International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 7, July 2013 ISSN 2319 - 4847

Security based Efficient Privacy Preserving Data Mining

Arpit Agrawal
Asst. Professor Department of CSE
Institute of Engineering & Technology
Devi Ahilya University
M.P., India

Abstract

Huge volume of detailed personal data is regularly collected and sharing of these data is proved to be beneficial for data mining application. Such data include shopping habits, criminal records, medical history, credit records etc. The development and penetration of data mining within different fields and disciplines, security and privacy concerns have emerged. Data mining technology which reveals patterns in large databases could compromise the information that an individual or an organization regards as private. The aim of privacy-preserving data mining is to find the right balance between maximizing analysis results that are useful for the common good and keeping the inferences that disclose private information about organizations or individuals at a minimum. In this paper we propose a new heuristic algorithm that improves the privacy of sensitive knowledge as itemsets by blocking more inference channels. The item count and increasing cardinality techniques based on item-restriction that hides sensitive itemsets. We demonstrate the efficiency of the algorithm, and also propose an efficient protocol that allows parties to share data in a private way with no restrictions and without loss of accuracy and we demonstrate the efficiency of the protocol. Our experimental shows the effectiveness comparison with existing works.

Key words: *Data Mining, PPDM, Association Rule, Public Key Cryptography.*

1. Introduction

Data mining deals with large database which can contain sensitive information. It requires data preparation which can uncover information or patterns which may compromise confidentiality and privacy obligations. Advancement of efficient data mining technique has increased the disclosure risks of sensitive data. A common way for this to occur is through data aggregation. Data aggregation is when the data are accrued, possibly from various sources, and put together so that they can be analyzed. This is not data mining per se, but a result of the preparation of data before and for the purposes of the analysis. The threat to an individual's privacy comes into play when the data, once compiled, cause the data miner, or anyone who has access to the newly compiled data set, to be able to identify specific individuals, especially when originally the data were anonymous.

What data mining causes is social and ethical problem by revealing the data which should require privacy? Providing security to sensitive data against unauthorized access has been a long term goal for the database security research community and for the government statistical agencies. Hence, the security issue has become, recently, a much more important area of research in data mining [1] and [2].

Data mining promises to discover unknown information. If the data is personal or corporate data, data mining offers the potential to reveal what others regard as private. This is more apparent as Internet technology gives the opportunity for data users to share or obtain data about individuals or corporations. In some cases, it may be of mutual benefit for two corporations (usually competitors) to share their data for an analysis task. However, they would like to ensure their own data remains private. In other words, there is a need to protect private knowledge during a data mining process. This problem is called Privacy Preserving Data Mining (PPDM).

The management of data for privacy has been considered in the context of releasing some information to the public while keeping private records hidden. It relates to issues in statistical databases as well as authorization and security access in databases. It is clear that hiding sensitive data by restricting access to it does not ensure complete protection. In many cases, sensitive data can be inferred from non-sensitive data based on some knowledge and/or skillful analysis.

The problem of protection against inference has been addressed in the literature of statistical databases since last decade. However, in the field of data mining and in particular for the task of association rules the focus has been more specific. Here, some researchers refer to the process of protection against inference as data disinfect. Data sanitization is defined as the process of making sensitive information in nonproduction databases safe for wider visibility. Others advocate a solution based on collaborators mining independently their own data and then sharing some of the resulting patterns. This second alternative is called rule disinfect. In this later case, a set of association rules is processed to block inference of so called sensitive rules.

This paper aims to contribute to the solution of two specific problems. First, the problem of sharing sensitive knowledge by sanitization (disinfect). Second, developing and improving algorithms for privacy in data mining tasks in scenarios which require multi-party computation.

2. Background Techniques

Data mining and Privacy

Data mining deals with large database which can contain sensitive information. It requires data preparation which can uncover information or patterns which may compromise confidentiality and privacy obligations. Advancement of efficient data mining technique has increased the disclosure risks of sensitive data. A common way for this to occur is through data aggregation.

Data aggregation is when the data are accrued, possibly from various sources, and put together so that they can be analyzed. This is not data mining privacy, but a result of the preparation of data before and for the purposes of the analysis. The threat to an individual's privacy comes into play when the data, once compiled, cause the data miner, or anyone who has access to the newly compiled data set, to be able to identify specific individuals, especially when originally the data were anonymous [2] and [3-5].

