INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 12, NO. 4, DECEMBER 2007, 351-358

Multi-Party Privacy-Preserving Decision Trees for Arbitrarily Partitioned Data

Shuguo HAN, and Wee Keong NG

***Abstract*— Privacy-preserving data mining seeks to empower conventional data mining techniques with the desirable property of preserving data privacy during the mining process. Given existing approaches on privacy-preserving decision tree induction for horizontally and vertically partitioned data involving multiple parties, we extend current work to multiple parties holding arbitrarily partitioned data. Although the extension is relatively straightforward, the difficulty lies in enabling multiple parties to securely perform the scalar product operation—a core operation in decision tree induction. In this paper, we propose the concept of Pseudo Scalar Product (PSP) to perform the secure scalar product operation more efficiently than existing approaches. PSP has computational and communication complexities of $O(n)$ and $O(mn)$ respectively, compared to $O(mn)$ and $O(mn^2)$ for existing approaches. The protocol for securely performing PSP is also more secure as it is able to protect each party's privacy against up to $n-2$ corrupted parties.**

***Index Terms*— Privacy-Preserving Data Mining, Decision Tree, Arbitrarily Partitioned data.**

1. INTRODUCTION

Conventional data mining generally assumes a centralized server that collects data from multiple parties before performing data mining on that server. It generally assumes that data on the server can be shared among all parties. As privacy issues become more prevalent, this assumption may no longer hold. For instance, a law enforcement agency may wish to study the pattern difference between the financial transactions of normal individuals and known money launderers to better investigate money laundering [12]. An individual's financial transactions may come from banks, credit card companies, tax collection agencies, and so on. If individuals' private data are disclosed to law enforcement agency, their privacy are infringed.

Privacy-preserving data mining (PPDM) was introduced [1], [9] to enable conventional data mining techniques to preserve data privacy during the mining process. Much work has been done to explore privacy-preserving data mining on horizontally and/or vertically partitioned data involving multiple parties so that no single party holds the overall data [15]. Figures 1 illustrates horizontally and vertically partitioned data. For horizontally partitioned data, two parties or more hold different objects for the same set of attributes. It means each object in the virtual database is completely owned by one party. For vertically partitioned data, two parties or more hold the different set of attributes for the same set of objects. In arbitrarily partitioned data, different disjoint portions are held by different parties. This is perhaps the most general form of data partitioning, as introduced by Jagannathan and Wright [8] for two parties. As argued by the authors, although extremely "patchworked" data is unlikely in practice, it is better suited to practical settings as a more general model of horizontally and vertically partitioned data.

The secure scalar product is a core operation in decision tree induction for vertically partitioned data. Much work has been done that discussed how the secure scalar product can be computed for two parties [4], [5], [11], [18]. Vaidya and Clifton introduced the Secure Set Intersection Cardinality method to perform secure scalar product for multiple parties [14]. This method has been applied to perform decision tree induction [13] and association rule mining [14] for vertically partitioned data, and SVM model construction [17] for horizontally partitioned data. A major weakness of the Secure Set Intersection Cardinality is its computational and communication complexities, which are $O(mn)$ and $O(mn^2)$ respectively, where n is the number of parties and m is the length of private vectors.

The contribution of this paper is a more efficient method to perform the secure scalar product operation for multiple parties. In this paper, we introduce another method called the Pseudo Scalar Product (PSP) to perform secure scalar product for multiple parties. We first illustrate how decision tree induction can be performed for arbitrarily partitioned data involving two parties, which is similar to the case for vertically partitioned data. Then, we show how PSP can be extended to n parties and propose a protocol to securely compute PSP with computational and communication complexities of $O(n)$ and $O(mn)$ respectively. This complexity is lower than that of the Secure Set Intersection Cardinality.

This paper is organized as follows: We first outline related work. In Section 3, we introduce the concept of the Pseudo Scalar Product for multi-binary vectors. Section 4 introduces the proposed approach for computing secure PSP. Section 5 analyzes the complexity and security of the proposed approach. Finally, we conclude the work in the last section.

2. RELATED WORK

As we are interested in improving the efficiency of the secure scalar product operation for multiple parties, we briefly review work on privacy preserving data mining that make use of secure scalar product for two parties or more. Most of these

This work is supported in part by grant P0520095 from the Agency for Science, Technology and Research (A*STAR), Singapore.

Shuguo Han is with Center for Advanced Information System, School of Computer Engineering, Nanyang Technological University, Singapore. (email:{hans0004}@ntu.edu.sg)

Wee Keong Ng is with Center for Advanced Information System, School of Computer Engineering, Nanyang Technological University, Singapore. (email:{awkng}@ntu.edu.sg)

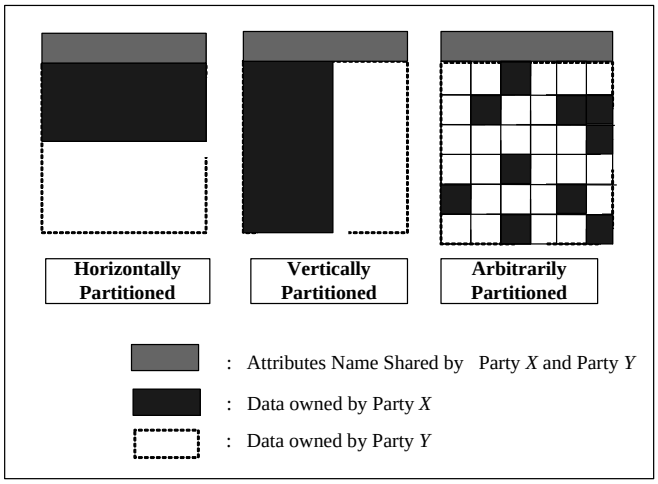


Fig. 1. Privacy-preserving data mining through the partitioning of data.

work are in decision tree induction and association rule mining where the scalar product is a basic operation.

