

Preemptive Measures against Malicious Party in Privacy-Preserving Data Mining

Shuguo Han*

Wee Keong Ng†

Abstract

Currently, many privacy-preserving data mining (PPDM) algorithms assume the semi-honest model and/or malicious model of multi-party interaction. However, both models are far from able to fully capture the complexity of events and data interactions among parties in the process of secure data mining. In the paper, we study the problem of security violations when a malicious party provides false data. We identify four privacy vulnerabilities of secure scalar product protocols that underlie many current PPDM algorithms. We propose a general more model of two-party interaction and demonstrate its applicability to securely compute $(x_1 + y_1)(x_2 + y_2)$ and $(x + y)\log_2(x + y)$ where x_i and y_i are private values held by each party respectively. We show how the proposed model can be used to securely compute four commonly used kernel functions and other common functions. We also propose two necessary conditions and two basic measures that should be adopted in the current malicious model.

1 Introduction

Data mining is widely used to extract knowledge from large amounts of data in various domains. As data mining becomes more and more pervasive, privacy concerns have drawn much attention from research community. To prevent the misuse of data, some laws and regulations have been proposed to prohibit companies or groups from sharing their data, such as the U.S. healthcare laws [20] and the 1996 administrative simplification provisions in HIPAA [9].

In the data mining community, much work has been done to perform data mining while preserving the data privacy of the participating parties. There are two general approaches in privacy-preserving data mining (PPDM): Randomization [1] and Secure Multi-party Computation [14]. The former approach alters the data before releasing the data for mining; however, the accuracy of results is affected. The latter approach incorporates Secure Multi-party Computations (SMC) [26] in

data mining [14]; this approach does not compromise the accuracy of results.

In the SMC approach, there are two broad models for categorizing the adversarial behavior of participants in a multi-party secure computation process [5]: The semi-honest model and malicious model. Loosely speaking, a party in the semi-honest model follows the protocol properly but it keeps a record of all the intermediate computations during the execution. After the protocol completes, the semi-honest party attempts to derive additional information from the intermediate computational results. A party in the malicious model is allowed to behave (and deviate) arbitrarily from the protocol. To force a malicious party to follow the protocol, zero-knowledge proofs [6] must be applied. Zero-knowledge proofs are proofs of the validity of an assertion made by a party without disclosing additional information.

Clearly, it is easier to adopt the semi-honest model than the malicious model in PPDM algorithms, as less effort is required to ensure the overall privacy-preserving property of the algorithm. To date, most PPDM algorithms use the semi-honest model. For the malicious model, zero-knowledge proofs are used to force a malicious party to abide by the protocols, such as secure equality protocol, secure scalar product protocol, secure set operations, and secure comparisons [12, 13]. Jiang and Clifton [11] proposed an Accountable Computing (AC) framework for the malicious model that is more efficient as it only initiates the identification and exposure of a malicious party when there is a security breach of an honest party.

To the best of our knowledge, almost all work in PPDM so far uses the semi-honest model and/or malicious model. As shown by Goldreich [5], there are inherent security problems that cannot be avoided in the malicious model for simple computations of SMC:

- *A malicious party substituting their local input and executing the protocol with a false input:* Any malicious party may use false values to probe the private data of honest parties. Any protocol for simple computations of SMC in the malicious model is not able to prevent a malicious party from

*School of Computer Engineering, Nanyang Technological University, Singapore, hans0004@ntu.edu.sg

†School of Computer Engineering, Nanyang Technological University, Singapore, awkng@ntu.edu.sg

providing false data.

- *A malicious party aborting the protocol prematurely (e.g., before sending their last message):* It is impossible to prevent any malicious party from suspending or aborting the execution of the protocol. In particular, the malicious party may abort at the first moment when it obtains the desired result, which causes unfairness for the honest parties.

The first problem—passing false values as true data with malicious intent—is a major source of vulnerability in a secure multi-party setting. Although the first problem cannot be avoided in simple computations of SMC, it is possible to address it in complex data mining algorithms. This paper discusses the first problem by making use of peculiarities and properties inherent in PPDM algorithms. We will pursue the second problem as part of our future work.

To prevent a malicious party from probing the inputs based on the outputs of secure building blocks in PPDM, the random shares technique [2, 5, 10, 14] has been used to split **all** the intermediate results into multiple random portions where each party holds one portion so that none of the parties is able to speculate anything about the intermediate results without all the portions. Originally, it was proposed for the semi-trusted model [2, 5, 10, 14]. Here we use the random shares as the main measure to fix the vulnerabilities when a malicious party is present.

