 1998.
8. 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*.
9. Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science*, volume 1880, 2000.
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.