Privacy-preserving distributed data mining

A Distributed Data Mining (DDM) model assumes that the data sources are distributed across multiple sites. The challenge here is: how can we mine the data across the distributed sources securely or without either party disclosing its data to the others? Most of the algorithms developed in this field do not take privacy into account because the focus is on efficiency. A simple approach to mining private data over multiple sources is to run existing data mining tools at each site independently and combine the results. However, this approach failed to give valid results for the following reasons:

• Values for a single entity may be split across sources. Data mining at individual sites will be unable to detect cross-site correlations.

• The same item may be duplicated at different sites, and will be over-weighted in the results.

• Data at a single site is likely to be from a homogeneous population. Important geographic or demographic distinctions between that population and others cannot be seen on a single site.

Recently, research has addressed classification using Bayesian Networks in vertically partitioned data, and situations where the distribution is itself interesting with respect to what is learned. In recently proposed an efficient algorithm for vertically mining association rules. Finally, data mining algorithms that partition the data into subsets have been developed. However, none of this work has directly addressed privacy issues and concerns.

Classification of PPDM

PPDM can be classified according to different categories. These are

Data Distribution- The PPDM algorithms can be first divided into two major categories, centralized and distributed data, based on the distribution of data. In a centralized database environment, data are all stored in a single database; while, in a distributed database environment, data are stored in different databases. Distributed data scenarios can be further classified into horizontal and vertical data distributions. Horizontal distributions refer to the cases where different records of the same data attributes are resided in different places. While in a vertical data distribution, different attributes of the same record of data are resided in different places. Earlier research has been predominately focused on dealing with privacy preservation in a centralized database. The difficulties of applying PPDM algorithms to a distributed database can be attributed to: first, the data owners have privacy concerns so they may not willing to release their own data for others; second, even if they are willing to share data, the communication cost between the sites is too expensive [4] and [8].

Hiding Purposes - The PPDM algorithms can be further classified into two types, data hiding and rule hiding, according to the purposes of hiding. Data hiding refers to the cases where the sensitive data from original database like identity, name, and address that can be linked, directly
or indirectly, to an individual person are hided. In contrast, in rule hiding, the sensitive knowledge (rule) derived from original database after applying data mining algorithms is removed. Majority of the PPDM algorithms used data hiding techniques. Most PPDM algorithms hide sensitive patterns by modifying data [2] and [9].

Data Mining Tasks / Algorithms - Currently, the PPDM algorithms are mainly used on the tasks of classification, association rule and clustering. Association analysis involves the discovery of associated rules, showing attribute value and conditions that occur frequently in a given set of data. Classification is the process of finding a set of models (or functions) that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. Clustering Analysis concerns the problem of decomposing or partitioning a data set (usually multivariate) into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups.

Privacy Preservation Techniques - PPDM algorithms can further be divided according to privacy preservation techniques used. Four techniques – sanitation, blocking, distort, and generalization -- have been used to hide data items

for a centralized data distribution. The idea behind data sanitation is to remove or modify items in a database to reduce the support of some frequently used item sets such that sensitive patterns cannot be mined. The blocking approach replaces certain attributes of the data with a question mark. In this regard, the minimum support and confidence level will be altered into a minimum interval. As long as the support and/or the confidence of a sensitive rule lie below the middle in these two ranges, the confidentiality of data is expected to be protected. Also known as data perturbation or data randomization, data distort protects privacy for individual data records through modification of its original data, in which the original distribution of the data is reconstructed from the randomized data. These techniques aim to design distortion methods after which the true value of any individual record is difficult to ascertain, but "global" properties of the data remain largely unchanged. Generalization transforms and replaces each record value with a corresponding generalized value [7-9].

