Enhancing Privacy for Frequent Itemset Mining in Vertically Distributed Data

Luong The Dung[1], Ho Tu Bao[2], Tran Dang Hung[3]
[1] VietNam Government Information Security Commission
[2] Japan Advanced Institute of Science and Technology
[3] Hanoi National University of Education
thedung@bcy.gov.vn, bao@jaist.ac.jp, hungtd@hnue.edu.vn

***Abstract*: In this paper, we propose some protocols which allow several parties to cooperate mining frequent itemsets on their joint data, while preserving privacy of the participants. The security of the proposed protocols are better than that of previous ones. More specifically, we achieve full privacy protection of the parties while preserving efficiency as the best existing protocol.**

Keywords: Privacy preserving data mining, cryptography

I. INTRODUCTION

This paper considers a scenario in which data are maintained by different parties. Those parties wish to share their data in certain ways to benefit each other for association rules mining that are standard algorithms are based on finding of frequent itemsets. We consider how to preserve privacy in frequent itemsets mining for distributed database [4]. That is, we study how a group of parties cooperatively mine frequent itemsets in distributed setting without revealing each party's portion of the data to the other.

Privacy-preserving association rules/itemsets mining is one of the most active topics in PPDM (Privacy Preserving Data Mining). Some examples are [5,15,20,25]. However, the purpose of these studies is to hide the sensitive rules, rather than privacy in distributed mining. In distributed setting, a simple method of PPDM in distributed databases is to perturb the original data. The procedure of transforming the original database into a new one that hides some information is called the perturbation process [1,2,3,6,7,8,14,17,18]. An association rules mining process on the perturbed database can reduce the risk of revealing the privacy information. Although these solutions are very efficient, the mining result on the perturbed database is less accurate than that of the original database.

Some other solutions are based on the secure distributed computation for the privacy frequent itemset mining. In [12] Kantarcioglu and Clifton presented a solution for horizontally partitioned data, this solution was designed based on Yao's generic secure computation protocol. However, the generic secure computation protocols are very highly expensive that showed in [9]. Thus, some other solutions for horizontally partitioned data have a better performance proposed in [26], [13].

In our work, we present some protocols for vertically partitioned data: a portion of each transaction is present at each party, but no party contains complete information for any transaction. There exists several works directly related to our works. Vaidya and Clifton proposed the secure set intersection cardinality method that was used for association rule mining in vertically partitioned data [23]. This method has been used to preserve privacy in decision tree learning [24] as well. The drawback of the secure set intersection cardinality method is that it has high computational and communication complexity. In [22], the authors presented an algebraic solution to dealing with vertically partitioned data. However, studies have shown that this solution can disclose many linear combinations from each party's private data. Zhong [27] proposed a frequent itemset mining protocol that is based on homomorphic encryption techniques for vertically partitioned data involving multiple parties. This protocol is more efficient than other protocols, but requires that one

party has to play the role as a trusted party. The protocol proposed in [10] is as efficient as Zhong's solution, but only resist the collusion of at most corrupted parties among participants. In [19], authors proposed a privacy preserving method for mining frequent itemsets; it works in presence of malicious participants and has property to be against collusion up to n-1 parties. However, it only obtains this property in computational models that their outputs are not revealed the support count information. Moreover, it requires two different protocols for two-party and multi-party, respectively. Thus, it has more complexity when using in a iteratively algorithm for mining all frequent itemsets since it requires the system have to set and store two difference set of parameters.

In this paper, we propose the protocols for privacy preserving frequent mining involving multiple parties. The proposed protocols do not require any trusted party while they can protect the privacy of each party against the collusion of any group of corrupted parties. These protocols also have the same communication cost and computational complexity as Zhong's protocol. In addition, we give two protocols that allow the parties to be able to select one of two privacy level corresponding to two protocols; one of them reveals only the support count, and the other reveals nothing.

The rest of this paper is organized as follows: Section 2 presents the problem fomulation. Section 3 presents the preliminaries. In section 4, we introduce a privacy preserving protocol for frequent itemset identifying without disclosing the support count. In Section 5, we present another protocol for identifying frequent itemset based on the support count computation, and after that we show that this protocol can run on the binary communication structure to obtain the better performance in Section 6. In each section, we prove the correctness and privacy properties of each protocol, evaluate the efficiency of each protocols based on estimate, and present experimental results of the efficiency as well. The last section concludes our work in this paper.

II. PROBLEM FORMULATION

A. Association rules and frequent itemset

The association rules mining problem can be formally stated in [4]. Let $I = \{I_1, I_2, ..., I_d\}$ be the set of all items; DB be a database of m transactions, where each transaction T_j ($j = 1, .., m$) is a set of items such that $T_j \subseteq I$. We say that the transaction T_j contains X , a set of some items in I , if and only if $X \subseteq T_j$. The problem is to find the association rules that have an implication of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$.

We say that such an association rule has a support $s\%$ if the percentage of records containing both X and Y in DB is $s\%$. The rule has confidence $c\%$ if the percentage of records containing both X and Y with regard to the total number of records containing X is $c\%$.

Let $X.count$ be the support count of the itemset X , which stands for the number of transactions containing the set X in DB , and $|DB|$ denotes the total number of transactions in DB . The strong association rules are required to meet a minimum support (s_{min}) and a minimum confidence (c_{min}) defined by the miner.

A set of items is referred to as an itemset. An item set that contains k items is a k -itemset. The support count of an itemset is the number of transactions containing the itemset. The minimum support count is defined as $t = s_{min} |DB|$. An itemset is frequent if its support count is not less than the minimum support count. Association rule mining is a two-step process: (1) Finding all frequent itemsets; (2) Generating strong association rules from the frequent itemsets. The main technical problem in association-rule mining is the frequent itemset identification.

B. Frequent item set identifying in vertically distributed data

In association rules mining, the transaction set DB is represented by a boolean matrix M . Each row

of the matrix corresponds to a transaction, while each column corresponds to an item. Denote an element of the matrix by $M(j,l)$, where $M(j,l)=1$ if the j^{th} transaction contains the l^{th} item I_l, it is 0 otherwise. Assume that M is vertically distributed on n parties $P_1, ..., P_n$. Each party owns the set of columns of the matrix M, denoted by $C_1, ..., C_n$, where $C_1 \cup ... \cup C_n = [1,m]$ and $C_i \cap C_j = \phi$ with any $i \neq j$.

