社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＣＳＥＣ－38（20）
2007／7／19

プライバシー保護した相関ルールマイニングに関する再考

蘇 春華† 周 建英†† 鮑 豊†† 高木 剛††† 櫻井 幸一†

† Department of Computer Science and Communication Engineering, Kyushu University
†† Institute for Infocomm Research (I^2R), Singapore
††† School of Systems Information Science, Future University-Hakodate
E-mail: †su@itslab.csce.kyushu-u.ac.jp, ††{jyzhou,baofeng}@i2r.a-star.edu.sg, †††takagi@fun.ac.jp ,
††††sakurai@csce.kyushu-u.ac.jp

Privacy-Preserving Association Rules Mining Scheme Revisited

Chunhua SU†, Jianying ZHOU††, Feng BAO††, Tsuyoshi TAKAGI†††, and Kouichi SAKURAI†

† 九州大学大学院システム情報科学府
†† シンガポール国立情報技術研究院
††† 公立はこだて未来大学システム情報科学部
E-mail: †su@itslab.csce.kyushu-u.ac.jp, ††{jyzhou,baofeng}@i2r.a-star.edu.sg, †††takagi@fun.ac.jp ,
††††sakurai@csce.kyushu-u.ac.jp

Abstract Assocaition Rules Mining is a frequently used technique which finds interesting associations and correlation relationships among large set of data items that occur frequently together in varieties of social and bussiness area. For the coopertional distributed assocaition rules mining, privacy-preserving techniques are strongly needed. In this paper, we employ frequent-pattern tree (FP-tree) structure storing compressed, crucial information about frequent patterns, and develop an efficient and secure FP-treebased mining method. We show that our protocol is collusion resistant, which means that even if all dishonest respondents collude with a dishonest data minerin an attempt to learn the associations between honest respondents and their responses, they will be unable to do so.

Key words association rule mining, privacy-preserving data mining, FP-tree, attributes-based encryption

1. Introduction

Nowadays, data mining is a widely used technology which has emerged as a means of identifying patterns or trends from large quantities of data. Finding association rules is one of the most frequently used data mining techniques. Association rules are used in different fields, such as: Marketing, Targeted Advertising, Floor Planning, Inventory Control, Churning Management. By using this technology, we can extract some useful information from huge amounts of data. The algorithms generally assume that necessary information from all clients is gathered at one central site. Most data mining tools operate by gathering all data into a central site, then running an algorithm on those gathered data. However, in this real world, many organization and the individual will have concern about their own privacy, they may be reluctant to share their own data to go on data mining unless there is privacy preserving data mining technology to make sure their privacy won't be violated or mis-used by other parties. For example, suppose some company want to do joint association rules mining over the participants' indivual databases for the maketing analysis. However, it will violate both companies and customers' privacy. The company don't want its commercial privacy to be known by other companies. At the same time, the customers are also reluctant to share their private information without any reasonable technique to protect their privacy.

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 transac-

tion implies (with a certain degree of confidence) the presence of other items. Association rules are statements of the form $X_1, X_2, ..., X_n \Rightarrow Y$, meaning that if we find all of $X_1, X_2, ..., X_n$ 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. We may also ask that the confidence be significantly higher than it would be if items were placed at random into baskets. In many (but not all) situations, we only care about association rules involving sets of items that appear frequently in baskets. For example, we cannot run a good marketing strategy involving items that no one buys anyway. Thus, 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 $X_1, X_2, ..., X_n \Rightarrow Y$ 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.

1.1 Association Rules Mining

The major steps in association rule mining are: frequent itemset generation and rule derivation. For mining the association rules, there is a most frequently used algorithm call A-Priori Algorithm propsoed in [1]. This algorithm proceeds levelwise, The A-Priori algorithm uses the downward closure property, to prune unnecessary branches for further consideration. It needs two parameters, *minSupp* and *minConf*. The *minSupp* is used for generating frequent itemsets and *minConf* is used for rule derivation. Here, we give a brief review of how to compute the support and confidence: Let $I = i_1, i_2, ..., i_n$ be an itemset. Let DB be set of transactions, where each transaction T is an itemset such that $T \subseteq I$. Given an itemset $X \subseteq I$, a transaction T contains X if and only if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$ where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has support s in the transaction database DB if s% of transactions in DB contain $X \cup Y$. The association rule holds in the transaction database DB with confidence c if c% of transactions in DB that contain X also contains Y. An itemset X with k items called $k-$itemset. So we difine as following:

$$support_{X \Rightarrow Y} = \frac{|T_{X \cup Y}|}{|DB|}$$

it 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:

$$confident_{X \Rightarrow Y} = \frac{support_{X \Rightarrow Y}}{support_X}$$

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. This algorithm contains a number of passes over the database. During pass k, the algorithm finds the set of frequent itemsets L_k of length k that satisfy the minimum support requirement. The algorithm terminates when L_k is empty. A pruning step eliminates any candidate, which has a smaller subset.

1.2 Privacy Problem in Distributed Association Rules Mining (ARM)

Most distributed ARM 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.