3. Related Works

Most methods for privacy computations use some form of transformation on the data in order to perform the privacy preservation. Typically, such methods reduce the granularity of representation in order to reduce the privacy. This reduction in granularity results in some loss of effectiveness of data management or mining algorithms. This is the natural trade-off between information loss and privacy. Some examples of such technique as:

Randomization method - The randomization technique uses data distortion methods in order to create private representations of the records .In this which noise is added to the data in order to mask the attribute values of records In most cases, the individual records cannot be recovered, but only aggregate distributions can be recovered. These aggregate distributions can be used for data mining purposes. Data mining techniques can be developed in order to work with these aggregate distributions. Two kinds of perturbation are possible with the randomization method:
➔Additive Perturbation - In this case, randomized noise is added to the data records. The overall data distributions can be recovered from the randomized records. Data mining and management algorithms re designed to work with these data distributions.
➔ Multiplicative Perturbation- In this case, the random projection or random rotation techniques are used in order to perturb the records [7].

The k-anonymity model and l-diversity-The *k*-anonymity model was developed because of the possibility of indirect identification of records from public databases. This is because combinations of record attributes can be used to exactly identify individual records. In the *k*-anonymity method, the granularity of data representation is reduced with the use of techniques such as generalization and suppression. This granularity is reduced sufficiently that any given record maps onto at least *k* other records in the data. The *l*-diversity model was designed to handle some weaknesses in the *k*-anonymity model since protecting identities to the level of *k*-individuals is not the same as protecting the corresponding sensitive values, especially when there is homogeneity of sensitive values within a group [3-6].

Distributed privacy preservation- In many cases, individual entities may wish to derive aggregate results from data sets which are partitioned across these entities. Such partitioning may be horizontal (when the records are distributed across multiple entities) or vertical (when the attributes are distributed across multiple entities). While the individual entities may not desire to hare their entire data sets, they may consent to limited information sharing with the use of an ariety of protocols. The overall effect of such methods is to maintain privacy for each individual entity, while deriving aggregate results over the entire data.

Downgrading Application Effectiveness - In many cases, even though the data may not be available, the output of applications such as association rule mining, classification or query processing may result in violations of privacy. This has lead to research in downgrading the effectiveness of applications by either data or application modifications [9-10].

4. Proposed Techniques

In this paper firstly we propose a new heuristic algorithm that improves the privacy of sensitive knowledge (as itemsets) by blocking more inference channels. We demonstrate the efficiency of the algorithm. Secondly we propose two techniques which are item count and increasing cardinality based on item-restriction that hide sensitive itemsets. Finally we propose an efficient protocol that allows parties to share data in a private way with no restrictions and without loss of accuracy.

Association Rule

The task of mining association rules over market basket data is considered a core knowledge discovery activity. Association rule mining provides a useful mechanism for discovering correlations among items belonging to customer transactions in a market basket database. Let D be the database of transactions and J = {J1, ..., Jn} be the set of items.

A transaction T includes one or more items in J (i.e., T □ J). An association rule has the form X → Y , where X and Y are non-empty sets of items (i.e. X □ J, Y □ J) such that X ∩ Y = Ø. A set of items is called an itemset, while X is called the antecedent. The support sprtD(x) of an item (or itemset) x is the percentage of transactions from D in which that item or itemset occurs in the database. In other words, the support sprt () of an association rule X → Y is the percentage of transactions T in a database where X ∪ Y □ T. The confidence or strength c for an association rule X → Y is the ratio of the number of transactions that contain X ∪ Y to the number of transactions that contain X. An itemset X □ J is frequent if at least a fraction sprt() of the transaction in a database contains X. Frequent itemsets are important because they are the building blocks to obtain association rules with a given confidence and support.

Proposed New Heuristic Algorithm

Both data-sharing techniques and pattern-sharing techniques face a challenge. That is, blocking as much inference channels to sensitive patterns as possible. As inputs a database D and a privacy support threshold σ. Let F(D, σ) be the set of frequent itemsets with support σ in D. We are also given B □ F as the set of sensitive itemsets that must be hidden based on some privacy/security policy. The set Sup(B) = B ∪ {X □ F|□b □ B and b □ X} is considered also sensitive because sensitive itemsets cannot be subsets of frequent non-sensitive itemsets. The task is to lower the support of the itemsets in B below σ and keep the impact on the non-sensitive itemsets A = F(D, σ)\Sup(B) at a minimum.

