Software Engineering
2013; 1(1): 1-6
Published online June 20, 2013 (http://www.sciencepublishinggroup.com/j/se)
doi: 10.11648/j.se.20130101.11

To allot secrecy-safe association rules mining schema using FP tree

S. Suresh[1], S. Uvaraj[2], N. Kannaiya Raja[3]

[1]Sri Venkateswara College of Engineering, Chennai
[2]Arulmigu Meenakshi Amman College of Engineering, Kanchipuram
[3]Defence Engineering College, Ethiopia

Email address:
ss12oct92@gmail.com(S. Suresh), ujrj@rediffmail.com(S. Uvaraj), kanniya13@hotmail.co.in(N. K. Raja)

To cite this article:
S. Suresh, S. Uvaraj, N. Kannaiya Raja. To Allot Secrecy-Safe Association Rules Mining Schema Using FP Tree. *Software Engineering*. Vol. 1, No. 1, 2013, pp. 1-6. doi: 10.11648/j.se.20130101.11

Abstract: Association rules mining is a frequently used technique which finds interesting association and correlation relationships among large set of data items which occur frequently together. Nowadays, data collection is ubiquitous in social and business areas. Many companies and organizations want to do the collaborative association rules mining to get the joint benefits. However, the sensitive information leakage is a problem we have to solve and privacy- preserving techniques are strongly needed. In this paper, we focus on the privacy issue of the association rules mining and propose a secure frequent-pattern tree (FP-tree) based scheme to pre- serve private information while doing the collaborative association rules mining. We display that our schema is secure and collusion-resistant for *n* parties, which means that even if *n* - 1 dishonest party collude with a dishonest data miner in an attempt to learn the associations' rules between honest respondents and their responses, they will be unable to success.

Keywords: Association Rules, Privacy-Preserving, Cryptographic Protocol

1. Introduction

Association rules mining techniques are generally applied to databases of transactions where each transaction consists of a set of items. In such a framework the problem is to discover all associations and correlations among data items where the presence of one set of items in a transaction implies (with a certain degree of confidence) the presence of other items. Association rules are statements of the form $X1,X2, ...,Xn \Rightarrow Y$, meaning that if we find all of $X1,X2, ...,Xn$ in the transactions, then we have a good chance of finding Y . The probability of finding Y for us to accept this rule is called the confidence of the rule. We normally would search only for rules that had confidence above a certain threshold. The problem is usually decomposed into two sub-problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Much data mining starts with the assumption that we only care about sets of items with high support, they appear together in many transactions. We then find association rules only involving a high-support set of items. That is to say that $X1,X2, ...,Xn \Rightarrow Y$ must appear in at least a certain percent of the transactions, called the support threshold. How to do the global support threshold counting with respecting clients' privacy is a major problem in privacy-preserving rules mining. $supportX \Rightarrow Y = |TX \cup Y| / |DB|$ means that the support is equal to the percentage of all transactions which contain both X and Y in the whole dataset. And then we can get that: $confidentX \Rightarrow Y = supportX \Rightarrow Y / support$. The problem of mining association rules is to find all rules whose support and confidence are higher than certain user specified minimum support and confidence. Distributed mining can be applied to many applications which have their data sources located at different places. In this paper, we assume that there are *n* parties possess their private databases respectively. They want to get the common benefit for doing association rules analysis in the joint databases. For the privacy concerns, they need a private preserving system to execute the joint association rules mining. The concern is solely that values associated with

an individual entity not being revealed.

1.1. Related Work

The research of privacy-preserving techniques for data mining began in Year 2000, R. Agrawal et al [12] proposed a reconstruction procedure which is possible to accurately estimate the distribution of original data values from the random noise perturbed data. However, Evfimievski et al[13] pointed out that the privacy breach will occur in R. Agrrawal's proposal and proposed a new randomization techniques to mine association rules from transactions consisting of categorical items where the data has been randomized to preserve privacy of individual transactions. The method presented in Kantarcioglu et al. [13] is the first cryptography-based solutions for private distributed association rules mining, it assumes three or more parties, and they jointly do the distributed Apriori algorithm with the data encrypted. In the recent research papers [15][7][8], some privacy-preserving association rules schemes are proposed. These papers are similar and developed a secure multi-party protocol based on homomorphic encryption.

1.2. Motivation and Our Contributions

The related works we mentioned above have three problems. The first one is low efficiency by using the Apriori algorithm. As we know, the Apriori algorithm is not so efficient because of its candidates generation scan. The second one is the accuracy problem in [13] in which there is a trade-off between the accuracy and security. The third one is the security problem; the schemes proposed in the related works are not collusion-resistant. We propose an improved scheme to overcome these three problems.