We consider a scenario that the parties wish to find the frequent itemsets from DB, where DB is called the joint data set of all parties. Our aim is to design distributed protocols to obtain the frequent itemsets while preserving privacy of each party's data. We consider privacy as protecting individual data records as well as protecting information about the local support count of the frequent itemsets of each party and even the global support count of the joint database.

For this problem, we study an itemset $X = \{I_l \mid l \in C\}$, where C be the set of columns corresponding to X. Note that the column set C of this itemset is partitioned into the subsets: $C \cap C_1, ..., C \cap C_n$, which are owned by parties $P_1, ..., P_n$, respectively. The support count s of X can be computed by

$$s = \sum_{j=1}^{m} \prod_{l \in C \cap C_1} M(j,l) ... \prod_{l \in C \cap C_n} M(j,l)$$

Let $u_{ij} = \prod_{l \in C \cap C_i} M(j,l)$. Note that each P_i can privately compute all u_{ij}'s. Therefore, identifying if the itemset X be frequent can be formulated as follows.

Definition 1 *Assume that there are n parties, each party P_i has a private vector $U_i = (u_{i1},...,u_{im})$, where each $u_{ij} \in \{0,1\}$, $i = 1,..,n$, and $j = 1,..,m$. For a public threshold $t = s_{min} \mid DB \mid$, the privacy-preserving frequent itmset identifying problem is to check if $s = \sum_{j=1}^{m} \prod_{i=1}^{n} u_{ij} \geq t$ without disclosing any privacy information of participants.*

III. COMPUTATIONAL AND PRIVACY MODEL

Privacy preserving protocols implement among a set of semi-honest parities. We assume that all parties are online, where each party has a communication channel with each other. To be applicable, we require that the protocol can ensure users' privacy in an environment that doesn't have any secure channels between the party.

The privacy preservation of this proposed protocol is based on the semi-honest security model [9]. Thus, we consider the possibility that some corrupted parties share their data with each other to derive the private data of the honest parties. One requirement is that no other private information about the honest parties should be revealed, except the final result.

In particular, we are giving the privacy definition for the protocol with the following parameter model. There are n parties involved in the protocol. Each party P_i has the private input U_i, where U_i is a the binary vector. We assume that prior to the protocol, each party has obtained the key pairs for the Elgamal encryption scheme: the private key x_i and the public key y_i. Each party's public key has known by members in the system, while the private key is secretly kept. The definition of privacy preserving in semi-honest model is as follows:

Definition 2 *A protocol for the frequent itemset mining problem in the definition 4.1 is said to protects each party's privacy against t corrupted parties in the semi-honest model if, for all $I \subset \{1,...,K\}$ such that $\mid I \mid = t$, there exists a probabilistic polynomial-time algorithm M such that*

$$\{M(f,[U_i,P_i]_{i\in I}, y_i \mid_{i\in I}\} \overset{c}{\equiv} \{\{P_i\}_{i\in I}[U_i,x_i]_{i=1}^{n}\}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability.

Basically, the definition 4.2 states that the computation is secure if the joint view of the corrupted parties during the execution of the protocol can be effectively simulated by a simulator. The corrupted parties have observed in the protocol using only the result f, the corrupted parties' knowledge, and the public keys. Therefore, the corrupted parties can not learn anything from f. By the definition, it indicates that there exists a simulator satisfied the above equation.

IV. SUPPORT COUNT PRESERVING PROTOCOL

A. Overview

Assuming that X is an frequent itemset, we have $t \le s \le m$. Thus, there exists a 0 in the list $\lambda = \{ \lambda_1 = s-1-t , \lambda_2 = s-2-t , ..., \lambda_k = s-k-t \}$, where $k = m-t$. If s is known by all parties, this problem can be solved immediately. However, for our purpose with strong privacy, this vector cannot be revealed.

Therefore, the basic idea of the protocol is follows. Let p and q be two primes such that $q \mid (p-1)$, let G be a subgroup of Z_p^* of order q, and g is a generator of G. All computations in this chapter always take in Z_p. The proposed protocol is to implement the following function

$$(U_1, U_2, ..., U_n) \mapsto (g^{r_1 \lambda_{\pi(1)}}, ..., g^{r_k \lambda_{\pi(k)}})$$

where $(\lambda_{\pi(1)}, ..., \lambda_{\pi(k)})$ is a random permutation of $(\lambda_1, ..., \lambda_k)$. Assume that $r_j = \sum_{i=1}^{n} r_{ij}$, where r_{ij} is uniformly generated from $[1, q-1]$ by Party i. Then, the parties can check whether existing $\lambda_j = 0$ that is equal with $g^{r_j \lambda_j} = g^0 = 1$. Clearly, when $\lambda_j \neq 0$, $g^{r_j \lambda_j}$ is a random number, the protocol does not leak any other information except the final result.

To achieve this goal, we use extensively ElGamal encryption together with a rerandomization technique and a joint decryption technique. These techniques have been used to some other problems, e.g., [11,28,29].

The joint decryption technique and ElGamal encryption. We assume that each party has a private key $x_i \in [1, q-1]$ corresponding to the public key $y_i = g^{x_i}$. Without loss of generality, we assume in the sequel that the parties are numbered from 1 to n. We define

$$x = \sum_{i=1}^{n} x_i$$

$$y = \prod_{i=1}^{n} y_i = g^x$$

In our protocol, the parties use the public value y as a public key to encrypt their data, and we also use variants of the ElGamal encryption in which the messages m is changed to g^m [11] before encrypting:

$$C = (a, h) = (g^m y^r, g^r)$$

Decrypting these encryptions needs the private key x, which is not known to any individual party. To decrypt a ciphertext C, all participants need to jointly perform the distributed decryption. That is, each participant computes and publishes h^{x_i}. The plaintext can then be decrypted by computing

$$\frac{a}{\prod h^{x_i}} = \frac{g^m g^{r \sum x_i}}{g^{r \sum x_i}} = g^m$$

The decryption algorithm is not efficient when m is large, however, in our protocol it is an efficient algorithm, because we only uses the private key to decide whether a ciphertext decrypts to 1 (or g^0), and we use this property at the final step of the protocol.

