Privacy-Preserving Decision Tree Classification Over Horizontally Partitioned Data

Justin Zhan[1], Stan Matwin[2], and LiWu Chang[3]
[1,2]School of Information Technology & Engineering, University of Ottawa, Canada
[3]Center for High Assurance Computer Systems, Naval Research Laboratory, USA
{zhizhan, stan}@site.uottawa.ca, lchang@itd.nrl.navy.mil

Abstract: Protection of privacy is one of important problems in data mining. The unwillingness to share their data frequently results in failure of collaborative data mining. This paper studies how to build a decision tree classifier under the following scenario: a database is horizontally partitioned into multiple pieces, with each piece owned by a particular party. All the parties want to build a decision tree classifier based on such a database, but due to the privacy constraints, neither of them wants to disclose their private pieces. We build a privacy-preserving system, including a set of secure protocols, that allows the parties to construct such a classifier. We guarantee that the private data are securely protected.

Keywords: Privacy, decision tree, classification.

I. Introduction

Business success often relies on collaboration. The collaboration is even more critical in the modern business world, not only because of mutual benefit it brings but also because the coalition of multiple parters will be more competitive than each individual. Assuming they trust each other to a degree that they can share their private data, the collaboration becomes straightforward. However, in many scenarios, sharing data are impossible because of privacy concerns. Thus, collaboration without sharing private data becomes extremely important.

In this paper, we study a prevalent collaboration scenario, the collaboration involving a data mining task: multiple parties, each having a private data set, want to conduct data mining on the joint data set that is the union of all individual data sets; however, because of the privacy constraints, no party wants to disclose its private data set to each other. The objective of this paper is to develop efficient methods that enable this type of computation while minimizing the amount of the private information that each party has to disclose.

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 470 - 476.

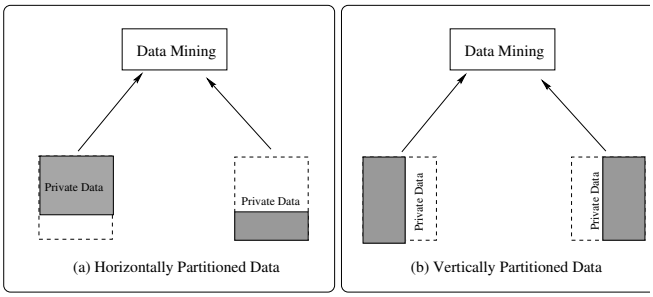

Figure 1:

Data mining includes various algorithms such as classification, association rule mining, and clustering. In this paper, we focus on classification. There are two types of classification between two collaborative parties: Figure 1(a) shows the data classification on the horizontally partitioned data, and Figure 1(b) shows the data classification on the vertically partitioned data. We study the classification on the horizontally partitioned data. In particular, we study how to build a decision tree classifier on private data. We have developed a privacy-preserving system that allows them to build a decision tree classifier based on their joint data.

II. Privacy-Preserving Decision-Tree Classification

Classification is an important problem in the field of data mining. In classification, we are given a set of example records, called the training data set, with each record consisting of several attributes. One of the categorical attributes, called the class label, indicates the class to which each record belongs. The objective of classification is to use the training data set to build a model of the class label such that it can be used to classify new data whose class labels are unknown.

Many types of models have been built for classification, such as neural networks, statistical models, genetic models, and decision tree models. The decision tree models are found to be the most useful in the domain of data

mining since they obtain reasonable accuracy and they are relatively inexpensive to compute. We define our problem as follows:

Problem 1 *We consider the scenario where n parties, each having a private data set (denoted by S_1, S_2, $\cdots$, and S_n respectively), want to collaboratively conduct decision tree classification on the union of their data sets. The data sets are assumed to be horizontally partitioned. Because they are concerned about the data privacy, neither party is willing to disclose its raw data set to others.*

Next, we give the notations that we will follow.

II.1 Notations

- e: public key.
- d: private key.
- P_i: the ith party.
- n: the total number of parties. Assuming $n > 2$.
- m: the total number of class.
- α is the number of bits for each transmitted element in the privacy-preserving protocols.
- N: the total number of records.