- We apply frequent-pattern tree (FP-tree) structure proposed in [2] to execute the association rules mining and extend it to distributed association rules mining framework. We use FP-tree to compress a large database into a compact FP-tree structure to avoid costly database scans.
- We present a privacy-preserving protocol which can overcoming the accuracy problem causes randomization-based techniques and improve the efficiency compared to those cryptography-based scheme [15][7][9].
- Our privacy-preserving protocol provides a perfect security and collusion resistant property. Our scheme uses the attribute-based encryption to create the global FP-tree for each party and then uses the homomorphism encryption to merger the FP-tree to get the final result of the global association rules.

2. Preliminaries

2.1. Problem Definition

We assume that there are n parties want to do cooperation on the joint databases $DB1 \cup DB2 \cup ... \cup DBn$ without revealing the private information of database. And we assume the standard synchronous model of computation in which n parties communicate by sending messages via point-to-point channels. There are some distributed parties who want to get the global result from their data transactions over the internet. Every party Pi has their private transaction $T\ 1\ i$. They all have serious concern about their privacy while they want to get the accurate result to help their following decision. No Party should be able to learn contents of a transaction of any other client. And we want to use some cryptographic toolkits to construct a secure multi-party computation protocol to perform this task. Let $I = \{a1, a2, ..., am\}$ be a set of items, and a transaction database $DB = \_T1, T2, ..., Tn\_$, where $Ti(i \in [1...n])$ is a transaction which contains a set of items in I. The support (or occurrence frequency) of a pattern A, where A is a set of items, is the number of transactions containing A in DB. A pattern A is frequent if A's support is no less than a predefined minimum support threshold $MinSupp$.

2.2. Cryptographic Primitives

Public Key Encryption with Homomorphic Property: In modern terms, a public-key encryption scheme on a message space M consists of three algorithms (K,E,D):

1. The key generation algorithm $K(1k)$ outputs a random pair of private/public keys (sk, pk), relatively to a security parameter k.

2. The encryption algorithm $Epk(m; r)$ outputs a cipher text c corresponding to the plaintext $m \in M$, using random value r.

3. The decryption algorithm $Dsk(c)$ outputs the plaintext m associated to the cipher text c. We will occasionally omit the random coins and write $Epk(m)$ in place of $Epk(m; r)$. Note that the decryption algorithm is deterministic.

In this paper we use Palliler encrytion as public key encryption. Paillier homomorphic encryption proposed by Pallier [11]. It is provably secure and one-way based on the Decisional Composite Residuosity Assumption and the Computational Composite Residuosity Assumption: If the public key is the modulus m and the base g, then the encryption of a message x is $E(x) = gxrm \bmod m2$.

Then it has the homomorphic property $E(x1) \cdot E(x2) = (gx1rm1\ )(gx2rm2\ ) = gx1+x2\ (r1r2)m = E(x1 + x2 \bmod m) \cdot$ and + denote modular multiplication and addition, respectively.

Attributes-based Encryption (ABE) The ABE scheme is developed from Identity based encryption(IBE) which introduced by Shamir [12], is a variant of encryption which allows users to use any string as their public key (for example, an email address). This means that the sender can send messages knowing only the recipient's identity (or email address), thus eliminating the need for a separate infrastructure to distribute public keys. In their scheme, there is one authority giving out secret keys for all of the

attributes. Each encryptor then specifies a list of attributes such that any user with at least d of those attributes will be able to decrypt. They show that the scheme they present is secure.

2.3. Security Definition and Adversary Model

This paper considers both semi-honest and malicious adversaries. For Semi-honest adversaries, every party are assumed to act according to their prescribed actions in the protocol. The security definition is straightforward, particularly as in our case where only one party learns an output. The definition ensures that the party does not get more or different information than the output of the function. This is formalized by considering an ideal implementation where a trusted third party (TTP) gets the inputs of the two parties and outputs the defined function. We require that in the real implementation of the protocol that is, one without a TTP the client C does not learn different information than in the ideal implementation. We say that π privately computes a function f if there exist probabilistic, polynomial-time algorithms SA and SB such that:

$$\{(SA(a, fA(x)), fB(x))\} \equiv \{(VIEW\pi A(x), OUPUT\pi B(x)) \quad (1)$$

$$\{(fA(x), SB(b, fB(x)))\} \equiv \{(OUPUT\pi A(x), VIEW\pi B(x))\} \quad (2)$$

where $\equiv$ denotes computational indistinguishability, which means that there is no probabilistic polynomial algorithm used by an adversary A can distinguish the probability distribution over two random string. It means that no matter how the adversary tries to derive the private information from the computation, what he can get only his inputs and the random values.

