Privacy Preserving K-nearest Neighbor Classification

Justin Zhan[1], LiWu Chang[2] and Stan Matwin[1]
(Corresponding author: Justin Zhan)

School of Information Technology & Engineering, University of Ottawa[1]
SITE building, 800 King Edward Ave. Ottawa, Ontario, K1N 6N5, Canada (Email: {zhizhan, stan}@site.uottawa.ca)
Center for High Assurance Computer Systems, Naval Research Laboratory[2]
4555 Overlook Avenue SW, Washington DC 20375-5337, USA (Email: lchang@itd.nrl.navy.mil)

(Received March 22, 2005; revised and accepted April 21, 2005)

Abstract

This paper considers how to conduct k-nearest neighbor classification in the following scenario: multiple parties, each having a private data set, want to collaboratively build a k-nearest neighbor classifier without disclosing their private data to each other or any other parties. Specifically, the data are vertically partitioned in that all parties have data about all the instances involved, but each party has its own view of the instances - each party works with its own attribute set. Because of privacy constraints, developing a secure framework to achieve such a computation is both challenging and desirable. In this paper, we develop a secure protocol for multiple parties to conduct the desired computation. All the parties participate in the encryption and in the computation involved in learning the k-nearest neighbor classifiers[1].

Keywords: k-nearest neighbor classifier, Privacy, security

1 Introduction

We address the following problem: multiple parties are cooperating on a data mining task. Each of the parties owns data pertinent to the aspect of the task addressed by this party. More specifically, the data consist of instances, all parties have data about all the instances involved, but each party has its own view of the instances - each party works with its own attribute set. The parties may be unwilling to release their attribute values to other parties due to privacy or confidentiality of the data . How can we structure information sharing between the parties so that the data will be shared for the purpose of data mining, while at the same time specific attribute values will be kept confidential by the parties to whom they belong? This is the task addressed in this paper. In the privacy-oriented data mining this task is known as data mining with vertically partitioned data.

In this paper, we focus on the following data mining algorithm: the k-nearest neighbor classification. The objective of k-nearest neighbor classification is to discover k nearest neighbors for a given instance, then assign a class label to the given instance according to the majority class of the k nearest neighbors. Our goal is not only to achieve the above objective, but also to preserve the data privacy. Our contribution is to develop a secure protocol based on homomorphic encryption and random perturbation techniques.

The paper is organized as follows: We describe the k-nearest neighbor classification procedure in Section 2. We then present our proposed secure protocols in Section 3. The related work is discussed in Section 4. We give our conclusion in Section 5.

[1]The preliminary version of this paper has been published in IEEE International Conference on Granular Computing (GrC 2005).

2 Build K-Nearest Neighbor Classifiers on Private Data

The k-nearest neighbor classification is an instance-based learning algorithm that has shown to be very effective for a variety of problem domains. The algorithm assumes that all instances correspond to points in the n-dimensional space. The key element of this scheme is the availability of a similarity measure that is capable of identifying neighbors. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance x be described by the feature vector $\langle a_1(x), a_2(x), \cdots, a_m(x)\rangle$, where $a_r(x)$ denotes the value of the rth attribute of instance x. Then the distance between two instances x_i and x_j is defined as $d(x_i, x_j)$, where

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{m}(a_r(x_i) - a_r(x_j))^2}. \quad (1)$$

2.1 Problem Definition

We consider the scenario where multiple parties, each having a private data set (denoted by D_1, D_2, $\cdots$ and D_n respectively), want to collaboratively build a k-nearest neighbor classifier on the concatenation of their data sets. Because they are concerned about their data privacy, neither party is willing to disclose its raw data set to others. Without loss of generality, we make the following assumptions about the data sets: (i) D_1, D_2, $\cdots$ and D_n contain the same number of records. Let N denote the total number of records for each data set; (ii) The identities of the ith (for $i \in [1, N]$) record in D_1, D_2, $\cdots$ and D_n are the same. (iii) The class labels are known by all the parties; (iv) The total number of parties, n, is greater than 2; (v) Each party receives only a portion of the query input according to her attribute set. In other words, one party does not know what values of the input query are received by other parties.

2.2 K-nearest Neighbor Classification Procedure

In this paper, we consider learning discrete-valued target functions of form $f : R^n \longrightarrow V$, where V is the finite set $v_1, v_2, \cdots, v_s$. The following is the procedure for building a k-nearest neighbor classifier on $[D_1 \cup D_2 \cdots \cup D_n]$.

