29

Privacy Preserving Data Mining

Xinjing Ge and Jianming Zhu
School of Information, Central University of Finance and Economics
Beijing, China

1. Introduction

With the development of network, data collection and storage technology, the use and sharing of large amounts of data has become possible. Once the data and information accumulated , it will become the wealth of information. Data mining, otherwise known as *knowledge discovery*, can extracted "meaningful information" or "knowledge" from the large amounts of data, so supports people's decision-making (Han & Kamber, 2006). However, traditional data mining techniques and algorithms directedly opreated on the original data set, which will cause the leakage of privacy data. At the same time, large amounts of data implicates the sensitive knowledge that their disclosure can not be ingored to the competitiveness of enterprise. These problems challenge the traditional data mining, so privacy-preserving data mining (PPDM) has become one of the newest trends in privacy and security and data mining research.

In privacy-preserving data mining (PPDM), data mining algorithms are analyzed for the side-effects they incur in data privacy, and the main objective in privacy preserving data mining is to develop algorithms for modifying the original data in some way, so that the private data and private knowledge remain private even after the mining process (Verykios et al., 2004a). A number of techniques such as Trust Third Party, Data perturbation technique, Secure Multiparty Computation and game theoretic approach, have been suggested in recent years in order to perform privacy preserving data mining.

However, most of these privacy preserving data mining algorithms such as the Secure Multiparty Computation technique, were based on the assumption of a semi-honest environment, where the participating parties always follow the protocol and never try to collude. As mentioned in previous works on privacy-preserving distributed mining (Lindell & Pinkas, 2002), it is rational for distributed data mining that the participants are assumed to be semi-honest, but the collusion of parties for gain additional benefits can not be avoided. So there has been a tendency for privacy preserving data mining to devise the collusion resistant protocols or algorithms, recent research have addressed this issue, and protocols or algorithms based on penalty function mechanism, the Secret Sharing Technique, and the Homomorphic Threshold Cryptography are given (Kargupta et al.,2007 ; Jiang et al., 2008; Emekci et al., 2007).

This chapter is organized as follows. In Section 2, we introduce the related concepts of the PPDM problem. In Section 3, we describe some techniques for privacy preserving data mining In Section 4, we discuss the collussion behaviors in Privacy Preserving Data Mining. Finally, Section 5 presents our conclusions.

2. The related concepts of PPDM

The concept of privacy is often more complex, In particular, in data mining, the definition of privacy preservation is refered to "getting valid data mining results without learning the underlying data values."(Clifton et al., 2002a), and (Stanley et al., 2004) also indicated PPDM encompasses the dual goal of meeting privacy requirements and providing valid data mining results, so the definition emphasizes the dilemma of balancing privacy preservation and knowledge disclosure.

2.1 Defining privacy preservation in data mining

Privacy-preserving data mining considers the problem of running data mining algorithms on confidential data that is not supposed to be revealed even to the party running the algorithm. The main consideration of PPDM is two fold (Verykios et al., 2004a). First, sensitive raw data like identifiers,names, addresses and so on, should be modified or trimmed out from the original database, in order for the recipient of the data not to be able to compromise another person's privacy. Second, sensitive knowledge which can be mined from a database by using data mining algorithms, should also be excluded, because such a knowledge can equally well compromise data privacy. So, privacy preservation occurs in two major dimensions: users' personal information and information concerning their collective activity. the former is referred to individual privacy preservation and the latter is refferred to collective privacy preservation (Stanley et al., 2004).

- **Individual privacy preservation:** The primary goal of data privacy is the protection of personally identifiable information. In general, information is considered personally identifiable if it can be linked, directly or indirectly, to an individual person. Thus, when personal data are subjected to mining, the attribute values associated with individuals are private and must be protected from disclosure. Miners are then able to learn from global models rather than from the characteristics of a particular individual.
- **Collective privacy preservation:** Protecting personal data may not be enough. Sometimes, we may need to protect against learning sensitive knowledge representing the activities of a group. We refer to the protection of sensitive knowledge as collective privacy preservation. The goal here is quite similar to that one for statistical databases, in which security control mechanisms provide aggregate information about groups and, at the same time, should prevent disclosure of confidential information about individuals. However, unlike as is the case for statistical databases, another objective of collective privacy preservation is to preserve strategic pattern that are paramount for strategic decisions, rather than minimizing the distortion of all statistics. In other words, the goal here is not only to protect personally identifiable information but also some patterns and trends that are not supposed to be discovered.

Privacy Preservation in Data Mining has some limitations: Privacy Preservation Data Mining techniques do not mean pefect privacy, for example, The SMC computation won't reveal the sensitive data, but the data mining result will enable all parties to estimate the value of the sensitive data. It isn't that the SMC was "broken", but that the result itself violates privacy.

2.2 Data distribution

In PPDM, How are the data available for mining: are they centralized or distributed across many sites? With distributed data, the way the data is distributed also plays an important role in defining the problem. The different partitioning poses different problems and can lead to different algorithms for privacy-preserving data mining.

Distributed data scenarios can be classified as horizontal data distribution and vertical data distribution (Verykios et al., 2004a). **Horizontal distribution** refers to these cases where different database records reside in different places, while **vertical data distribution**, refers to the cases where all the values for different attributes reside in different places.

2.3 Models of PPDM

In the study of privacy-preserving data mining (PPDM), there are mainly four models as follows:

1. Trust Third Party Model

The goal standard for security is the assumption that we have a trusted third party to whom we can give all data. The third party performs the computation and delivers only the results – except for the third party, it is clear that nobody learns anything not inferable from its own input and the results. The goal of secure protocols is to reach this same level of privacy preservation, without the problem of finding a third party that everyone trusts.

2. Semi-honest Model

In the semi-honest model, every party follows the rules of the protocol using its correct input, but after the protocol is free to use whatever it sees during execution of the protocol to compromise security.

3. Malicious Model

In the malicious model, no restrictions are placed on any of the participants. Thus any party is completely free to indulge in whatever actions it pleases. In general, it is quite difficult to develop efficient protocols that are still valid under the malicious model. However, the semi-honest model does not provide sufficient protection for many applications.

4. Other Models - Incentive Compatibility

While the semi-honest and malicious models have been well researched in the cryptographic community, other models outside the purview of cryptography are possible. One example is the interesting economic notion of incentive compatibility. A protocol is incentive compatible if it can be shown that a cheating party is either caught or else suffers an economic loss. Under the rational model of economics, this would serve to ensure that parties do not have any advantage by cheating. Of course, in an irrational model, this would not work.

We remark, in the "real world", there is no external party that can be trusted by all parties, so the Trust Third Party Model is a ideal model.

2.4 Evaluation of privacy preserving algorithms

An important aspect in the development and assessment of algorithms and tools, for privacy preserving data mining is the identification of suitable evaluation criteria and the development of related benchmarks. It is often the case that no privacy preserving algorithm exists that outperforms all the others on all possible criteria. Rather, an algorithm may perform better that another one on specific criteria, such as performance and/or data utility. It is thus important to provide users with a set of metrics which will enable them to select the most appropriate privacy preserving technique for the data at hand, with respect to some specific parameters they are interested in optimizing.

A preliminary list of evaluation parameters to be used for assessing the quality of privacy preserving data mining algorithms, is given below: (Verykios et al., 2004a)

- the ***performance*** of theney, proposed algorithms in terms of time requirements, that is the time needed by each algorithm to hide a specified set of sensitive information;
- the ***data utility*** after the application of the privacy preserving technique, which is equivalent with the minimization of the information loss or else the loss in the functionality of the data;
- the ***level of uncertainty*** with which the sensitive information that have been hidden can still be predicted;
- the ***resistance*** accomplished by the privacy algorithms, to different data mining techniques.

3. Privacy preserving data mining techniques

Data mining includes various algorithms such as classification, association rule mining, and clustering. In recent years, a number of techniques have been proposed to preserve privacy. Privacy preserving data mining techniques can be classified by different assumptions or domain knowledge.for example, data distribution, data modification, data mining algorithm, data or rule hiding, privacy preservation (Verykios et al., 2004a). In this section, we will introduce some mainly privacy preserving data mining techniques, which include the Trust Party tehnique, randomization technique, Secure Multiparty Computation and anonymity techniques.

3.1 The trust party tehnique

The typical approach to data mining of distributed privacy preserving data is to build a data warehouse containing all the data, then mine the warehouse. This requires that the warehouse be trusted to maintain the privacy of all parties - since it knows the source of data, it learns site specific information as well as global results. For privacy preserving data mining, if we can find a fully trusted third party, then all parties give their input to the trust third party, and the trust third party computes the output and returns it to the parties. For example, current e-commerce transactions have a trusted (central) third party with access to all the information. The "trust" is governed by legal contracts enjoining the improper release of information. In some cases, the third party is dispensed with and contracts exist between the interested parties themselves. This is obviously insecure from the technical perspective.

Trusted third parties are, however, difficult to find, especially when the number of participants increases. Though SMC enables this *without* the trusted third party, but the computation and/or communication required may be high. so the protocols obtained by this general construction are inefficient and useless in the case of a large number of participants. Other factors, such as the need for continual online availability of the parties, create further restrictions and problems in realworld settings such as a web-based survey. So we need to extend the fully trusted party technique. For example, refernce (Gilburd et al., 2004) proposed a new privacy model: *k*-privacy --by means of an innovative, yet natural generalization of the accepted trusted third party model. This allows implementing cryptographically secure efficient primitives for real-world large scale distributed systems. As an example for the usefulness of the proposed model, we employ *k*-privacy to introduce a technique for obtaining knowledge --by way of an association-rule mining algorithm from large-scale data basement, while ensuring that the privacy is cryptographically secure.