3. Secure Multi-party Protocol for Association Rules Mining Based on FP-tree

3.1. Problem in Apriori-Based Distributed Association Rules Mining

Most distributed association rules mining algorithms are adaptations of existing sequential (serial) algorithms. Generally speaking two strategies for distributing data for parallel computation can be identified:

1. Data distribution: The data is apportioned amongst the processes, typically by"horizontally" segmenting the dataset into sets of records. Each process then mines its allocated segment (exchanging information on-route as necessary).

2. Task distribution: Each process has access to the entire dataset but is responsible for some subset of the set of candidate itemsets.

The Apriori heuristic achieves good performance gained by (possibly significantly) reducing the size of candidate sets. However, in situations with a large number of frequent patterns, long patterns, or quite low minimum support thresholds, an Apriori-like algorithm may suffer from the following two nontrivial costs:

– It is costly to handle a huge number of candidate sets. For example, if there are 10 power 4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10power 7 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as $\{a1... a100\}$, it must generate $2100 - 2 \approx 1030$ candidates in total. This is the inherent cost of candidate generation, no matter what implementation technique is applied.

– It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns.

3.2. The General Description of Our Proposal

Our protocol construction is based on secure multi-party computation techniques. The history of the multi-party computation problem is extensive since it was introduced by Yao [14] and extended by Goldreich, Micali, and Wigderson [7]. Secure multi-party computation (MPC) protocols allow a set of n players to securely compute any agreed function on their private inputs, where the following properties must be satisfied: *privacy*, meaning that the corrupted players do not learn any information about the other players' inputs. and *correctness*, meaning that the protocol outputs the correct function value, even when the malicious players treat. In Secure Multi-party Computation, we always assume that *semi-honest model* exists.

1. Support count among the common k-item sets privately using the homomorphic encryption scheme.

2. Using attribute-based encryption scheme, every party executes FP-tree construction and prepares for the global FP-tree construction.

3. Using the private matching scheme, every party merges the conditional FP trees to get common frequent itemsets.

4. Secure global support count computation in the merged FP-trees.

5. Output the final result of association rules from the merged global FP-trees.

Initialization: A multiparty ABE system with homomorphic property is composed of K attribute authorities and one central authority. Each attribute authority is also assigned a value dk. The system uses the following algorithms: Setup: A randomized algorithm which must be run by some trusted party (e.g. central authority). Takes k as input the security parameter. Outputs a public key, secret key pair for each of the attribute authorities, and also outputs a system public key and master secret key which will be used by the central authority.

(1) Attribute Key Generation: A randomized algorithm run by an attribute authority. Takes as input the authority's secret key, the authority's value dk, a user's GID, and a set

of attributes in the authority's domain *AkC*. (We will assume that the user's claim of these attributes has been verified before this algorithm is run). Output secret key for the user.

(2) Central Key Generation: A randomized algorithm runs by the central authority. Takes as input the master secret key and a user's GID and outputs secret key for the user.

(3) Encryption: A randomized algorithm runs by a sender. Takes as input a set of attributes for each authority, a message, and the system public key. Outputs the cipher text.

(4) Decryption: -A deterministic algorithm runs by a user. Takes as input a cipher- text, which was encrypted under attribute set AC and decryption keys for an attribute set *Au*. Outputs a message *m* if $|AkC \cap Aku| > dk$ for all authorities *k*.

3.3. Distributed Association Mining with FP-Tree

FP-growth is a divide-and-conquer methodology proposed by [8] which decomposes the association rules mining tasks into smaller ones. It only scans the database twice and does not generate candidate itemsets. The algorithm substantially reduces the search costs. At first, we let the parties build a global FP-tree together and then do the association rules mining on the global FP-tree. FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Efficiency of mining is achieved with three techniques: (1) a large database is compressed into a condensed, smaller data structure, FP-tree which avoids costly, repeated database scans, (2) our FP-tree-based mining adopts a pattern-fragment growth method to avoid the costly generation of a large number of candidate sets, and (3) a partitioning-based, divide-and-conquer method is used to decompose the mining task into a set of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space.

