

# An Architecture for Privacy-preserving Mining of Client Information

Murat Kantarcioglu\*

Jaideep Vaidya †

Department of Computer Sciences  
Purdue University  
1398 Computer Sciences Building  
West Lafayette, IN 47906, USA  
Email: {kanmurat, jsvaidya}@cs.purdue.edu

## Abstract

Due to privacy concerns, clients of some services may not want to reveal their private information. Even in these situations, data mining is feasible without sacrificing user privacy. Prior approaches to this problem generally trade off accuracy for security, without giving provable bounds on security. Alternatives to the randomization technique are required to enable accurate data mining while strictly preserving privacy. In this paper, we present a general architecture that enables privacy-preserving mining of client information. Under some reasonable assumptions, we show that our methods are secure, while maintaining the accuracy of the results.

*Keywords:* Privacy, Data Mining, Security

## 1 Introduction

Data Mining is a well known technique for extracting useful information from huge amounts of data. The algorithms generally assume that necessary information from all clients is gathered at one central site. Due to increasing privacy concerns, all clients may not trust a single central party with their information. Indeed, they may not want to reveal any information to any one party. In certain sectors like health care, even if a certain party has all the data for some clients, it may be legally restricted from revealing this information to any other party. Therefore, privacy preserving mining is becoming an increasingly important problem.

It is obvious that data cannot be revealed without any modification. The straightforward approach to solving this problem is to add noise to the data, and approximately remove the effect of this noise from the results. The accuracy of the results obviously depends on the amount of noise added. While increasing the noise improves the security, it usually reduces the overall accuracy. Worse, in some data mining environments adding noise to the data is not an option. For example, consider mining health care data to find the efficacy of certain medicines. Here, both privacy of data and accuracy of results are extremely important.

Even allowing addition of noise, extraneous information collected from other sources could seriously reduce the security of the noise addition approach. Consider the following scenario in which salary of users is protected information. Assume that noise uniformly

distributed over the range  $-20k$  to  $20k$  is added to the salary information. Also, after the addition of noise an instance is found with the salary being  $60k$ . Now, if from some other information (e.g. IP address), we find that this particular client is from West Lafayette, IN. Assuming we know that West Lafayette is a city that has a high percentage of students with income below  $20k$ , while Professors easily earn more than  $60k$ . So combining all the available information leads us to believe that with high probability, the particular client is actually a Professor at Purdue University, with the age being above 30. It is clear from this example that noise addition approach is not necessarily secure in the global sense (even though exact estimation of attribute values may not be possible). In the next section we detail the various reasons requiring a different solution from the data perturbation model.

Individual parties are interested in protecting their own local data, while still being able to infer useful results from the combined data of all parties. Though, theoretically, Secure Multiparty Computation enables just this, in practical terms, issues regarding consistency of data, software engineering issues etc. exist. A typical SMC protocol consists of several rounds of local computation interspersed with communication between the interested parties. All of the parties need to be online during the whole protocol. No party would like to be permanently "on call". As far as they are concerned, it would be better if the data could be shipped off in some form to another site where the actual mining of data takes place.

**Organization.** Section 2 motivates the need for a new architecture and also outlines the related work in the area. We give a detailed description of the architecture in Section 3. Section 4 gives an example of using the architecture to do a common data mining task. Finally, we conclude the paper and give directions for future work in Section 5.

## 2 Motivation and Background

We will now motivate this work by describing some of the potential problems with the randomization approach and then discuss the related work in the area.

### 2.1 Security

Generally the privacy of a randomization model is calculated on the basis of the original data distribution and the perturbation function. However, this definition is insufficient in protecting privacy. Realistically, there can be external knowledge, which when coupled with data revealed in computation, improves the estimates of actual data values (Agrawal &

Aggarwal 2001). Consider the following hypothetical example. Assume that there exists an attribute  $Y$ , which is distorted by a uniform random variable ranging over  $[-2, +2]$ . Let the distorted values be represented by  $Y'$ . Now, if we observe the distorted value for some record, i.e.  $Y'_i$  (perturbed value of attribute  $Y$  of the  $i^{\text{th}}$  record), has the value 5, this implies  $Y_i$  (the original value)  $\in [3, 7]$ . A key component of the privacy preserving data mining method of (Agrawal & Srikant 2000), is to approximately reconstruct the original distribution.