The new heuristic algorithm improves the privacy of sensitive knowledge (as itemsets) by blocking more inference channels in two phases. In Phase 1, the input to the privacy process is defined. In particular, the user provides a privacy threshold σ. The targeted database D is mined and the set F(D, σ) of frequent itemsets is extracted. Then, the user specifies sensitive itemsets B. The algorithm removes all supersets of a set in B because if we want to hide an itemset, then the supersets of that itemset must also be hidden (in a sense we find the smallest B so that Sup(B) = Sup(B′), if B′ was the original list of sensitive itemsets). In Phase 1, we also compute the size (cardinality) kbk and set TD(b) □ T of transactions in D that support b, for each frequent itemset b □ B. Then, we sort the sensitive itemsets in ascending order by cardinality first and then by support (size-k contains the frequent itemsets of size k where k is the number of items in the itemset). One by one, the user specifies lb for each sensitive itemset. Users may specify a larger lb value for higher sensitivity of that itemset. Note that as a result of Phase 1 we have a data structure that can, given a sensitive itemsets b □ B, retrieve TD′(b), kbk, and sprtD′(b) (initially, D′ = D, TD′(b) = TD(b), and sprtD′(b) = sprtD(b)).

Phase 2 applies the heuristic algorithm

Procedure heuristic algorithm

Begin
1. for each b □ B
1.1 while Condition 1 is not satisfied for b
1.1.1 Greedily find frequent 2-itemset b′ □ b.
1.1.2 Let T (b′,A) the transactions in TD′(b′) that affects the minimum number of 2- temsets.
1.1.3 Set the two items in b′ to nil in T(b′,A).
1.1.4 Update TD′(b), kbk, and sprtD′(b)
1.2 end //while
2. end //for
End

The algorithm takes as input the set of transactions in a targeted database D, a set B of frequent sensitive itemsets with their labels lb and the target support t, and a set A of frequent non-sensitive itemsets. Then, the algorithm takes the sensitive itemsets one by one. The strategy we follow in the experiments is to start with the sensitive itemsets with the smallest cardinality.

If there are more than one b □ B with the same cardinality we rank them by support, and select next the itemset with highest support1. The algorithm enters a loop where we follow the same methodology of the algorithm to greedily find a 2-itemset to eliminate items from transactions. This consists of finding the (k − 1)-frequent itemset of highest support that is a subset of a current k-itemset. We start with b □ B as the top k-itemset; that is, the algorithm finds a path in the lattice of frequent itemsets, from b □ B to a 2-itemset, where every child in the path is the smaller proper subset with highest support among the proper subsets of cardinality one less. Then, the database of transactions is updated with items removed from those specific transactions.

Techniques for Hiding Sensitive Itemsets

Many privacy preserving data mining algorithms attempt to hide what database owners consider as sensitive. Specifically, in the association-rules domain, many of these algorithms are based on item-restriction methods; that is, removing items from some transactions in order to hide sensitive frequent itemsets. Frequent itemsets are important because they are the building blocks to obtain association rules with a given confidence and support. Typically,

algorithms to find frequent itemsets use the anti-monotonicity property, and therefore, we must find first all frequent itemsets of size k before proceeding to find all itemsets of size k + 1. We refer to the set of all frequent itemsets of size k as depth k. We assume a context where parties are interested in releasing some data but they also aim to keeping some patterns private. We identify patterns with frequent itemsets. Patterns represent different forms of correlation between items in a database. Sensitive itemsets are all the itemsets that are not to be disclosed to others. While no sensitive itemset is to become public, the non-sensitive itemsets are to be released. One could keep all itemsets private, but this would not share any knowledge. The aim is to release as many non-sensitive itemsets as possible while keeping sensitive itemsets private. This is an effort to balance privacy with knowledge discovery. It seems that discovery of itemsets is in conflict with hiding sensitive data. Sanitizing algorithms at Level 1 take (as input) raw data or database D and modify it to construct (as output) a database D′ where sensitive itemsets are hidden. The alternative scenario at Level 3 is to remove the sensitive itemsets from the set of frequent itemsets and publish the rest. This scenario implies that a database D does not need to be published. However, this prevents data miners from applying other discovery algorithms of learning models to data, and therefore, reduces the options for knowledge discovery. Pattern-sharing algorithms are Level 3 algorithms and are also called rule restriction-based algorithms. Here, parties usually share a set of rules after removing the sensitive rules. Thus, parties avoid sharing data and it has been argued that this approach reduces the hazards of concluding any sensitive rules or discovering private data. However, they are typically over-protected. They correspond to the approach in statistical databases where data is released from a data generator based on a learned model. While the learned model is based on the original data, users of the generated data can only learn the generator model and therefore may miss many patterns in the data.