Lindell and Pinkas [9] proposed a method to induce a decision tree using the ID3 algorithm for horizontally partitioned data based on various cryptographic techniques such as the oblivious transfer protocol, and oblivious polynomial evaluation to preserve the privacy of each party's data. Du and Zhan [4] addressed the decision tree construction problem for vertically partitioned data involving only two parties using the secure scalar product based on a semi-trusted party. The semi-trusted third party should not collude with either of the parties and should not learn anything about the private data of the two parties.

Vaidya and Clifton [11] mined association rules for vertically partitioned data having two parties involving the secure scalar product. An algebraic solution is given for secure scalar product that hides private vectors using random values. Vaidya and Clifton also presented a Naïve Bayes classifier [12] that involves the secure scalar product operation to determine the probability estimate of each class label.

Du, Han, and Chen [3] proposed secure versions of multivariate linear regression and multivariate classification for vertically partitioned data involving two parties. They developed a practical security model based on a number of building blocks, such as the secure matrix product protocol and the secure matrix inverse protocol. Barni, Orlandi and Piva [2] presented a privacy-preserving neural network involving two parities under the scenario where one party holds the private data and the other only has a private neural network that is used to process the private data. The protocol preserves the privacy of two parties by secure scalar product protocol, private polynomial evaluation [10], and so on. Wan *et al.* [16] proposed a generic formulation of gradient descent methods for secure computation by defining the target function as a composition. They demonstrated its feasibility in specific gradient descent methods, such as linear regression, and neural network. The privacy is preserved using secure scalar product protocols.

Han and Ng [6] proposed a protocol for secure genetic algorithms for the following scenario: Two parties, each holding an arbitrarily partitioned data set, seek to perform genetic algorithms to discover a better set of rules without disclosing their own private data. The challenge for privacy-preserving genetic algorithms is to allow the two parties to securely and jointly evaluate the fitness value of each chromosome using each party's private data without compromising their data privacy. They proposed a new protocol to address this challenge that is correct and secure. The protocol preserves the privacy of two parties using secure scalar product protocol.

Han and Ng [7] proposed a protocol for privacy-preserving self-organizing map for vertically partitioned data involving two parties. Self-organizing map (SOM) is a widely used algorithm for transforming data sets to a lower dimensional space to facilitate visualization. The challenges in preserving data privacy in SOM are (1) to securely discover the inner neuron from data privately held by two parties; (2) to securely update weight vectors of neurons; and (3) to securely determine the termination status of SOM. The authors proposed protocols to address the above challenges using secure scalar product protocols as well.

The secure set intersection cardinality was proposed by Vaidya and Clifton to securely compute the scalar product for association rule mining [14] and decision tree induction [13] involving multiple parties. A protocol was proposed by the authors to compute secure set intersection cardinality; the computational and communication cost are $O(mn)$ and $O(mn^2)$ respectively for n parties with vectors of length m. The protocol has strong privacy-preserving property as all vectors are encrypted by all parties based on the commutative one-way hash function.

Zhong [18] proposed an association rule mining protocol that is based on homomorphic encryption techniques for vertically partitioned data involving multiple parties. The computational and communication complexity are $O(n)$ and $O(mn)$ respectively for n parties with vectors of length m. However, the protocol has one privacy problem: The protocol

requires one party P to generate a key pair (i.e., a public key and private key) based on a public-key encryption algorithm. Private data from all other parties are encrypted based on the public key from Party P. If Party P is dishonest and colludes with other dishonest parties, the honest party's private data may be disclosed.

To the best of our knowledge, the protocol by Zhong [18] has the lowest computational complexity and communication cost for performing secure scalar product involving multiple parties. However, the protocol by Vaidya and Clifton [14] has the best privacy-preserving feature. In this paper, we propose a technique for computing secure scalar product involving multiple parties that has the same communication cost as Zhong's protocols and that has as strong privacy as Vaidya and Clifton's protocol.

3. Problem Statement

In the section we show how the secure scalar product is involved in the induction of decision tree for arbitrarily partitioned data held by two parties. In Section 3.1, we introduce a binary vector notation for a column of data set held by different parties. This notation allows us to express the scalar product of two column vectors of data sets from different parties—an operation used in evaluating attribute splits in decision tree induction, as described in Section 3.2. In Section 3.3, we show how the two-party case is extended to multiple parties using the concept of PSP.