3.2 Data perturbation technique

Data perturbation technique, first proposed in (Agrawal & Srikant, 2000), represents one common approach in privacy preserving data mining, where the original (private) dataset is perturbed and the result is released for data analysis. Data perturbation includes a wide variety of techniques including (but not limited to): additive, multiplicative (Kim & Winkler, 2003), matrix multiplicative, k-anonymization (Sweeney, 2002), micro-aggregation (Li & Sarkar, 2006), categorical data perturbation (Verykios, 2004b), data swapping (Fienberg & McIntyre, 2004), resampling (Liew, 1985), data shuffling (Muralidhar & Sarathy, 2006). Now we mostly focus on two types of data perturbation that apply to continuous data: additive and matrix multiplicative, other detailed data perturbation techniques can refer to the related literatures.

3.2.1 Additive perturbation

The additive perturbation is a technique for privacy-preserving data mining in which noise is added to the data in order to mask the attribute values of records (Agrawal & Srikant, 2000). The noise added is sufficiently large so that individual record values cannot be recovered. Therefore, techniques are designed to derive aggregate distributions from the perturbed records. Subsequently, data mining techniques can be developed in order to work with these aggregate distributions. The method of randomization can be described as follows.

Consider a set of data records denoted by $X = \{x_1, x_2, \cdots, x_N\}$. For record $x_i \in X$, we add a noise component which is drawn fromthe probability distribution $f_Y(y)$. These noise components are drawn independently, and are denoted $Y = \{y_1, y_2, \cdots y_N\}$. Thus, the new set of distorted records are denoted by $x_1 + y_1, \cdots x_N + y_N$. We denote this new set of records by $Z = \{z_1, \cdots z_N\}$. So the data owner replaces the original dataset X with $Z = X + Y$. where Y is a noise matrix with each column generated independently from a n-dimensional random vector Y with mean vector zero. As is commonly done, we assume throughout that $\sum_Y$ equals $\sigma^2 I$, *i.e.*, the entries of Y were generated independently from some distribution with mean zero and variance σ^2 (typical choices for this distribution include Gaussian and uniform). In this case, Y is sometimes referred to as *additive white noise*. Thus, the original records cannot be recovered, but the distribution of the original records can be recovered.

3.2.2 Matrix multiplicative perturbation

The most common method of data perturbation is that of additive perturbations. However, matrix multiplicative perturbations can also be used to good effect for privacy-preserving data mining.

The data owner replaces the original data X with $Y = MX$ where M is an $n' \times n$ matrix chosen to have certain useful properties. If M is orthogonal ($n' = n$ n and $M^T M = I$), then the perturbation exactly preserves Euclidean distances, *i.e.*, for any columns x_1, x_2 in X ,their corresponding columns y_1, y_2 in Y satisfy $\|x_1 - x_2\| = \|y_1 - y_2\|$. If each entry of M is generated independently from the same distribution with mean zero and variance σ^2 (n' not necessarily equal to n), then the perturbation approximately preserves Euclidean distances on expectation up to constant factor $\sigma^2 n'$. If M is the product of a discrete cosine transformation matrix and a truncated perturbation matrix, then the perturbation approximately preserves Euclidean distances.

3.2.3 Evaluation of data perturbation technique

The data perturbation technique have the benefits of efficiency, and does not require knowledge of the distribution of other records in the data. This is not true of other methods such as k-anonymity which require the knowledge of other records in the data. this technique does not require the use of a trusted server containing all the original records in order to perform the anonymization process. While this is a strength of the data perturbation technique, it also leads to some weaknesses, since it treats all records equally irrespective of their local density. Therefore, outlier records are more susceptible to adversarial attacks as compared to records in more dense regions in the data. In order to guard against this, one may need to be needlessly more aggressive in adding noise to all the records in the data. This reduces the utility of the data for mining purposes.

Reference (Liu et al., 2006) provides a detailed survey of attack techniques on the data perturbation, especially additive and matrix multiplicative perturbation. These attacks offer insights into vulnerabilities data perturbation techniques under certain circumstances. In summary, the following information could lead to disclosure of private information from the perturbed data.

1. Attribute Correlation: Many real world data has strong correlated attributes, and this correlation can be used to filter off additive white noise.

2. Known Sample: Sometimes, the attacker has certain background knowledge about the data such as the *p.d.f.* or a collection of independent samples which may or may not overlap with the original data.

3. Known Inputs/Outputs: Sometimes, the attacker knows a small set of private data and their perturbed counterparts. This correspondence can help the attacker to estimate other private data.

4. Data Mining Results: The underlying pattern discovered by data mining also provides a certain level of knowledgewhich can be used to guess the private data to a higher level of accuracy.

5. SampleDependency: Most of the attacks assume the data as independent samples from some unknown distribution. This assumption may not hold true for all real applications. For certain types of data, such as the time series data, there exists auto correlation/dependency among the samples. How this dependency can help the attacker to estimate the original data is still an open problem.

At the same time, a "privacy/accuracy" trade-off is faced for the data perturbation technique. On the one hand, perturbation must not allow the original data records to be adequately recovered. On the other, it must allow "patterns" in the original data to be mined.

Data perturbation technique is needed for situations where accessing the original form of the data attributes is mandatory. It happens when, for instance, some conventional off-the-shelf data analysis techniques are to be applied. While this approach is more generic, some inaccuracy in the analysis result is to be expected.

3.2.4 Application of the data perturbation technique

The randomization method has been extended to a variety of data mining problems. Refercence (Agrawal & Srikant, 2000) firstly discussed how to use the approach for solving the privacy preserving classification problem classification. Refercence (Zhang et al. 2005; Zhu & Liu, 2004) have also proposed a number of other techniques which seem to work well over a variety of different classifiers.

There has been research considering preserving privacy for other type of data mining. For instance, reference (Evfimievski et al., 2002) proposed a solution to the privacy preserving distributed association mining problem.The problem of association rules is especially challenging because of the discrete nature of the attributes corresponding to presence or absence of items. In order to deal with this issue, the randomization technique needs to be modified slightly. Instead of adding quantitative noise, random items are dropped or included with a certain probability. The perturbed transactions are then used for aggregate association rule mining. The randomization approach has also been extended to other applications, for example, SVD based collaborative filtering (Polat & Du, 2005).

3.3 Secure multiparty computation technique

3.3.1 Background

In privacy preserving distributed data mining, two or more parties owning confidential databases wish to run a data mining algorithm on the union of their databases without revealing any unnecessary information. For example, consider separate medical institutions that wish to conduct a joint research while preserving the privacy of their patients. One way to view this is to imagine a trusted third party-- everyone gives their input to the trusted party, who performs the computation and sends the results to the participants. However, this is exactly what we don't want to do, for example, hospitals are not allowed to hand their raw data out, security agencies cannot afford the risk, and governments risk citizen outcry if they do. Thus, the question is how to compute the results without having a trusted party, and in a way that reveals nothing but the final results of the data mining computation. Secure Multiparty Computation enables this *without* the trusted third party. The concept of Secure Multiparty Computation was introduced in (Yao, 1986) and has been proved that there is a secure multi-party computation solution for any polynomial function (Goldreich,1998). The basic idea of Secure Multiparty Computation is that a computation is secure if at the end of the computation, no party knows anything except its own input and the results. This approach was first introduced to the data mining community by Lindell and Pinkas (Lindell & Pinkas, 2002), with a method that enabled two parties to build a decision tree without either party learning anything about the other party's data, except what might be revealed through the final decision tree. Now this technique have been developed for association rules, clustering, k-nearest neighbor classification, and are working on others.

Allowed adversarial behavior: there are two main types of adversaries. (Lindell & Pinkas, 2002)

a. **Semi-honest adversaries:** In semi-honest adversarial model, it correctly follows the protocol specification, yet attempts to learn additional information by analyzing the transcript of messages received during the execution. This is a ratherweak adversarial model. However, there are some settings where it can realistically modelthe threats to the system. Semi-honest adversaries are also called "honest-but-curious"and "passive".

b. **Malicious adversaries:** In malicious adversarial model , a party may arbitrarily deviate from the protocol specification .In general, providing security in the presence of malicious adversaries is preferred, as it ensures that no adversarial attack can succeed. Malicious adversaries are also called "active".

We remark that although the semi-honest adversarial model is far weaker than the malicious model, it is often a realistic one. This is because deviating from a specified program which may be buried in a complex application is a non-trivial task.

3.3.2 Techniques for building secure multiparty computation protocols

In this section,we describe here some simple protocols that are often used as basic building blocks, or primitives, of secure computation protocols.

Oblivious Transfer: Oblivious transfer is a simple functionality involving two parties. It is a basic building block of many cryptographic protocols for secure computation. The notion of 1-out-2 oblivious transfer was suggested by (Even et al., 1985) (as a variant of a different but equivalent type of oblivious transfer that has been suggested by (Rabin, 1981)). The protocol involves two parties, the sender and the receiver. and its functionality is defined as follows:

- **Input:** The sender's input is a pair of strings (x_0, x_1) and the receiver's input is a bit $\sigma \in \{0,1\}$.
- **Output:** The receiver's output is x_σ (and nothing else), while the sender has no output.