Rerandomization technique. Another key technique in our scheme is known as rerandomization that is introduced in [16] for privacy protection. A rerandomization is multi-party protocol that involves several mix servers. The input to the protocol is a list of ciphertext items and the output is a new, permutated list of those ciphertext items such. The

related operations are the permutation of the order of items and the re-encryption. Therefore, it randomly rearranges the order of items. If we re-randomise and permute a sequence of cipher-texts, then we get another sequence of cipher-texts with the same multiset of clear-texts but in a different order. The security of this technique is characterized by Looking at these two sequences of cipher-texts, the adversary cannot determine any information about the correspondence between the new cipher-text corresponding and the old cipher-text.

In the proposed protocol, we use a rerandomization technique based on the ElGamal cipher, in which each party plays the role as a mix server. Each party implements the permutation task. In the following, we describe how this technique works. The input to this technique consists of a list of ElGamal ciphertext $\{(a_1,h_1),...(a_m,h_m)\}$ encrypted by the joint public key y. The corresponding private key is x. Then, each participant performs a random permutation on this input list in turn by re-encrypting every ciphertext and output them randomly. The final output then be a sequence $\{(a_{\pi(1)}',h_{\pi(1)}'),...(a_{\pi(m)}',h_{\pi(m)}')\}$, this represents a random re-encryption of $\{(a_1,h_1),...(a_m,h_m)\}$, and π is a random permutation function on the set of k elements.

B. Protocol design

Basically, the idea of our protocol consists of four phases as follows (The detailed protocol is implemented in Figure 4.2). The first phase is to obtain the joint encryption of the dot product (support count s) of all vectors from parties using multiplicative monomorphic property and additive monomorphic property of the encryption scheme. The second phase creates encryptions of the masked values $r_j\lambda_j$ ($j=\{1,...k\}$). To obtain this result, each party divide the first component of the encryption of s by values $\{g^{t+1},...,g^{t+k}\}$, and randomize it by the random values r_{ij}. Then, the parties follow one communication rounds to connect all encryptions to obtain the encryptions of $r_j\lambda_j$. The third phase, the parties execute the n round to permute and randomise the set of the encryptions using the re-randomisation technique. The final phase, the parties jointly decrypt the received new encryptions, which are in an independent order of the original encryptions. They check if existing a decryption are equal to 1 (g^0). The support count preserving protocol is presented as follows.

Input: There are n parties, each party P_i has $U_i=(u_{i1},...u_{im})$ $(u_{ij}\in\{0,1\})$

Output: $Check\in\{True,False\}$

• **Phase 1**. Encryption and connection. For $j=1,...,m$

- For $i=1,...,n$: P_i computes

$C_i(j)\overset{def}{=}(a_{ij},h_{ij})=(y^{\alpha_{ij}},g^{\alpha_{ij}})$, where α_{ij} is randomly picked from $[1,q-1]$.

- P_1 computes $C_j\overset{def}{=}(a_j,h_j)=(g^{u_{1j}}a_{1j},h_{1j})$ and sends C_j to P_2.

- For $i=2,...,n$: P_i computes $C_j=(a_j,h_j)=(a_j^{u_{ij}}a_{ij},h_j^{u_{ij}}h_{ij})$ and sends C_j to $P_{i+1(\bmod n)}$

- P_1 computes $C=(a,h)=(\prod_{j=1}^{m}a_j\prod_{j=1}^{m}h_j)$ and broadcasts it.

• **Phase 2.** Encryption randomization. For $j=1,...,k$

- For $i=1,...,n$: P_i computes $C_i(j)=(a_{ij},h_{ij})=(a/g^{j+t})^{r_{ij}},h^{r_{ij}})$, where r_{ij} is uniformly chosen from $[1,q-1]$,

- P_1 sets $C_j=(a_j,h_j)=(a_{1j},h_{1j})$ and sends C_j to P_2,

- For $i=2,...,n$: P_i computes $C_j=(a_j,h_j)=(a_ja_{ij},h_jh_{ij})$ and sends C_j to $P_{i+1(\bmod n)}$

• **Phase 3**. Randomization and permutation. For $i=1,...,n$

- P_i computes: for $j=1,...,k$,

$$R_j=(R_j^{(1)},R_j^{(2)})=(a_{\pi_i(j)}y^{\delta_{\pi_i(j)}},h_{\pi_i(j)}g^{\delta_{\pi_i(j)}})$$

where π_i is an permutation on $\{1,..,k\}$

and $\delta_{\pi_i(j)}$ is uniformly chosen from $[1,q-1]$.

- P_i sets: for $j=1,...,k$, $C_j=R_j$ and sends C_j to $P_{i+1(\mathrm{mod}\,n)}$

• **Phase 4**. Decryption. For $j=1,...,k$

- P_1 broadcasts h_j to the other parties

- For $i=1,...,n$: P_i computes $h_{ij}=(h_j)^{x_i}$

- P_1 sets $h_j=h_{1j}$ and sends it to P_2

- For $i=2,...,n$: P_i computes $h_j=h_{ij}h_j$, and sends h_j to $P_{i+1(\mathrm{mod}\,n)}$

- P_1 computes $d_j=a_j/h_j$, if $d_j=1$ then $Check=True$ else $Check=False$

- P_1 outputs $Check$

C. Correctness Analysis

Theorem 1 *If all participants follow the protocol and there exists one plaintext ``1'' in the decryption list* $\{d_1, ..., d_m\}$, *then* $t\le s\le n$. *If there is no plaintext ``1'' existing in the decryption list, then* $s<t$.

Proof. To complete the proof of theorem, we need to prove that $d_j=g^{r_{\pi(j)}\lambda_{\pi(j)}}$, $j=\{0,...,k\}$. Indeed,

In the phase 1, starting from P_1, we have $C_j=(a_j,h_j)=(g^{u_{1j}}y^{\alpha_{1j}},g^{\alpha_{1j}})$. Each P_i ($i=2,...,n$) receives $C_j=(a_j,h_j)$ from P_{i-1} and computes a new value $C_j=(a_j,h_j)=(a_j^{u_{ij}}a_{ij},h_j^{u_{ij}}h_{ij})$. Therefore at the end of the phase 1, using the inductive method, we have

$$a=\prod_{j=1}^{m}a_j=g^{\sum_{j=1}^{m}\prod_{i=1}^{n}u_{ij}}y^{\sum_{j=1}^{m}(\sum_{i=1}^{n-1}\prod_{k=i+1}^{n}u_{kj}\alpha_{ij}+\alpha_{nj})}=g^{s}y^{\theta}$$

$$h=\prod_{j=1}^{m}h_j=g^{\sum_{j=1}^{m}(\sum_{i=1}^{n-1}\prod_{k=i+1}^{n}u_{kj}\alpha_{ij}+\alpha_{nj})}=g^{\theta}$$

where, denote $s=\sum_{i=1}^{n}\prod_{j=1}^{m}u_{ij}$ and $\theta=\sum_{j=1}^{m}(\sum_{i=1}^{n-1}\prod_{k=i+1}^{n}u_{kj}\alpha_{ij}+\alpha_{nj})$.