3.1. Binary Vector Notation for Arbitrarily Partitioned Data

In two-party decision tree induction, each party holds an arbitrary partition of the original data set $\mathbf{S}$, as shown in Fig. 2. To represent a column of the data set $\mathbf{S}$ held by different parties, we introduce a binary vector notation below. Column vectors of data set are involved in the computation of scalar products, as required in computing measures such as the information gain or entropy. For the column corresponding to attribute A of the original data set, we represent the portion held by Party X (and likewise Party Y) as a set of $\|A\|$ binary column vectors. Each of these binary column vectors corresponds to a particular value of A. For instance, for $A = \alpha$, the binary column vector for data set $\mathbf{X}$ in Party X is denoted $\mathbf{X}_{[A=\alpha]} = [x_1, x_2, \ldots, x_m]^T$ (m is the number of tuples in data set $\mathbf{X}$) where

- $\mathbf{X}_{[A=\alpha]}(i) = x_i = 1$ if (i) attribute A's value in the i-th tuple of $\mathbf{S}$ is held by another party or (ii) attribute A's value is α and is held by Party X.
- $\mathbf{X}_{[A=\alpha]}(i) = x_i = 0$ if attribute A's value is held by Party X and is not α.

For example, the "Humidity" attribute has two possible values "High" and "Normal". Given that there are two parties X and Y, the "Humidity" column of the original data set is represented by four binary vectors; two in X and two in Y. Using the above notation, we represent the portion held by Party X for the humidity value "High" as $\mathbf{X}_{[\text{Humidity}=\text{"High"}]} = [1, 1, 1, 1, 0]^T$, since the values of the first, third and fourth tuples are held by Party Y but not Party X; the second tuple held by Party X has value "High"; and the last tuple has value "Normal" rather than "High". Note that the class attribute is known to all parties.

When more than one attributes are involved, we use a logical conjunction of their corresponding binary vectors to obtain a single binary vector:

$$\mathbf{X}_{[A=\alpha, B=\beta]} = \mathbf{X}_{[A=\alpha]} \wedge \mathbf{X}_{[B=\beta]}$$

where

$$\mathbf{X}_{[A=\alpha, B=\beta]}(j) = \mathbf{X}_{[A=\alpha]}(j) \wedge \mathbf{X}_{[B=\beta]}(j)$$

for all $1 \leqslant j \leqslant m$.

Using the above notation for binary column vectors, we are able to represent the arbitrary column partitions of the original data set held by each party.

3.2. Decision Tree Construction

In a two-party setting, each party holds an arbitrary partition of the original data set that is known to the other party. The objective is for each party to be able to induce a decision tree based on its own partition and the other party's partition, without compromising the privacy of the other party.

We now show how the binary column vectors are used in decision tree induction. In incremental decision tree induction, at each stage of splitting a tree node, attributes are evaluated to determine its "goodness". The concepts of *entropy index* and *information gain* are commonly used to evaluate the "goodness" of each possible splits. Given a data set $\mathbf{S}$ having L classes,

$$\text{Entropy}(\mathbf{S}) = -\sum_{i=1}^{L} \left(\frac{|\sigma_{C=i}(\mathbf{S})|}{|\mathbf{S}|} \log \frac{|\sigma_{C=i}(\mathbf{S})|}{|\mathbf{S}|} \right) \quad (1)$$

where C is the class label and σ is the selection operator (as in relational algebra). In the example in Fig. 2, C is "PlayBall". In a two-party setting, each party may compute the entropy of the original data set as the class information is known. So, Party X computes

$$\begin{aligned} \text{Entropy}(\mathbf{S}) &= -3/5 \log(3/5) - 2/5 \log(2/5) \\ &= 0.971 \end{aligned}$$

where 3 are 2 are the number of value "No" and "Yes" respectively.

The information gain for attribute A is:

$$\begin{aligned} \text{Gain}(\mathbf{S}, A) &= \text{Entropy}(\mathbf{S}) \\ &\quad - \sum_{\alpha \in A} \left(\frac{|\sigma_{A=\alpha}(\mathbf{S})|}{|\mathbf{S}|} \times \text{Entropy}(\sigma_{A=\alpha}(\mathbf{S})) \right) \end{aligned} \quad (2)$$

where α is a value of A. We illustrate how Party X securely computes the information gain $\text{Gain}(\mathbf{S}, \text{Outlook})$ of attribute "Outlook" using its own partition and Party Y's partition without compromising Y's data privacy. As "Outlook" has two possible values "Rain" and "Sunny", we need to compute the entropy of $\mathbf{S}$ with respect to these two attribute values. For "Outlook=Rain", $|\sigma_{[\text{Outlook}=\text{"Rain"}]}(\mathbf{S})|$ is the number of

Day	Outlook	Humidity	Wind	PlayBall
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Rain	High	Weak	Yes
D4	Rain	Normal	Weak	Yes
D5	Rain	Normal	Strong	No

Original data set $\mathbf{S}$

Day	Outlook	Humidity	Wind	PlayBall
D1	Sunny	-	Weak	No
D2	-	High	-	No
D3	Rain	-	-	Yes
D4	Rain	-	-	Yes
D5	-	Normal	Strong	No

Data set $\mathbf{X}$ held by Party X

Day	Outlook	Humidity	Wind	PlayBall
D1	-	High	-	No
D2	Sunny	-	Strong	No
D3	-	High	Weak	Yes
D4	-	Normal	Weak	Yes
D5	Rain	-	-	No