So even if this obfuscation seems sufficient, from the approximate reconstruction of the distribution of  $Y$  we may find out that  $Pr\{3 \leq Y \leq 4\} \approx 0$ . From this we can conclude that  $Y_i \in [5, 7]$ . Even worse we could possibly learn more from the joint distributions of some attributes. For example, assume that we infer a joint probability distribution function for  $Y$  and  $T$ . Using this joint distribution, we may learn that  $Pr\{6 \leq Y \leq 7 | 0.5 \leq T \leq 1.0\} \approx 0.9$ . Given that  $Pr\{0.5 \leq T_i \leq 1.0\} \approx 0.9$ , we can infer that  $Pr\{6 \leq Y_i \leq 7\} \approx 0.8$ . This unexpectedly leads to a major breach in security. Since, it is rather difficult to anticipate all possible additional information which could be revealed at some point, it is difficult to select the optimum range for the additive noise, and it is highly probable that security breaches may occur.

### 2.1.1 Approximate information is sufficient for breaching security

In most cases, perfectly accurate information is unnecessary for breaching security. Assume that we can divide an attribute coarsely into some categories. Now, it may be sufficient to map data into one of those categories, which is much easier. (i.e. For certain advertising purposes, it is sufficient to find out that person A falls in the higher income bracket, with salary above 60k per annum, without revealing the exact salary). Knowing that with high probability this person is in the high income bracket, the marketing department of some company may send this person many e-mail promotions in order to sell some expensive merchandise.

## 2.2 Problems in getting accurate Data Mining Results

Ultimately, in randomization methods, there is a trade off between accuracy of data mining results and data security. Hence, these methods may not be suitable for mining data in situations requiring both high accuracy and high security. For example, consider the mining of health data. With this kind of critical data both security and accuracy is important.

Randomization approaches work quite well for mining frequent itemsets and general classification problems. However, an increasingly important class of problems involves accurate classification of rare classes. For rare classes, it is difficult to quantify the effect of data perturbation on the classifier built. The effect of a small change in their probability distribution could lead to a comparatively large change in the classifier built. It would be useful to be able to quantify the difference between the classifier built with the original data and classifier built with the perturbed data, but no tight guarantees currently exist. Hence, investigating other approaches is a worthwhile direction for research.

## 2.3 Related Work

Recently there has been a spate of work addressing privacy preserving data mining. (Agrawal & Srikant 2000) uses the data perturbation technique for classification. (Agrawal & Aggarwal 2001) extends this work, giving better privacy definitions and using the EM algorithm for distribution reconstruction. (Evfimievski, Srikant, Agrawal & Gehrke 2002) also extend the work to do association rule mining on categorical attributes. While the prior papers discuss the modification of data, (Lindell & Pinkas 2000) use the orthogonal technique of multiparty computation technique to enhance the classification algorithm *ID3*, into a privacy preserving one. (Vaidya & Clifton 2002) addresses privacy issues in association rule mining for vertically partitioned data. (Rizvi & Haritsa 2002) also apply the data distortion approach to boolean association rule mining. One interesting feature of this work, is the flexible definition of privacy; e.g. The ability to correctly guess a value of '1' from the distorted data can be considered a greater threat to privacy than correctly learning '0'. (Kantarcioglu & Clifton 2002) address privacy issues in association rule mining for horizontally partitioned data.

There is a whole body of work addressing Secure Multiparty Computation (Yao 1986, Goldreich 2000), but much of this work is theoretical and needs to be applied to the data mining domain.

## 3 Description of the architecture

In this section we describe an architecture where secure multi-party techniques developed in the cryptography domain can be easily used for data mining purposes. In the next section, we show how association rule mining can be done in this architecture.

Although Secure Multiparty protocols can be used directly to implement totally secure data mining algorithms, a key requirement is that all individuals should stay online and participate in a complex SMC protocol. Also, for any new data mining task, an entire SMC style protocol must be executed, and the information providers have to remain online for this. Even assuming that all of the information providers are online, the communication complexity of the current SMC style data mining algorithms (i.e (Lindell & Pinkas 2000), (Kantarcioglu & Clifton 2002)) cannot be effectively scaled to a large number of participants. Therefore it is clear that current SMC protocols cannot be directly used and need to be enhanced for individual privacy-preserving data mining.