As part of our contribution in this paper, we propose a general model for two participating parties together with a protocol to securely split all *numerical* intermediate results. To illustrate the applicability of the proposed model, we show two secure approaches to split $(x_1+y_1)(x_2+y_2)$ and $(x+y)\log_2(x+y)$ respectively into two random portions where x_i and y_i are private values held by each party without disclosing x_i or y_i to the other party. The proposed approaches are more efficient than the existing approaches respectively. We also show that four commonly used kernel functions (i.e., linear kernel, Gaussian kernel, Polynomial kernel, and Sigmoidal kernel) and other common functions can be securely splitted into two portions based on the proposed model.

Although the random shares technique is able to fix the vulnerabilities, the measure cannot restrict and restraint the scope of activity of the malicious party. To do that, we identify two necessary conditions that an honest parties can use to check against the vulnerabilities of the malicious model in PPDM: (1) all parties must maintain consistent data values for the same attributes across different iterations of a data mining algorithm; and (2) inherent properties of the PPDM algorithm must be preserved in the same or different

iterations.

In order to fulfill these two conditions, we propose the other two basic measures for enforcing security. The first measure is *input consistency check*—it ensures that each party is forbidden from providing different values for the same attributes. The second measure—the *property-preservation check*—ensures that certain inherent properties of the algorithm are preserved across the same or different iterations.

The objective of this paper is to identify the severity of the problem when the malicious party is present in PPDM and to propose the random shares technique as a main measure and other two proposed novel measures to fix the vulnerabilities. The organization of the paper is as follows: In Section 2, we present an overview of related work on secure scalar product protocols and a specific secure scalar product protocol that is used in this paper. To illustrate the severity of the problem, Section 3 identifies four possible privacy breaches of secure scalar product protocols if a malicious party provides false data. In Section 4, we generalize a model together with a protocol based on the random shares technique as a main measure to fix the privacy breaches. To restrict and restraint the scope of activity of the malicious party, Section 5 defines two necessary conditions to check against the vulnerabilities together with other two measures to ensure these conditions are fulfilled. The final section concludes the paper.

2 Background

Section 2.1 reviews related work on various PPDM algorithms that preserve data privacy using secure scalar product protocols to perform basic secure computations. Section 2.2 presents a specific secure scalar product protocol that is used in the paper.

2.1 Related Work The work by Lindell and Pinkas [14] is among the pioneer works in PPDM; they proposed a protocol to construct decision trees for horizontally partitioned data. The protocol is based on a sub-protocol for the secure computation of $(x+y)\ln(x+y)$, where x and y are private values held respectively by two parties. Since the publication of this work, various data mining algorithms using secure multi-party computations [26] have been proposed.

To construct decision trees for vertically partitioned data, a secure scalar product protocol was proposed to discover the number of records that contains a set of attribute values [3]. The criteria to evaluate the “goodness” of each possible splits, such as information gain and entropy or gini, are securely computed. To mine association rules, various secure scalar product protocols with different complexity and levels of secu-

ity [21, 28, 29] have been proposed to securely compute the support and confidence to determine if an association rule is frequent.

Jagannathan and Wright performed privacy-preserving k -means clustering [10] to securely compute the closest cluster for a given arbitrarily partitioned data input based on secure scalar product protocols. A Naïve Bayes classifier for vertically partitioned data was proposed by Vaidya and Clifton [22] where secure scalar product protocols are used to determine the probability estimate of each class label. Wright and Yang [25] proposed a secure protocol to learn the Bayesian network structure for vertically partitioned data. An efficient and privacy-preserving version of the $K2$ algorithm was given based on secure scalar product protocols. As shown by Yu *et al.* [27], secure scalar product can be used to compute the global Gram matrix between two parties for horizontally partitioned data held by two or more parties.

Recently, secure scalar product protocols have been applied in gradient descent proposed by Wan *et al.* [24] to update the weight vector attached the input vector, essentially to minimize the prediction error. Genetic algorithm proposed by Han and Ng [7] has used secure scalar product protocols to securely compute the fitness during evaluation for rule discovery. Secure scalar product protocols have also been used in self-organizing map proposed by Han and Ng [8] to securely compute the closest cluster and securely detect the termination status of the protocol.

In summary, secure scalar product protocols are a fundamental secure building block for many data mining algorithms in conjunction with the semi-honest model. Recently, a secure scalar product protocol in conjunction with the malicious model was proposed by Kantarcioglu and Kardes [12] using zero-knowledge proofs. Jiang and Clifton [11] presented the Accountable Computing (AC) framework to detect malicious behaviors with the help of a third independent entity. The approach is more efficient as it only initiates the identification and exposure of a malicious party when there is security breach of an honest party. The authors extended a secure scalar product protocol to the malicious model using the AC framework.