Data set $\mathbf{Y}$ held by Party Y

Fig. 2. The private data sets of Party X and Party Y. Note that "PlayBall" is the class attribute.

tuples in S whose Outlook attribute has value "Rain". It is computed using the secure scalar product as follows:

$$\begin{aligned} & |\sigma_{[\text{Outlook}=\text{“Rain”}]}(\mathbf{S})| \\ = \; & \mathbf{X}_{[\text{Outlook}=\text{“Rain”}]} \bullet \mathbf{Y}_{[\text{Outlook}=\text{“Rain”}]} \\ = \; & [0,1,1,1,1]^T \bullet [1,0,1,1,1]^T \\ = \; & 3. \end{aligned}$$

Similarly

$$|\sigma_{[\text{Outlook}=\text{“Rain”},\text{PlayBall}=\text{“No”}]}(\mathbf{S})|$$

and

$$|\sigma_{[\text{Outlook}=\text{“Rain”},\text{PlayBall}=\text{“Yes”}]}(\mathbf{S})|$$

can be computed using the secure scalar product, whose values are 1 and 2 respectively. According to Equation 1, we have

$$\begin{aligned} & \text{Entropy}(\sigma_{\text{Outlook}=\text{“Rain”}}(\mathbf{S})) \\ = \; & -1/3\log(1/3) - 2/3\log(2/3) \\ = \; & 0.917. \end{aligned}$$

$\text{Entropy}(\sigma_{\text{Outlook}=\text{“Sunny”}}(\mathbf{S}))$ can be computed in a similar manner, whose value is 0. According to Equation 2,

$$\begin{aligned} & \text{Gain}(\mathbf{S}, \text{Outlook}) \\ = \; & 0.971 - 2/5 \times 0 - 3/5 \times 0.917 \\ = \; & 0.42. \end{aligned}$$

$\text{Gain}(\mathbf{S}, \text{Humidity})$ and $\text{Gain}(\mathbf{S}, \text{Wind})$ are computed securely by Party X in a similar manner. We choose the attribute (i.e., *Outlook*) which has the highest information gain (value 0.42) as the best splitting attribute to split data sets $\mathbf{S}$ into partitions $\sigma_{\text{Outlook}=\text{“Rain”}}(\mathbf{S})$ and $\sigma_{\text{Outlook}=\text{“Sunny”}}(\mathbf{S})$.

If the data set $\sigma_{\text{Outlook}=\text{“Rain”}}(\mathbf{S})$ is to be partitioned further, $\text{Gain}(\sigma_{\text{Outlook}=\text{“Rain”}}(\mathbf{S}), \text{Humidity})$ and $\text{Gain}(\sigma_{\text{Outlook}=\text{“Rain”}}(\mathbf{S}), \text{Wind})$ are computed in a similar way to select the attribute that yields the highest information gain. This process can be repeated until the whole tree is built. For more details, please refer to [4], [13].

3.3. Pseudo Scalar Product

As shown above, the secure scalar product is a core operation in decision tree induction for arbitrarily partitioned data. If this core operation can be extended to multiple parties, we can construct a decision tree for arbitrarily partitioned data involving multiple parties [13].

Below, we make an observation regarding how the scalar product for multiple binary vectors is computed. We observe that the scalar product of two binary vectors V_1 and V_2 of length m is equivalent to the number of j's ($1 \leqslant j \leqslant m$) satisfying $V_1(j) + V_2(j) = 2$. For instance, in the following equation, the value of scalar product is 3 as there are three j's ($j = 3, 4, 5$) satisfying $\mathbf{X}_{[\text{Outlook}=\text{“Rain”}]}(j) + \mathbf{Y}_{[\text{Outlook}=\text{“Rain”}]}(j) = 2$:

$$\begin{aligned} & \mathbf{X}_{[\text{Outlook}=\text{“Rain”}]} \bullet \mathbf{Y}_{[\text{Outlook}=\text{“Rain”}]} \\ = \; & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \bullet \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\ = \; & 3 \end{aligned} \tag{3}$$

This observation provides us an idea of how binary scalar products for two parties can be extended to multiple binary vectors as follows:

definition 3.1: **Pseudo Scalar Product** (PSP): Given n binary vectors $V_1, V_2, \ldots, V_n$ corresponding to n parties where each vector is of size m, the number of j's, $1 \leqslant j \leqslant m$, such that $\sum_{i=1}^{n} V_i(j) = n$ (where $V_i(j)$ denotes the jth element of vector V_i) is the Pseudo Scalar Product of these n binary vectors.

We note that the value of PSP is equivalent to the cardinality of set intersection [14] of multiple binary vectors. If PSP can be computed securely and efficiently, then the scalar product for multiple parties can be determined. In the next section, we present a protocol for computing PSP securely and efficiently.

4. Protocol for Computing Secure PSP

This section introduces a protocol to compute PSP for n parties without compromising each participating party's data

vector. The protocol requires each party to encrypt its own private vector before sending to another party. Section 4.1 describes an existing encryption scheme that is used in the proposed protocol. Section 4.2 presents the secure protocol.