Also, it is quite possible that all clients cannot be online during the entire secure distributed data mining process. To cater for such situations, we would like to use some kind of intermediate parties where the collected information can be stored securely. To achieve that, we add two more parties different from the sites collecting the original information. We assume that these parties are *non-colluding* and *semi-honest*. (For a detailed description of the semi-honest model, refer to (Goldreich 2000)) In effect, all parties correctly follow the protocols, but then are free to use whatever information they see during the execution of the protocols in any way. While the assumption of non-colluding parties may seem unnatural at first, it seems to make sense upon further thought. In fact, in real life, this situation often occurs when using non technical solutions to problems. e.g. When people hire a lawyer, they assume that lawyers is not colluding with other related parties against them. It is assumed that e-bay is not colluding with the seller

of the auctioned goods against the potential buyers. In fact, there are legal safeguards against this. Companies assume that their consultants do not collude with their competitors.

The function of the sites in this architecture can be summarized as below:

- **Original Site:(OS)** This is the site that collects the information and will learn the final result of the data mining process.
- **Non-Colluding Storage Site:(NSS)** This site stores the shared part of the user information.
- **Processing Site:(PS)** Although this site can be eliminated, we use this site to do the data mining efficiently.

To simplify the explanation, let us assume that only one bit of information is collected from every user. However, the process used is still generally applicable.

Assume that some unique random number is assigned to a user when the user first connects to the original site (assume that number is  $i$ ). Now, the user must send exactly  $(i, x_i)$  to the information collecting site. However this would reveal the  $x_i$  to that site. To avoid this, user will send  $(i, \bar{x}_i = x_i \oplus r_i)$  to the original site and  $(i, r_i)$  to the non-colluding storage site. Note that neither the original site nor the non-colluding storage site will be able to predict the  $x_i$ . The Non-colluding storage site sees only a random number, and since the value the Original Site sees is XOR-ed  $\oplus$  with a random variable, it is indistinguishable from random. Given any data mining task ( $f$ ) defined on  $X = [x_1, x_2, \dots, x_n]$ , it suffices to evaluate  $f(\bar{X} \oplus R) = f(X)$  since  $R = [r_1, r_2, \dots, r_n]$  and  $\bar{X} \oplus R = [\bar{x}_1 \oplus r_1, \bar{x}_2 \oplus r_2, \dots, \bar{x}_n \oplus r_n]$ . It is a known fact that with the assumption of existence of trapdoor permutations (RSA is assumed to be a trapdoor permutation), any functionality  $g$ , ( $g : \{0, 1\}^* \times \{0, 1\}^* \mapsto \{0, 1\}^* \times \{0, 1\}^*$ ) can be evaluated privately in the semi-honest model. (Goldreich 2000) Since the initial XOR operation can be easily represented as a circuit, given functionality  $f$ , we can define a functionality  $g(X, R) = f(\bar{X} \oplus R)$ . Using the above fact, we can conclude that

**Theorem 3.1** *In this architecture, any data mining functionality can be evaluated privately without revealing any information other than the final data mining result.*

**Proof Sketch.** *Since each user XORs his information with a random sequence, the result is indistinguishable from a random sequence. This is equivalent to using a one time bit pad which is known to be perfectly secure in the information theoretic sense. Secure Multiparty Computation techniques can be used to securely compute any functionality. If we assume that the desired functionality is  $f$ , we can compose a new secure functionality  $g$  which gives us the desired result.  $g$  can be derived from  $f$  as described above.  $g$  can now be securely computed using the SMC techniques without revealing any information in the semi-honest model.*

## 4 Individual Privacy-preserving Mining of Association Rules

In this section we give an algorithm for privacy preserving association rule mining. Our main goal is to preserve the privacy of the individual users and leak

as little information as possible about the final association rules to *Processing site* (PS) and *Non-colluding Storage Site* (NSS). In order to show that privacy of the individual users is protected, it suffices to show that *original site* (OS) learns nothing to differentiate between any individual users. Thus, any partial information can be revealed, as long as it does not help the *original site* in differentiating between two users. Note that this is somewhat different from the definition of privacy in secure multi-party computation in which nothing other than the final result is allowed to be revealed.

Also we will show that NSS will learn nothing and PS will only learn an upper bound on support counts.