II.2 Decision Tree Classification Algorithm

Classification is one of the forms for data analysis that can be used to extract models describing important data classes or to predict future data. It has been studied extensively by the community in machine learning, expert system, and statistics as a possible solution to knowledge discovery problems. The decision tree is one of the classification methods. A decision tree is a class discriminator that recursively partitions the training set until each partition entirely or dominantly consists of examples from one class. A well known algorithm for building decision tree classifiers is ID3 [13]. We describe the algorithm below where S represents the training samples and AL represents the attribute list:

ID3(S, AL)

1. Create a node V.
2. **If** S consists of samples with all the same class C **then** return V as a leaf node labelled with class C.
3. **If** AL is empty, **then** return V as a leaf-node with the majority class in S.
4. Select test attribute (TA) among the AL with the highest information gain.
5. Label node V with TA.
6. **For** each known value a_i of TA
   (a) Grow a branch from node V for the condition $TA = a_i$.
   (b) Let s_i be the set of samples in S for which $TA = a_i$.
   (c) **If** s_i is empty **then** attach a leaf labelled with the majority class in S.
   (d) **Else** attach the node returned by **ID3(s_i, $AL - TA$)**.

According to ID3 algorithm, each non-leaf node of the tree contains a splitting point, and the main task for building a decision tree is to identify an attribute for the splitting point based on the information gain. Information gain can be computed using *entropy*. In the following, we assume there are m classes in the whole training data set. $Entropy(S)$ is defined as follows:

$$Entropy(S) = -\sum_{j=1}^{m} Q_j \log Q_j, \tag{1}$$

where Q_j is the relative frequency of class j in S. Based on the entropy, we can compute the information gain for any candidate attribute A if it is used to partition S:

$$Gain(S, A) = Entropy(S) - \sum_{v \in A} (\frac{|S_v|}{|S|} Entropy(S_v)), \tag{2}$$

where v represents any possible values of attribute A; S_v is the subset of S for which attribute A has value v; $|S_v|$ is the number of elements in S_v; $|S|$ is the number of elements in S. To find the best split for a tree node, we compute information gain for each attribute. We then use the attribute with the largest information gain to split the node.

II.3 Cryptography Tools

In this paper, we use the concept of homomorphic encryption which was originally proposed in [18]. Since then, many such systems have been proposed [3, 15, 16, 17]. We observe that some homomorphic encryption schemes, such as [4], are not robust against chosen cleartext attacks. However, we base our secure protocols on [17], which is semantically secure [9].

In our secure protocols, we use additive homomorphism offered by [17]. In particular, we utilize the following characterizer of the homomorphic encryption functions: $e(a_1) \times e(a_2) = e(a_1 + a_2)$ where e is an encryption function; a_1 and a_2 are the data to be encrypted. Because of the property of associativity, $e(a_1 + a_2 + .. + a_n)$ can be computed as $e(a_1) \times e(a_2) \times \cdots \times e(a_n)$ where $e(a_i) \neq 0$. That is

$$d(e(a_1 + a_2 + \cdots + a_n)) = d(e(a_1) \times e(a_2) \times \cdots \times e(a_n)) \tag{3}$$

$$d(e(\alpha_1)^{\alpha_2}) = d(e(\alpha_1 \alpha_2)) \tag{4}$$

II.4 Privacy-Preserving Decision Tree Classification System

The privacy-preserving classification system contains several secure protocols that multiple parties need follow. There are five major steps:

- To compute $Entropy(S_v)$.
- To compute $\frac{|S_v|}{|S|}$.
- To compute $\frac{|S_v|}{|S|}Entropy(S_v)$.
- To compute information gain for each candidate attribute.
- To compute the attribute with the largest information gain.

The goal of our privacy-preserving classification system is to disclose no private data in every step. We firstly select a key generator who produces the encryption and decryption key pairs. The computation of the whole system is under encryption. For the purpose of illustration, let's assume that P_n is the key generator who generates a homomorphic encryption key pair (e, d). Next, we will show how to conduct each step.

II.4.1 Computation of $e(Entropy(S_v))$

Protocol 1 *To compute* $e(Q_j)$

1. *Each party computes their own share of* $|S_v|$. *Assuming* P_1 *gets* c_1, P_2 *gets* c_2, $\cdots$, P_n *gets* c_n.
2. P_n *sends* $e(c_n)$ *to* P_1.
3. P_1 *computes* $e(c_n) \times e(c_1) = e(c_1 + c_n)$ *and sends it to* P_2.
4. *Repeat until* P_{n-1} *obtains* $e(c_1 + c_2 + \cdots + c_n)$.
5. P_{n-1} *computes* $e(\sum_{i=1}^{n} c_i)^{\frac{1}{N}} = e(Q_j)$.