4.1. Homomorphic Encryption

The proposed secure protocol in Section 4.2 requires each party to send private data to each other. The private data is encrypted in order to preserve privacy. We use the homomorphic encryption scheme, which is a variant of the ElGamal encryption scheme [18], define as:

$$\digamma_y(m,k) = (g^m y^k \bmod p,\ g^k \bmod p) = (G,H) = M \quad (4)$$

where

- $y = g^x \mod p$ is a public key; x is a private key in the range [1, $p-2$] and p is a large random prime number;
- m is the message to be encrypted;
- k is chosen uniformly at random in the range [1, $p-2$];
- g is a generator for a cyclic group $\mathbb{G}$ whose discrete logarithm is hard to compute; and
- G, H are the pair of ciphertexts M (encrypted texts).

The encryption scheme is based on the computational difficulty of computing the *discrete logarithm* of group $\mathbb{G}$. Given a prime number p, a generator g and an expression $g^a \mod p$, it is computationally difficult to find the value a. The encryption scheme has the indistinguishability property [18] that any two different messages will have different ciphertexts. In addition, it has the additive homomorphic property [18] that can be used to perform computation on ciphertexts from different parties:

Property 1: $\digamma$ is *additively homomorphic* such that for any two messages m_1, m_2, two random numbers k_1, k_2 and public key y,

$$\digamma_y(m_1,k_1) \uplus \digamma_y(m_2,k_2) = \digamma_y(m_1+m_2, k_1+k_2)$$

, where $\uplus$ denote the homomorphic addition operator.

Proof:

$$\begin{aligned} & \digamma_y(m_1,k_1) \uplus \digamma_y(m_2,k_2) \\ \Leftrightarrow\ & (G_1,H_1) \uplus (G_2,H_2) \\ \Leftrightarrow\ & (G_1 \times G_2, H_1 \times H_2) \\ \Leftrightarrow\ & (g^{m_1+m_2} y^{k_1+k_2} \bmod p, g^{k_1+k_2} \bmod p) \\ \Leftrightarrow\ & \digamma_y(m_1+m_2, k_1+k_2) \end{aligned}$$

Using the above property, two parties who wish to encrypt the sum of their private data may accomplish this by encrypting their private data independently.

4.2. Secure PSP Protocol

In this section, we describe a protocol to compute PSP securely. We assume the following assumptions/tasks are performed:

1) Non-sensitive information such as the number of parties n and the length of each vector m is known to all the parties.
2) An global ordering is imposed on all parties such they are globally numbered from 1 to n.
3) All parties use the same public key y, the generator g and prime number p.

Recalling the definition of PSP (Section 3.3), to determine the scalar product of n binary vectors V_i ($1 \leqslant i \leqslant n$), we only need to determine the number of j's such that $\sum_{i=1}^{n} V_i(j) = n$. In the multi-party scenario, each binary vector V_i is held by Party P_i. In order to perform the required summation, each party P_i sends a partial sum of its binary vector element $V_i(j)$ and the summation of binary vectors of all preceding parties $\sum_{u=1}^{i-1} V_u(j)$ to the next party in the global party order, starting with P_1, for each j.

As this is to be done securely, each party P_i introduces a random number $r_i(j)$ to each binary vector element $V_i(j)$ to prevent the next party from guessing his binary vector elements. The random number is arithmetically added to the binary vector element and the sum is encrypted before sending to the next party. At the same time, party P_i also sends an encrypted partial sum of n, $r_i(j)$, and all the preceding random numbers $r_1(j), r_2(j), \ldots, r_{i-1}(j)$.

Party P_1 fires up this process and after one round of secure information passing, the two encrypted sums (binary vector elements + random numbers, and n + random numbers) are routed back to P_1, for each j. Party P_1 now compares the two encrypted sums. If they are the same, then we have just found, in a secure manner, a j value that satisfies $\sum_{i=1}^{n} V_i(j) = n$.

However, using the same set of random numbers for the binary vector elements and n may compromise P_1's data privacy, as P_2 is able to guess P_1's random number $r_1(j)$ and subsequently, $V_i(j)$. This is unfairness to the first party. Hence, no party will want to assume the role of the first party. To overcome this problem, we use a different set of random numbers $r'_1(j), r'_2(j), \ldots, r'_n(j)$ for passing the encrypted value of n. The protocol is described below:

Protocol: (Secure Pseudo Scalar Product Protocol)

Input: n parties, each having a private binary vector V_i ($1 \leqslant i \leqslant n$), public key y, generator g and prime p.

Output: Pseudo-Scalar Product, which corresponds to the scalar product of the n binary vectors.