Our two new heuristics

Our two new heuristics focus on building an itemset g so that attacking items in g would affect the support of sensitive itemsets. We first describe the process of attacking items from g J. Note that g is not necessarily sensitive itself; that is, we do not require g B. In fact, it may be that g is not even frequent. We describe two ways of selecting transactions to attack; these will be called methods. We also describe two ways of building the set g, and we refer to these as techniques. We present the methods first.

The methods

The methods presented here determine what item and what transaction to attack given an itemset g J. The two methods hide one sensitive itemset related to itemset g. How sensitive itemsets are related to g will become clear in the next subsection. For now, suffice it to say that g will contain items that have high support in sensitive itemsets (and hopefully low support in non-sensitive itemsets). In both methods, we attack an item x g until one itemset b B becomes hidden. Then, a new itemset g is selected. We perform the attack on sensitive itemsets by attacking the item x g with the highest support. We determine the list of transactions Lg D that support g (i.e. Lg = {T D|g T}. We remove the item x from the transactions T Lg until the support of some b B is below the required privacy support threshold. The difference between our two methods is that the transactions T Lg are sorted in two different orders. Thus, which transactions are attacked and which ones are left untouched, even though they include x, is different for the two methods.

Method 1 sorts the transactions T Lg in ascending order based on kT \ gk (the number of items in T not in g). Notice that kT \gk can be zero. The guiding principle is that if kT \ gk is small, then removing x from T would impact the support of sensitive itemsets but rarely the support of other itemsets, thus non-sensitive itemsets will remain mostly untouched.

Method 2 sorts the transactions T Lg in ascending order based on k(Y ∩ T) \ gk (recall that Y is all items in A and A is all the non-sensitive frequent itemsets). Again, k(Y ∩ T) \ gk can be zero. The second method makes even a stronger effort to make sure that those transactions that are attacked have very few items involved in non-sensitive itemsets.

How to select the itemset g in the techniques

We present here the techniques to prioritize the order in which itemsets b B are attacked so that their support is below t. The techniques build the itemset g used by the methods described before.

Technique 1 item count

➔ First sort the items in X based on how many itemsets in B contain the item. Recall that X J is all items in the sensitive itemsets. Thus, this sorts the items in X is according to

$$\text{Item\_Count}(x) = \sum \|\{x\} \cap b\| \text{ where } b \in B$$

Let x0 X be the item with the highest count. If we have more than one item with the same Item Count(x) value, we select the item with the highest support. If the tie persists, we select arbitrarily. Then,

g = U b, Where b□B and x0□b.

That is, g is the union of sensitive itemsets that include the item x0.

➔ If the construction of g results in hiding a sensitive itemset (using either Method 1 or Method 2), then the hidden itemset is removed from B.
(a) If B is empty, then the technique stops.
(b) Otherwise the technique is applied again, building a new g from the very beginning.
3. If the construction of g does not result in hiding a sensitive itemset, we find x □ g with lowest support, and replace g with g \ {x}. We then reapply Method 1 or Method 2 again.

Technique 2 increasing cardinality
The technique first sorts the itemsets in B based on their cardinality. Starting from the smallest cardinality, the technique selects an itemset g that in this case is an itemset b □ B. The technique can then have a Method 1 variant or a Method 2 variant. If we have more than one itemset of the same cardinality, we attack the itemset with the highest support. This technique also iterates until all sensitive itemsets are hidden. Every time an itemset b is hidden, a new g is calculated. Note that because g □ B, the entire application of Method 1 or Method 2, must result in g being hidden, and therefore a sensitive itemset is hidden.