In other words, 1-out-of-2 oblivious transfer implements the function $((x_0, x_1), \sigma) \mapsto (\lambda, x_\sigma)$, where λ denotes the empty string (i.e., no output).

Oblivious transfer protocols have been designed based on virtually all known assumptions which are used to construct specific trapdoor functions (i.e. public key cryptosystems), and also based on generic assumptions such as the existence of enhanced trapdoor permutations. There are simple and eficient protocols for oblivious transfer which are secure only against semi-honest adversaries (Even et al., 1985).

Oblivious Polynomial Evaluation: The problem of "oblivious polynomial evaluation" (OPE) involves a sender and a receiver. The sender's input is a polynomial Q of degree k over some finite field F, namely a polynomial $Q(z) = \sum_{i=0}^{k} a_i z^i$ (the degree k of the polynomial, wis public). The receiver's input is an element . The protocol is such that the receiver obtains $Q(z)$ without learning anything else about the polynomial Q, and the sender learns nothing. That is, the problem considered is the private computation of the function $(Q, z) \mapsto (\lambda, Q(z))$ .where λ is the empty output.

The major motivation for oblivious polynomial evaluation is the fact that the output of a k egree random polynomial is $k+1$ wise independent; this is very useful in the construction of cryptographic protocols. Another motivation is that polynomials can be used for approximating functions that are defined over the Real numbers.

Homomorphic Encryption:A homomorphic encryption scheme is an encryption scheme which allows certain algebraic operations to be carried out on the encrypted plaintext, by applying an eficient operation to the corresponding ciphertext. In particular, we will be interested in additively homomorphic encryption schemes (Paillier ,1999) that is comparable with the encryption process of RSA in terms of the computation cost, while the decryption process of the additive homomorphism is faster than the decryption process of RSA.

An additively homomorphic cryptosystem has the nice property that for two plain text message m_1 and m_2, it holds $e(m_1) \times e(m_2) = e(m_1 + m_2)$,where $\times$ denotes multiplication. This essentially means that we can have the sum of two numbers without knowing what those numbers are. Moreover, because of the property of associativity, $e(m_1 + m_2 + \cdots + m_s) = e(m_1) \times e(m_2) \times \cdots \times e(m_s)$, where $e(m_i) \neq 0$.

And we can easily have the following corollary: $e(m_1)^{m_2} = e(m_2)^{m_1} = e(m_1 \times m_2)$

An eficient implementation of an additive homomorphic encryption scheme with semantic security was given by Paillier (Paillier ,1999).

Threshold decryption: Threshold decryption is an example of a multiparty functionality. The setting includes m parties and an encryption scheme. It is required that any $m' < m$ of the parties are able to decrypt messages, while any coalition of strictly less than m' parties learns nothing about encrypted messages. This functionality can, of course, be implemented using generic constructions, but there are specific constructions implementing it for almost any encryption scheme, and these are far more eficient than applying the generic constructions to compute this function ality. Interestingly, threshold decryption of homomorphic encryption can be used as a primitive for constructing a very eficient generic protocol for secure multiparty computation, with a communication overhead of only $O(mk|c|)$ bits (Franklin & Haber (1996) for a construction secure against semi-honest adversaries, and Cramer et al. (2001) for a construction secure against malicious adversaries).

Other Cryptographic Tools:

Many basic security operations now have been applied to Secure protocols of privacy preserving data mining, such as Secure Sum, Secure Set, Secure Size of Set Intersection Union, Scalar Product (Clifton et al., 2002b)..

3.3.3 Application of the secure multiparty computation technique

Secure Multi-party Computation (SMC) technique is a common approach for distributed privacy preserving data mining, and now has been extended to a variety of data mining problems. For example, Lindell & Pinkas (2002) introduced a secure multi-party computation technique for classification using the ID3 algorithm, over horizontally partitioned data. Specifically, they consider a scenario in which two parties owning confidential databases wish to run a data mining algorithm on the union of their databases, without revealing any unnecessary information. Du & Zhan (2002) proposed a protocol for making the ID3 algorithm privacy-preserving over vertically partitioned data. Vaidya & Clifton (2002) presented the component scalar product protocol for privacy-preserving association rule mining over vertically partitioned data in the case of two parties; Wright & Yang (2004) applied homomorphic encryption to the Bayesian networks induction for the case of *two* parties. Zhan et al., (2007) proposed a cryptographic approach to tackle collaborative association rule mining among multiple parties.

3.3.4 Common errors of the secure multiparty computation

There are common errors which often occur when designing secure protocols, here we would like to use this section to introduce some of these errors briefly, interested reader can refer to (Lindell & Pinkas, 2009).

- **Semi-honest Behavior does not Preclude Collusions:** Assuming that adversaries are semi-honest does not ensure that no two parties collude. The "semi-honest adversary" assumption merely ensures that an adversary follows the protocol, and only tries to learn information from messages it received during protocol execution. It is still possible, however, that the adversary controls more than a single party and might use the information it learns from all the parties it controls.
- **Deterministic Encryption Reveals Information:** A common misconception is that encrypting data, or hashing it, using any encryption system or hash function, keeps the data private. The root of the problem is the use of a deterministic function (be it a hash function or a deterministic encrypting scheme such as textbook RSA). One should therefore never apply a deterministic function to an item and publish the result.

Instead, a semantically secure encryption scheme must be used. Unfortunately, this rules out a number of "simple and eficient" protocols that appear in the literature (indeed, these protocols are not and cannot be proven secure).

- **Input Dependent Flow:** the flow of the protocol (namely, the decision which parts of it to execute), must not depend on the private input of the parties. Otherwise, The protocolis not secure
- **Security Proofs:** It is tempting to prove security by stating what constitutes a "bad behavior" or an "illegitimate gain" by the adversary, and then proving that this behavior is impossible. Any other behavior or gain is considered benign and one need not bother with it. This approach is often easier than the use of simulation based proofs. However , it is hard to predict what type of corrupt behavior an adversary might take and thus dangerous to disregard any other behavior that we have not thought of as useless for the adversary. Indeed, real world attackers often act in ways which were not predicted by the designers of the system they attack. It is also hard to define what constitutes a legitimate gain by the adversary, and allow it while preventing illegitimate or harmful gains. The notion of "harmful" might depend on a specific application or a specific scenario, and even then it might be very hard to define. So the protocol designers must prove security according to the simulation based proof (Lindell & Pinkas, 2009), which prevent any attack which is not possible in an idealized scenario.

3.3.5 Evaluation of the secure multiparty computation technique

Secure Multiparty Computation enables distributed privacy preserving data mining *without* the trusted third party, Moreover, the secure multiparty computation technique make the result of data mining correct without information loss. The shortcoming of the technique is the computation and communication overhead of protocol is very high, espericaly for the large database, which hinder its application in practice. So secure multiparty computation, due to its high computational requirement, is most suitable for situations where the number of distributed sources is relatively small and the global analysis to be supported can be derived by the given set of primitives.

3.4 A game theoretic approach to privacy preserving data mining

Game theory has been widely applied in many different domains like economics, finance, etc. Recently, it has also been applied for managing distributed computing environment. Applications of game theory in secure multi-party computation and privacy preserving distributed data mining is relatively new. Kleinberg et al. (1998) proposed a microeconomic view of data mining and illustrated how data clustering for customer segmentation in a market with two players could be modeled as a two-player game so that the segmentation was driven by the objective of deriving best marketing strategies. Kleinberg et al., (2001) tried to justify the fairness of disclosing private information as part of a transaction by the compensation of gaining better services using a game theoretic approach, for applications like marketing survey and collaborative filtering. In addition, there are some recent studies based on game theory to address the collusion problem of privacy preserving data mining that use secure multiparty computation (Abraham et al., 2006; Kargupta et al., 2007). Kargupta et al.(2007) offers a game-theoretic framework for the PPDM problem as a multi-party game where each party tries to maximize its own objectives, and also presents equilibrium-analysis of such PPDM-games and outlines a game-theoretic solution based on the concept of "cheap-talk" borrowed from the economics and the game theory literature.

4. The collussion behaviors in privacy preserving data mining

Based on cryptographic techniques and secure multi-party computations, privacy preserving protocols or algorithms have been designed for Privacy preserving data mining . However, many of these algorithms make strong assumptions about the behavior of the participating entities, such as, they assume that the parties are semi-honest, that is, they always follow the protocol and never not try to collude or sabotage the process.

As mentioned in previous works on privacy-preserving distributed mining (Lindell & Pinkas, 2002), the participants are assumed to be semi-honest that is rational for distributed data mining, but these kind of assumptions fall apart in real life and the collusion of parties happen easily to gain additional benefits. For example (Kargupta et al., 2007), the US Department of Homeland Security funded PURSUIT project involves privacy preserving distributed data integration and analysis of network traffic data from different organizations. However, network traffic is usually privacy sensitive and no organization would be willing to share their network traffic with a third party. PPDM offers one possible solution which would allow comparing and matching multi-party network traffic for detecting common attacks, stealth attacks and computing various statistics for a group of organizations without necessarily sharing the raw data. However, participating organization in a consortium like PURSUIT may not all be ideal. Some may decide to behave like a "leach" exploiting the benefit of the system without contributing much. Some may intentionally try to sabotage the multi-party computation. Some may try to collude with other parties for exposing the private data of a party.