A major limitation of deploying protocols such as secure scalar product, secure sum, secure set intersection, and so on, in PPDM algorithms is that although each protocol is secure as it is in the context of the malicious model, the overall security and privacy cannot be guaranteed when considering the overall data mining process. In Section 3, we show four security vulnerabilities of secure scalar product protocols in the malicious model.

Protocol 1 Secure Scalar Product Protocol

Input: Party A has a private vector \mathbf{a} and Party B has a private vector \mathbf{b} .

Output: Party A and B hold private portions r_a and r_b respectively where $r_a + r_b = \mathbf{a} \cdot \mathbf{b}$

- 1: Party A generates a private and public key pair (sk, pk) , and sends pk to Party B.
 - 2: For every element i , Party A encrypts a_i and sends the encrypted text $c_i = E_{pk}(a_i)$ to Party B.
 - 3: Party B computes $\omega = \prod_{i=1}^n (c_i^{b_i}) \cdot E_{pk}(-r_b)$ where r_b is generated randomly and sends ω to Party A.
 - 4: Party A decrypts ω and obtains $r_a = D_{sk}(\omega) = \mathbf{a} \cdot \mathbf{b} - r_b$.
-

2.2 Secure Scalar Product Protocol This section introduces a specific secure scalar product protocol that is proposed based on two properties of some public key encryption schemes.

Two Properties: The Secure Scalar Product protocol introduced here uses the properties of a probabilistic public key encryption scheme E that are summarized as follows: (1) The encryption function E is *additive homomorphic*; i.e., $E(a) * E(b) = E(a+b)$ where a and b are the messages to be encrypted. This property means we are able to compute the encryptions of the sum of two messages without decrypting the individual messages. (2) E has the *semantic security* property. Informally, it means whatever the adversary could compute from the ciphertexts could be computed without ciphertexts.

Secure Scalar Product Protocol: The Secure Scalar Product protocol was proposed by Geothals *et al.* [4]. Party A and Party B have private vectors \mathbf{a} of length n and \mathbf{b} of length n respectively. They wish to compute the scalar product of \mathbf{a} and \mathbf{b} without disclosing private vectors. At the end of the protocol, each party holds private random portions r_a and r_b of the scalar product respectively. (This is an example where the random shares technique is applied to enforce security. We will show security vulnerabilities in Section 3 if \mathbf{a} and \mathbf{b} are both known by one party.) The protocol is based on the public key encryption system mentioned above, such as the Naccache-Stern cryptosystem [15], Paillier Cryptosystem [18], Pohlig-Hellman [19] and so on. Based on the additive homomorphic property, we have

$$E_{pk}(\mathbf{a} \cdot \mathbf{b} - r_b) = E_{pk}\left(\sum_{i=1}^n (a_i^{b_i}) - r_b\right) = \prod_{i=1}^n (c_i^{b_i}) \cdot E_{pk}(-r_b)$$

where pk is the public key shared by two parties, c_i is

the encrypted text (ciphertext) of message a_i , and r_b is the random share known only by Party B.

Party B securely computes $E_{pk}(\mathbf{a} \cdot \mathbf{b} - r_b)$ based on its private vectors and the encrypted text c_i from Party A and sends $E_{pk}(\mathbf{a} \cdot \mathbf{b} - r_b)$ to Party A. Party A uses its private key sk to decrypt $E_{pk}(\mathbf{a} \cdot \mathbf{b} - r_b)$ and obtains its private portions r_a . The details of the Secure Scalar Product protocol are shown in Protocol 1. The computational complexity and communication cost are both $O(n)$ [4] as Party A performs n encryptions and sends n ciphertexts to Party B in Protocol 1 in Step 2 of the protocol.

3 Security Breaches of Secure Scalar Product Protocols

Probing attack is a common form of security breaches in a multi-party setting. By substituting true data by false data (repeatedly), a malicious party is able to probe an honest party's private data.

Probing attack is particularly attractive to a malicious party when the number of possible values to be probed is bounded. This is the case for binary vectors when computing scalar products that involve only two parties. For simplicity and without loss of generality, we assume Party A with private vector \mathbf{a} is the malicious party and Party B with private vector \mathbf{b} is the honest party in this section.