Justification of two techniques
We need to mention that in the field of association-rule mining, we could not find any existing methodology that discusses the best way to hide a set of itemsets using item-restriction methods.
Our first technique (Item Count) is based on the idea that attacking the item with the highest count (amongst the sensitive itemsets) has two advantages: a) an item with the highest count means it is shared by most of the sensitive itemsets and thus, attacking that item reduces the support of all these sensitive itemsets that share that item at the same time, and b) the item with the highest count usually has a high support and attacking an item with a high support reduces the chances that frequent non-sensitive itemsets might be hidden as a side effect. Notice that we supposed here that the frequent itemset that includes an item with a high support usually has a high support compared to the other frequent itemsets in the same level. Of course, that might not be the case always.
The second technique (Increasing Cardinality) is based on the idea that hiding the sensitive itemsets in the lower levels will lead to hiding all the sensitive item-sets in the higher levels that include the attacked itemset in the lower level. in the field of association-rule mining, there is no existing methodology that discusses the best ways to hide a set of itemsets using item-restriction methods.

Secure Multi-party Computation
Today, most of the sensitive data maintained by organizations is not encrypted. This is especially true in database environments, where sensitive data or sensitive patterns are usually included. In the past, parties who sought privacy were hesitant to implement database encryption because of their high cost, complexity, and performance degradation. Recently, with the ever growing risk of data theft and emerging legislative requirements, parties have become more willing to compromise efficiency for privacy.
While traditional ways of database encryption were indeed costly, very complex and created significant performance degradation, fortunately, there are now newer, better ways to handle database encryption. In this section, we demonstrate how database encryption can allow parties to share data for the common good without jeopardizing the privacy of each party.
Computation tasks based on data distributed among several parties have privacy implications. The parties could be trusted parties, partially trusted parties or even competitors. To perform such a computation, one of the parties must know the input of all the other parties. When the task is a data mining task, the absence of a trusted third party or a specific mechanism to exchange data in a way that does not reveal sensitive data (for example, by adding noise or perturbing the data so that it does not affect the final desired result), is a challenge for privacy-preserving data mining. When two or more parties want to conduct analysis based on their private data and each party wants to conceal their own data from the others, the problem falls into the area known as Secure Multi-Party Computation (SMC).

Public-key cryptosystems (asymmetric ciphers)
A cipher is an algorithm that is used to encrypt plaintext into cipher text and vice versa (decryption). Ciphers are said to be divided into two categories: private key and public key.
Private-key (symmetric key) algorithms require a sender to encrypt a plaintext with the key and the receiver to decrypt the ciphertext with the key. A problem with this method is that both parties must have an identical key, and somehow the key must be delivered to the receiving party.
Public-key (asymmetric key) algorithms use two separate keys: a public key and a private key. The public key is used to encrypt the data and only the private key can decrypt the data. A form of this type of encryption is called RSA

(discussed below), and is widely used for secured websites that carry sensitive data such as username and passwords, and credit card numbers.

Problem statement and solution techniques

Let P = {P0, . . . , Pn} be a set of N parties where |N| ≥ 3. Each party Pi has a database DBi. We assume that parties running the protocol are semi-honest. The goal is to share the union of DBi as one shuffled database DBComp = Sn i=0 DBi and hide the link between records in DBComp and their owners.

Our protocol employs a public-key cryptosystem algorithm on horizontally partitioned data among three or more parties. In our protocol, the parties can share the union of their data without the need for an outside trusted party. The information that is hidden is what data records where in the possession of which party. Our protocol is described for one party as the protocol driver. We call this first party Alice.

Step1: Alice generates a public encryption key kPA. Alice makes kPA known to all parties (for illustration we use another two parties called Bob and Vismay).

Step2: Each party (including Alice) encrypts its database DBi with Alice's public key. This means the encryption is applied to each row (record or transaction) of the database. Parties will need to know the common length of rows in the database. We denote the result of this encryption as kPA(DBi). Note that, by the properties of public cryptosystems, only Alice can decrypt these databases.

Step3: Alice passes her encrypted transactions kPA(DB1) to Bob. Bob cannot learn Alice's transactions since he does not know the decryption key.

Step4: Bob mixes his transactions with Alice's transactions. That is, he produces a random shuffle of kPA(DB1) and kPA(DB2) before passing all these shuffled transactions to Vismay.

Step5: shuffles the transactions kPA(DB3) to the transactions received from Bob.

Step6: The protocol continues in this way, each subsequent party receiving a database with the encrypted and shuffled transaction of all previous parties in the enumeration of the parties. The i-th party mixes randomly his encrypted transactions kPA(DBi) with the rest and passes the entries shuffled transaction to the (i + 1)-th party.

Step7: The last party passes the transactions back to Alice.