Steps:

1) Set PSP = 0.
2) **for** $j = 1, 2, \ldots, m$
   a) Party P_1 generates two random numbers $r_1(j)$ and $k_1(j)$ which are uniformly distributed in interval [1, $p-2$]. It computes the sum of its binary vector element and the random number $s_1(j) = V_1(j) + r_1(j)$. It encrypts the sum as $\digamma_y(s_1(j), k_1(j)) = S_1(j)$ (Eq. 4). Party P_1 sends $S_1(j)$ to Party P_2. Party P_1 now generates another random number $r'_1(j)$ which is uniformly distributed in interval [1, $p-2$]. It computes the sum $t_1(j) = n + r'_1(j)$. It encrypts the sum as $\digamma_y(t_1(j), k_1(j)) = T_1(j)$. Party P_1 sends $T_1(j)$ to Party P_2.
   b) **for** i = 2, …, n (may be performed concurrently):

Party P_i generates two random numbers $r_i(j)$ and $k_i(j)$ which are uniformly distributed in interval $[1, p-2]$. It computes the sum of its binary vector element and the random number $s_i(j) = V_i(j) + r_i(j)$. It encrypts the sum as $\digamma_y(s_i(j), k_i(j)) = S_i(j)$.
Party P_i now generates another random number $r'_i(j)$ which is uniformly distributed in interval $[1, p-2]$. It encrypts it $t_i(j) = r'_1(j)$ as $\digamma_y(t_i(j), k_i(j)) = T_i(j)$.

c) **for** i = 2, ..., n: After Party P_i receives $S_{i-1}(j)$ and $T_{i-1}(j)$ from the preceding Party P_{i-1}, it computes $S_i(j) = S_i(j) \uplus S_{i-1}(j)$ and $T_i(j) = T_i(j) \uplus T_{i-1}(j)$ and then sends them to the next Party $P_{((i+1) \mod n)}$.

d) **for** i = 1, 2, ..., n (may be performed concurrently): Party P_i encrypts the previously generated random numbers $u_i(j) = r_i(j)$ (as $\digamma_y(u_i(j), k_i(j)) = U_i(j)$), and $v_i(j) = r'_i(j)$ (as $\digamma_y(v_i(j), k_i(j)) = V_i(j)$).

e) **for** i = 1, 2, ..., n: After Party P_i receives $S_{((i-1+n) \mod n)}(j)$ and $T_{((i-1+n) \mod n)}(j)$ from Party $P_{((i-1+n) \mod n)}$, it computes $S_i(j) = V_i(j) \uplus S_{((i-1+n) \mod n)}(j)$ and $T_i(j) = U_i(j) \uplus T_{((i-1+n) \mod n)}(j)$ and then sends $S_i(j)$ and $T_i(j)$ to $P_{((i+1) \mod n)}$.

f) After receiving $S_n(j)$, $T_n(j)$ from Party P_n, Party P_1 checks if $S_n(j) = T_n(j)$. If so, Party P_1 increases PSP by 1.

end for

3) return PSP

5. Protocol Analysis

In this section, we show that the protocol for computing the Pseudo Scalar Product is correct and privacy-preserving.

5.1. Correctness

We prove that the output from the protocol is the PSP that is defined in Section 4.2. For each element j of the binary vectors, Party P_1 eventually receives the two ciphertexts $S_n(j)$ and $T_n(j)$:

$$\begin{aligned} S_n(j) &= \digamma(V_1(j) + \cdots + V_n(j) + r_1(j) + \cdots + r_n(j) + \\ &\quad r'_1(j) + \cdots + r'_n(j), 2k_1(j) + \cdots + 2k_n(j)) \\ T_n(j) &= \digamma(n + r_1(j) + \cdots + r_n(j) + \\ &\quad r'_1(j) + \cdots + r'_n(j), 2k_1(j) + \cdots + 2k_n(j)) \end{aligned}$$

Based on the indistinguishability property of the encryption scheme, $S_n(j) = T_n(j)$ if and only if $V_1(j) + \cdots + V_n(j) = n$. Party P_1 will check whether the two ciphertexts are the same. If so, the value of PSP is increased by 1 according to the definition in Section 4.2. Therefore, PSP is correctly computed.

5.2. Complexity

We derive the computational complexity and communication cost of the protocol.

Computational Complexity: To compute the PSP for m elements of n binary vectors, the protocol iterates m times (Step 2). At each iteration, Step 2(a) requires two encryptions; its complexity is $O(1)$. Step 2(b) encrypts $2(n-2)$ times. However, these encryptions can be performed concurrently as described; so the complexity is $O(1)$. Step 2(c) performs the $\uplus$ operation $2(n-2)$ times; its complexity is $O(n)$. Step 2(d) performs $2n$ encryptions. Again, these operations can be done concurrently; so the complexity is $O(1)$. Step 2(e) performs the $\uplus$ operation $2n$ times; its complexity is $O(n)$. All the m iterations can be computed concurrently as well. Therefore, the overall computational complexity is $O(n)$, which is much lower than the $O(mn)$ complexity of Vaidya and Clifton [14] and is equivalent to the one by Zhong [18].

Communication Complexity: At each iteration, only Steps 2(c) and 2(e) transfer ciphertexts $4n - 2 = 2(n-1) + 2n$ times. If the number of bits to encode each ciphertext pair is c, then the overall communication cost is $4cmn$ bits, with a complexity of $O(mn)$. This is much lower than the $O(mn^2)$ complexity of Vaidya and Clifton [14] and is equivalent to the one by Zhong [18].

5.3. Privacy Preservation

We first show a privacy problem of the protocol given in [18] for vertically partitioned data involving multiple parties as it is competitive with ours in the term of complexity. While Zhong has only illustrated his idea for three parties, we generalize the key idea of the protocol for n parties as follows to show the privacy problem.

To set up the protocol, P_1 generates the public key y and private key x and sends the public key y to all other parties.