If the same vector of the honest party has been used several times over, the malicious party is able to eliminate the number of possibilities by *intersecting sets of possibilities* based on its true vectors, as illustrated below. Suppose the scalar product between vector \mathbf{b} and one true vector $\mathbf{a} = [1, 1, 0]^T$ of the malicious party is 1, then there are two possible values for the first two elements of \mathbf{b} :

$$\{ [1, 0, *]^T, [0, 1, *]^T \}$$

where $*$ can be 1 or 0. If the scalar product between the same vector \mathbf{b} and another true vector $\mathbf{a}' = [0, 1, 1]^T$ is 1, then there are two other possible values for the last two elements of \mathbf{b} :

$$\{ [* , 0, 1]^T, [* , 1, 0]^T \}$$

By intersecting (logical AND) the sets of possible vectors, Party A is able to limit the possibilities of \mathbf{b} to

$$\{ [1, 0, 1]^T, [0, 1, 0]^T \}$$

Hence, we show that the privacy of binary vectors in a scalar product operation may be violated if the same vector is used more than once even just with the true vectors of the malicious party.

If Party A probes the private vector of Party B by providing any false data, the problem is more serious. Vaidya and Clifton [23] has identified simple probing as a vulnerability attack. We identify three more possible probings: number system probing, dual probing and shrink probings. Other than shrink probing, which is specific to binary vector, the other three probing attacks apply to general vectors.

3.1 Number System Probing Assuming all elements of private vector \mathbf{b} of Party B are binary values. Here we want to show that if the scalar product is fully known, all elements of vector \mathbf{b} may be discovered by the malicious party (Party A).

In number system probing, Party A sets its vector $\mathbf{a} = [2^0, 2^1, \dots, 2^{n-1}]^T$ and computes the desired scalar product of vectors \mathbf{a} and \mathbf{b} with Party B as follows:

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= \begin{bmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{n-2} \\ 2^{n-1} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} \\ &= 2^{n-1}b_n + 2^{n-2}b_{n-1} + \dots + 2^1b_2 + 2^0b_1 \end{aligned}$$

Finally, the scalar product is known by Party A. We note that any base-10 value can be uniquely converted its binary numeral based on the binary number system. Bits of the binary numerical can be easily mapped to elements of the vector \mathbf{b} . Therefore Party A is able to find all elements of private vector \mathbf{b} of Party B. For instance, if $\mathbf{a} \cdot \mathbf{b} = 29$ and $n = 5$, the binary numerical of 29 is 11101_2 . Hence, it can be clearly discovered that vector \mathbf{b} is $[1, 0, 1, 1, 1]^T$ which is obtained by reading the sequence of the binary numerical from the right.

The case of probing a binary vector can be generalized to cases for probing a non-binary vector. As long as the value of the base used in the elements of vector \mathbf{a} (the base is 2 in the above example) is larger than the maximum value of elements of vector \mathbf{b} , the scalar product can be easily transformed into another new base format. Hence, the malicious party is able to detect the private vector of the honest party.

This probing is very hard to detect in the current malicious model as Party A is able to perform the data mining algorithms normally and produce the real data mining output without being detected as he/she knows Party B's private vector. However, when the length of vector \mathbf{b} is large enough, the bit length required by false elements of \mathbf{a} becomes impractical. For instance when $n = 10,000$, the last element a_n of false vector \mathbf{a} becomes as large as $a_n = 2^{10,000} - 2$ just to probe a binary vector.

3.2 Simple Probing In simple probing, a malicious party generates a false vector to probe one element of the other party's private vector [23]. Only one element value of the false vector is set to be 1. After executing secure scalar product protocols, the malicious party is able to determine the corresponding element value of the honest party's vector; the value is in fact the scalar product. Party A could likewise generate all such probing vectors to probe the entire Party B's vector.

Although simple probing is theoretically feasible, its practicality depends on the length of vectors n . In vertically partitioned data, n is the number of records in the data set. Hence, this is generally a very large number. So, it may not be possible for a malicious party to successfully probe all elements of the honest party's vector during one data mining task. However, it is still possible for the malicious party to successfully probe selected elements of the honest party's vector; these elements may correspond to sensitive information of a person, such as his/her medical conditions, salary, and so on.

Vaidya and Clifton have proposed a simple method to counter simple probing [23]: If the scalar product is 0 or 1 and is less than some threshold r , abort the protocol. In this way, the malicious party cannot successfully probe the honest's private vector. This technique has been proposed for privacy-preserving association rule mining from vertically partitioned data and is useful when the threshold r is less than the minimum support and minimum confidence. If the frequency of an itemset is less than the threshold, then it is less than the minimum support and minimum confidence as well; i.e., it is not interesting or frequent.

3.3 Dual Probing In dual probing, a malicious party uses two vectors to probe one element of the honest party's private vector. One vector is a real vector from the malicious private data; the other is the false vector with only one element having value reduced by 1 from the corresponding element of the real vector. After repeating secure scalar product protocols twice with the two vectors, the difference of the two scalar products is the corresponding element of the honest party's private vector.