Step8: Alice decrypts the complete set of transaction with her secret decrypt key. He/ She can identify her own transactions. However, Alice is unable to link transactions with their owners because transactions are shuffled.

Step9: Alice publishes the transactions to all parties. If the number of parties is N, then N − 1 of the parties need to collude to associate data to their original owners.

It may seem that the protocol above is rather elaborate, for the seemingly simple task of bringing the data of all parties together while removing information about what record (transaction) was contributed by whom. We now show how to apply this protocol to improve on the privacy-preserving data mining of association rules.

The task of mining association rules over market basket data is considered a core knowledge discovery activity since it provides a useful mechanism for discovering correlations among items belonging to customer transactions in a market basket database.

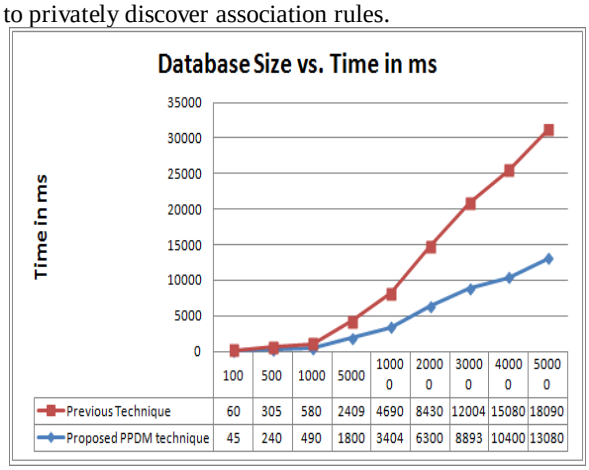
5. Results Analysis

We proposed a flexible and easy-to-implement protocol for privacy-preserving data sharing based on a new heuristic algorithm, item count and increasing cardinality based on item-restriction that hide sensitive itemsets and a private way with no restrictions and without loss of accuracy by public-key cryptosystem. The protocol is efficient in practical settings and it requires less machinery than previous approaches (where commutative encryption was required). This protocol ensures that no data can be linked to a specific user. The protocol allows users to conduct private mining analyses without loss of accuracy. Our protocol works under the common and realistic assumption that parties are semi-honest, or honest but curious, meaning they execute the protocol exactly as specified, but they may attempt to infer hidden links and useful information about other parties. A privacy concern of this protocol is that the users get to see the actual data. But previous research has explored whether parties are willing to trade off the benefits and costs of sharing sensitive data. The results of this research showed that parties are willing to trade-off privacy concerns for economic benefits. There are several issues that may influence practical usage of the presented protocol. While the

protocol is efficient, it may be still a heavy overhead for parties who want to share huge multimedia databases. This can be solved with slightly more cost; if there are N parties, each party plays the data distributor with 1/N share of the data, and we conduct N rounds.

We also showed that our protocol is efficient, and especially more efficient than the protocol presented. We showed that by using our protocol not to share entire local databases, but local itemsets with their local support values, we can mine privately association rules on the entire database without revealing any single transaction to other parties. We showed that the overhead to security is reduced as we do not need commutative encryption1, the sharing process was reduced from 6 steps to 4 steps, and the protocol is more secure as we share fewer local frequent itemsets that may not result in global frequent itemsets. Privacy concerns may discourage users who would otherwise participate in a jointly beneficial data mining task.

The proposed efficient protocol that is allows parties to share data in a private way with no restrictions and without loss of accuracy. Our method has the immediate application that horizontally partitioned databases can be brought together and made public without disclosing the source/owner of each record. At another level, we have an additional benefit that we can apply our protocol to privately discover association rules.



	100	500	1000	5000	10000	20000	30000	40000	50000
Previous Technique	60	305	580	2409	4690	8430	12004	15080	18090
Proposed PPDM technique	45	240	490	1800	3404	6300	8893	10400	13080

Figure 1 Plot of time requirements for PPDM.

Our protocol is more efficient than previous methods where to privately share association rules, the requirements like each party can identify only their data, no party is able to learn the links between other parties and their data, no party learns any transactions of the other parties' databases.