1) Party P_1 encrypts each element $V_1(j)$ individually using the public key and sends encryption pairs of all elements $V_1(j)$ $(1 \le j \le m)$ to Party P_2.
2) for $i = 2, \ldots n-1$. After receiving the encryptions from Party P_{i-1}, Party P_i computes the encryption pairs of $V_1(j)V_2(j)\ldots V_i(j)$ $(1 \le j \le m)$ for element j by proposed steps and sends the encryptions pairs to Party P_{i+1}.
3) Party P_n computes the encryption pair of the sum of all elements: $s = \sum_{j=1}^{m}\{(V_1(j)V_2(j)\ldots V_n(j)\}$.
4) Party P_n computes the encryption pairs of $r(s-t-k)$ for $1 \le k \le m$. It sends these m encryption pairs back to P_1 where r is a private constant known only by Party P_n and t is the predefined threshold. And P_1 decrypts $r(s-t-k)$ $(1 \le k \le m)$, checks whether one of them is 0. If so, the s will be larger than t.

Please refer to [18] for the details of the correctness of the protocol. Here we analyze the privacy problem. If we have a closer look, we find the role of Party P_1 who has the private key is very crucial. If the first party is dishonest, and colludes with two neighbors P_{i-1} and P_{i+1} of the Party P_i, the private vector of Party P_i will be easily disclosed.

Next, we show the privacy preservation analysis of the proposed protocol. According to the manner in which data is arbitrarily partitioned, each party holds a number of binary vectors for each column of the original data set (Section 3.1). To compute the scalar product of the set of binary vectors corresponding to a particular value α of an attribute A, the proposed protocol is executed to determine the PSP. To show that data privacy is preserved, we show that the binary vector elements (0 or 1) are not compromised.

Each time the protocol is executed for element j of a set of n binary vectors, random numbers are generated and used at each of its n iterations to hide the vector elements $V_i(j)$ $(1 \leqslant i \leqslant n)$. Furthermore, only the encrypted sum of vector elements and random numbers are sent to another party. In conjunction with the encryption scheme, it is impossible for any colluding parties to guess the value of the vector elements. Therefore, each party's data privacy is protected against up to $n-1$ colluding parties, assuming there is at least one honest party.

In the protocol, Party P_1 plays the role of checking whether $\sum_{i=1}^{n} V_i(j) = n$. We want to show further that the protocol protects data privacy against up to $n-2$ colluding parties. We consider two cases when P_1 is not a colluding party and when P_1 is a colluding party.

For the first case, since P_1 is not colluding, it will not disclose whether $\sum_{i=1}^{n} V_i(j) = n$. So, data privacy is preserved and we may conclude that each party's data privacy is protected against up to $n-1$ other colluding parties.

For the second case when P_1 is colluding, privacy is not an issue if $\sum_{i=1}^{n} V_i(j) \neq n$. If $\sum_{i=1}^{n} V_i(j) = n$, then P_1 knows that the vector element of each of the n parties is 1. For each of the other $n-1$ parties having vector element 1, its private data partition may either contain or not contain the actual attribute value (Section 3.1). In either case, the party may be honest or colluding. Hence, there are 4 sub-cases to explore. Assuming that all binary vectors in consideration pertain to attribute A and value α, we have:

1) Party Z holds α in its private data partition and it is honest. As α may only be held by one party, if Party Z is the only honest party, then P_1 will know that Z has α in its partition. Hence, Party Z's data privacy is lost. As such, the protocol requires that there is at least two honest parties (equivalently, at most $n-2$ colluding parties) so that P_1 will not know which of the honest parties holds α in its partition.
2) Party Z holds α in its private data partition and it is colluding. Since Z is colluding, we generally disregard its need for data privacy.
3) Party Z does not hold α in its private data partition and it is honest. Since Z does not hold α, there is no privacy issue.
4) Party Z does not hold α in its private data partition and it is colluding. Since Z is colluding, we generally disregard its need for data privacy.

On the whole, even when P_1 is colluding, the protocol protects data privacy against up to $n-2$ colluding parties.

5.4. Other Application: Association Rule Mining

To mine association rules, the measures of *support* and *confidence* are commonly used to determine whether an association rule $\{(A = \alpha) \Rightarrow (B = \beta)\}$ is interesting. Both measures are defined as follows:

$$\text{support} = \frac{|\sigma_{[\text{A}=\alpha,\text{B}=\beta]}(S)|}{m}$$

$$\text{confidence} = \frac{|\sigma_{[\text{A}=\alpha,\text{B}=\beta]}(S)|}{|\sigma_{[\text{A}=\alpha]}(S)|}$$