Dual probing is not limited to binary vectors. To probe the honest party's private vector of length n , a total of $2n$ probes and executions of secure scalar product protocols are required. Clearly, this is even more impractical than simple probing. Compared to simple probing, dual probing yields correct scalar products. More importantly, the threshold-based method by Vaidya and Clifton [23] is not able to detect and halt dual probing.

3.4 Shrink Probing In shrink probing, the malicious party (Party A) begins by setting portions of its private vector \mathbf{a} to 1. Let $\mathbf{a}^{\{p\sim q\}}$ denote the vector where by elements from index p to q are 1 and all others elements are 0. Party A uses $\mathbf{a}^{\{p\sim q\}}$ to determine how many elements indexed from p to q of the honest party's vector \mathbf{b} are 1's or 0's. In the next probing, it shrinks the number of 1 elements in its probe vector by half, by changing the value of p or q appropriately.

To illustrate, let the honest party vector $\mathbf{b} = [1, 1, 0, 0, 1, 1, 1, 1]^T$. Using shrink probing, Party A sets its initial private vector \mathbf{a} to $\mathbf{a}^{\{1\sim 8\}} = [1, 1, 1, 1, 1, 1, 1, 1]^T$. After computing the scalar product, Party A knows that the honest party has six 1's in its vector, but it does not know the position of these 1's elements. In the next probing, Party A uses $\mathbf{a}^{\{1\sim 4\}} = [1, 1, 1, 1, 0, 0, 0, 0]^T$ in its scalar product computation with Party B to determine that Party B has two 1's within the first four positions. In the third probing, Party A uses $\mathbf{a}^{\{1\sim 2\}} = [1, 1, 0, 0, 0, 0, 0, 0]^T$ and determines that Party B has two 1's in the first two positions in the vector. Gradually, Party A is able to form the distribution of the 1 values in vector \mathbf{b} . In fact, using only three probes, Party A is able to guess all the elements of the honest party's vector.

Shrink probing focuses on determining the distribution of the 1 values in the honest party's vector. Although a malicious party may not discover all elements of the vector, it may discover additional information about the distribution of the element values.

3.5 Summary This section discussed four probings on the honest party's private values: Number system probing, simple probing, dual probing, and shrink probing. Given the weaknesses and vulnerabilities of conventional malicious model, we discuss a measure—random shares—to address them in the next section.

4 Random Shares

Many PPDM algorithms make use of secure building blocks such as secure scalar product, secure sum, secure permutation, and secure set intersection in every iteration to execute the overall data mining tasks. Various protocols have been proposed to execute these operations securely in the semi-honest and/or malicious model.

As shown in Fig. 1, any computation on the private input data sets are performed using secure building blocks f . The outputs from the secure building blocks f are used by a data mining algorithm g to make decision in the current or next iteration. As an example, in decision tree induction or association rule mining for vertically partitioned data, the secure building block f

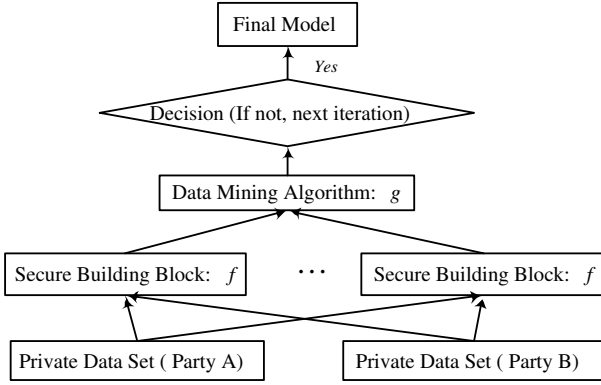


Figure 1: General model of privacy-preserving data mining algorithms for two parties.

is secure scalar product protocols; g is the computation of the information gain or Gini index for attribute selection criteria in decision tree induction or the computation of the support and confidence in association rule mining respectively that form part of the overall PPDM algorithm.

Fig. 1 shows that the outputs of the secure building blocks f are sent to the data mining algorithm g for further computation. This output is a form of intermediate results that are vulnerable to privacy violation as discussed in Section 3. To further prevent such security breaches by a malicious party, the *random shares* technique [2, 5, 10, 14] is used as a measure to split **all numerical** intermediate results of g into two random values held separately by each party; the sum of these two random values is the value of the intermediate result. Neither party is able to speculate anything about the intermediate results using only its private portion. Therefore the security is preserved for the data mining algorithm g .