6. Conclusion

In this paper the heuristic algorithm proposed for solution to the forward-attack inference problem. The heuristic algorithm added a very important feature that did not exist in the other algorithms. The heuristic algorithm allows users to customize their own depths of security. The algorithm adds this flexibility with a reasonable cost and blocking more inference channels. The order in which the sensitive itemsets should be hidden may influence the results of the heuristic algorithm.

Then we have presented the two techniques for hiding sensitive itemsets based on item-restriction that hide sensitive itemsets. We have also shown that rather simple new data structures implement these techniques with acceptable cost since we avoid the expensive steps of mining the database several times during the sanitization process. We have implemented the code needed to test both techniques and conducted the experiments on a real data set. Our results show that both techniques have an effect on the frequent non-sensitive itemsets, but Technique 1 (Item Count) has about less effect than Tech- unique 2 (Increasing Cardinality). Also, using Method 2 rather than Method 1 lowers the effect on the frequent non-sensitive itemsets.

Finally we have proposed a flexible and easy-to-implement protocol for privacy-preserving data sharing based on a public-key cryptosystem. The protocol is efficient in practical settings and it requires less machinery than previous approaches (where commutative encryption was required). This protocol ensures that no data can be linked to a specific user. The protocol allows users to conduct private mining analyses without loss of accuracy. Our protocol works under the common and realistic assumption that parties are semi-honest, or honest but curious, meaning they execute the protocol exactly as specified, but they may attempt to infer hidden links and useful information about other parties. A privacy concern of this protocol is that the users get to see the actual data. But previous research has explored whether parties are willing to trade off the benefits and costs of sharing sensitive data. The results of this research showed that parties are willing to trade-off privacy concerns for economic benefits. There are several issues that may influence practical usage of the presented protocol. While the protocol is efficient, it may be still a heavy overhead for parties who want to share huge multimedia databases.

References

[1.] MohammadReza Keyvanpour and Somayyeh Seifi Moradi, "Classification and Evaluation the Privacy Preserving Data Mining Techniques by using a Data Modification–based Framework", International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 2 Feb 2011, pp. 862-870.

[2.] Lambodar Jena, Narendra ku.Kamila, Ramakrushna Swain,Tanmay ku. Das, "Secure Privacy Preserving Association Rule Data Mining: A Cryptographic Approach", International Journal of Computer Application, Issue 1, Volume 2 (December' 2011), pp. 1-14.

[3.] S.Vijayarani and S.Nithya, "Sensitive Outlier Protection in Privacy Preserving Data Mining", International Journal of Computer Applications (0975 – 8887) Volume 33– No.3, November 2011, pp. 19-27.

[4.] Hemanta Kumar Bhuyan, Narendra Kumar Kamila and Sanjit Kumar Dash, " An Approach for Privacy Preservation of Distributed Data in Peer-to-Peer Network using Multiparty Computation", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 2, July 2011, pp. 424-429.

[5.] B. V. Harita and P.G.Chaitanya, "Preserving the Privacy and Sharing the Data Using Classification on Perturbed Data", International Journal of Engineering Science and Technology (IJEST), Vol. 3 No.12 December 2011, pp. 8334-8341.

[6.] Deepika Saxena, "Privacy of Data, Preserving in Data Mining", International Journal of Scientific & Engineering Research Volume 2, Issue 3, March-2011, pp. 1-5.

[7.] Dharminder Kumar and Deepak Bhardwaj, " Rise of Data Mining: Current and Future Application Areas", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 1, September 2011, pp. 256-260.

[8.] Rajesh Shrivastava, Rashmi Awasthy, Bharat Solanki, "New Improved Algorithm for Mining Privacy - Preserving Frequent Itemsets", International Journal of Computer Science & Informatics, Volume-1, Issue-1,2011, pp. 1-7.

[9.] GAYATRI NAYAK and SWAGATIKA DEVI, "A SURVEY ON PRIVACY PRESERVING DATA MINING: APPROACHES AND TECHNIQUES", International Journal of Engineering Science and Technology (IJEST), Vol. 3 No. 3 March 2011, pp. 2127-2133.

[10.] Alka Gangrade, Durgesh Kumar Mishra, Ravindra Patel, "Classification Rule Mining through SMC for Preserving Privacy Data Mining: A Review", 2009 International Conference on Machine Learning and Computing IPCSIT vol.3, 2011, Singapore, pp. 431-434.