Protocol 2 *To compute* $e(Q_j log(Q_j))$

1. P_{n-1} *generates a set of random numbers* r_1, r_2, $\cdots$, *and* r_t.
2. P_{n-1} *sends the sequence of* $e(Q_j)$, $e(r_1)$, $e(r_2)$, $\cdots$, $e(r_t)$ *to* P_n *in a random order.*
3. P_n *decrypts each element in the sequence, and sends* $log(Q_j)$, $log(r_1)$, $log(r_2)$, $\cdots$, $log(r_t)$ *to* P_1 *in the same order as* P_{n-1} *did.*
4. P_1 *adds a random number* R *to each of the elements, then sends them to* P_{n-1}.
5. P_{n-1} *obtains* $log(Q_j) + R$ *and computes* $e(Q_j)^{(log(Q_j)+R)} = e(Q_j log(Q_j) + RQ_j)$.
6. P_{n-1} *sends* $e(Q_j)$ *to* P_1.
7. P_1 *computes* $e(Q_j)^{-R} = e(-RQ_j)$ *and sends it to* P_{n-1}.
8. P_{n-1} *computes* $e(Q_j log(Q_j) + RQ_j) \times e(-RQ_j) = e(Q_j log(Q_j))$.

Protocol 3 *To compute* $e(Entropy(S_v))$

1. *Repeat protocol 1-2 to compute* $e(Q_j log(Q_j))$ *for all* j*'s.*
2. P_{n-1} *computes* $e(Entropy(S_v)) = \prod_j e(Q_j log(Q_j)) = e(\sum_j Q_j log(Q_j))$.

Theorem 1 *(Correctness). Protocol 1-3 correctly compute Entropy.*

Proof In protocol 1, P_{n-1} obtains $e(Q_j)$. In protocol 2, P_{n-1} gets $e(Q_j log(Q_j))$. These two protocols are repeatedly used until P_{n-1} obtains $e(Q_j log(Q_j))$ for all j's. In protocol 3, P_{n-1} computes the entropy by all the terms previously obtained. Notice that although we use $Entropy(S_v)$ to illustrate, $Entropy(S)$ can be computed following the above protocols with different input attributes. ∎

Theorem 2 *(Privacy-Preserving). Assuming the parties follow the protocol, the private data are securely protected.*

Proof In protocol 1, all the data transmission are hidden under encryption. The parties who are not the key generator can't see other parties' private data. On the other hand, the key generator doesn't obtain the encryption of other parties's private data. Therefore, protocol 1 discloses no private data. In protocol 2, although P_{n-1} sends $e(Q_j)$ to P_n, Q_j is hidden by a set of random numbers known only by P_{n-1}. Thus private data are not revealed. In protocol 3, the computations are still under encryption, no private data are disclosed either. ∎

Theorem 3 *(Efficiency). The computation of Entropy is efficient from both computation and communication point of view.*

Proof To prove the efficiency, we need conduct complexity analysis of the protocol. The bit-wise communication cost of protocol 1 is $\alpha(n-1)$, of protocol 2 is $\alpha(3t+5)$. The total communication cost has the upper bound of $\alpha m(n+3t+4)$. The computation cost of protocol 1 is nN, of protocol 2 is $5t+3$. The total computation cost is upper bounded by $mnN + 5mt + 4m$. Therefore, the protocols are sufficient fast. ∎

II.4.2 The Computation of $\frac{|S_v|}{|S|}Entropy(S_v)$

Protocol 4 *To Compute* $\frac{|S_v|}{|S|}$

1. P_{n-1} *sends* $e(|S_v|)$ *to* P_1.
2. P_{n-1} *generates a set of random numbers:* r_1, r_2, $\cdots$, r_t.
3. P_{n-1} *sends* $e(|S|)$, $e(r_1)$, $\cdots$, *and* $e(r_t)$ *to* P_n *in a random order. Note that* $e(|S|)$ *can be computed following the first four steps of protocol 1.*
4. P_n *decrypts each element, then sends the sequence of* $\frac{1}{|S|}$, $\frac{1}{r_1}$, $\cdots$, *and* $\frac{1}{r_t}$ *to* P_1 *in the same order as* P_{n-1} *did.*
5. P_1 *computes* $e(|S_v|)^{\varpi}$ *where* ϖ *denotes for each decrypted element, then sends the sequence to* P_{n-1} *in the same order as* P_n *did.*
6. P_{n-1} *obtains* $e(\frac{|S_v|}{|S|})$ *since he knows the original permutations.*

Up to now, P_{n-1} has obtained $e(\frac{|S_v|}{|S|})$ and $e(Entropy(S_v))$. Next, we discuss how to compute $\frac{|S_v|}{|S|}Entropy(S_v)$.