Below, we propose a general model to securely compute g without disclosing random shares (outputs) from secure building blocks f in Section 4.1. To show the applicability of the general model, Section 4.2 illustrates a case where g is directly computed by the proposed model while Section 4.3 illustrates a case where g cannot be securely computed by directly applying the proposed model. In Section 4.4, we show other applications of the proposed model, including secure computations of four kernel functions and other common functions. The final section summarizes Section 4.

4.1 General Model Based on Random Shares

Now that the input to algorithm g is in the form of two values privately held by two parties whose sum is the actual input value, we propose a general model of secure

computation for g based on the private portions held by each party without disclosing any private portions to the other party.

Formally, Party A and Party B each hold private data a (a value, vector or matrix) and b (a value, vector or matrix) respectively to be sent as inputs to a secure building block f yielding random shares x and y as outputs. Only Party A knows x and only Party B knows y such that $(x + y) = f(a, b)$. There are one or more secure building blocks supplying input data to algorithm g :

$$g(f(a_1, b_1), f(a_2, b_2), \dots, f(a_n, b_n))$$

where n is the total number of the secure building blocks and f 's are the same or different secure building blocks.

It is very common that g cannot be computed directly without knowing the private portions of the other party, such as $g(x, y) = (x + y) \ln(x + y)$, $(x + y)^n$ and $\sqrt{x + y}$. To address the problem, the general Yao's circuit evaluation protocol [26] can be applied. However, this is not sufficiently efficient for large quantities of data. Here we propose a general model together with a protocol to efficiently compute g , rather than to apply the circuit evaluation protocol.

First, we make the following transformation for g :

$$\begin{aligned} & g(f(a_1, b_1), f(a_2, b_2), \dots, f(a_n, b_n)) \\ &= g(x_1 + y_1, x_2 + y_2, \dots, x_n + y_n) \\ &\Leftrightarrow g' \\ &= \sum_{p=1}^m (c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n)) \end{aligned}$$

where c_p is the constant factor of the p_{th} term of g' (m terms) and $J_{i,p}$ is a specific function. It is clear that the new g' is a linear sum expression where every p_{th} ($1 \leq p \leq m$) term can be expressed as follows:

$$c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n)$$

where $J_{1,p}(x_1, \dots, x_n)$ is obtained by Party A only and $J_{2,p}(y_1, \dots, y_n)$ is obtained by Party B only.

The method to securely compute g' is shown in Protocol 2. If the terms are only held by one party, the party can securely compute the sum of the terms individually (Steps 2 and 3 of Protocol 2). If the p_{th} term is held by two parties, Party A and Party B construct vectors \mathbf{a} and \mathbf{b} separately by appending $J_{1,p}(x_1, \dots, x_n)$ to vector \mathbf{a} and $J_{2,p}(y_1, \dots, y_n)$ to vector \mathbf{b} respectively (Steps 5 and 6 of Protocol 2).

By applying the Secure Scalar Product protocol in Section 2.2 (Protocol 1) to \mathbf{a} and \mathbf{b} , Party A and Party B each obtain two private portions r_c and r_d respectively (Step 8 of Protocol 2). At the end of the protocol, Party

Protocol 2 Random Shares Protocol

Input: $g' = \sum_{p=1}^m (c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n))$.

Output: Party A and Party B obtain private portions R_a and R_b respectively where $R_a + R_b = g'$.

- 1: The terms $c_p \times J_{1,p}(x_1, \dots, x_n) \times J_{2,p}(y_1, \dots, y_n)$ for $1 \leq p \leq m$ are classified into four types:
 - (a) Terms of the first type only contain the random shares from Party A where $J_{2,p}(y_1, \dots, y_n) = 1$.
 - (b) Terms of the second type only contain the random shares from Party B where $J_{1,p}(x_1, \dots, x_n) = 1$.
 - (c) Terms of the third type are the constants. We denote the total sum of terms in this type by c .
 - (d) Terms of the fourth type are multiplications of random shares from two parties.
 - 2: Party A securely computes the sum r_a of all the terms of the first type.
 - 3: Party B securely computes the sum r_b of all the terms of the second type.
 - 4: **for** every p_{th} term of the fourth type **do**
 - 5: The element $c_p \times J_{1,p}(x_1, \dots, x_n)$ is appended to Party A's private vector **a**
 - 6: The element $J_{2,p}(y_1, \dots, y_n)$ is appended to Party B's private vector **b**.
 - 7: **end for**
 - 8: Two parties jointly and securely perform the Secure Scalar Product protocol in Section 2.2 to vectors **a** and **b**. After the protocol, Party A and Party B each obtain random share r_c and r_d respectively where $r_c + r_d$ is the sum of all the terms in the fourth types.
 - 9: Party A obtains $R_a = r_a + r_c + c$ and Party B obtains $R_b = r_b + r_d$.
-

A and Party B easily obtain private portions R_a and R_b respectively where $R_a + R_b = g'$ (Step 9 of Protocol 2).