In Phase 2, each party firstly obtains a set of encryptions of $\{\lambda_1=s-t-1,\ \lambda_2=s-t-2,\ ...,\ \lambda_k=s-t-k\}$ by computing $(a_j,h_j)=(a/g^{j+t})^{r_{ij}},(h)^{r_{ij}})$, and then the parties follow a round to compute the product of all encryptions, thus the final result of this phase is k randomized encryptions denoted by $(C_1,...,C_k)$ as follows.

$$C_j=(a_j,h_j)=(\prod_{i=1}^{n}(a/g^{j+t})^{r_{ij}},\prod_{i=1}^{n}(h)^{r_{ij}}))$$

$$=(\prod_{i=1}^{n}g^{\lambda_j r_{ij}}y^{\alpha_{ij}},\prod_{i=1}^{n}g^{\alpha_{ij}})$$

$$=(g^{\lambda_j\sum_{i=1}^{n}r_{ij}}y^{\theta\sum_{i=1}^{n}r_{ij}},g^{\theta\sum_{i=1}^{n}r_{ij}})$$

$$=(g^{\lambda_j r_j}y^{\alpha_j},g^{\alpha_j})$$

where $r_j=\sum_{i=1}^{n}r_{ij}$.

In Phase 3, the protocol receives the set $(C_1,...,C_k)$ and permutes this set n times. Thus, the parties obtain $C_j=(a_{\pi(j)},h_{\pi(j)})$, where π is the final result of the permutation of the n times.

$$a_{\pi(j)}=g^{\lambda_{\pi(j)}r_{\pi(j)}}y^{\theta_{\pi(j)}+\sum_{i=1}^{n}\delta_{\pi_j(i)}};$$

$$h_{\pi(j)}=g^{\theta_{\pi(j)}+\sum_{i=1}^{n}\delta_{\pi_j(i)}}$$

Therefore, in Phase 4, the parties obtain

$$d_j = a_j / \prod_{i=1}^{n} (h_j)^{x_i}$$

$$= \frac{g^{r_{\pi(j)}\lambda_{\pi(j)}} y^{(\theta_{\pi(j)} + \sum_{i=1}^{n} \delta_{\pi_i(j)})}}{g^{(\theta_{\pi(j)} + \sum_{i=1}^{n} \delta_{\pi_i(j)}) \sum_{i=1}^{n} x_i}}$$

$$= \frac{g^{r_{\pi(j)}\lambda_{\pi(j)}} g^{\sum_{i=1}^{n} x_i (\theta_{\pi(j)} + \sum_{i=1}^{n} \delta_{\pi_i(j)})}}{g^{(\theta_{\pi(j)} + \sum_{i=1}^{n} \delta_{\pi_i(j)}) \sum_{i=1}^{n} x_i}}$$

$$= g^{r_{\pi(j)}\lambda_{\pi(j)}}$$

This finishes the proof.

D. Privacy Analysis

In the proposed protocol, only the final results are published at the end of the process, and other information, such as each party' the binary vector and even the support count of the itemset are always kept secret. In such a way, we ensure the privacy protection of each party in maximum degree. In the protocol, each party publishes the binary vector in a particular ElGamal encrypted vector form using a public key y, instead of plaintexts. Without the knowledge of the corresponding private key x, nobody can get the values of parties without breaking the ElGamal cryptosystem, which is assumed to be infeasible. Making use of the homomorphic properties of ElGamal encryption, we get the randomized encryption list of λ_j. By employing the re-randomization technique, and the joint decryption technique we can identify if an itemset is frequent, without leaking any further information. Since the basic building blocks of the encryption scheme and the re-randomization technique are proved to be secure, it is easily to deduce that our scheme is secure against any passive attack.

The important security feature of our protocol which is better than the previous method is that we achieve the full privacy protection of the parties. That is, we do not assume the existence of any kind of trusted parties. Moreover, no collusion of parties can possibly lead to the revelation of any private information, unless all parties together form a single collusion, which is not significant.

Theorem 2 *The protocol in Subsection IV-B preserves the privacy of the honest parties against the collusion, up to* $n-1$ *corrupted parties.*

Proof. To prove this theorem, we will design a simulator M that simulates the joint view of the miner and the corrupted users by a probabilistic polynomial-time algorithm. Subsequently, this simulator is combined with a simulator for the ElGamal cipher-texts to obtain a completed simulator. To do so, basically we show a polynomial-time algorithm for computing the joint view of the corrupted parties. The computation of the algorithm is based on what the corrupted parties have observed in the protocol using only the final result, the corrupted parties' information, and the public keys. The algorithm outputs the simulated values for the encryptions generated by a simulator of ElGamal encryptions. Without loss of generality, it suffices to consider the case in which only one party is honest, indexed by k. We will show the simulator that its steps follow the protocol's steps.

For each $C_k(j)$ of P_k in Phase 1, M computes

$$C_k(j) = (a_{kj}, h_{kj}) = \left(\frac{a_{kj}}{h_{kj}^{(\sum_{i=1,\dots n, i\neq k} x_i)}}, \frac{h_{kj}}{g^{(\sum_{i=1,\dots n, i\neq k} \alpha_{ij})}}\right)$$

$$= (g^{u_{kj}} g^{\gamma_j}, g^{\gamma_j})$$

where γ_j denotes a random number in $[1,..,q-1]$. Note that Elgamal encryption is semantically secure under the DDH, thus, M can simulates C_j by a random ciphertext of the Elgamal encryption.

To simulate all $C_j = (a_j, h_j)$ of P_k in Phases 2, we note that given $(g^{e_1}, g^{e_2}, g^{re_1}, g^{re_2})$, since the computational indistinguishability follows from the

semantic security of ElGamal encryption, which is well known under DDH, $(g^{e_1}, g^{e_2}, g^{re_1}, g^{re_2})$ and $(g^{e_1}, g^{e_2}, g^{e_3}, g^{e_4})$ are not computationally distinguishable. Therefore, M can simulate each C_{1j} by $C_{ij}' = (g^{e_3}, g^{e_4})$, where e_3 and e_4 are randomly and independently chosen in $[1, q-1]$.