Algorithms have been proposed for distributed data mining. Cheung et al. proposed a method for horizontally parti tioned data. Algorithms that apply these approaches include those described by Agrawal and Schafer (1996), Becuzzi et al (1999), and Cheung and Xiao (1999). The "count distribution algorithm" (Agrawal and Shafer 1996) is an example of the data distribution approach. The algorithm operates as follows:

(1) Divide the dataset amongst the available processes so that each process is responsible for a particular horizontal segment.

(2) Each process determines the local support counts in its segment for the candidate 1-itemsets.

(3) Exchange information between processes so that each process obtains the global support counts for all 1-itemsets (a process which Agrawal refers to as global reduction).

(4) Each process then prunes the 1-itemsets, generates a set of candidate 2-itemsets from the supported 1-itemsets, and then determines the local support for each of these candidate sets, and so on.

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 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as $\{a_1, ..., a_{100}\}$, it must generate $2^{100} - 2 \approx 10^{30}$ 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.

1.3 Related Works

The research of privacy-preserving techniques for data mining began in Year 2000, R. Agrawal et al [?] proposed a reconstruction procedure which is possible to accurately estimate the distribution of original data values from the random noise perturbed data. It showed possible to develop accurate approximation models while respecting users' privacy concerns. However, Evfimievski et al [4] 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. A privacy breach is a situation when, for some clients, the disclosure of its randomized private information to the server reveals that a certain property of unrandomized private information holds with high probability. The method presented in Kantarcioglu et al. [10] is the first cryptography-based solutions for private distributed association rules mining, it assumes three or more parties, and they jointly do the distributed Aprori algorithm with the data encrypted. In the recent papers [6] [14] [16], some privacy-preserving association rules those paper are similar and developed a secure multi-party protocol and a secure multi-party protocol based on homomorphic encryption.

1.4 Our Contributions

- 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 [6] [14] [16].
- We apply frequent-pattern tree (FP-tree) structure to execute the association rules mining and extend it to distributed association rules mining framework.
- Our privacy-preserving protocol provide a perfect anonymity and it is collusion-resistant.

2. Preliminaries

2.1 Problem Formulation

The distributed database in our model is a horizontally partitioned database. The database schema of all the partitions are the same, i.e., their records are transactions on the same set of items. (It can be modified for the case in which the schema at different sites are not completely identical.) Many distributed databases are horizontally partitioned. For example, a retail chain may have several regional data centers, each manages the transaction records in its own region. It is important to mine the association rules based on data from all the centers. Distributed mining can be applied to many applications which have their data sources located at dierent 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 clustering 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.

2.2 Cryptographic Premitives

Homomorphic Encryption with Scalaring We review multiplicative and additive homomorphic encryption schemes with the property of scalaring. Multiplicative homomorphic encryption schemes are usually more efficient than additive homomorphic encryption schemes, An encryption scheme is multiplicative homomorphic if and only if $E(m_1) \cdot E(m_2) = E(m_1 \times m_2)$, where is an operator. An encryption scheme is additive homomorphic if and only if $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$. An encryption is scalarable if $c = E(m)$ can be mapped randomly to a ciphertext $E(m_k)$ or $c' = E(km)$ for a random k. The ElGamal encryption scheme is a multiplicative homomorphic encryption scheme with the scalaring property.

Attributes-based Encryption (ABE) The ABE scheme is developed from Identity based encryption(IBE) which introduced by Shamir [13], 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 sep- arate 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 sid secure.

3. Assoication Mining With 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.

Let $I = \{a_1, a_2, ..., a_m\}$ be a set of items, and a transaction database $DB = \langle T_1, T_2, ..., T_n \rangle$, where $T_i (i \in [1...n])$ 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 $MinSupp$. 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, $\langle (f:4), (c:4), (a:3), (b:3), (m:3), (p:3) \rangle$ (the number after ":" indicates the support), in which items are ordered in frequency descending order.

Second, the root of a tree is created and labeled 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: $\langle (f:1), (c:1), (a:1), (m:1), (p:1) \rangle$. 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 $\langle f, c, a, b, m \rangle$ shares a common prefix $\langle f, c, a \rangle$ with the existing path $\langle f, c, a, m, p \rangle$, 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 $\langle f, b \rangle$ shares only the node $\langle f \rangle$ 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, $\langle (c:1), (b:1), (p:1) \rangle$.

(5) For the last transaction, since its frequent item list $\langle c, a, m, p \rangle$ 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 subtrees as the children of the root, and a frequent-item-header table.

(2) Each node in the item-prefix subtree 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) item-name 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 a_i by traversing the FP-tree once following a_i'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-tree, 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.

4. Private Multi-party Protocol for Association Rules Mining

ur 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 [15] and extended by Goldreich, Micali, and Wigderson [8]. Secure multi-party computation (MPC) protocols allow a set of n players to securely compute any agreed func-

tion 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) Every party executes FP-tree construction and find out conditional pattern-bases and the conditional FP-trees.

(2) Merge the conditional FP-trees using Attribute-based encryption scheme.