1) Training algorithm:
   - For each training example (x, f(x)), add the example to the list *training-examples*.
2) Classification algorithm: Given a query instance x_q to be classified,
   - Let x_1, $\cdots$ x_k denote the k instances from *training-examples* that are nearest to x_q
   - Return

$$\hat{f}(x_q) \longleftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i)),$$

   where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

2.3 How to Obtain the K-nearest Neighbor Instances

Given a query instance x_q, we want to compute the distance between x_q and each of the N training instances. Since each party holds only a portion of a training instance, each party computes her portion of the distance measure (called the *distance portion*) according to her attribute set. To decide the k-nearest neighbors of x_q, all the parties need to sum their distance portions together. For example, assume that the distance portions for the first instance are s_{11}, s_{12}, $\cdots$, s_{1n}; and the distance portions for the second instance are s_{21}, s_{22}, $\cdots$, s_{2n}. To compute whether the distance between the first instance and x_q is larger than the distance between the second instance and x_q, we need compute whether $\sum_{i=1}^{n} s_{1i} \geq \sum_{i=1}^{n} s_{2i}$. How to obtain this result without compromising the data privacy? A naive solution is that those parties disclose their distances portions to each other, and they can then easily decide the k nearest neighbors by comparing the distances. However, the naive solution will lead to private data disclosure. The reasons are as follows: one party can make multiple queries, and if he gets the distance portions from each query, then he can identify the private data. Let's use an example to illustrate this problem. Assume that the query instance contains 2 non-zero values, e.g., $x_q = 1.2, 4.3, 0, \cdots, 0$, and P_1 holds the first two attributes. Then, the query requester can learn the private values of P_1 with two queries. Firstly, he uses x_q to get a d value. He uses another $x'_q = 5.6, 4.8, 0, \cdots, 0$ to get a value d'. He can solve the following two equations to get the first and second elements (denoted by y_1 and y_2) of x_i which are supposed to be private:

$$d = \sqrt{(y_1 - 1.2)^2 + (y_2 - 4.3)^2)},$$

$$d' = \sqrt{(y_1 - 5.6)^2 + (y_2 - 4.8)^2}.$$

How to securely compute the k nearest neighbors without the help of a trusted party presents a challenge. In next section, we develop a secure protocol to tackle this challenge.

3 Secure Computing Protocol

3.1 Introducing Homomorphic Encryption

In our secure protocols, we use homomorphic encryption [9] keys to encrypt the parties' private data. In particular, we utilize the following property 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

$$e(a_1 + a_2 + \cdots + a_n) = e(a_1) \times e(a_2) \times \cdots \times e(a_n) \quad (2)$$

3.2 Protocol

Without loss of generality, assuming P_l has a private distance portion of the ith training instance, s_{il}, for $i \in [1, N], l \in [1, n]$. The problem is to decide whether $\sum_{l=1}^{n} s_{il} \leq \sum_{l=1}^{n} s_{jl}$ for $i, j \in [1, N] (i \neq j)$ and select

k smallest values, without disclosing each distance portion. We will provide a solution which uses homomorphic encryption and random perturbation techniques. Before describing the protocol, we randomly select a key generator, e.g., P_n.

Protocol 1 *(Secure Computing Protocol)*

Step I: Compute $e(\sum_{l=1}^{n} s_{il})$ for $i \in [1, N]$.

1) Key and random number generation

a. P_n generates a cryptographic key pair (d, e) of a semantically-secure homomorphic encryption scheme and publishes its public key e. Let $e(.)$ denote encryption and $d(.)$ denote decryption.

b. P_l generates N random numbers r_{il}, for all $i \in [1, N], l \in [1, n]$.

2) Forward transmission

a. P_1 computes $e(s_{i1} + r_{i1})$, for $i \in [1, N]$, and sends them to P_2.

b. P_2 computes $e(s_{i1}+r_{i1}) \times e(s_{i2}+r_{i2}) = e(s_{i1} + s_{i2}+r_{i1}+r_{i2})$, where $i \in [1, N]$, and sends them to P_3.

c. Repeat (a) and (b) until P_{n-1} obtains $e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + r_{i1} + r_{i2} + \cdots + r_{i(n-1)})$, for all $i \in [1, N]$.

d. P_n computes $e(s_{in})$, $i \in [1, N]$, and sends them to P_{n-1}.