### 4.1 Brief Description of Association Rule Mining

Association rule mining has become very popular in the past few years. It has many applications from market basket data analysis to intrusion detection. The problem can be defined as follows: (Agrawal, Imielinski & Swami 1993) Let  $I = \{i_1, i_2, \dots, i_n\}$  be a set of items. Let  $DB$  be set of transactions, where each transaction  $T$  is an itemset such that  $T \subseteq I$ . Given an itemset  $X \subseteq I$ , a transaction  $T$  contains  $X$  if and only if  $X \subseteq T$ . An association rule is an implication of the form  $X \Rightarrow Y$  where  $X \subseteq I, Y \subseteq I$  and  $X \cap Y = \emptyset$ . The rule  $X \Rightarrow Y$  has support  $s$  in the transaction database  $DB$  if  $s\%$  of transactions in  $DB$  contain  $X \cup Y$ . The association rule holds in the transaction database  $DB$  with confidence  $c$  if  $c\%$  of transactions in  $DB$  that contain  $X$  also contain  $Y$ . An itemset  $X$  with  $k$  items called  $k$ -itemset. The problem of mining association rules is to find all rules whose support and confidence are higher than certain user specified minimum support and confidence.

In this simplified definition of the association rules, missing items, negatives and quantities are not considered. In this respect, transaction database  $DB$  can be seen as 0/1 matrix where each column is an item and each row is an transaction. In this paper, we will mainly deal with this view of association rules.

### 4.2 An individual privacy-preserving protocol for support count

In our architecture, we will assume that the user information is stored as a 0/1 matrix. We make this assumption to have consistent transaction sizes, since different itemset sizes may reveal some information about the client shopping habits. Client  $i$ 's information consists of a  $m$  bit string that has the  $j^{th}$  bit as 1 if  $j^{th}$  item is in the transaction. Client  $i$  randomly generates a  $m$  bit string  $R \in \{0, 1\}^m$  and sends  $(i, X \oplus R)$  to original site, and sends  $(i, R)$  to non-colluding storage site. Note that  $(X \oplus R) \oplus R = X$ .

In this paper, we will only show how the support of the any given itemset  $X$  can be found without revealing any information about the individual clients. Clearly, using this as a sub-protocol, the original algorithm (Agrawal et al. 1993) can be implemented in a privacy preserving manner, as long as the sub-protocol does not reveal any information about the individual user.

In order to implement our algorithm, we assume that both *original site* (OS) and the *non-colluding storage site* (NSS) have access to a commonly decided secure pseudo-random number generator. So given the same initial seed, both sites can generate the same random sequence. Also assume that there are  $n$  users with  $m$  bit inputs.

Before protocol starts, OS and NSS decide on an  $\epsilon$  that will be fixed during the execution of the entire data mining process. This  $\epsilon$  will determine how much noise will be added in each support count protocol execution. Basically,  $\epsilon \times n$  fake itemset will be inserted to the message sent to the **processing site**(PS).

Assume that  $X = \{i_{k_1}, i_{k_2}, \dots, i_{k_j}\}$  is the itemset whose support count we would like to learn. OS projects the  $k_1, k_2, \dots, k_j$  columns of the data and randomly generates  $\epsilon \times n \times j$  bits. The same random sequence can also be generated by the NSS by just sending the initial seed to NSS.

After this step, OS selects another random number seed and starts the following process: for each  $j$  bit in the previous sequence, it randomly decides with probability  $p$  whether it will generate fake itemset that supports the  $X$  or not. Based on that it will either XOR it with  $j$  bit that is all 1's or some other random  $j$  bit. OS keeps the count of how many times it the XOR'ed with  $j$  bit all 1's sequence. Let this number be  $t$ , and  $t < \epsilon \times n$ . In the next stage, the OS assigns a random ordering to the  $(1 + \epsilon)n$  items. It then sends this random ordering to the NSS. Both sites send their randomly ordered complete data sets (including both original and fake transactions) to the processing site.

After receiving the messages from the both sites, PS calculates the XOR of the corresponding items and counts the ones that has all 1's. (Basically, all 1's means that transaction supports the itemset  $X$ ) Then PS sends this count to OS and OS learns the original count by subtracting  $t$  from it.

The corresponding pseudo-code is given in protocol 4.2.