Protocol 5 *To Compute* $\frac{|S_v|}{|S|}Entropy(S_v)$

1. P_{n-1} *sends* $e(\frac{|S_v|}{|S|})$ *to* P_1.
2. P_1 *computes* $e(\frac{|S_v|}{|S|}) \times e(R') = e(\frac{|S_v|}{|S|} + R')$ *where* R' *is a random number only known by* P_1, *then sends* $e(\frac{|S_v|}{|S|} + R')$ *to* P_n.
3. P_n *decrypts it and sends* $\frac{|S_v|}{|S|} + R'$ *to* P_{n-1}.
4. P_{n-1} *computes* $e(Entropy(S_v))^{(\frac{|Sv|}{|S|}+R')} = e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v))$.
5. P_{n-1} *sends* $e(Entropy(S_v))$ *to* P_1.
6. P_1 *computes* $e(Entropy(S_v))^{-R'} = e(-R'Entropy(S_v))$, *and sends it to* P_{n-1}.
7. P_{n-1} *computes* $e(\frac{|S_v|}{|S|}Entropy(S_v) + R'Entropy(S_v)) \times e(-R'Entropy(S_v)) = e(\frac{|S_v|}{|S|}Entropy(S_v))$.

Theorem 4 *(Correctness). Protocol 4-5 correctly computes* $\frac{|S_v|}{|S|}Entropy(S_v)$.

Proof In protocol 4, P_{n-1} obtains $e(\frac{|S_v|}{|S|})$. In protocol 5, P_{n-1} gets $\frac{|S_v|}{|S|}Entropy(S_v)$. The computation uses the both properties of homomorphic encryption. ∎

Theorem 5 *(Privacy-Preserving). Assuming the parties follow the protocol, the private data are securely protected.*

Proof In protocol 4, P_n sends $e(|S|)$ to P_n but it is hidden by a set of random numbers known only by P_{n-1}. Although P_1 receives a decrypted sequence, $\frac{1}{|S|}$ is also hidden by a set of random numbers. Thus, P_1 and P_n are prevented from knowing $|S|$. On the other hand, P_{n-1} is also hidden from knowing $|S|$ since all the terms he holds are under encryption. In protocol 5, although P_1 sends $e(\frac{|S_v|}{|S|} + R')$ to P_n, $\frac{|S_v|}{|S|}$ is hidden by a random number known only by P_1. Therefore, private data are not revealed. ∎

Theorem 6 *(Efficiency). The computation of protocol 4 and protocol 5 is efficient from both computation and communication point of view.*

Proof To prove the efficiency, we need conduct complexity analysis of the protocol. The total communication cost is $\alpha(3t + 4)$. The total computation cost is upper bounded by $6t$. Therefore, the protocols are very efficient. ∎

II.4.3 The Computation of the Attribute With the Largest Information Gain

Following the above protocols, we can compute $e(Entropy(S))$, $\frac{|S_v|}{|S|}Entropy(S_v)$. What left is to compute information gain for each attribute and select the attribute with the largest information gain.

Protocol 6 *To Compute Information Gain for An Attribute*

1. P_{n-1} *computes* $\prod_{v\in A} e(\frac{|S_v|}{|S|}Entropy(S_v)) = \sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v)$.
2. *He computes* $e(\sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v))^{-1} = e(-\sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v))$.
3. *He computes* $e(Gain(S, A)) = e(Entropy(S)) \times e(-\sum_{v\in A}\frac{|S_v|}{|S|}Entropy(S_v))$.

Once we compute the information gain for each candidate attribute, we then compute the attribute with the largest information gain. Without loss of generality, assuming there are k information gains: $e(g_1)$, $e(g_2)$, $\cdots$, and $e(g_k)$, with each corresponding to a particular attribute.