In Phase 3, M simulates all $R_j = (R_j^{(1)}, R_j^{(2)})$ by using a randomly permuted vector of m ElGamal cipher-texts. If X is frequent, among these m cipher-texts, the one of encryptions of g^0, all the remaining cipher-texts should be the random number. If $s < t$, all the cipher-texts should be the encryptions of random numbers.

In Phase 4, each message d_j in the protocol can be simulated as in Phase 3, it is similar to the choosing a uniformly random element of G. This finishes the simulation algorithm.

E. Performance analysis

Communication Analysis. Communication protocols of this type are generally analyzed either based on the communication cost or number of modular operations performed.

The basic communication cost for a single party is as follows. In Phase 1, a set of $4m$ messages is transmitted in each of the n rounds and broadcasted by P_1. In each round i, the party i sends the set to the next party. Similarity, Phase 2 requires the transmission of $2k$ messages. Phase 3 transmits $2k$ messages and Phase 4 is $2k$ messages. The entire protocol is symmetric, so all of the parties transmit the equal amounts of data. To calculate the total communication cost, we simply multiply the single party cost by n. Thus, let us assume that the size of the parties's key is K bits, the upper bound on the total communication cost of the protocol is in $O(nmK)$. Note that, this is much lower than the $O(mn^2K)$ complexity of Vaidya and Clifton [23] and is equivalent to the one by Zhong [27]. The detailed communication cost is showed in Table 1.

Table 1. The communication cost

The number of rounds	$n = O(n)$
The number of messages	$n(4m+6k) = O(mn)$
The number of bits	$n(4m+6k)K = O(nmK)$

Computational complexity evaluation. In this section, we show the results of the complexity estimation of the protocol and the efficiency measurement of the protocol in practice. In the proposed protocol, the computational cost of all parties in Phase 1 is $2nm$ modular exponentiations and $2nm$ modular multiplication, and at Phase 2 is $2nk$ modular exponentiations and kn modular multiplication inversions. The computational cost of the parties is at most of all $2nk$ modular multiplications and $2nk$ modular exponentiations in Phase 3, and the parties use kn modular multiplication inversions for Phase 4. We conclude that the total computational complexity is bounded by $O(nm)$ modular exponentiations, $O(mn)$ modular multiplications and $O(nm)$ modular multiplication inversions (Table 1). Note that these computational costs do not include the overhead of key generation and computing the parameter y. However, generating these parameters belongs to the preparation period of the mining process. Therefore, it can be implemented before the protocol is executed without affecting the computation time of the protocol.

Table 2. The complexity of the support count preserving protocol

	Exp.	**Mul.**	**Inv.**
# Operations	*n(2m+4k)*	*n(2m+4k)*	*2kn*
# Rounds	*n*	*n*	*n*
The average	*(2m+4k)*	*(2m+4k)*	*2k*
Estimation	*O(nm)*	*O(nm)*	*O(nk)*

Note that our protocol follows a communication ring with n parties, a mount of calculations of each parties is equal. Thus it is regarded as the protocol run n rounds, each round consists of computational complexities of $O(m)$ modular multiplications, $O(m)$ modular exponentiations and $O(k)$ modular inversions.

Note that time for modular multiplications is much lower than exponentiation and inversion operations, thus we can say that the complexity of the protocol is bounded by $O(mn)$ and exponentiations and $O(mn)$ inversions. However, these operations can be computed concurrently. Therefore, the overall computational complexity is $O(m)$, which is much lower than the $O(mn)$ complexity of Vaidya and Clifton [23] and is equivalent to the one by Zhong [27].

For evaluating the efficiency of the protocol in practice, we build an experiment on the privacy preserving frequent itemset mining in the C# language of Microsoft Visual Studio 2005 environment. All experiments are performed on the Window XP operating system with Intel core 2 duo E7500 2.93GHz and 2GB memory. As communication complexity depends on the network performance and physical distance of two parties, we simply considered parties as threads that exchange data directly by shared memory method. The used cryptographic functions are derived from Open OPenGPGBBlackbox Library.

We measure the computation cost of the protocol for the number of the difference parties from 2 to 10. Before executing the protocol, we generate a pair of keys for each party with $p = 1024$ and $q = 160$ bits.

Table 2 illustrates our measurements of all parties's computation time all of phases: it is in regard to m and n, for a typical scenario, where $m = 1000$, $n = 10$, and we fix $k = \frac{20}{100} m = 200$. The computation time of all parties is about 26.5 seconds.

Table 3. The parties's time for the support count preserving protocol

	m				
n	**200**	**400**	**600**	**800**	**1000**
2	1.4	2.7	3.8	4.8	5.1
3	1.5	3.1	4.9	6.3	7.7
5	2.2	4.4	7.5	10.4	13.9
10	6.0	10.2	15.2	20.3	26.5

V. SUPPORT COUNT COMPUTATION-BASED PROTOCOL

A. Overview

Recall that, the problem is to decide if $s = \sum_{j=1}^{m} \prod_{i=1}^{n} u_{ij} \geq t$, where each $U_i = (u_{i1}, ..., u_{im})$ ($u_{ij} \in \{0,1\}$) is owned by party P_i. Let $\lambda_j = \sum_{i=1}^{n} u_{ij} - n$ ($j = 1, ..., m$), note that $\prod_{i=1}^{n} u_{ij} = 1$ as long as $\lambda_j = 0$. Therefore, s is the number of $\lambda_j = 0$. We design a protocol to compute this value, our basis idea is that, the parties should obtain a random permutation of the set ($g^{\lambda_{\pi(1)}}$, .., $g^{\lambda_{\pi(m)}}$), where ($\lambda_{\pi(1)}$, ..., $\lambda_{\pi(m)}$) is a random permutation of (λ_1, ..., λ_m). In other words, We need a protocol to implement the following function

$$(U_1, U_2, ..., U_n) \mapsto (g^{\lambda_{\pi(1)}}, ..., g^{\lambda_{\pi(m)}})$$

If we obtain this result, we obtain two goals. First, the parties can be counted the number of $\lambda_j = 0$ that is equal with $g^{\lambda_j} = g^0 = 1$. After obtaining support counts we can compare it to the threshold t. Second, when $\lambda_j \neq 0$, g^{λ_j} is a random number, so we obtain the privacy goal as well, because the parties can not know g^{λ_j} generated from which λ_j.