1. Since only the frequent items will play a role in the frequent-pattern mining, it is necessary to perform one scan of transaction database DB to identify the set of frequent items (with frequency count obtained as a by-product).

2. If the set of frequent items of each transaction can be stored in some compact structure, it may be possible to avoid repeatedly scanning the original transaction database.

3. If multiple transactions share a set of frequent items, it may be possible to merge the shared sets with the number of occurrences registered as count.

It is easy to check whether two sets are identical if the frequent items in all of the transactions are listed according to a fixed order. Given a transaction database DB and a minimum support threshold *MinSupp*, the problem of finding the complete set of frequent patterns is called the frequent-pattern mining problem. With the above observations, one may construct a frequent-pattern tree as follows. First, a scan of *DB* derives a list of frequent items, {(*f*: 4), (*c* : 4), (*a* : 3), (*b* : 3), (*m* : 3), (*p* : 3)_}(the number after ":" indicates the support), in which items are ordered in frequency descending order. Second, the root of a tree is created and labelled with"null". The FP-tree is constructed as follows by scanning the transaction database DB the second time.

1. The scan of the first transaction leads to the construction of the first branch of the tree: _(*f* : 1), (*c* : 1), (*a* : 1), (*m* : 1), (*p* : 1)_. Notice that the frequent items in the transaction are listed according to the order in the list of frequent items.

2. For the second transaction, since its (ordered) frequent item list *_f, c, a, b,m_* shares a common prefix *_f, c, a_* with the existing path *_f, c, a,m, p_*, the count of each node along the prefix is incremented by 1, and one new node (*b* : 1) is created and linked as a child of (*a* : 2) and another new node (*m* : 1) is created and linked as the child of (*b* : 1).

3. For the third transaction, since its frequent item list *_f, b_* shares only the node *_f_* with the *f*-prefix subtree, *f*'s s count is incremented by 1, and a new node (*b* : 1) is created and linked as a child of (*f* : 3).

4. The scan of the fourth transaction leads to the construction of the second branch of the tree, *{*(*c* : 1), (*b* : 1), (*p* : 1)*}*

5. For the last transaction, since its frequent item list *_c, a, m, p_* is identical to the first one, the path is shared with the count of each node along the path incremented by 1.

A frequent-pattern tree (or FP-tree in short) is a tree structure defined below:

1. It consists of one root labeled as "null", a set of item-prefix sub-trees as the children of the root, and a frequent-item-header table.

2. Each node in the item-prefix sub-tree consists of three fields: item-name, count, and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none.

3. Each entry in the frequent-item-header table consists of two fields, (1) itemname and (2) head of node-link (a pointer pointing to the first node in the FP-tree carrying the item-name).

Based on this definition, we have the following FP-tree construction algorithm. This property is directly from the FP-tree construction process, and it facilitates the access of all the frequent-pattern information related to *ai* by traversing the FP-tree once following *ai*'s node-links. We can generate the conditional pattern-bases and the conditional FP-trees generated from the existing FP-tree. Construction of a new FP-tree from a conditional pattern-base obtained during the mining of an FP- ree, the items in the frequent itemset should be ordered in the frequency descending order of node occurrence of each item instead of its support (which represents item occurrence). This is because each node in an FP-tree may represent many occurrences of an item but such a node represents a single unit (i.e., the itemset whose elements always occur together) in the construction of an item- associated FP-tree. We can

develop a distributed frequent-pattern mining algorithm with distributed FP-trees. When one party has received the tree string from another party, he generates new string by merging the received string and its own string.

4. The Details of Multi-party Mining Scheme

In our ABE scheme, we assume that the universe of attributes can be partitioned into K disjoint sets. Each will be monitored by a different authority. As mentioned above, we also have one trusted central authority who does not monitor any attributes.

4.1. Verifiable Secret Sharing

Secret-sharing schemes are used to divide a secret among a number of parties. The information given to a party is called the share (of the secret) for that party. It realizes some access structure that defines the sets of parties who should be able to reconstruct the secret by using their shares First, let's consider a very simplified scheme based on the Feldman Verifiable Secret Sharing scheme. Recall that, given *d* points *p*(1), ..., *p*(*d*) on a *d* − 1 degree polynomial, we can use Lagrange interpolation to compute *p*(*i*) for any *i*. However, given only *d*−1 points, any other points are information theoretically hidden. According to the Lagrange formula, *p*(*i*) can be computed as a linear combination of *d* known points. Let *Δj* (*i*) be the coefficient of *p*(*j*) in the computation of *p*(*i*). Then *p*(*i*) = _*j*∈*S p*(*j*)*Δj* (*i*) where *S* is a set of any d known points and *Δj*(*i*) = pi*k*∈*S,jnot equal k*(*i* − *k*)/(*j* − *k*). Note that any set of *d* random numbers defines a valid polynomial, and given these numbers we can find any other point on that polynomial.