In the following, the general model of party interaction using Protocol 2 is applied to securely compute several specific computations g based on random shares from f without disclosing any private portion.

4.2 Secure Computation of $g = (x_1 + y_1) \times (x_2 + y_2)$

Consider the following specific computation

$$(4.1) \quad g = (x_1 + y_1) \times (x_2 + y_2)$$

where x_i and y_i are the random private portions held by each party respectively. The values x_i and y_i may be outputs of some secure building blocks f , such as secure scalar product protocols used in privacy-preserving genetic algorithms for classification [7] to securely compute the fitness value of each chromosome.

After expanding g in Eq. 4.1, we have

$$g = (x_1 + y_1) \times (x_2 + y_2)$$

$$= x_1 \times x_2 + x_1 \times y_2 + x_2 \times y_1 + y_1 \times y_2$$

$$= g'$$

It is clear that g is equivalent to g' . Then we compute g by applying Protocol 2 as follows:

$$g = x_1 \times x_2 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_2 \\ y_1 \end{bmatrix} + y_1 \times y_2$$

$$= x_1 \times x_2 + r_a + r_b + y_1 \times y_2$$

$$= (x_1 \times x_2 + r_a) + (r_b + y_1 \times y_2)$$

$$= R_a + R_b$$

where R_a and R_b are two private portions held by Party A and Party B respectively after Protocol 2 where $R_a + R_b = (x_1 + y_1) \times (x_2 + y_2)$.

Analysis: The computational and communication overheads mainly come from the execution of Protocol 1 in Step 8 of Protocol 2. The computational overheads to securely compute $R_a + R_b = (x_1 + y_1) \times (x_2 + y_2)$ are 3 encryptions, 2 exponentiations, and one decryption: (1) Two encryptions are used to encrypt x_1 and x_2 by Party A in Step 2 of Protocol 1; (2) One encryption is used to encrypt $-r_b$ to get $E(-r_b)$ by Party B in Step 3 of Protocol 1; (3) Two exponentiations are used to compute $c_i^{b_i}$ by Party B in Step 3 of Protocol 1; and (4) One decryption is used to decrypt $E(\mathbf{a} \cdot \mathbf{b} - r_b)$ by Party A in Step 4 of Protocol 1.

If the Paillier Cryptosystem [18] is applied, two exponentiations are required for each encryption, where one of them can be pre-computed and one exponentiation is required for each decryption. Therefore, nine exponentiations ($3 \times 2 + 2 + 1$) are required where three of them can be pre-computed.

Communication overheads are three times the bit length used (typically 512 or 1,024 bits long) where two of them are used to send ciphertexts c_1 and c_2 from Party A to Party B and one is used to send ciphertexts $E(r_a)$ from Party B to Party A.

Our approach is more efficient and general than the existing approach proposed by Goldreich [5] that reduces the same problem to 1-out-of-4 Oblivious Transfer (OT) protocol. 1-out-of-4 OT (OT_1^4) protocol is a special case of 1-out-of- N OT (OT_1^N) protocol. In OT_1^N , one party (the sender) has N inputs a_1, a_2, \dots, a_N and the other party (the receiver) can choose to get a_i for some $1 \leq i \leq N$ of its choice, without learning anything about the other inputs and without the sender learning anything about i . If the receiver is a malicious party, he/she may select the undesired input for the malicious purpose from the sender, meaning the receiver discovers something that should not be known by him/her so that security may be breached.

The computational complexity of OT_1^N can be efficient as $O(\log N)$ calls of the OT_1^2 protocol [17] and the communication cost of OT_1^N protocol is $O(N)$. For one execution of OT_1^2 protocol, the computational and communication overheads using a random oracle [16] are 6 exponentiations and three times the bit length respectively. The approach by Goldreich [5] to securely split $(x + y) \log_2(x + y)$ has more overheads than the proposed protocol. Besides, the approach by Goldreich may only handle binary inputs [5]. Our approach is more general and works for non-binary inputs.

Although the above involves only two secure building blocks f , it can be easily extended to the general case:

$$g = (x_1 + y_1)(x_2 + y_2) \dots (x_n + y_n)$$

Due to the lack of space, we will not describe the generalization here.

4.3 Secure Computation of $g = (x + y) \log_2(x + y)$
Next we present a more efficient manner (using the proposed model) to securely compute

$$(x_1 + y_1) \log_2(x_1 + y_1) + \dots + (x_n + y_n) \log_2(x_n + y_n)$$

that is based on n pairs of random shares from n secure building blocks f . It has been used in privacy-preserving decision tree induction for horizontally partitioned data [14] and vertically partitioned data [3] to securely compute the entropy and information gain where f is secure scalar product protocols.