Protocol 7 *To Compute the Largest Information Gain*

	g_1	g_2	g_3	$\cdots$	g_k
g_1	+1	+1	-1	$\cdots$	-1
g_2	-1	+1	-1	$\cdots$	-1
g_3	+1	+1	+1	$\cdots$	+1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
g_k	+1	+1	-1	$\cdots$	+1

Table 1:

	S_1	S_2	S_3	S_4	Weight
S_1	+1	-1	-1	-1	-2
S_2	+1	+1	-1	+1	+2
S_3	+1	+1	+1	+1	+4
S_4	+1	-1	-1	+1	0

Table 2:

1. *P_{n-1} computes $e(g_i) \times e(g_j)^{-1} = e(g_i - g_j)$ for all $i, j \in [1, k], i > j$, and sends the sequence denoted by φ to P_n in a random order.*

2. *P_n decrypts each element in the sequence φ. He assigns the element $+1$ if the result of decryption is not less than 0, and -1, otherwise. Finally, he obtains a $+1/-1$ sequence denoted by φ'.*

3. *P_n sends $+1/-1$ sequence φ' to P_{n-1} who computes the largest element.*

Theorem 7 *(Correctness). Protocol 6-7 correctly computes the attribute with the largest information gain.*

Proof In protocol 6, P_{n-1} obtains $e(Gain(S, A))$. In protocol 7, P_{n-1} gets the attribute with the largest information. We discuss the details as follows:

P_{n-1} is able to remove permutation effects from φ' (the resultant sequence is denoted by φ'') since she has the permutation function that she used to permute φ, so that the elements in φ and φ'' have the same order. It means that if the qth position in sequence φ denotes $e(g_i - g_j)$, then the qth position in sequence φ'' denotes the evaluation results of $g_i - g_j$. We encode it as $+1$ if $g_i \geq g_j$, and as -1 otherwise. P_{n-1} has two sequences: one is the φ, the sequence of $e(g_i - g_j)$, for $i, j \in [1, k](i > j)$, and the other is φ'', the sequence of $+1/-1$. The two sequences have the same number of elements. P_{n-1} knows whether or not g_i is larger than g_j by checking the corresponding value in the φ'' sequence. For example, if the first element φ'' is -1, P_{n-1} concludes $g_i < g_j$. P_{n-1} examines the two sequences and constructs the index table (Table 1) to compute the largest element.

In Table 1, $+1$ in entry ij indicates that the information gain of the row (e.g., g_i of the ith row) is not less than the information gain of a column (e.g., g_j of the jth column); -1, otherwise. P_{n-1} sums the index values of each row and uses this number as the weight of the information gain in that row. She then selects the one that corresponds to the largest weight.

To make it clearer, let's illustrate it by an example. Assume that: (1) there are 4 information gains with $g_1 < g_4 < g_2 < g_3$; (2) the sequence φ is $[e(g_1 - g_2), e(g_1 - g_3), e(g_1 - g_4), e(g_2 - g_3), e(g_2 - g_4), e(g_3 - g_4)]$. The sequence φ'' will be $[-1, -1, -1, -1, +1, +1]$. According to φ and φ'', P_{n-1} builds the Table 2. From the table, P_{n-1} knows g_3 is the largest element since its weight, which is $+4$, is the largest.

∎

Theorem 8 *(Privacy-Preserving). Assuming the parties follow the protocol, the private data are securely protected.*

Proof In protocol 6, there is no data transmission. In protocol 7, we need prove it from two aspects: (1) P_{n-1} doesn't get information gain (e.g.,g_i) for each attribute. What P_{n-1} gets are $e(g_i - g_j)$ for all $i, j \in [1, k], i > j$ and $+1/-1$ sequence. By $e(g_i - g_j)$, P_{n-1} cannot know each information gain since it is encrypted. By $+1/-1$ sequence, P_{n-1} can only know whether or not g_i is greater than P_j. (2) P_n doesn't obtain information gain for each attribute either. Since the sequence of $e(g_i - g_j)$ is randomized before being send to P_n who can only know the sequence of $g_i - g_j$, he can't get each individual information gain. Thus private data are not revealed. ∎

Theorem 9 *(Efficiency). The computation of protocol 6 and protocol 7 is efficient from both computation and communication point of view.*

Proof The total communication cost is upper bounded by αm^2. The total computation cost is upper bounded by $m^2 + m + 1$. Therefore, the protocols are very fast. ∎

III. Overall Discussion

Our privacy-preserving classification system contains several components. In Section , we show how to correctly compute $e(Entropy(S_v))$. In Section , we present protocols to compute $\frac{|S_v|}{|S|} Entropy(S_v)$. In Section , we show how to compute information gain for each candidate attribute; we then describe how to obtain the attribute with the largest information gain. We discussed the correctness of the computation in each section. Overall correctness is also guaranteed.