### 4.3 Communication complexity of the protocol

Clearly for each itemset of size  $j$ ,  $O((1 + \epsilon)nj)$  bit must be sent during the execution of the protocol. So, for association rule mining, if there are  $C_j$  candidate itemsets of size  $j$ , the association rule mining algorithm will need to send at most  $O(\sum_{i=0}^m (C_i * (1 + \epsilon)ni))$  bits, where  $m$  is the size of the largest itemset which can be found.

### 4.4 Security of the protocol

To analyze the security of the entire protocol, let us first analyze the information revealed to each site participating in the protocol. Note that the  $\epsilon$  value is used to generate some number of fake transactions. The  $p$  value is used to probabilistically make some (random) number of those fake transactions supporting the itemset. Both the Original Site (OS) and the Non-colluding Storage Site (NSS) know the value of  $\epsilon$ , while only the Original Site knows the value of  $p$ .

#### 4.4.1 NSS view

During the execution of the protocol, NSS does not learn anything, because all it sees are random values from a uniform distribution. These are indistinguishable from any other values generated from a uniform distribution. Also, the NSS cannot refine its estimate of the value of  $p$  in any way, since no useful information pertaining to  $p$  is revealed.

#### 4.4.2 OS view

The values that the OS sees from the users are the user information bits XOR'ed with random numbers from a uniform distribution. Thus, these

---

### Protocol 1 An Individual privacy-preserving protocol for support count

---

**Require:** Given itemset  $X = \{i_{k_1}, i_{k_2}, \dots, i_{k_j}\}$  and  $|X| = j$ , noise parameter  $\epsilon$ , transaction size  $m$  and database size  $n$

{Actions for each site are given below}

**for** site "OS" **do**

generate random numbers  $seed_1, seed_2, seed_3$  ;  
generate  $\epsilon \times n, j$  bit random number using  $seed_1$  as the initial seed of the pseudo-random generator, call the list  $R_1$ ;

$t = 0$  ;

**for** each  $j$  bit number  $r \in R_1$  **do**

choose a random number  $v$  between  $[0, 1]$

**if**  $v < p$  **then**

$r = r \oplus \{1\}^j$

**else**

$r = r \oplus x_r$  { $x_r$  is a random generated from  $seed_3$ }

**end if**

**if**  $r$  XOR'ed with  $\{1\}^j$  **then**

$t++$ ;

**end if**

**end for**

project the  $k_1, \dots, k_j$  columns of the initial user data matrix.

assign random id numbers to each  $(1 + \epsilon)n, j$  bit numbers using  $seed_2$ .

permute the list and name it as  $L$ .

send  $seed_1, seed_2$  to NSS.

send  $L$  to PS.

wait for the result  $rs$ , from PS.

output  $rs - t$  as the support count of  $X$  to the user

**end for**

**for** site "NSS" **do**

get random numbers  $seed_1, seed_2$  from OS ;

generate  $\epsilon \times n, j$  bit random number using  $seed_1$  as the initial seed of the pseudo-random generator;

project the  $k_1, \dots, k_j$  columns of the initial user data matrix.

assign random id numbers to each  $(1 + \epsilon)n, j$  bit numbers using  $seed_2$ .

permute the list and name the list as  $\bar{L}$

send  $\bar{L}$  to PS.

**end for**

**for** site "PS" **do**

$cnt = 0$

**for** each  $(i, x) \in L$  and  $(i, r) \in \bar{L}$  **do**

**if**  $x \oplus r = \{1\}^j$  **then**

$cnt++$ ;

**end if**

**end for**

send  $cnt$  to OS

**end for**

---

values are indistinguishable from random values. Through the protocol, OS does learn the support count of the  $X$  but it has no idea which users support the  $X$ . Basically  $\forall i, j, Pr\{\text{user}_i \text{ supports } X\} = Pr\{\text{user}_j \text{ supports } X\}$ . Therefore nothing is revealed related to the individual privacy of customers.

#### 4.4.3 PS view

During the association rule mining, PS only learns that for some itemset (note that it does not know what the itemset is) support count is less than the result but again it has no idea which user really supports the itemset because of the added noise. Clearly, it gains no information that can damage individual user privacy. Furthermore, it does not even learn the exact distribution of support counts. At no point of time in the protocol is either  $p$  or  $\epsilon$  revealed to the PS. So the PS does not *know* the exact values of  $\epsilon$  or  $p$ .