4.1 The collussion analysis of PPDM based on the game theory

Applications of game theory in secure multi-party computation and privacy preserving distributed data mining is relatively new (Abraham et al., 2006 ; Kargupta et al., 2007). Kargupta et al. (2007) argues that large-scale multi-party PPDM can be thought of as a game where each participant tries to maximize its benefit by optimally choosing the strategies during the entire PPDM process. With a game theoretic framework for analyzing the rational behavior of each party, authors present detailed equilibrium analysis of the well known secure sum computation (Clifton et al., 2002b) as an example. A new version of the secure sum is proposed as follows and interested readers can find a detailed anysis in Kargupta et al. (2007).

Secure Sum Computation (Clifton et al., 2002b)**:** Suppose there are n individual nodes organized in a ring topology, each with a value v_j , $j = 1,2,\cdots,n$ .It is known that the sum $v = \sum_{j=1}^{n} v_j$ (to be computed) takes an integer value in the range $[0, \quad N-1]$.

The basic idea of secure sum is as follows. Assuming nodes do not collude, node 1 generates a random number R uniformly distributed in the range $[0, \quad N-1]$, which is independent of its local value v_1 . Then node 1 adds R to its local value v_1 and transmits $(R+v_1) \bmod N$ to node 2. In general, for $i = 2,\cdots n$, node i performs the following operation: receive a value z_{i-1} from previous node $i-1$, add it to its own local value v_i and compute its modulus N. In other words, $z_i = (z_{i-1} + v_i) \bmod N = (R + \sum_{j=1}^{i} v_j) \bmod N$, where z_i is the perturbed version of local value v_i to be sent to the next node i + 1. Node n performs the same step and sends the result z_n to node 1. Then node 1, which knows *R*, can subtract *R* from z_n to obtain the actual sum. This sum is further broadcasted to all other sites.

Collusion Analysis (Kargupta et al., 2007)**:** **i**t can be shown that any z_i has an uniform distribution over the interval $[0,\ \ N-1]$ due to the modulus operation. Further, any z_i and v_i are statistically independent, and hence, a single malicious node may not be able to launch a successful privacy-breaching attack. Then how about collusion?

Assume that there are $k(k \geq 2)$ nodes acting together secretly to achieve a fraudulent purpose. Let v_i be an honest node who is worried about her privacy. We also use v_i to denote the value in that node. Let v_{i-1} be the immediate predecessor of v_i and v_{i+1} be the immediate successor of v_i. The possible collusion that can arise are:

- If $k = n-1$, then the exact value of v_i will be disclosed.
- If $k \geq 2$ and the colluding nodes include both v_{i-1} and v_{i+1}, then the exact value of v_i will be disclosed.
- If $n-1 > k \geq 2$ and the colluding nodes contain neither v_{i-1} nor v_{i+1}, or only one of them, then v_i is disguised by $n-k-1$ other nodes' values.

The first two cases need no explanation. Now let us investigate the third case. Without loss of generality, we can arrange the nodes in an order such that $v_1\ , v_2 \cdots v_{n-k-1}$ are the honest sites, v_i is the node whose privacy is at stake and $v_{i+1}, \cdots v_{i+k}$ form the colluding group. We have

$$\underbrace{\sum_{j=1}^{n-k-1} v_j}_{denoted\ by\ X} + \underbrace{v_i}_{denoted\ by\ Y} = \underbrace{v - \sum_{j=i+1}^{i+k} v_j}_{denoted\ by\ W}$$

where *W* is a constant and is known to all the colluding nodes. Now, it is clear that the colluding nodes will know v_i is not greater than *W*, which is some extra information contributing to the utility of the collusions. To take a further look, the colluding nodes can compute the posteriori probability of v_i and further use that to launch a maximum a posteriori probability (MAP) estimate-based attack. It can be shown that, this posteriori probability is:

$$f_{posterior}(v_i) = \frac{1}{(m+1)(n-k-1)} \times \sum_{j=0}^{r} (-1)^j c_j^{(n-k-1)} \times c_{(n-k-1)+(r-j)(m+1)+t-1}^{(r-j)(m+1)+t}$$

where $v_i \leq W, r = \left\lfloor \frac{W - v_i}{m+1} \right\rfloor$ and $t = W - v_i - \left\lfloor \frac{W - v_i}{m+1} \right\rfloor (m+1)$. When $v_i > W, f_{posterior}(v_i) = 0$.

Due to space constraints, we have not included the proof of this result here.

Game Analysis (Kargupta et al., 2007)**:** In a multi-party PPDM environment, each node has certain responsibilities in terms of performing their part of the computations, communicating correct values to other nodes and protecting the privacy of the data. Depending on the characteristics of these nodes and their objectives, they either perform their duties or not, sometimes, they even collude with others to modify the protocol and reveal others' private information. Let M_i denote the overall sequence of computations node *i* has performed, which may or may not be the same as what it is supposed to do defined by the PPDM protocol. Similarly, let R_i be the messages node *i* has received, and S_i he messages it has sent. Let G_i be a subgroup of the nodes that would collude with node *i*. The strategy of each node in the multi-party PPDM game prescribes the actions for such computations, communications, and collusions with other nodes, *i.e.*, $\sigma_i = (M_i, R_i, S_i, G_i)$. Further let $c_{i,m}(M_i)$ be the utility of performing M_i, and similarly we can define $c_{i,r}(R_i), c_{i,s}(S_i), c_{i,g}(G_i)$. Then the overall utility of node *i* will be a linear or nonlinear function of utilities obtained by the choice of strategies in the respective dimensions of computation, communication and collusion. Without loss of generality, we consider an utility function which is a weighted linear combination of all of the above dimensions:

$$u_i(\{\sigma_i, \sigma_{-i}\}) = \omega_{i,m}c_{i,m}(M_i) + \omega_{i,s}c_{i,s}(S_i) + \omega_{i,r}c_{i,r}(R_i) + \omega_{i,g}c_{i,g}(G_i)$$

where $\omega_{i,m}, \omega_{i,s}, \omega_{i,r}, \omega_{i,g}$ represent the weights for the corresponding utilityfactors. Note that we omitted other nodes' strategies in the above expression just for simplicity.

In secure sum computation, the derived posteriori probability can be used to quantify the utility of collusion, *e.g.*,

$$g(v_i) = Posteriori - \Pr ior = f_{posterior}(v_i) - \frac{1}{m+1}$$

We see here that this utility depends on $W - v_i$ and the size of the colluding group *k*. Now we can put together the overall utility function for the game of multi-party secure sum computation:

$$u_i(\{\sigma_i, \sigma_{-i}\}) = \omega_{i,m}c_{i,m}(M_i) + \omega_{i,s}c_{i,s}(S_i) + \omega_{i,r}c_{i,r}(R_i) + \omega_{i,g} \sum_{j \in P - G_i} g(v_j)$$

where *P* is the set of all nodes and G_i is the set of nodes colluding with node *i*.

Now considering a special instance of the overall utility where the node performs all the communication and computation related activities as required by the protocol. This results in a function: $u_i(\{\sigma_i, \sigma_{-i}\}) = \omega_{i,g} \sum_{j \in P - G_i} g(v_j)$, where the utilities due to communication and computation are constant and hence can be neglected for determining the nature of the function. Through studying the plot of the overall utility of multi-party secure sum as a function of the distribution of the random variable $W - v_i$ and the size of the colluding group *k*, it shows that the utility is maximum for a value of *k* that is greater than 1. Since the strategies opted by the nodes are dominant, the optimal solution corresponds to the Nash equilibrium. This implies that in a realistic scenario for multi-party secure sum computation, nodes will have a tendency to collude. Therefore the non-collusion (*k* = 1) assumption of the classical secure multi-party sum is sub-optimal.

From the above analysis we can see, the collusion of parties happen easily to gain additional benefits in multi-party privacy preserving data mining, because the strategies of following protocol is not always optimal. Based on the penalty mechanism without having to detect collusion, a cheap-talk protocol is proposed to offer a more robust process,and the optimal strategy is to following the protocol for secure computation with punishment strategy.

In a word, the semi-honest adversarial model is often a realistic one (Lindell & Pinkas ,2002), but it sometimes deviate from the real-life application of privacy preserving distributed data mining, so it will a new trend for privacy preserving data mining is to make the collusion resistant protocols or algorithms algorithm work well in real life. Recent research have addressed this issue, and collusion resistant protocols or algorithms based on penalty function mechanism, the Secret Sharing Technique, and the Homomorphic Threshold Cryptography are given.

4.2 Collusion resistant protocols based on penalty function mechanism

For distributed privacy preserving data mining, to achieve a Nash equilibrium with no collusions, the game players can adopt a punishment strategy to threaten potential deviators. Kargupta et al., (2007) design a mechanism to penalize colluding nodes in various ways:

1. Policy I: Remove the node from the application environment because of protocol violation. Although it may work in some cases, the penalty may be too harsh since usually the goal is to have everyone participate in the process and faithfully contribute to the data mining process.

2. Policy II: Penalize by increasing the cost of computation and communication. For example, if a node suspects a colluding group of size k' (an estimate of k), then it may split the every number used in a secure sum among $\alpha k'$ different parts and demand $\alpha k'$ rounds of secure sum computation one for each of these $\alpha k'$ parts, here $\alpha > 0$ is a constant factor. This increases the computation and communication cost by $\alpha k'$ fold. This linear increase in cost with respect to k', the suspected size of colluding group, may be used to counteract possible benefit that one may receive by joining a team of colluders. The modified utility function is given by $\tilde{u}_i(\{\sigma_i, \sigma_{-i}\}) = u_i(\{\sigma_i, \sigma_{-i}\}) - \omega_{ip} * \alpha k'$ .The last term in the equation accounts for the penalty due to excess computation and communication as a result of collusion.