(3) Support count among the common k-item sets privately using the scalar homomorphic encryption scheme.

(4) Secure global support count computation.

(5) Output the final result of association rules

A Multipaprty 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 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. Attribute Key Generation : A randomized algorithm run by an attribute authority. Takes as input the authority's secret key, the authority's value d_k, a user' s GID, and a set of attributes in the authority ' s domain A_C^k. (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. Central Key Generation : A randomized algorithm run by the central au- thority. Takes as input the master secret key and a user ' s GID and outputs secret key for the user. Encryption : A randomized algorithm run by a sender. Takes as input a set of attributes for each authority, a message, and the system public key. Outputs the ciphertext. Decryption : A deterministic algorithm run 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 $|A_C^k \cap A_u^k| > d_k$ for all authorities k.

The security is based on Bilinear Diffie-Hellman(BDH) Assumption, as defined as following:

[Definition 1] **(Bilinear Diffie-Hellman(BDH) Assumption)**. Let G be a group of prime order q and generator g where $|q|$ is proportional to the security parameter k. There exists a negligible function *neg* such that for all adversaries A, given G, q, g, g^a, g^b, g^c and bilinear map e for randomly chosen $a, b, c \leftarrow Z_q$, A can distinguish $e(g,g)^{abc}$ from $e(g,g)^R$ for random $R \leftarrow Z_q$ with probability at most $neg(k)$

5. 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 men- tioned above, we also have one trusted central authority who does not monitor any attributes.

5.1 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 $\Delta_j(i)$ be the coefficient of $p(j)$ in the computation of $p(i)$. Then $p(i) = \sum_{j \in S} p(j)\Delta_j(i)$ where S is a set of any d known points and $\Delta_j(i) = \prod_{k \in S, j \neq k}(i-k)/(j-k)$. 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.

5.2 Specifying Attributes

If we take this approach, any user with any d attributes 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 $e, g \in G_1$, and $a, b \in Zq$, $e(g^a, g^b) = e(g,g)^{ab}$. Now, suppose instead of giving each user $g^{p(i)}$ for each attribute i, we choose a random value t_i and give $g^{p(i)/t_i}$. 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 g^{t_i} for each attribute i in the desired transaction set.

5.3 Multiple Encryptions

Thus our first attempt Multi Authority Scheme is as follows:

System

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: $\left\{D_{k,i} = g^{p(i)/t_{k,i}}\right\}_{i \in A_u}$.

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 $m = E/Y_0^s$.

a) 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 attributes between them would be able to combine their keys to find p(0). However, if we instead give each user u a different polynomial p_u (but still with the same zero point $p_u(0) = p(0)$), 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. Then our first scheme runs a follows:

5.4 FP-Tree Aggregation Computing

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 are merged in this step.

6. Conclusions and Future Works

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, we hope to improve the efficiency of this secure multi-party computing approach more and develop a family of problems and solutions to privacy-preserving data mining. 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.

文　献

[1] R. Agrawal, T. Imielinski, and A. N. Swami. *"Mining association rules between sets of items in large databases"*. Proceedings of the ACM SIGMOD International Conference on Management of Data, 1993

[2] R. Agrawal, R. Srikant. *Privacy-Preserving Data Mining.* ACM SIGMOD Int'l Conf. on Management of Data, Dallas, May 2000.

[3] M. Chase, *Multi-Authority Attribute Based Encryption*, Fourth IACR Theory of Cryptography Conference(TCC 2007), Holland, 2007.

[4] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, Johannes Gehrke *Privacy Preserving Mining of Association Rules.* Proc. of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD), 2002

[5] P. Feldman. *A practical scheme for non-interactive verifiable secret sharing.* In Proc. of FOCS pp.427-437 1987.

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

[7] Oded Goldreich. *Foundations of Cryptography Volume 2, Chapt.7*, Cambridge Univ. Press, 2004.

[8] O. Goldreich, S. Micali and A. Wigderson. emphHow to play any mental game. In Proceedings of the 19th annual ACM symposium on Theory of computing, 1987

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

[10] M. Kantarcioglu and C. Clifton. *"Privacy-Preserving Distributed Mining of association rules on horizontally partitioned data"*. Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. 2002.

[11] Y. Lindell and B. Pinkas. *Privacy preserving data mining.* In Advances in Cryptology CRYPTO '00, volume 1880 of Lecture Notes in Computer Science, pp. 36-54. Springer-Verlag, 2000.

[12] Pascal Paillier. *Public-key cryptosystems based on composite degree residuosity classes.* In EUROCRYPT, Prague, Czech Republic,1999.

[13] Adi Shamir. Identity-based cryptosystems and signature schemes. In Proc. of CRYPTO 1984, volume 196, Springer LNCS, pp. 47-53, 1984.

[14] J.S. Vaidya and C. Clifton. *"Privacy Preserving Association Rule Mining in Vertically Partitioned Data"*. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2002.

[15] A.C Yao, *Protocols for Secure Computation*, In 23rd FOCS, 1982

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