To achieve this goal, we use variants of the ElGamal encryption as used in the above sections. Basically, the algorithm works as follows.

B. Protocol Design

Basically, the idea of protocol is as follows. For each value u_{ij}, the party holding the value u_{ij} encrypts this value using the joint public key of parties. Note that, without the help of all parties, nobody can decrypt any of this encryption. By additional monomorphic property of Elgamal scheme, the parties follow n communication rounds to connect all encryptions of the binary values u_{ij} to obtain the

encryptions of $\sum_{i=1}^{n} u_{ij}$. At the end of this step, the parties obtain the m encryptions of $\lambda_1, ..., \lambda_m$.

The parties execute the n round to permute and randomise the set of the encryptions of $\lambda_1', ..., \lambda_m'$. Finally, the parties jointly decrypt the received new encryptions, which are in an independent order of the original encryptions. They count how many of the decryptions are equal to 1 (g^0). This number is equal to the support count. The detailed protocol is implemented as follows.

Input: There are n parties, each party P_i has $U_i = (u_{i1}, ...u_{im})$ $(u_{ij} \in \{0,1\})$

Ouput: $s = \sum_{i=1}^{m} \prod_{j}^{n} u_{ij}$.

• **Phase 1**. Encryption and connection: For $j = 1, ..., m$

- For $i = 1$, ..., n : each P_i computes $C_i(j) = (a_{ij}, \tilde{h}_{ij}) = (g^{u_{ij}} y^{\alpha_{ij}}, g^{\alpha_{ij}})$, where α_{ij} is randomly chosen from $[1, q-1]$.
- P_1 computes $C_j = (a_j, h_j) = (g^{-n} a_{1j}, h_{1j})$ and sends C_j to P_2
- For $i = 2, ..., n$: each P_i computes $C_j = (a_j, h_j) = (a_j a_{ij}, h_j h_{ij})$ and sends it to $P_{i+1(\mathrm{mod}\, n)}$

• **Phase 2**. Randomization and permutation.
For $i = 1, ..., n$,

- P_i computes: for $j = 1, ..., m$,

$R_j = (R_j^{(1)}, R_j^{(2)}) = (a_{\pi_i(j)} y^{\delta_{\pi_i(j)}}, h_{\pi_i(j)} g^{\delta \pi_i(j)})$

where π_i is an permutation on $\{1, .., m\}$ and δ_j is uniformly chosen from $[1, q-1]$.

- P_i sets: for $j = 1, ..., m$, $C_j = R_j$ and sends C_j to $P_{i+1(\mathrm{mod}\, n)}$.

• **Phase 3**. The key component computation.
For $j = 1, ..., m$,

- P_1 broadcasts h_j to the other parties
- For $i = 1, ..., n$: each P_i computes $h_{ij} = (h_j)^{x_i}$
- P_1 sets $h_j = h_{1j}$, and sends h_j to P_2
- For $i = 2, ..., n$: each P_i computes $h_j = h_j h_{ij}$ and sends h_j to $P_{i+1(\mathrm{mod}\, n)}$

• Phase 4. Decryption. P_1 does:

- $s = 0$
- For $j = 1, ..., m$: $d_j = a_j / h_j$ if $d_j = 1$ then $s = s + 1$;
- Output s

C. Correctness Analysis

Theorem 3 *If all participants follow the protocol, the number of plantexts ``1" in the decryption list $\{d_1, d_2, ..., d_m\}$ is the support count.*

Proof. The end of phases 1, the parties obtain

$$C_j = (a_j, h_j) = (g^{\sum_{i=1}^{n} u_{ij} - n}, y^{\sum_{i=1}^{n} \alpha_{ij}}, g^{\sum_{i=1}^{n} \alpha_{ij}})$$

$$= (g^{\lambda_j} y^{\theta_j}, g^{\theta_j})$$

Where, denote $\lambda_j = \sum_{i=1}^{n} u_{ij} - n$ and $\theta_j = \sum_{i=1}^{n} \alpha_{ij}$

In Phase 2, the protocol receives the set (C_1, ..., C_m) and permutes this set n times. Thus, the parties obtain $C_j = (a_{\pi(j)}, h_{\pi(j)})$, where, $\pi(j)$ is the final result of the permutation of the n times.

$$a_{\pi(j)} = g^{\lambda_{\pi(j)}} y^{\theta_{\pi(j)} \sum_{i=1}^{n} \delta_{\pi_i(j)}};$$

$$h_{\pi(j)} = g^{\theta_{\pi(j)} \sum_{i=1}^{n} \delta_{\pi_i(j)}}$$

The result of phase 3 is $\prod_{i=1}^{n} (h_j)^{x_i}$, where h_j is received from Phase 2. Therefore, in Phase 4, the parties obtain

$$d_j = a_j / \prod_{i=1}^{n} (h_j)^{x_i}$$

$$= \frac{g^{\lambda_{\pi(j)}} y^{(\theta_{\pi(j)} \sum_{i=1}^{n} \delta_{\pi_i(j)})}}{g^{(\theta_{\pi(j)} \sum_{i=1}^{n} \delta_{\pi_i(j)}) \sum_{i=1}^{n} x_i}}$$

$$= \frac{g^{\lambda_{\pi(j)}} g^{\sum_{i=1}^{n} x_i (\theta_{\pi(j)} \sum_{i=1}^{n} \delta_{\pi_i(j)})}}{g^{(\theta_{\pi(j)} \sum_{i=1}^{n} \delta_{\pi_i(j)}) \sum_{i=1}^{n} x_i}}$$

$$= g^{\lambda_{\pi(j)}}$$

If $d_j = 1$, C_j is the encryption of g^0 that means $\lambda_j = 0$. It further follows that the tuple occurs at record of the data set and so the frequency f is increased to 1. The final result in Phase 4 is correct.