The new secure sum with penalty (SSP) protocol is as follows (Kargupta et al., 2007):

Consider a network of *n* nodes where a node can either be *good* (honest) or *bad* (colluding). Before the secure sum protocol starts, the good nodes set their estimate of bad nodes in the network $k' = 0$ and bad nodes send invitations for collusions randomly to nodes in the network. Every time a good node receives such an invitation, it increments its estimate of k'. Bad nodes respond to such collusion invitations and form collusions. If a bad node does not receive any response, it behaves as a good node. To penalize nodes that collude, good nodes split their local data into $\alpha k'$ random shares. This initial phase of communication is cheap talk in our algorithm. The secure sum phase consists of $O(\alpha k')$ rounds of communication for every complete sum computation. This process converges to the correct sum in $O(n\alpha k)$ time. Note that, the SSP protocol does not require detecting all the colluding parties. Raising k' based on a perception of collusion will do. If the threat is real, the parties are expected to behave as long they are acting rationally to optimize their utility.

4.3 Collusion resistant protocols based on the secret sharing technique

Now, we are particularly interested in the mining of association rule in a scenario where the data is vertically distributed among different parties. To mine the association rule, these parties need to collaborate with each other so that they can jointly mine the data and produce results that interest all of them. And we will provide a secure, efficient and collusion resistant distributed association rules mining algorithm based on the Shamir's secret sharing technique (Shamir,1979).

4.3.1 Shamir's secret sharing technique

Shamir's secret sharing method (Shamir,1979) allows a dealer D to distribute a secret value v_s among n peers $P_1, P_2, \cdots P_n$, such that the knowledge of any $k(k \le n)$ peers is required to reconstruct the secret. The method is described in Algorithm 1.

Algorithm 1(Shamir's secret sharing algorithm):

```
Require: $v_s$ : Secret value,
P: Set of parties $P_1, P_2, \cdots P_n$ to distribute the shares,
$k$ : Number of shares required to reconstruct the secret.
1: Select a random polynomial
   $q(x) = a_{k-1}x^{k-1} + \cdots + a_1 x^1 + v_s$ ,where $a_{k-1} \ne 0, q(0) = v_s$ .
2: Choose $n$ publicly known distinct random values $x_1, \cdots x_n$ such that $x_i \ne 0$ .
3: Compute the share of each peer, $P_i$, where $share_i = q(x_i)$ .
4: for $i = 1$ to $n$ do
5:    Send $share_i$ to peer $P_i$ .
6: end for.
```

Shamir's method is information theoretically secure, in order to construct the secret value v_s, at least k shares are required to determine the random polynomial $q(x)$ of degree $k-1$, so the complete knowledge of up to $k-1$ peers does not reveal any information about the secret.

4.3.2 Privacy-preserving distributed association rule mining problem

Party $P_1, P_2, \cdots P_n$ have private data set $DB_1, DB_2, \cdots, DB_n$ respectively, and $DB_i \cap DB_j = \Phi, for \ \forall \ i, j \in n$. The data set $DB_1, DB_2, \cdots, DB_n$ forms a database DB , namely $DB = DB_1 \bigcup DB_2 \bigcup \cdots \bigcup DB_n$, let N denote the total number of transactions for each data set. The n parties want to conduct association rule mining on $DB = DB_1 \bigcup DB_2 \bigcup \cdots \bigcup DB_n$ and find the association rule with support and confidence being greater than the given thresholds.

During the mining of association rule, we assume all parties follow the protocol, and the object of the paper is to propose a protocol of distributed association rules mining in vertically partitioned data based on the Shamir's secret sharing technique (Shamir,1979), which can prevent effectively the collusion behaviors and conduct the computations across the parties without compromising their data privacy, simultaneously, the security of the protocol refer to semantic security (Goldreich, 2001).

4.3.3 Distributed association rule mining algorithm

In order to learn association rule, one must compute confidence and support of a given candidate itemset, and given the values of the attributes are 1 or 0, to judge whether a particular itemset is frequent, we only need to find out the number of records (denote ***c.count***) where the values for all the attributes in the itemset are 1. if $\text{c.count} \ge \text{Ns}\%$ ，then the candidate itemset is the frequent itemset. The following is the algorithm to find frequent itemsets:

Algorithm 2: The algorithm to find frequent itemsets

```
1. $L_1 = \{\text{ large } 1\_itemsets\}$
2. for $(k = 2; L_{k-1} \neq \Phi; k++)$ do begin
3. $C_k = \text{apriori-gen}(L_{k-1})$
4.     for all candidates $c \in C_k$ do begin
5.         if all the attributes in c are entirely the same party that party
           independently compute c.count
6.         else
           collaboratively compute c.count (We will show how to compute it in
           Section 4.3.)
7.     end
8. $L_k = L_k \bigcup \{c \mid c.count \geq \min\sup\}$
9. end
10. Return $L = \bigcup_k L_k$
```

In step 3, the function $C_k = \text{apriori-gen}(L_{k-1})$ can generate the set of candidate itemsets C_k, which is discussed in (Agrawal & Srikant, 1994). Given the counts and frequent itemsets, we can compute all association rules with $\text{support} \geq \text{minsup}$.

In the procedure of association rule mining, step 1, 3, 6 and 8 require sharing information. In step 3 and 8, we use merely attribute names, in step 1, to compute large 1-itemsets, each party elects her own attributes that contribute to large $1\_itemsets$, where only one attribute forms a large 1-itemset, there is no computation involving attributes of other parties, therefore, data disclosure across parties is not necessary. At the same time, since the final result $L = \bigcup_k L_k$ is known to all parties, step 1, 3 and 8 reveal no extra information to either party. However, to compute *c.count* in step 6, a computation accessing attributes belonging to different parties is necessary. How to conduct these computations across parties without compromising each party's data privacy is the challenge we are faced with. If the attributes belong to different parties, they then construct vectors for themselves attributes, for example, for the some candidate itemset, party P_i have p attributes $a_1, a_2, \cdots, a_p$, then party P_i can construct vector A_i, the jth element denote $A_{ij} = \prod_{k=1}^{p} a_k$ in vector A_i. Subsequently, they can apply our secure algorithm to obtain *c.count*, which will be discussed in Section 4.3.4

4.3.4 Privacy-preserving algorithm to collaboratively compute *c.count*

The fact that the distributed parties jointly compute ***c.count*** without revealing their raw data to each other presents a great challenge. In this section, we show how to privately compute ***c.count*** based on Shamir's secret sharing algorithm (Shamir, 1979) for the case of multiple parties without revealing the secret values to others.

Without loss of generality, assuming party P_1 has a private vector A_1, party P_2 , a private vector A_2 ,…, and party P_n, a private vector A_n, we use A_{ij} to denote the jth element in vector A_i, the value of A_{ij} is the attribute value of the P_i in the jth transaction of the database. Given that the absence or presence of an attribute is represented as 0 or 1, the value of A_{ij} is equal to 0 or 1, for example $A_i = (1, 0, 1, 1 \cdots 0)^T$.

Assuming all parties follow the algorithm and do the computations honestly, the whole process is summarized in Algorithm 3.

Algorithm 3: Privacy-Preserving Algorithm to Collaboratively Compute *c.count*

Require: P: Set of parties $P_1, P_2, \cdots P_n$.

$A_{i\,j}$: Secret value of P_i ,

X : A set of n publicly known random values $x_1, x_2, \cdots x_n$.

k : Degree of the random polynomial $k = n-1$.

```
for each transaction $j = 1$ to $N$ do
  for each party $P_i$ ($i = 1, \cdots n$) do
    Select a random polynomial $q_i(x) = a_{n-1}x^{n-1} + \cdots + a_1 x^1 + A_{i\,j}$
    Compute the share of each party $P_t$, where $sh(A_{i\,j}, P_t) = q_i(x_t)$
      for $t = 1$ to n do
        Send $sh(A_{i\,j}, P_t)$ to party $P_t$
    Receive the shares $sh(A_{i\,j}, P_t)$ from every party $P_t$.
    Compute $S(x_i) = q_1(x_i) + q_2(x_i) + \cdots + q_n(x_i)$
      for $t = 1$ to n do
        Send $S(x_i)$ to party $P_t$
    Receive the results $S(x_i)$ from every party $P_i$.
    Solve the set of equations to find the sum $\sum_{i=1}^{n} A_{i\,j}$ of secret values
    if the $\sum_{i=1}^{n} A_{i\,j} = n$, let $m_j = 1$, otherwise $m_j = 0$.
Each party computes $c.count = \sum_{j=1}^{N} m_j$ .
```

4.3.5 Analysis of the privacy-preserving algorithm to obtain *c.count*

In this section, we give the correctness, complexity and security analysis of the privacy-preserving algorithm 3.