3) Backward transmission

a. P_{n-1} computes $e(-r_{i(n-1)})$, for $i \in [1, N]$ and sends them to P_{n-2}.

b. P_{n-2} computes $e(-r_{i(n-1)}) \times e(-r_{i(n-2)}) = e(-r_{i(n-1)}-r_{i(n-2)})$, $i \in [1, N]$, and sends them to P_{n-3}.

c. Repeat (a) and (b) until P_1 obtains $e_{i1} = e(-r_{i1} - r_{i2} - \cdots - r_{i(n-1)})$, for all $i \in [1, N]$.

d. P_1 sends e_{i1}, for $i \in [1, N]$, to P_{n-1}.

4) Computation of $e(\sum_{l=1}^{n} s_{il})$, for $i \in [1, N]$

a. P_{n-1} computes $e_{i(n-1)} = e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + r_{i1} + r_{i2} + \cdots + r_{i(n-1)}) \times e(s_{in}) = e(s_{i1} + s_{i2} + \cdots + s_{i(n-1)} + s_{in} + r_{i1} + r_{i2} + \cdots + r_{i(n-1)})$, $i \in [1, N]$.

b. P_{n-1} computes $e_{i(n-1)} \times e_{i1} = e(\sum_{l=1}^{n} s_{il})$, for $i \in [1, N]$ and $l \in [1, n]$.

Step II: Compute $e(\sum_{l=1}^{n} -s_{jl})$ for $j \in [1, N]$.

1) Random number generation

a. P_l generates N random numbers r'_{jl}, for all $j \in [1, N], l \in [1, n]$.

2) Forward transmission

a. P_1 computes $e(-s_{j1} + r'_{j1})$, for $j \in [1, N]$, and sends them to P_2.

b. P_2 computes $e(-s_{j1} + r'_{j1}) \times e(-s_{j2} + r'_{j2}) = e(-s_{j1} - s_{j2} + r'_{j1} + r'_{j2})$, where $j \in [1, N]$, and sends them to P_3.

c. Repeat (a) and (b) until P_{n-1} obtains $e(-s_{j1} - s_{j2} - \cdots - s_{j(n-1)} + r'_{j1} + r'_{j2} + \cdots + r'_{j(n-1)})$, for all $j \in [1, N]$.

d. P_n computes $e(-s_{jn})$, $j \in [1, N]$, and sends them to P_{n-1}.

3) Backward transmission

a. P_{n-1} computes $e(-r'_{j(n-1)})$, for $j \in [1, N]$ and sends them to P_{n-2}.

b. P_{n-2} computes $e(-r'_{j(n-1)}) \times e(-r'_{j(n-2)}) = e(-r'_{j(n-1)}-r'_{j(n-2)})$, $j \in [1, N]$, and sends them to P_{n-3}.

c. Repeat (a) and (b) until P_1 obtains $e_{j1} = e(-r'_{j1} - r'_{j2} - \cdots - r'_{j(n-1)})$, for all $j \in [1, N]$.

d. P_1 sends e_{j1}, for $j \in [1, N]$, to P_{n-1}.

4) Computation of $e(\sum_{l=1}^{n} -s_{jl})$, for $j \in [1, N]$

a. P_{n-1} computes $e_{j(n-1)} = e(-s_{j1} - s_{j2} - \cdots - s_{j(n-1)} + r'_{j1} + r'_{j2} + \cdots + r'_{j(n-1)}) \times e(-s_{jn}) = e(-s_{j1} - s_{j2} - \cdots - s_{j(n-1)} - s_{jn} + r'_{j1} + r'_{j2} + \cdots + r'_{j(n-1)})$, $j \in [1, N]$.

b. P_{n-1} computes $e_{j(n-1)} \times e_{j1} = e(\sum_{l=1}^{n} -s_{jl})$, for $j \in [1, N]$ and $l \in [1, n]$.

Step III: Compute the k nearest neighbors

1) P_{n-1} computes $e_{i(n-1)} \times e_{j(n-1)} = e(\sum_{l=1}^{n} s_{il} - \sum_{l=1}^{n} s_{jl})$, for $i, j \in [1, N]$, and collects the results into a sequence Φ which contains $N(N-1)$ elements.

2) P_{n-1} randomly permutes this sequence and obtains the permuted sequence denoted by Φ', then sends Φ' to P_n.

3) P_n decrypts each element in 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''.

4) P_n sends Φ'' to P_{n-1} who computes k smallest elements. (Details are given in Section 3.3.) They are the k nearest neighbors for a given query instance x_q. He then decides the class label for x_q.

3.3 How to Compute the Smallest k Elements

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.