D. Privacy Analysis

Theorem 4 *The protocol in Subsection V-B preserves the privacy of the honest parties against the collusion, up to $n-1$ corrupted parties.*

Sketch of proof. We recall that in our scheme, each party submits an encrypted vector of m ElGamal encryptions of $u_i j$. The only one way to know the true value of a party is to decrypt encryptions using the joint private key x. In proposed protocol, we assume all parties is honest without trusted party to create the key pairs, which means that each party self generates its key pair and after that all parties jointly compute the joint public key y. Therefore, no individual knows the private key (x) corresponding to y, and the only way to decrypt the encrypted vector is that all parties participate in the computing using their secret keys. Therefore, any group of colluding parties cannot open any specific encrypted value, without the cooperation of all other parties. While employing the re-randomization technique, we use a similar idea. That is, we do not suggest the existence of any mix server and the random permutation is instead performed by all parties one by one. In such way, even $n-1$ parties collude, they still cannot decide the correspondence between the input ciphertexts and the output re-encrypted ciphertexts. Note that this theorem can be also proved by constructing a simulator M based the simulators of the Elgamal encryption.

E. Performance analysis

Communication Analysis. The main communication occurs at Phases 1, 2, and 3. The $2m$ messages are transmitted in each of the n rounds of Phase 1. In each round i, the party i sends a set of encryptions to the next party. Similarity, the transmission in Phase 2 is $2m$ messages. Phase 3 transmits $2m$ messages. The entire protocol is symmetric, so all of the parties transmit equal amounts of data. So, to calculate the total communication cost, we simply multiply the single party cost by n. The upper bound on the total communication cost of the protocol is in Table 3. We can see that the communication overhead of this protocol is less than the support count preserving protocol.

Table 4. The communication cost

The number of rounds	n = O(n)
The number of messages	6nm = O(nm)
The number of bits	6nmK = O(nmK)

Computational complexity evaluation. The computational cost of all parties in Phase 1 and Phase 2 is $4nm$ modular exponentiations and $4nm$ modular multiplication. The computational cost of the parties is at most of all nm modular multiplications, and nm modular exponentiations in Phase 3. The party 1 uses m modular multiplication inversions for Phase 4. We conclude that the total computational complexity is $O(nm)$ modular exponentiations, $O(nm)$ modular multiplications and $O(m)$ modular multiplication inversions. These computational costs do not include the overhead of key generation and computing the parameters y. Table 4 shows the computational complexity of the protocol.

Table 5. The complexity of the support count computation protocol

	Exp.	Mul.	Inv.
The number of operations	*5nm*	*5nm*	*m*
The number of rounds	*n*	*n*	*1*
The average number/round	*5m*	*5m*	*m*
Estimation	*O(nm)*	*O(nm)*	*O(m)*

Similarity to the first protocol, this protocol follows a communication ring with n parties, a mount of calculations of each parties is equal. Thus it is regarded as the protocol run n rounds, each round consists of computational complexities of $O(nm)$ modular multiplications and $O(nm)$ modular exponentiations.

For evaluating the efficiency of the protocol in practice, we build an experiment in the environment as the first experiment. We measure the computation cost of the protocol for the number of the difference parties from 2 to 10. Table 5 illustrates our measurements of all parties's computation time: it is in regard to m and n, for a typical scenario, where $m = 1000$, $n = 10$. The computation time of all parties is about 15.1 seconds.

Table 6. The parties's time for the support count computation protocol

	m				
n	**200**	**400**	**600**	**800**	**1000**
2	0.7	1.4	2.7	3.8	4.1
3	1.3	2.6	3.7	4.2	5.5
5	2.5	3.8	5.4	7.4	9.5
10	4.4	7.0	10.6	13.9	15.1

VI. USING BINARY TREE COMMUNICATION STRUCTURE

The homomorphic addition and multiplication operations may be expensive, because of the computational complexity of the cryptographic operations and the relatively high latency and low bandwidth of Internet connections. In order to make certain secure functions faster, we can use a communication construction presented in [21]. The authors show that any function that can be represented as $y = f(x_1,..,x_n) = x_1 \otimes x_2 \otimes ... \otimes x_n$ with being an associative operation can be securely computed using a more efficient tree structure. There many operations satisfying this description (i.e., set union, intersection, multiplication, addition).

In the protocol of Section V.B, Phase 1 uses both the homomorphic addition and multiplication operations, thus this phase can be used in the tree structure. Phase 2 is to permute and re-encrypt the elements, thus we can divide the set of elements into many partitions. Note that the permutation of the element set is similar to hierarchically merge all partitions into a random permutation of elements. Phase 3, the decryption of elements can be done parallel, thus they can be divided into many partitions to independently do. Therefore, this protocol can be improved to use in the binary tree communication structure.

VII. CONCLUSION

In this paper, we have proposed some new protocols for privacy-preserving frequent itemset mining in vertically distributed data. Basically, the proposed protocols are based on the ElGamal encryption scheme, ensure strong privacy without loss of accuracy. We showed that the proposed protocols can resist the collusion of $n-1$ corrupted parties among n parties in the protocol without using the trusted party. Moreover, they are quite efficient in comparison with some existing protocols. In addition, we gave two protocols that allow the parties to be able to select one of two privacy level corresponding to two protocols, one of them reveals only the support count, and the other reveals nothing. Note that the second protocol can apply to designing other privacy-preserving data mining algorithms, e.g, privacy preserving for learning of decision trees ID3 in arbitrary distributed data.

Currently, experiments in this paper are only simulated on PC computers without the condition of the big distributed systems. Due to these reasons we cannot run experiment on the big databases. This is one of problems that we will tackle in a future version. In addition, the current protocols only use for semi-honest model, Therefore it is hard to compare this work with the protocols proposed in malicious

model. Our intention is to extend the proposed protocols for malicious model.

REFERENCES

[1] Dakshi Agrawal and Charu C. Aggarwal, On the design and quantification of privacy preserving data mining algorithms. Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 247-255, 2001. ACM.

[2] Agrawal, Rakesh and Srikant, Ramakrishnan and Thomas, Dilys. Privacy preserving OLAP. SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 251-262, 2005. ACM.

[3] Agrawal, Shipra and Haritsa, Jayant R. A Framework for High-Accuracy Privacy-Preserving Mining. ICDE '05: Proceedings of the 21st International Conference on Data Engineering, pages 193-204, 2005. IEEE Computer Society.

[4] Cheung, David W. and Han, Jiawei and Ng, Vincent T. and Fu, Ada W. and Fu, Yongjian. A fast distributed algorithm for mining association rules. DIS '96: Proceedings of the fourth international conference on on Parallel and distributed information systems, pages 31-43, 1996. IEEE Computer Society.