Correctness Analysis: Assuming party P_i has a private vector $\mathrm{A_i}$, so for arbitrary transaction j in database DB, party P_i has a private value $\mathrm{A_{i\,j}}$, according algorithm 3, the sum $\sum_{i=1}^{n} A_{i\,j}$ of secret values is the constant term of the sum polynomial $S(x) = q_1(x) + q_2(x) + \cdots + q_n(x)$, so we need to solve the following liner equations:

$$\begin{cases} b_{n-1}{x_1}^{n-1} + b_{n-2}{x_1}^{n-2} + \cdots + b_1 x_1 + \sum_{i=1}^{n} A_{i\,j} = s(x_1) \\ b_{n-1}{x_2}^{n-1} + b_{n-2}{x_2}^{n-2} + \cdots + b_1 x_2 + \sum_{i=1}^{n} A_{i\,j} = s(x_2) \\ \cdots\cdots \\ b_{n-1}{x_n}^{n-1} + b_{n-2}{x_n}^{n-2} + \cdots + b_1 x_n + \sum_{i=1}^{n} A_{i\,j} = s(x_n) \end{cases}$$

Noted that there are n unknown coefficients and n equations, determinant of coefficient $D = \begin{vmatrix} {x_1}^{n-1} & {x_1}^{n-2} & \cdots & x_1 & 1 \\ {x_2}^{n-1} & {x_2}^{n-2} & \cdots & x_1 & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ {x_n}^{n-1} & {x_n}^{n-2} & \cdots & x_n & 1 \end{vmatrix}$ is the Vander monde determinant，when $D = \prod_{1 \le j \le i \le n} (x_i - x_j) \ne 0$, that is $x_i \ne x_j$, the equations has a unique solution, and each party P_i can solve the set of equations and determine the value of $\sum_{i=1}^{n} A_{i\,j}$, however it cannot determine the secret values of the other parties since the individual polynomial coefficients selected by other parties are not known to P_i. If $A_{1\,j}, A_{2\,j}, \cdots, A_{n\,j}$ are all equal to 1, that is $\sum_{i=1}^{n} A_{i\,j} = n$, this means the transaction has the whole attributes and supports the association rule, we let $m_j = 1$. Otherwise, if some attributes of $A_{1\,j}, A_{2\,j}, \cdots, A_{n\,j}$ are not equal to 1, that is, $\sum_{i=1}^{n} A_{i\,j} \ne n$, this means the transaction has not the whole attributes and does not support the association rules, we let $m_j = 0$. To compute the number of transactions which support the association rule, we only count the number of $m_j = 1$, then $c.count = \sum_{j=1}^{N} m_j$, so the algorithm 3 can compute ***c.count*** correctly under the condition of all parties doing the computations honestly during the mining of association rule.

Complexity Analysis: Assuming there are N transactions and n parties, the communication cost is $2n(n-1)$ from step 5,6 and step 9,10 of algorithm 3, so the communication cost of algorithm 3 is $2Nn(n-1)$.

The following contribute to the computational cost of each transaction: (1) the generation of the random polynomial $q_i(x)$,$i = 1, \cdots n$ from step 3; (2) the total number of n^2 computations on the share of each party from step 4; (3) the total number of $n(n-1)$ additions from step 8; (4) the computational cost of solving the set of equations to find the sum $\sum_{i=1}^{n} A_{i\,j}$ of secret values from step 12; (5)the computational cost of letting $m_j = 1\, or\, 0$ according the sum $\sum_{i=1}^{n} A_{i\,j}$ from step 13.

Compared to the other technique, for example, commutative encryption and secure multi-party computations, although these techniques are very secure, the excessive computation and communication cost associated render them impractical for scenarios involving a large number of parties. However the algorithm proposed by our paper is scalable in terms of computation and communication cost, and therefore it can be run even when there is a large number of parties involved. So our algorithm is efficient and practical.

Security Analysis: Proposition 1: Algorithm 3 is semantic security (Goldreich, 2001) for the network attackers.

Proof: A network attackers listening to the network traffic of the parties cannot learn any useful information, such as the private values or the sum of those values, expect for all the shares and the intermediate values, however these values cannot be used to determine the coefficients of the sum polynomial and each party's secretly random polynomial without knowing random values $x_1, x_2, \cdots x_n$ for which the intermediate results are calculated, and it can not be concluded whether the transaction is support the association rule.

Proposition 2: Algorithm 3 is semantic security and can prevent effectively the collusion behaviors for the collaborative parties under the condition of the number of the collusion parties $t < n-1$.

Proof: Firstly, algorithm 3 is semantic security for the collaborative parties. Compared with the network attackers, the collaborative parties know random values $x_1, x_2, \cdots x_n$. At algorithm 3, P_i computes the value of its polynomial at *n* points as shares, and then keeps one of these shares for itself and sends the remaining $n-1$ shares to other parties, so if the collaborative party gets all other parties shares and intermediate values through listening to the network traffic of the parties, except for the value of the corresponding sum polynomial at *n* different points, he can get, for example, the value of that party P_i's secretly random polynomial at $n-1$ different point $x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n$. And because the degree of each party P_i's secretly random polynomial is $k = n-1$ and have *n* unkown coefficients, in order to compute the coefficients of the corresponding party P_i's secretly random polynomial and get the party P_i's private value, the value at *n* different points are needed, so party P_i's private value can not be achieved.

Secondly, algorithm 3 can prevent effectively the collusion behaviors for the collaborative parties under the condition of the number of the collusion parties $t < n-1$. If there are $n-1$ parties collusion, for example, $P_2, \cdots P_n$, they can get the value $s(x_1)$ of sum polynomial at x_1 and $q_1(x_2), \cdots, q_1(x_n)$ which is the value of party P_1's secretly random polynomial $q_1(x)$ at $n-1$ different points $x_2, \cdots x_n$, note that $S(x_1) = q_1(x_1) + q_2(x_1) + \cdots + q_n(x_1)$, so they can compute $q_1(x_1) = S(x_1) - q_2(x_1) - \cdots - q_n(x_1)$, then can conclude the party P_1's private value through solving the following liner equations:

4.4 Collusion resistant protocols based on the homomorphic threshold cryptography

Now we have known homomorphic encryption and threshold decryption often are used as basic building blocks of secure computation protocols for PPDM. In this section , we will study the privacy preserving distributed association rule mine problem(§4.3.2) base on homomorphic encryption and threshold decryption, where homomorphic encryption and threshold decryption can refer to §3.3.3.

In order to learn association rule, we should find frequent itemsets, Algorithm 2 give how to find frequent itemsets, where the key step is step 6 (compute c.count) . How to conduct these computations across parties without compromising each party's data privacy is the challenge we are faced with. If the attributes belong to different parties, they then construct vectors for themselves attributes, for example, for the some candidate itemset, party P_i have p attributes $a_1, a_2, \cdots, a_p$, then party P_i can construct vector A_i , the jth element denote $A_{ij} = \prod_{k=1}^{p} a_k$ in vector A_i . Subsequently, they can also apply our secure algorithm to obtain *c.count*, which will be discussed in Section 4.4.1.

4.4.1Collusion resistant protocol based on the homomorphic threshold cryptography

The fact that the collaborative parties jointly compute ***c.count*** without revealing their raw data to each other presents a great challenge. In this section, we develop secure protocol to compute ***c.count*** for the case of multiple parties. Without loss of generality, assuming Party P_1 has a private vector A_1 , Party P_2 , a private vector A_2 ,…, and Party P_n , a private vector $\mathrm{A_n}$. we use A_{ij} to denote the jth element in vector A_i , so the value of A_{ij} is the attribute value of the P_i in the jth transaction of the database. Given that the absence or presence of an attribute is represented as 0 or 1, the value of $A_{i\,j}$ is equal to 0 or 1, for example $A_i = (1, 0, 1\ ,1 \cdots 0)^T$.

1. $P_1, P_2, \cdots P_n$ **perform the following:**

a. $P_1, P_2, \cdots P_t\ (1 \le t \le n)$ jointly generate a threshold cryptographic key pair $(d(d_1, d_2, \cdots, d_t), e)$ of a homomorphic encryption scheme. That is, a secret key associated with a single public key is distributed among a group of parties. For simplicity and without loss of generality, let $t = n$, then only if all parties cooperate, can they decrypt the ciphertext and prevent the collusion of parties. Let $e(\cdot)$ denote encryption and $d_i(\cdot)$ denote party i decryption. Meanwhile, the threshold cryptographic key pair $(d(d_1, d_2, \cdots, d_t), e)$ is semantic security. They also generate the number, X, where X is an integer which is more than n.