As for the privacy protection, all the communications between the parties are encrypted, therefore, the parties who has no decryption key cannot gain anything out of the communication. On the other hand, there are some communication between the key generator and

other parties. Although the communications are still encrypted, the key generator may gain some useful information. However, we guarantee that the key generator cannot gain the private data by adding random numbers in the original encrypted data so that even if the key generator get the intermediate results, there is little possibility that he can know the intermediate results. Therefore, the private data are securely protected with overwhelming probability.

IV. Conclusion

Prior to conclude this paper. We describe the most related works. In early work on privacy-preserving data mining, Lindell and Pinkas [14] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by secure multi-party computation (SMC) research [22, 10]. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [21]. Randomization approaches were firstly proposed by Agrawal and Srikant in [2] to solve privacy-preserving data mining problem. Researchers proposed more random perturbation-based techniques to tackle the problems (e.g., [5, 19, 7]). In addition to perturbation, aggregation of data values [20] provides another alternative to mask the actual data values. In [1], authors studied the problem of computing the kth-ranked element. Dwork and Nissim [6] showed how to learn certain types of boolean functions from statistical databases in terms of a measure of probability difference with respect to probabilistic implication, where data are perturbed with noise for the release of statistics.

The problem we are studying is actually a special case of a more general problem, the Secure Multi-party Computation (SMC) problem. Briefly, a SMC problem deals with computing any function on any input, in a distributed network where each participant holds one of the inputs, while ensuring that no more information is revealed to a participant in the computation than can be inferred from that participant's input and output [12]. The SMC problem literature is extensive, having been introduced by Yao [22] and expanded by Goldreich, Micali, and Wigderson [11] and others [8]. It has been proved that for any function, there is a secure multi-party computation solution [10]. The approach used is as follows: the function F to be computed is first represented as a combinatorial circuit, and then the parties run a short protocol for every gate in the circuit. Every participant gets corresponding shares of the input wires and the output wires for every gate. This approach, though appealing in its generality and simplicity, means that the size of the protocol depends on the size of the circuit, which depends on the size of the input. This is highly inefficient for large inputs, as in data mining. It has been well accepted that for special cases of computations, special solutions should be developed for efficiency reasons.

In this paper, we provide a novel solution for decision tree classification over horizontally partitioned private data. Instead of using data transformation, we define a protocol using homomorphic encryption to exchange the data while keeping it private. Our classification system is quite efficient that can be envisioned by the communication and computation complexity. The total communication complexity is upper bounded by $\alpha(m^2+nm+3tm+4m+3t+4)$. The computation complexity is upper bounded by $mnN+m^2+5mt+5m+6t+1$.

References

[1] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k th-ranked element. In *EUROCRYPT pp 40-55*, 2004.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[3] J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography, pp. 120-128, Kingston, Ontario, May*, 1994.

[4] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Information Security Conference, 471-483*, 2002.

[5] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proceedings of The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, August 24-27 2003.

[6] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO 2004 528–544*.

[7] A. Evfmievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 211-222, San Diego, CA, June 9-12*, 2003.

[8] M. Franklin, Z. Galil, and M. Yung. An overview of secure distributed computing. Technical Report TR CUCS-00892, Department of Computer Science, Columbia University, 1992.

[9] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On secure scalar product computation for privacy-preserving data mining. In *Proceedings of The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004), volume 3506 of Lecture Notes in Computer Science, pages 104–120, Seoul, Korea, December 2–3, 2004, Springer-Verlag*, 2004.

[10] O. Goldreich. Secure multi-party computation (working draft). http://www.wisdom.weizmann.ac.il /home/oded/public_html/foc.html, 1998.

[11] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[12] S. Goldwasser. Multi-party computations: Past and present. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, Santa Barbara, CA USA, August 21-24 1997.

[13] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[14] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science, Volume 1880*, 2000.

[15] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM conference on Computer and Communication Security, pp. 59-66, San Francisco, California, United States*, 1998.

[16] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Eurocrypt'98, LNCS 1403, pp.308-318*, 1998.

[17] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *In Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592, pages 223-238. Springer-Verlag*, 1999.

[18] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation, eds. R. A. DeMillo et al., Academic Press, pp. 169-179.*, 1978.

[19] Shariq Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[20] L. Sweeney. k-anonymity: a model for protecting privacy. In *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10 (5), pp 557–570*, 2002.

[21] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 639-644, Edmonton, Alberta, Canada, July 23-26*, 2002.

[22] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.