4.2. Specifying Transaction's Attributes

If we take this approach, any user with any *d* attributes which are specified will be able to decrypt. But we want each encryptor to be able to give a specific subset of attributes such that at least *d* are necessary for decryption. In order to do this, we need an extra tool: bilinear maps, for bilinear map *e, g* ∈ *G*1, and *a, b* ∈ *Zq*, *e*(*ga, gb*) = *e*(*g, g*)*ab*. Now, suppose instead of giving each user *gp*(*i*) for each attribute *i*, we choose a random value *ti* and give *gp*(*i*)/*ti* . If the user knew *gti* for at least *d* of these attributes, he could compute *e*(*g, g*)*p*(*i*) for each i and then interpolate to find the secret *e*(*g, g*)*p*(0). Then if our encryption includes *e*(*g, g*)*p*(0)*m*, the user would be able to find *m*. Thus, the encryptor can specify which attributes are relevant by providing *gti* for each attribute *i* in the desired transaction set.

4.3. Multiple Encryptions

First, let's consider a very simplified scheme based on the Feldman Verifiable Secret Sharing scheme. Recall that, given *d* points *p*(1), ..., *p*(*d*) on a *d* − 1 degree polynomial, we can use Lagrange interpolation to compute *p*(*i*) for any *i*. However, given only *d*−1 points, any other points are information theoretically hidden. According to the Lagrange formula, *p*(*i*) can be computed as a linear combination of *d* known points. Let *Δj*(*i*) be the coefficient of *p*(*j*) in the computation of *p*(*i*). Then *p*(*i*) = _*j*∈*S p*(*j*)*Δj*(*i*) where *S* is a set of any d known points and *Δj*(*i*) = pi*k*∈*S,j not =k*(*i* − *k*)/(*j* − *k*). Note that any set of *d* random numbers defines a valid polynomial, and given these numbers we can find any other point on that polynomial. Thus our first attempt Multi-Authority Scheme is as follows:

Preparing for the Global FP-tree Construction

Init First fix $y_1 ... y_k$, $\{t_{k,i}\}_{i=1...n,k=1...K} \leftarrow Z_q$. Let $y_0 = \sum_{k=1}^{K} y_k$. **System Public Key** $Y_0 = e(g,g)_{y_0}$.

Attribute Authority k

Authority Secret Key The SW secret key: $y_k, t_{k,1} ... t_{k,n}$.
Authority Public Key $T_{k,i}$ from the SW public key: $T_{k,1} ... T_{k,n}$ where $T_{k,i} = g^{t_{k,i}}$.
Secret Key for User u **from authority k** Choose random $d-1$ degree polynomial p with $p(0) = y_k$. Secret Key: $\{D_{k,i} = g^{p(i)/t_{k,i}}\}_{i \in A_u}$.

Encryption for attribute set A_C

Choose random $s \leftarrow Z_q$.
Encryption: $E = Y_0^s m, \left\{E_{k,i=T_{k,i}^s}\right\}_{i \in A_C^k, \forall k}$
Decryption: For each authority k, for d attributes $i \in A_C^k \cap A_u$, compute $e(E_{k,i}, D_{k,i}) = e(g,g)^{p(i)s}$. Interpolate to find $Y_k^s = e(g,g)^{p(0)s} = e(g,g)y_k^s$. Combine these values to obtain $\prod_{k=1}^{K} Y_k^s = Y_0^s$. Then $m = E/Y_0^s$.

4.4. Private FP-Tree Matching

Here, we apply the secure matching protocol proposed by Freedman et al. [16] to merge the FP-tree between every two parties. We propose a framework whereby all parties participate to a secure aggregation mechanism without having access to the protected data. In order to ensure end to end confidentiality, the framework uses additive homomorphic encryption algorithms. All the count of the conditional FP-tree is merged in this step.