b. P_1 generates a set of random integers $R_{1\,1}, R_{1\,2}, \cdots, R_{1\,N}$ and sends $e(A_{1\,1} + R_{1\,1}X), e(A_{1\,2} + R_{1\,2}X), \cdots, e(A_{1\,N} + R_{1\,N}X)$ to P_n ; P_2 generates a set of random integers $R_{2\,1}, R_{2\,2}, \cdots, R_{2\,N}$ and sends $e(A_{2\,1} + R_{2\,1}X), e(A_{2\,2} + R_{2\,2}X), \cdots, e(A_{2\,N} + R_{2\,N}X)$ to P_n ;… 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 ; P_n generates a set of random integers $R_{n\,1}, R_{n\,2}, \cdots, R_{n\,N}$ and encrypts his private vector

$$e(A_{n\,1} + R_{n\,1}X), e(A_{n\,2} + R_{n\,2}X), \cdots, e(A_{n\,N} + R_{n\,N}X)$$

c. P_n computes:

$$\begin{aligned} E_1 &= e(A_{1\,1} + R_{1\,1}X) \times e(A_{2\,1} + R_{2\,1}X) \times \cdots \times e(A_{n\,1} + R_{n\,1}X) \\ &= e(A_{1\,1} + A_{2\,1} + \cdots A_{n\,1} + (R_{1\,1} + R_{2\,1} + \cdots R_{n\,1})X) \end{aligned}$$

$$\begin{aligned} E_2 &= e(A_{1\,2} + R_{1\,2}X) \times e(A_{2\,2} + R_{2\,2}X) \times \cdots \times e(A_{n\,2} + R_{n\,2}X) \\ &= e(A_{1\,2} + A_{2\,2} + \cdots A_{n\,2} + (R_{1\,2} + R_{2\,2} + \cdots R_{n\,2})X) \end{aligned}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\begin{aligned} E_N &= e(A_{1\,N} + R_{1\,N}X) \times e(A_{2\,N} + R_{2\,N}X) \times \cdots \times e(A_{n\,N} + R_{n\,N}X) \\ &= e(A_{1\,N} + A_{2\,N} + \cdots A_{n\,N} + (R_{1\,N} + R_{2\,N} + \cdots R_{n\,N})X) \end{aligned}$$

d. P_n randomly permutes $E_1, E_2, \cdots, E_N$ and obtains the permuted sequence $D_1, D_2, \cdots, D_N$.

e. From computational balance point of view, P_n divides $D_1, D_2, \cdots, D_N$ into n parts with each part having approximately equal number of elements.

f. P_n decrypts using himself private key d_n and sends $d_n(D_1), d_n(D_2), \cdots, d_n(D_{N/n})$ to P_{n-1} ; P_{n-1} decrypts $d_{n-1}d_n(D_1), d_{n-1}d_n(D_2), \cdots, d_{n-1}d_n(D_{N/n})$, send it to P_{n-2} ,…, P_1 ; P_1 decrypts $d_1 d_2 \cdots d_{n-1} d_n(D_1) = M_1$, $d_1 d_2 \cdots d_{n-1} d_n(D_2) = M_2, \cdots, d_1 d_2 \cdots d_{n-1} d_n(D_{N/n}) = M_{N/n}$;

g. P_n decrypts using himself private key d_n and sends

$d_n(D_{N/n+1}), d_n(D_{N/n+2}), \cdots, d_n(D_{2N/n})$ to P_{n-1}; P_{n-1} decrypts
$d_{n-1}d_n(D_{N/n+1}), d_{n-1}d_n(D_{N/n+2}), \cdots, d_{n-1}d_n(D_{2N/n})$, sends it to $P_{n-2}, \ldots, P_3, P_1, P_2$,
P_2 decrypts
$d_2d_1\, d_3\cdots d_{n-1}d_n(D_{N/n+1}) = M_{N/n+1}$, $d_2d_1\, d_3\cdots d_{n-1}d_n(D_{N/n+2}) = M_{N/n+2}, \cdots,$
$d_2d_1\, d_3\cdots d_{n-1}d_n(D_{2N/n}) = M_{2N/n}$;

h. Continue until P_n decrypts using himself private key d_n and sends $d_n(D_{(n-2)N/n}+1), d_n(D_{(n-2)N/n+2}), \cdots, d_n(D_{(n-1)N/n})$ to P_{n-2}, P_{n-2} decrypts and sends it to $P_{n-3}, \ldots, P_2, P_1, P_{n-1}$, P_{n-1} decrypts

$d_{n-1}d_1\, d_2\cdots d_{n-2}d_n(D_{(n-2)N/n}+1) = M_{(n-2)N/n+1}, d_{n-1}d_1\, d_2\cdots d_{n-2}d_n(D_{(n-2)N/n+2}) = M_{(n-2)N/n+2}, \cdots,$
$d_{n-1}d_1\, d_2\cdots d_{n-2}d_n(D_{(n-1)N/n}) = M_{(n-1)N/n}$

i. P_n sends $D_{(n-1)N/n+1}, D_{(n-1)N/n+2}, \cdots, D_N$ to P_{n-1}, P_{n-1} decrypts
$d_{n-1}(D_{(n-1)N/n+1}), d_{n-1}(D_{(n-1)N/n+2}), \cdots, d_{n-1}(D_N)$, sends it to $P_{n-2}, \ldots, P_2, P_1, P_n$
P_n decrypts
$d_nd_1\, d_2\cdots d_{n-2}d_{n-1}(D_{(n-1)N/n+1}) = M_{(n-1)N/n+1}, d_nd_1\, d_2\cdots d_{n-2}d_{n-1}(D_{(n-1)N/n+2}) = M_{(n-1)N/n+2},$
$\cdots, d_nd_1\, d_2\cdots d_{n-2}d_{n-1}(D_N) = M_N$

2. Compute *c.count*

a. $P_1, P_2, \cdots P_n$ make $M_1, M_2, \cdots M_N$ module X respectively, note that if a decrypted term M_i is equal to $n \bmod X$, it means the values of $P_1, P_2, \cdots P_n$ are all 1, then let $m_i = 1$, otherwise $m_i = 0$ For example, if the transaction j is permuted as position i ,then $M_i \bmod X = (A_{1j} + A_{2j} + \cdots A_{nj} + (R_{1j} + R_{2j} + \cdots R_{nj})X) \bmod X = A_{1j} + A_{2j} + \cdots A_{nj}$

Consequently, compare whether each decrypted term $M_i \bmod X$ is equal to $n \bmod X$. If yes, then let $m_i = 1$, otherwise $m_i = 0$.

b. P_1 computes $c_1 = \sum_{i=1}^{N/n} m_i$, P_2 computes $c_2 = \sum_{i=N/n+1}^{2N/n} m_i$, …, P_n computes

$$c_n = \sum_{i=(n-1)N/n+1}^{N} m_i \ .$$

c. all parties $P_i\ (i = 1, 2, \cdots, n-1)$ encrypted $e(c_i)$ and send it to P_n .

d. P_n computes $e(c_1) \times e(c_2) \times \cdots e(c_n) = e(c_1 + c_2 + \cdots c_n)$,then decrypts $d_n(e(c_1 + c_2 + \cdots c_n))$ and sends it to P_{n-1}, P_{n-1} decrypts $d_{n-1}d_n(e(c_1 + c_2 + \cdots c_n))$ and sends it to $P_{n-2}, \ldots,$ P_1 decrypts $d_1 \cdots d_{n-1}d_n(e(c_1 + c_2 + \cdots c_n)) = c_1 + c_2 + \cdots c_n = c.count$.

4.4.2 Analysis of collusion resistant protocol based on the homomorphic threshold cryptography

Correctness Analysis: Assume all of the parties follow the protocol, in which the threshold cryptographic system is a additively homomorphic cryptosystem, which enable us to get

$$E_i = e(A_{1i} + R_{1i}X) \times e(A_{2i} + R_{2i}X) \times \cdots \times e(A_{ni} + R_{ni}X)$$
$$= e(A_{1i} + A_{2i} + \cdots A_{ni} + (R_{1i} + R_{2i} + \cdots R_{ni})X) \quad i = 1, 2, \cdots n$$

And given $X > n$, so

$$M_i \bmod X = (A_{1j} + A_{2j} + \cdots A_{nj} + (R_{1j} + R_{2j} + \cdots R_{nj})X) \bmod X = A_{1j} + A_{2j} + \cdots A_{nj}$$

If $A_{1j}, A_{2j}, \cdots, A_{nj}$ are all equal to 1, this means the transaction has the whole attributes and supports the association rule, we let $m_i = 1$. Otherwise, if some attributes of $A_{1j}, A_{2j}, \cdots, A_{nj}$ are not equal to 1, this means the transaction has not the whole attributes and does not support the association rules, we let $m_i = 0$, to compute the number of transactions which support the association rule, we only count the number of $m_i = 1$, then

$$c.count = c_1 + c_2 + \cdots c_n = \sum_{i=1}^{N} m_i \; .$$

Meanwhile, in the protocol, P_n permutes $E_i, i = 1, 2, \cdots N$ before sending them to other parties, permutation does not affect $c.count$, and summation is not affected by a permutation. Therefore, the final $c.count$ is correct.

Complexity Analysis: The bit-wise communication cost of this protocol is upper bounded by $2\alpha[(n-1)N + n]$, where a is the number of bits for each encrypted element. It consist of (1) the maximum cost of $(n-1)N$ from step 1(b); (2) the maximum cost of $(n-1)N$ from step 1(f)-1(i); (3) the maximum cost of $2n$ from step 2(c) and 2(d).

The following contributes to the computational cost: (1) the generation of a threshold cryptographic key pair, the integer X and nN random integers (2) the total number of $nN + n$ encryptions; (3) the total number of $(n-1)(N+1)$ multiplications; (4) the generation of permutation function; (5) the total number of N permutations; (6) the total number of $(n+1)N$ decryptions; (7) the total number of N modulo operations; (8) the total number of $(n+1)N$ additions; (9) dividing N numbers into n parts.