As far as estimation of the values goes, note that even the NSS cannot really *know* the value for  $p$ , and even more, it definitely won't know the exact number of fake supporting transactions. Let us analyze the information that the PS receives during the protocol. The PS receives only one list of numbers each from both the OS and the NSS. All of these numbers look completely random and indistinguishable from each other. The only thing that the PS finds out is what it computes, viz. how many of the transactions support the itemset and which are these transactions. However, note that all identifying information from these transactions has been removed, and also the PS has no idea of what are the attributes for the itemset, so it is wholly unable to distinguish between true and false transactions. It cannot even make any assumptions about the product  $\epsilon * p$ . The only reasonable guess that it can make, is that in the interests of efficiency, the value of  $\epsilon$  is likely to be less than 1. Also, though the value of  $\epsilon$  is fixed, the value of  $p$  is changed in every iteration. Thus at worst, the PS can get an upper bound on the value of  $\epsilon$  as 1 which it anyway knew. In any case, in terms of individual privacy, nothing is revealed. Thus the Processing Site gains *no* additional information.

#### 4.4.4 Security of whole protocol

Since neither the OS, PS nor the NSS receive any additional information beyond what they are already supposed to know, the entire protocol is secure. Of course, in all of these analyses, we assume that no collusion occurs between these parties and these parties follow the protocols. This assumption is realistic in the sense, with some certain legal binding contracts this may be enforced.

#### 4.5 Accuracy of the protocol

Now, we analyze the accuracy of the protocol. Our protocol exactly calculates the support count of an itemset. The only cause for inaccuracy could be the randomization at the start where the OS adds a certain number of fake transactions before sending the list to the PS. However, the Original Site *knows* exactly how many fake supporting transactions have been added and can subtract this off before returning the final result to the user. Thus the effect of the randomization on the result can be completely negated. Randomization is done only to preserve security while sending data to the processing site. So, in our case, randomization enhances the security of the protocol

rather than degrading it and ensures that the Processing Site cannot find out any additional information.

## 5 Conclusion and Future Work

The main contributions of this paper have been to present a novel architecture for privacy preserving data mining. We prove that any kind of data mining can be done securely with this architecture without sacrificing accuracy. We demonstrate how association rule mining can be done using this architecture. However, the whole database needs to be transmitted at least once, which may turn out to be prohibitive. In the future, we hope to improve the efficiency of this approach. As a first direction, we plan to explore securely generating representative samples from the database. This would be an orthogonal technique for applications not requiring perfect accuracy, but very high security.

## 6 Acknowledgments

We thank Chris Clifton and Xiaodong Lin for several useful discussions on the subject.

## References

- Agrawal, D. & Aggarwal, C. C. (2001), On the design and quantification of privacy preserving data mining algorithms, *in* 'Symposium on Principles of Database Systems'.
- Agrawal, R., Imielinski, T. & Swami, A. N. (1993), Mining association rules between sets of items in large databases, *in* P. Buneman & S. Jajodia, eds, 'Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data', Washington, D.C., pp. 207–216.
- Agrawal, R. & Srikant, R. (2000), Privacy-preserving data mining, *in* 'Proceedings of the 1997 ACM SIGMOD Conference on Management of Data', ACM, Dallas, TX.
- Evfimievski, A., Srikant, R., Agrawal, R. & Gehrke, J. (2002), Privacy preserving mining of association rules, *in* 'Proceedings of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery in Databases and Data Mining'.
- Goldreich, O. (2000), 'Secure multi-party computation', Working Draft.
- Kantarcioğlu, M. & Clifton, C. (2002), Privacy-preserving distributed mining of association rules on horizontally partitioned data, *in* 'Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery'.
- Lindell, Y. & Pinkas, B. (2000), Privacy preserving data mining, *in* 'Advances in Cryptology – CRYPTO 2000', Springer-Verlag, pp. 36–54.
- Rizvi, S. J. & Haritsa, J. R. (2002), Privacy-preserving association rule mining, *in* 'Proceedings of 28th International Conference on Very Large Data Bases (VLDB)'.
- Vaidya, J. S. & Clifton, C. (2002), Privacy preserving association rule mining in vertically partitioned data, *in* 'Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.

Yao, A. C. (1986), How to generate and exchange secrets, in 'Proceedings of the 27th IEEE Symposium on Foundations of Computer Science', IEEE, pp. 162–167.