Preventing Collusion. Note that we can easily extend this to prevent collusion: If we give all our users points from the same polynomial, any group with at least *d* transaction between them would be able to combine their keys to find p(0). However, if we instead give each user u a different polynomial *pu* (but still with the same zero point *pu*(0) = *p*(0)), then one user's points will give no information on the polynomial held by the other (as long as neither has more than d.1 points). To see this, note that, given any *d*−1 points on polynomial *p*1 and any *d*−1 points on polynomial *p*2, with the requirement that these polynomials must intersect at 0, it is still the case that any value for *y* = *p*1(0) = *p*2(0) will define a valid pair of polynomials. Thus, *y* is information theoretically hidden. Ideally, a public-key encryption scheme should be semantically secure against adaptive chosen-cipher text

attack. Informally, this means that an adversary can learn nothing about the plaintext corresponding to a given cipher text *c*, even when the adversary is allowed to obtain the plaintext corresponding to cipher texts of its choice. We henceforth denote this security notion as IND-CCA (indistinguishability against adaptive chosen-cipher text attacks). That is to say that our scheme's security, which is under the semi-honest model, relies on the encryption's strength.

Private Merging Protocol

Input: Party A's input is a set $T_A = \{T_A^1, ..., T_A^k\}$, party B's input is a set $T_B = \{T_B^1, ..., T_B^l\}$. The elements in the input sets are taken from a domain of size N.

1. Party performs the following:
   (a) He chooses the secret-key parameters for a semantically-secure homomorphic encryption scheme, and publishes its public keys and parameters. The plaintexts are in a field that contains representations of the N elements of the input domain, but is exponentially larger.
   (b) She uses interpolation to compute the coefficients of the polynomial $P(y) = \sum_{i=0}^{k} \alpha_i T_B^i$ of degree k with roots $x_1, ..., x_k$.
   (c) She encrypts each of the $(k+1)$ coefficients by the semantically-secure homomorphic encryption scheme and sends to party B the resulting set of ciphertexts, $Enc(\alpha_0), ..., Enc(\alpha_k)$.
2. Party B performs the following for every $T_B^i \in T_B$:
   (a) He uses the homomorphic properties to evaluate the encrypted polynomial at T_B^i. That is, he computes $Enc(P(y)) = Enc(\sum_{i=0}^{k} \alpha_i T_B^i)$.
   (b) He chooses a random value r and computes $Enc(rP(y)+y)$. (One can also encrypt some additional payload data py by computing $Enc(rP(y)+(T_B^i|p_B))$. Party obtains p_B iff T_B^i is in the intersection.) He randomly permutes this set of kS ciphertexts and sends the result back to the client .
3. Party A decrypts all l ciphertexts received. She locally outputs all values $x \in X$ for which there is a corresponding decrypted value .

5. Conclusions

The main contribution of this paper is proposing a general framework for privacy preserving association rules mining. For that the randomization methodologies are not good enough to attain the high accuracy and protect clients' information from privacy breach and the malicious attack, we show that how association rules mining can be done in this framework and prove that is secure enough to keep the clients' privacy. We also show that our protocols works with less communication complexity and communication complexity compared to other related schemes. In the future research, a common framework with more formal and reliable for privacy preservation will enable next generation data mining technology to make substantial advances in alleviating privacy concerns.

References

[1] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the 19th annual ACM symposium on Theory of computing (1987)

[2] Han, J., Pei, J., Yin, Y., Mao, R.: Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery, 53–87 (2004)

[3] Kantarcioglu, M., Clifton, C.: Privacy-Preserving Distributed Mining of association rules on horizontally partitioned data. In: Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (2002)

[4] Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000)

[5] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592. Springer, Heidelberg (1999)

[6] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

[7] Vaidya, J.S., Clifton, C.: Privacy Preserving Association Rule Mining in Vertically Partitioned Data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002)

[8] Yao, A.C.: Protocols for Secure Computation. In: 23rd FOCS (1982)

[9] Zhan, J.Z., Matwin, S., Chang, L.: Privacy-Preserving Collaborative Association Rule Mining. In: Procceding of DBSec 2005, pp. 153–165 (2005)

[10] Freedman, M., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)

[11] Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1993)

[12] Agrawal, R., Srikant, R.: Privacy-Preserving Data Mining. In: ACM SIGMOD Int'l Conf. on Management of Data, Dallas (May 2000)

[13] Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy Preserving Mining of Association Rules. In: Proc. of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD) (2002)

[14] Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proc. of FOCS, pp. 427–437 (1987)

[15] Fukazawa, T.,Wang, J., Takata, T., Miyazak, M.: An Effective Distributed Privacy- Preserving Data Mining Algorithm. In: Fifth International Conference on Intelligent Data Engineering and Automated Learning, UK (2004)

[16] Goldreich, O.: Foundations of Cryptography, vol. 2, ch.7. Cambridge Univ. Press, Cambridge (2004)