Security Analysis: Given that P_n obtains all the encrypted terms from other parties and the cryptographic system is a semantic security, the ciphertext does not leak any useful information about the plaintext and P_n can not get other useful information of the plaintext from the ciphertext. Meanwhile, since the cryptographic system is a threshold cryptosystem, those parties will not able to decrypt and get the plaintext unless they cooperate. That is, P_n will not have access to the original values of other parties without cooperating with those parties. As a result, the collusion behaviors can be prevented effectively. $P_1, P_2, \cdots P_n$ in our protocol jointly generate a threshold cryptographic key pair $(d(d_1, d_2, \cdots, d_n), e)$ of a homomorphic encryption scheme, which means the protocol is secure under the condition of the number of the collusion parties is less than n. Generally, given $P_1, P_2, \cdots P_t \; (1 \le t \le n)$ jointly generate a threshold cryptographic key pair $(d(d_1, d_2, \cdots, d_t), e)$ of a homomorphic encryption scheme, which means the protocol is secure under the condition of the number of the collusion parties is less than t.

Meanwhile, Each party of $P_1, P_2, \cdots P_{n-1}$ obtains some plaintexts of all D_i. Since D_i are in permuted form and those n-1 parties don’t know the permutation function, so they cannot know which transaction support the association rule. And each party only knows a part of transactions supporting the association rules, which lead to trivial benefit for them.

5. Conclusion

Due to the right to privacy in the information ear, privacy-preserving data mining (PPDM) has become one of the newest trends in privacy and security and data mining research. In this chapter, we intrduced the related concepts of privacy-preserving data mining and some privacy preserving techniques such as Trust Third Party , Data perturbation technique, Secure Multiparty Computation and game theoretic approach. Moreover, we discussed the collussion behaviors in privacy-preserving data mining (PPDM) and gave the collusion resistant protocols or algorithms based on penalty function mechanism, the Secret Sharing Technique, and the Homomorphic Threshold Cryptography.

6. References

Abraham I., Dolev D., Gonen R. &Halpern J. (2006). Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, pp. 53–62, ISBN:1-59593-384-0, New York, NY, USA,. ACM Press.

Agrawal, R. & Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases, *Proceedings of 20th International Conference on Very Large Data Bases*，pp.487-499, ISBN 55860-153-8, Santiago, Chile.

Agrawal R. & Srikant R. (2000). Privacy-Preserving Data Mining. *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp.439–450, ISBN 1-58113-217-4.

Clifton C., Kantarcioglu M. &Vaidya J.(2002a). Defining privacy for data mining. *Proceeding of the National Science Foundation Workshop on Next Generation Data Mining*, pp.126-133, Baltimore, MD, USA.

Clifton C., Kantarcioglu M., Vaidya J., Lin X. & Zhu M.Y. (2002b). Tools for Privacy Preserving Distributed Data Mining. *ACM SIGKDD Explorations*, Vol 4, No 2, pp. 28-34, ISSN 1931-0145.

Cramer R., Damgard I. & Nielsen J. B. (2001) . Multiparty Computation from Threshold Homomorphic Encryption. *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology,* pp. 280-299, ISBN:3-540-42070-3, Springer-Verlag.

Du W. & Zhan Z. (2002). Building decision tree classifier on private data. *Proceedings of the IEEE international conference on Privacy, security and data mining,* pp. 1-8, ISSN 0-909-92592-5, Maebashi City, Japan.

Emekci F., Sahin O. D., Agrawal D. & Abbadi A. El. (2007). Privacy preserving decision tree learning over multiple parties. *Data and Knowledge Engineering*, Vol.63, No 2, pp.348–361,ISSN 0169-023X.

Even S., Goldreich O. & Lempel A.(1985). A Randomized Protocol for Signing Contracts, *Communications of the ACM,* vol. 28, Issue 6, pp. 637–647, ISSN 0001-0782.

Evfimievski A., Srikant R., Agrawal R. & Gehrke J. (2002). Privacy-Preserving Mining of Association Rules. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.217–228, ISBN:1-58113-567-X, Edmonton, Alberta, Canada.

Fienberg S. E. & McIntyre J.(2004). Data Swapping: Variations on a Theme by Dalenius and Reiss. *Privacy in Statistical Databases (PSD)*, pp.14-29, Barcelona, Spain.

Franklin M. K & Haber S. (1996). Joint Encryption and Message-Efficient Secure Computation. *Journal of Cryptology*, Vol 9, No 4, pp.217-232, ISSN 0933-2790 .

Gilburd B. ; Schuster A. & Wolff. R. (2004). k-TTP: A New Privacy Model for Largescale Distributed Environments. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 563–568, ISBN 1-58113-888-1, Seattle, WA, USA.

Goldreich O. (1998). Secure Multi-party Computation, http://www.wisdom weizmann.ac.il/.

Goldreich, O. (2001). *Foundations of cryptography*, Volume Basic Tools, ISBN 13:978-0521791724, Cambridge Universiy Press.

Han J.& Kamber M.(2006). *Data Mining: Concepts and Techniques*. 2nd edition, San Francisco: Morgan Kaufmann Publishers.

Jiang W., Clifton C. & Kantarcioglu M. (2008). Transforming semi-honest protocols to ensure accountability. *Data and Knowledge Engineering*, Vol.65, pp.57–74, ISSN 0169-023X .

Kargupta H., Das K.& Liu d K.(2007). Multi-party, privacy-preserving distributed data mining using a game theoretic framework. *PKDD,* Vol.4702, pp.523–531, Springer.

Kim J. J. & Winkler W. E.(2003). Multiplicative Noise for Masking Continuous Data. *Technical Report Statistics #2003-01*, Statistical Research Division, U.S. Bureau of the Census, Washington D.C.

Kleinberg J., Papadimitriou C.& Raghavan P. (1998). A microeconomic view of data mining. *Data Mining and Knowledge Discovery*, Vol 2, No 4, pp.311–324, ISSN 1384-5810.

Kleinberg J., Papadimitriou C.& Raghavan P. (2001). On the value of private information. *Proceedings of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge,* pp. 249–257, ISBN:1-55860-791-9, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Li X.B.& Sarkar S.(2006). A Tree-based Data Perturbation Approach for Privacy-Preserving Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol 18, No 9, pp.1278–1283 , ISSN 1041-4347.

Liew C. K., Choi U. J. & Liew C. J.(1985). A Data Distortion by Probability Distribution. *ACM Transactions on Database Systems (TODS)*, Vol 10, No 3, pp.395–411. [3] Lindell Y. & Pinkas B.(2002). Privacy preserving data mining. *Journal of Cryptology*, Vol.15, No 3, pp.177–206, ISSN 0933-2790 .

Lindell Y. & Pinkas B.(2009). Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, Vol 1, No 1, pp.59-98.

Liu K., Giannella C. & Kargupta H.(2006) An Attacker's View of Distance Preserving Maps for Privacy-Preserving Data Mining. *PKD 2006*, pp.297-308, LNCS.

Muralidhar K.& Sarathy R.(2006). Data shuffling a new masking approach for numerical data. *Management Science,* Vol 52, No 5, pp.658–670.

Paillier P. (1999). Public-key Cryptosystems based on Composite Degree Residuosity Classes. *Proceedings of the 17th international conference on Theory and application of cryptographic techniques,* pp. 223–238, ISSN 3-540-65889-0, Prague, Czech Republic.

Polat H. & Du W. (2005). SVD-based Collaborative Filtering with Privacy. *Proceedings of the 2005 ACM symposium on Applied computing*, pp.791-795, ISBN1-58113-964-0, Santa Fe, New Mexico.

Rabin M. O. (1981). How to Exchange Secrets by Oblivious Transfer, *Technical Report* TR-81, Aiken Computation Laboratory.

Shamir, A. (1979). How to share a secret. *Communications of the ACM,* Vol 22, No.11, pp 612-613, ISSN:0001-0782.

Stanley R. M. Oliveira and Osmar R. Zaïane. (2004). Toward standardization in privacy-preserving data mining, *ACM SIGKDD 3rd Workshop on Data Mining Standards*, pp. 7–17, Seattle, WA, USA.

Sweeney L.(2002). k-Anonymity: a Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, Vol 10, No 5, pp.557–570,

Vaidya J. & Clifton C.W. (2002) . Privacy preserving association rule mining in vertically partitioned data. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.639-644, ISBN 1-58113-567-X, Edmonton, Alberta, Canada.

Verykios V.S., Bertino E., Fovino I.N., Provenza L.P., Saygin, Y. & Theodoridis Y.(2004a). State-of-the-art in privacy preserving data mining, *SIGMOD Record,* Vol. 33, No. 1, pp.50-57.

Verykios V. S., Elmagarmid A. K., Bertino E., Saygin Y. & Dasseni E.(2004b) . Association Rule Hiding. *IEEE Transactions on Knowledge and Data Engineering*, Vol 16, Issue 4, pp.434–447, ISSN 1041-4347.

Wright R. & Yang Z. (2004). Privacy-preserving bayesian network structure computation on distributed heterogeneous data. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 713-718, ISBN:1-58113-888-1. ACM, New York, NY, USA.

Yao A. C. (1986). How to Generate and Exchange Secrets. *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science,* pp. 162-167, ISSN 0-8186-0740-8.

Zhan Z., Matwin S. & Chang L. (2007). Privacy-preserving collaborative association rule mining. *Journal of Network and Computer Applications,* Volume 30, Issue 3, pp. 1216-1227, ISSN:1084-8045.

Zhang P., Tong Y., Tang S.& Yang D.(2005). Privacy-Preserving Naive Bayes Classifier. *Advanced Data Mining and Applications*, Vol 3584， pp. 744-752, LNCS.

Zhu Y.& Liu L. (2004).Optimal Randomization for Privacy-Preserving Data Mining. *ACM KDD Conference*, pp.761-766, ISBN1-58113-888-1, Seattle, WA, USA.