## A Discussion of Reviewers Comments

### A.1 Efficiency

**Comment:** Why do we need a new architecture if it is not “very” efficient?

**Response:** Although Secure Multiparty protocols can be used to implement totally secure algorithms, a key requirement is that all individuals should stay online and participate in a complex SMC protocol. Also, for any new data mining task, an entire SMC style protocol must be executed, and the information providers have to remain online for this. Even assuming that all of the information providers are online, the communication complexity of the current SMC algorithms cannot be scaled to that many participants effectively. Therefore it is clear that current SMC protocols cannot be directly used and need to be enhanced for individual privacy-preserving data mining.

One way of going about this, is to store some of the user information at some site before engaging in the data mining process. Note that data cannot be given as-is to any site. We must obscure the data in some manner and then store it. One way is to use the data perturbation approach. However, as discussed in the motivation section, these data perturbation approaches may not be suitable for applications requiring high security and high accuracy. Therefore we propose to store the data in two or more sites that learns nothing from the data (assuming they do not collude.) This new architecture is necessary even if it is not extremely efficient, since it will enable data mining with any desired privacy level without a loss in the accuracy. We do believe that an interesting research direction would be to search for very efficient algorithms under the proposed architecture to do privacy preserving data mining at the desired level of security. Also the proposed architecture is resilient to hacking. In order to reveal actual information, all data holders must be hacked.

### A.2 Information Leakage

**Comment:** The suggested protocol for computing the support count might leak some data, as the PS should probably know, or be able to estimate, what are the epsilon and p values that are used by the protocol. Using these values it can learn information about the actual support count. Please comment.

**Response:** The  $\epsilon$  value is used to generate some number of fake transactions. The  $p$  value is used to probabilistically make some (random) number of those fake transactions supporting the itemset. Both the Original Site (OS) and the Non-colluding Storage Site (NSS) know the value of  $\epsilon$ , while only the Original Site knows the value of  $p$ . At no point of time are either of these values revealed to the Processing Site (PS). So, the PS does not *know* the exact values of  $\epsilon$  or  $p$ .

As far as estimation of the values goes, note that even the NSS cannot really *know* the value for  $p$ , and even more, it definitely wont know the exact number of fake supporting transactions. Let us analyze the information that the PS receives during the protocol. The PS receives only one list of numbers each from

both the OS and the NSS. All of these numbers look completely random and indistinguishable from each other. The only thing that the PS finds out is what it computes, viz. how many of the transactions support the itemset and which are these transactions. However, note that all identifying information from these transactions has been removed, and also the PS has no idea of what are the attributes for the itemset, so it is wholly unable to distinguish between true and false transactions. It cannot even make any assumptions about the product  $\epsilon * p$ . The only reasonable guess that it can make is that the value of  $\epsilon$  is likely to be less than 1 in the interests of efficiency. Also, though the value of  $\epsilon$  is fixed, the value of  $p$  is changed in every iteration. Thus at worst, the PS can get an upper bound on the value of  $\epsilon$  as 1 which it anyway knew. In any case, in terms of individual privacy, nothing is revealed. Thus the Processing Site gains *no* additional information.

### A.3 Scalability

**Comment:** The secure data mining algorithms earlier developed were intended for mining over a set of distributed datasets, where the number of datasets is small (say at most hundreds). They do not scale to having each person be a separate dataset. Thus while they can be applied in theory, please qualify that statement by adding that the execution and communication complexity is likely to be so high that in practice the algorithms will not be useful in this context.

**Response:** It is indeed true that the algorithms developed earlier do not scale well to having each person being a separate dataset. In fact, this is one of the primary motivations for our architecture. Our architecture reduces the potential  $n$  party secure protocol ( $n$  being the number of users) into a three party protocol. So the scalability problem for a large number of participants in SMC style protocols is automatically solved.

### A.4 Randomization

**Comment:** When you start randomizing data for PS, don't you run into all the issues you covered in Section 2?

**Response:** Problems with the randomization approach generally lead to either inadequate security or inaccurate results. Our randomization does *not* affect the accuracy of the results at all. Note that the Original Site *knows* exactly the effect of the randomization on the result and subtracts this effect before releasing the final result. Randomization is done only to preserve security while sending data to the processing site. So, in our case, randomization enhances the security of the protocol rather than degrading it and ensures that the Processing Site cannot find out any additional information.