Table 2: An example

	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

It means that if the qth position in sequence Φ denotes $e(\sum_{l=1}^n s_{il} - \sum_{l=1}^n s_{jl})$, then the qth position in sequence Φ''' denotes the decrypted result of $\sum_{l=1}^n s_{il} - \sum_{l=1}^n s_{jl}$. We encode it as +1 if $\sum_{l=1}^n s_{il} \geq \sum_{l=1}^n s_{jl}$, and as -1 otherwise. P_{n-1} has two sequences: one is Φ, the sequence of $e(\sum_{l=1}^n s_{il} - \sum_{l=1}^n s_{jl})$, for $i, j \in [1, N](i \neq j)$, 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 $\sum_{l=1}^n s_{il}$ is larger than $\sum_{l=1}^n s_{jl}$ by checking the corresponding value in the Φ''' sequence. For example, if the first element Φ''' is -1, P_{n-1} concludes $\sum_{l=1}^n s_{il} < \sum_{l=1}^n s_{jl}$. P_{n-1} examines the two sequences and constructs the index table (Table 1) to compute the k nearest neighbors.

In Table 1, $+1$ in entry ij indicates that the distance measure of the row (e.g., $\sum_l^n s_{il}$ of the ith row) is not less than the distance measure of a column (e.g., $\sum_l^n s_{jl}$ 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 distance measure in that row. She then selects the instances, that correspond to the k smallest weights, as the k nearest neighbors.

To make it clearer, let us illustrate it by an example. Assume that: (1) there are 4 elements denoted by $S_1 = \sum_{l=1}^n s_{1l}$, $S_2 = \sum_{l=1}^n s_{2l}$, $S_3 = \sum_{l=1}^n s_{3l}$, and $S_4 = \sum_{l=1}^n s_{4l}$. (2) $S_1 < S_4 < S_2 < S_3$; (3) the sequence Φ is $[e(S_1 - S_2), e(S_1 - S_3), e(S_1 - S_4), e(S_2 - S_3), e(S_2 - S_4), e(S_3 - S_4)]$. 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 $S_1 < S_4 < S_2 < S_3$ according to their weights, e.g., S_1 is the smallest element since its weight, which is -2, is the smallest. Thus, P_{n-1} knows the k nearest neighbors for a given query instance x_q.

In the next section, we show the correctness of protocol, the preservation of data privacy, and its complexity.

3.4 Analysis of Correctness, Privacy and Complexity

Correctness analysis Assuming all of the parties follow the protocol, the protocol correctly finds the k-nearest neighbors for a given query instance x_q. In step I, P_{n-1} obtains $e(\sum_{l=1}^n s_{il})$ for $i \in [1, N]$. In step II, P_{n-1} gets $e(\sum_{l=1}^n s_{jl})$ for $j \in [1, N]$. The key issue is that P_{n-1} actually obtains the k-nearest neighbors in step III. This property directly follows the discussion of Section 3.3.

Privacy analysis Assuming all of the parties follow the protocol, one party's distance portion cannot be disclosed to another party. In the protocol, before one party sends his private data to any other parties, she firstly adds in a random number known by her and then uses a semantically secure scheme to encrypt the data. Therefore, other parties cannot identify her actual data. Even though P_n has the decryption key d, what he can obtain is a permuted sequence of $\sum_{l=1}^n s_{il} - \sum_{l=1}^n s_{jl}$, for $i, j \in [1, N](i \neq j)$. By knowing this sequence, each single private value cannot be identified. since each private element is from the real domain. Neither can P_{n-1} obtain private values. This is because P_{n-1} knows only the following sequences: Φ, Φ'' and Φ'''. By knowing these information, private data values cannot be obtained because the equations discussed in Section 2.3 can't be constructed.

Collusion Resistance The protocol tolerates colluding parties for a single query only if the number of non-colluding parties is at least one. If P_n doesn't collude with other parties, then there is no information disclosure. We are interested in the case where P_n and P_{n-1} are the colluding parties. In this case, the colluding parties know $\sum_{l=1}^n s_{il} - \sum_{l=1}^n s_{jl}$. To exactly know a specific s_{il} or s_{jl}, for ($l \in [1, n-2]$), one needs to know exact $n-2$ distance portions. However, if there are at least one party (e.g., P_1) who doesn't collude with other parties, then the colluding parties can only obtain $s_{i1} - s_{j1}$. By knowing this value, one cannot know s_{i1} or s_{j1}. Hence, the portion s_{il} of a particular party l is not disclosed. In practice, there must be at least one party who doesn't collude with other parties, otherwise, all the parties collude with each other and they can simply share their private data. Therefore, the protocol can tolerate the colluding parties. For the multiple-query case, where each query requester can make many queries, P_n can't be the requester of a query. P_n and P_{n-1} can't collude with other parties to ensure the data privacy.