For instance, to compute the support and confidence of the rule

$$\{(\text{Outlook} = \text{“Rain”}) \Rightarrow (\text{PlayBall} = \text{“Yes”})\}$$

$$|\sigma_{[\text{Outlook}=\text{“Rain”},\text{PlayBall}=\text{“Yes”}]}(S)|$$

and

$$|\sigma_{[\text{Outlook}=\text{“Rain”}]}(S)|$$

have to be computed. Similarly with the manner in Section 3.2, they can be computed using the secure scalar product protocol whose values are 3 and 2 respectively. So,

$$\begin{aligned} \text{support} &= \mathbf{X}_{[\text{Outlook}=\text{“Rain”},\text{PlayBall}=\text{“Yes”}]} \bullet \\ &\quad \mathbf{Y}_{[\text{Outlook}=\text{“Rain”},\text{PlayBall}=\text{“Yes”}]} \\ &= \frac{2}{5} = 0.4 \end{aligned}$$

and

$$\begin{aligned} \text{confidence} &= \mathbf{X}_{[\text{Outlook}=\text{“Rain”}} \bullet \mathbf{Y}_{[\text{Outlook}=\text{“Rain”}} \\ &= \frac{2}{3} = 0.67. \end{aligned}$$

Using the secure scalar product, we are able to compute the confidence and support of rule $\{(\text{Outlook} = \text{“Rain”}) \Rightarrow (\text{PlayBall} = \text{“Yes”})\}$ without disclosing any private data set to the other. To extend privacy-preserving association rule mining, protocol to compute PSP can be applied.

6. Conclusions

Secure scalar product is a known core operation in decision tree induction and other data mining techniques for vertically partitioned data where each data partition is held privately by one party. Although extensions have been made to multiple parties and arbitrarily partitioned data, we discussed limitations of existing work in terms of their computational complexity and privacy-preserving quality.

In this paper, we proposed a secure protocol to compute the Pseudo-Scalar Product for multiple parties holding arbitrarily partitions of data. We showed that determining the PSP yields the scalar product of binary vectors. The proposed protocol is as efficient as the best existing protocols for performing the same task, and as secure against colluding parties as the most secured protocols for performing the same task so far.

REFERENCES

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM international Conference on Management of Data*, pages 439–450, Dallas, Texas, United States, 2000.

[2] M. Barni, C. Orlandi, and A. Piva. A privacy-preserving protocol for neural-network-based computation. In *Proceedings of the 8th ACM Workshop on Multimedia and Security*, pages 146–151, Geneva, Switzerland, 2006.

[3] W. Du, Y. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 222–233, Lake Buena Vista, Florida, April 22–24 2004.

[4] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, pages 1–8, Maebashi City, Japan, 2002.

[5] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the 7th Annual International Conference in Information Security and Cryptology*, pages 104–120, Seoul, Korea, December 2–3 2004.

[6] S. Han and W. K. Ng. Privacy-preserving genetic algorithms for rule discovery. In *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, pages 407–417, Regensburg, Germany, September 2007.

[7] S. Han and W. K. Ng. Privacy-preserving self-organizing map. In *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, pages 428–437, Regensburg, Germany, September 2007.

[8] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery in Data Mining*, pages 593–599, Chicago, Illinois, USA, 2005.

[9] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–53. Springer-Verlag, 2000.

[10] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 245–254, Atlanta, Georgia, United States, 1999.

[11] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 23-26 2002.

[12] J. Vaidya and C. Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *Proceedings of the SIAM International Conference on Data Mining*, pages 522–526, Lake Buena Vista, Florida, 2004.

[13] J. Vaidya and C. Clifton. Privacy-preserving decision trees over vertically partitioned data. In *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Storrs, Connecticut, 2005. Springer.

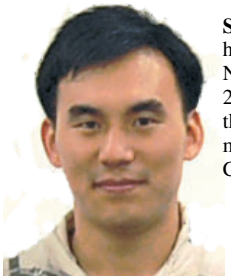
[14] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), 2005.

[15] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 3(1):50–57, March 2004.

[16] L. Wan, W. K. Ng, S. Han, and V. C. S. Lee. Privacy-preservation for gradient descent methods. In *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 775–783, San Jose, California, USA, August 2007.

[17] H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the ACM Symposium on Applied Computing*, pages 603–610, Dijon, France, 2006.

[18] S. Zhong. Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sciences*, 177(2):490–503, 2007.



Shuguo Han received the Bachelor's degree (with honors) from School of Computer Engineering, Nanyang Technological University, Singapore, in 2005. Currently, he is pursuing a Ph.D. degree from the same university. His research interests are data mining with privacy issues, machine learning, and Cryptography technologies.

Wee Keong Ng is Associate Professor at the School of Computer Engineering, Nanyang Technological University, Singapore. He received his PhD from the University of Michigan at Ann Arbor, USA. He has published more than 170 refereed journal and conference articles in the area of data mining, web information systems, database and data warehousing, and software agents. He is currently a member of the Editorial Review Board of the Journal of Database Management (JDM), the International Journal of Applied Decision Sciences (IJADS), the International Journal of Intelligent Data Analysis (IJIDA), the International Journal of Intelligent Information and Database Systems (IJIIDS), and the International Journal of Intelligent Information Technologies (IJIIT). Dr. Ng actively participates in international conference activities. In recent years, he is Asia Pacific Liaison Chair of the 7th IFIP WG6.11 Conference on e-Commerce, e-Business, and e-Government (I3E 2008), Area Chair of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2008), Program Co-Chair of the 5th International Conference on Service Science and Service Management (ICSSSM'08), Area Chair of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2007), Program Chair of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2006), Program Vice Chair of the 9th Pacific Rim International Conference on Artificial Intelligence (PRICAI2006), and Industrial Chair of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2005). Dr. Ng is a Member of IEEE and ACM.