[5] Dasseni, Elena and Verykios, Vassilios S. and Elmagarmid, Ahmed K. and Bertino, Elisa. Hiding Association Rules by Using Confidence and Support. IHW '01: Proceedings of the 4th International Workshop on Information Hiding, pages 369-383, London, UK, 2001. Springer-Verlag.

[6] Jim Dowd and Shouhuai Xu and Weining Zhang. Privacy-preserving decision tree mining based on random substitutions. In International Conference on Emerging Trends in Information and Communication Security, pages 145-159, 2005. Springer Berlin / Heidelberg.

[7] Du, Wenliang and Zhan, Zhijun. Using randomized response techniques for privacy-preserving data mining. KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 505--510, 2003. ACM.

[8] Evfimievski, Alexandre and Srikant, Ramakrishnan and Agrawal, Rakesh and Gehrke, Johannes. Privacy preserving mining of association rules. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 217-228, 2002. ACM.

[9] Oded Goldreich. The Foundations of Cryptography, volume 2, chapter 7: General Cryptographic Protocols. 2004.

[10] Shuguo Han and Wee Keong Ng. Multi-Party Privacy-Preserving Decision Trees for Arbitrarily Partitioned Data. International Journal of Intelligent Control and Systems, 12(4):351-358, 2007.

[11] Martin Hirt and Kazue Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. Proceedings of EuroCrypt 2000, LNCS series, pages 539-556, 2000. Springer-Verlag.

[12] Kantarcioglu, Murat and Clifton, Chris. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. IEEE Trans. on Knowl. and Data Eng., 16(9):1026-1037, 2004.

[13] Kapoor, V. and Poncelet, P. and Trousset, F. and Teisseire, M. Privacy preserving sequential pattern mining in distributed databases. CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management, pages 758-767, 2006. ACM.

[14] Kargupta, Hillol and Datta, Souptik and Wang, Qi and Sivakumar, Krishnamoorthy. On the Privacy Preserving Properties of Random Data Perturbation Techniques. ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining, pages 99, 2003. IEEE Computer Society.

[15] Li, Yu-Chiang and Yeh, Jieh-Shan and Chang, Chin-Chen. MICF: An effective sanitization algorithm for hiding sensitive patterns on data mining. Adv. Eng. Inform., 21(3):269-280, 2007.

[16] Michels Markus and Horster Patrick. Some Remarks on a Receipt-Free and Universally Verifiable Mix-Type Voting Scheme. Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology, pages 125-132, 1996. Springer-Verlag.

[17] Oliveira, Stanley R. M. and Zaane, Osmar R. Privacy preserving frequent itemset mining. CRPIT '14: Proceedings of the IEEE international conference on Privacy, security and data mining, pages 43-54, 2002. Australian Computer Society, Inc.

[18] Rizvi, Shariq J. and Haritsa, Jayant R. Maintaining data privacy in association rule mining. VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, pages 682-693, 2002. VLDB Endowment.

[19] Yoones Asgharzadeh Sekhavat and Mohamad Fathian. Mining frequent itemsets in presence of malicious participants. Journal of IET Information Security, 4(2):80-92, 2010.

[20] Shyue Liang Wang, Yu Huei Lee, Billis, S., Jafari, A. Hiding sensitive items in privacy preserving association rule mining. Systems, Man and Cybernetics, 2004 IEEE International Conference on, pages 3239 - 3244, 2005. IEEE Computer Society.

[21] Vaidya, Jaideep and Clifton, Chris. Leveraging the "Multi" in secure multi-party computation. WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society, pages 53--59, 2003. ACM.

[22] Vaidya, Jaideep and Clifton, Chris. Privacy preserving association rule mining in vertically partitioned data. KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 639-644, 2002. ACM.

[23] Vaidya, Jaideep and Clifton, Chris. Secure set intersection cardinality with application to association rule mining. J. Comput. Secur., 13(4):593-622, 2005.

[24] Vaidya, Jaideep and Clifton, Chris and Kantarcioglu, Murat and Patterson, A. Scott. Privacy-preserving decision trees over vertically partitioned data. ACM Trans. Knowl. Discov. Data, 2(3):1-27, 2008.

[25] Wu, Yi-Hung and Chiang, Chia-Ming and Chen, Arbee L. P. Hiding Sensitive Association Rules with Limited Side Effects. IEEE Trans. on Knowl. and Data Eng., 19(1):29-42, 2007.

[26] Yi, Xun and Zhang, Yanchun. Privacy-preserving distributed association rule mining via semi-trusted mixer. Data Knowl. Eng., 63(2):550--567, 2007.

[27] Sheng Zhong. Privacy-preserving algorithms for distributed mining of frequent itemsets. Information Sciences, 177(2):49-503, 2007.

[28] Zhong, Sheng and Yang, Zhiqiang and Chen, Tingting. k-Anonymous data collection. Inf. Sci., 179(17):2948-2963, 2009.

[29] Zhong, Sheng and Yang, Zhiqiang and Wright, Rebecca N. Privacy-enhancing k-anonymization of customer data. Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 139-147, 2005. ACM.

AUTHORS' BIOGRAPHIES

Luong The Dung received the B.Tech. degree in Information Technology from Le Quy Don Technical University (2001), Ph.D. degree in Computer Science from Military Institute of Science and Technology. He is currently a lecturer at Information Security Faculty of Academy of Cryptography techniques. He works in the areas of computer network; he is specialized in network security technologies as a designer. His main research interests include Privacy Preserving Data Mining and Security Issues in Data Mining.

Ho Tu Bao received a B.Tech. degree in applied mathematics from Hanoi University of Technology (1978), M.S. and Ph.D. degrees in Computer Science from Pierre and Marie Curie University, Paris (1984, 1987), and Habilitation from Paris Dauphine University (1998). He was a Ph.D. candidate (1983-1987) at INRIA (the French National Institute for Research in Computer Science and Control, France), visiting fellow (1992) at Wisconsin-Madison University (USA), associate professor (1991) at the Institute of Information Technology, Vietnamese Academy of Science and Technology (VAST), visiting associate professor (1993-1997) at the School of Information Science, and professor at the School of Knowledge Science (since April 1998) of JAIST.

Dang Hung Tran received the BS in computer science from Hanoi National University of Education in 2001, MS in computer science from Vietnam National University, Hanoi in 2006, and PhD in knowledge science from Japan Advanced Institute of Science and Technology in 2009. He is currently a lecturer of Hanoi National University of Education. His research interests include machine learning and its applications on molecular biology.