Complexity The complexity are dominated by step III. The total computation and communication costs are both $O(N^2)$.

4 Related Work

Secure Multi-Party Computation A Secure Multi-party Computation (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. The SMC problem literature was introduced by Yao [12]. It has been proved that for any polynomial function, there is a secure multi-party computation solution [6]. The approach used is as follows: the function F to be computed is firstly 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

Table 1: Index table of distance measures of N instances

	$\sum_l^n s_{1l}$	$\sum_l^n s_{2l}$	$\sum_l^n s_{3l}$	$\cdots$	$\sum_l^n s_{Nl}$
$\sum_l^n s_{1l}$	+1	+1	-1	$\cdots$	-1
$\sum_l^n s_{2l}$	-1	+1	-1	$\cdots$	-1
$\sum_l^n s_{3l}$	+1	+1	+1	$\cdots$	+1
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\sum_l^n s_{Nl}$	+1	+1	-1	$\cdots$	+1

in its generality and simplicity, is highly impractical for large datasets.

Privacy-Preserving Data Mining In early work on privacy-preserving data mining, Lindell and Pinkas [8] propose a solution to privacy-preserving classification problem using oblivious transfer protocol, a powerful tool developed by secure multi-party computation (SMC) research. The techniques based on SMC for efficiently dealing with large data sets have been addressed in [11]. In [13], a secure protocol for multi-party computation of boolean vector product with a commodity server [3] was developed. 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., [4, 14]). In addition to perturbation, aggregation of data values [10] provides another alternative to mask the actual data values. In [1], authors studied the problem of computing the kth-ranked element. Dwork and Nissim [5] 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. In this paper, we focus on privacy-preserving among the inter-party computation. In [7], Kantarcioglu and Clifton proposed an approach to solve the problem of private computation of a distributed k-nn classifier over horizontally partitioned data. In this paper, we provide a solution for building k-nn classifiers on vertically partitioned data using homomorphic encryption and random perturbation techniques.

5 Conclusion

In this paper, we consider the problem of privacy-preserving collaborative k-nearest neighbor classification. In particular, we study how multiple parties can collaboratively build a k-nearest neighbor classifier over the vertically partitioned data. We develop a secure collaborative k-nearest neighbor classification protocol based on homomorphic encryption scheme. In our protocol, the parties do not need to send all their data to a central, trusted party. Instead, we use the homomorphic encryption and random perturbation techniques to conduct the computations across the parties without compromising their data privacy. Correctness of our protocols is shown and complexity of the protocols is addressed as well.

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*, pp. 439–450. ACM Press, May 2000.

[3] D. Beaver, "Commodity-based cryptography (extended abstract)," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, El Paso, TX USA, May 4-6 1997.

[4] 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.

[5] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *CRYPTO 2004 528–544*.

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

[7] M. Kantarcioglu and C. Clifton, "Privately computing a distributed k-nn classifier," in *the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), pp.279-290, Pisa, Italy, Sept.20-24,2004.*

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

[9] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptography - EUROCRYPT '99, pp 223-238, Prague, Czech Republic*, May 1999.

[10] 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.

[11] J. Vaidya and C. W. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada.*

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

[13] Z. Zhan, L. Chang, and S. Matwin, "Privacy-preserving collaborative data mining," in *Foundation and Novel Approach in Data Mining, edited by T.Y. Lin, S. Ohsuga, C.J. Liau, and X. Hu Springer-Verlag, to appear.*

[14] Z. Zhan, L. Chang, and S. Matwin, "Privacy-preserving multi-party decision tree induction," in *the 18th annual IFIP WG 11.3 working conference on data and application security, Sitges, Catalonia, Spain*, 2004.

Justin Zhan is a part-time professor at the School of Information Technology and Engineering, University of Ottawa, Canada. His research interest contains privacy and security issues in data mining, network security and wireless network security.

LiWu Chang is a research scientist at Center for High Assurance Computer Systems of Naval Research Laboratory, USA. His research interest includes methodologies of intelligent computation, decision analysis, and secure computations.

Stan Matwin is a professor at the School of Information Technology and Engineering, University of Ottawa, Canada. His research is in machine learning, data mining, and their applications, as well as in technological aspects of Electronic Commerce.