First, we show how to transform $g = (x + y) \log_2(x + y)$ to the expression of g' and then securely split g' into two portions. According to the Taylor series of the natural logarithm, we have

$$(4.2) \quad \ln(x) = \sum_{k=1}^{\infty} \frac{(-1)^{(k-1)}}{k} (x-1)^k \\ = (x-1) + \frac{-1}{2}(x-1)^2 + \dots$$

The convergence condition of Eq. 4.2 is $|x-1| < 1$, i.e., $0 < x < 2$. The error for a partial evaluation with the first n terms of the series is as follows:

$$(4.3) \quad \left| \ln(x) - \sum_{k=1}^n \frac{(-1)^{k-1}(x-1)^k}{k} \right| \\ < \left(\frac{|x-1|^{n+1}}{n+1} \right) \left(\frac{1}{1-|x-1|} \right) \\ \leq \delta$$

which shows the error shrinks exponentially as n grows. Within the predefined error δ , we can find n such that

$$\ln(x) \approx \sum_{k=1}^n \frac{(-1)^{k-1}(x-1)^k}{k}$$

We transform $(x + y) \log_2(x + y)$ as follows:

$$(x + y) \log_2(x + y) \\ = C \left(\frac{x}{C} + \frac{y}{C} \right) \log_2 \left(C \left(\frac{x}{C} + \frac{y}{C} \right) \right) \\ = C(x' + y') \log_2(C(x' + y')) \\ = C(x' + y') \log_2 C + C(x' + y') \log_2(x' + y') \\ (4.4) = C(x') \log_2 C + C(y') \log_2 C \\ + \frac{C}{\ln 2} (x' + y') \ln(x' + y')$$

where C is large enough to let $x'_k = x/C$ and $y' = y/C$ be in the range of $(0, 1)$. Then, $x' + y'$ is in the range of $(0, 2)$, which satisfies the convergence condition of Eq. 4.2.

From the preceding transformation, it shows that if $(x' + y') \ln(x' + y')$ in Eq. 4.4 is the expression of g' , $(x + y) \log_2(x + y)$ is the expression of g' as other terms in Eq. 4.4 are held by one party only.

By applying the Taylor series in Eq. 4.2 to $(x' + y') \ln(x' + y')$ in Eq. 4.4, we have

$$(x' + y') \ln(x' + y') \\ \approx (x' + y') \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k \\ = (x' + y' - 1) \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k \\ + \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k \\ (4.5) = \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^{(k+1)} \\ + \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (x' + y' - 1)^k$$

Next, we show two methods to securely split the terms in Eq. 4.5. In Method 1, we transform the terms in Eq. 4.5 to the expression of g' , and then directly apply Protocol 2. In Method 2, we propose a more efficient approach based on a proposed provable theorem.

Method 1: Based on Binomial formula

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} \\ = \begin{bmatrix} \binom{n}{0} a^0 \\ \binom{n}{1} a^1 \\ \vdots \\ \binom{n}{n} a^n \end{bmatrix} \cdot \begin{bmatrix} b^n \\ b^{n-1} \\ \vdots \\ b^0 \end{bmatrix}$$

we have

$$(x' + y') \ln(x' + y')$$

$$\begin{aligned}
&= \sum_{k=1}^n \left(\frac{(-1)^{(k-1)}}{k} \sum_{i=0}^{k+1} \binom{k+1}{i} (x'-1)^i (y')^{k+1-i} \right) \\
&\quad + \sum_{k=1}^n \left(\frac{(-1)^{(k-1)}}{k} \sum_{i=0}^k \binom{k}{i} (x'-1)^i (y')^{k-i} \right) \\
&= \sum_{k=1}^n \frac{(-1)^{(k-1)}}{k} (\mathbf{v}_k^a \cdot \mathbf{v}_k^b + \mathbf{w}_k^a \cdot \mathbf{w}_k^b)
\end{aligned}$$

where

$$\mathbf{v}_k^a = \begin{bmatrix} \binom{k+1}{0} (x'-1)^0 \\ \binom{k+1}{1} (x'-1)^1 \\ \vdots \\ \binom{k+1}{k+1} (x'-1)^{k+1} \end{bmatrix}, \quad \mathbf{v}_k^b = \begin{bmatrix} (y')^{k+1} \\ (y')^k \\ \vdots \\ (y')^0 \end{bmatrix}, \\
\mathbf{w}_k^a = \begin{bmatrix} \binom{k}{0} (x'-1)^0 \\ \binom{k}{1} (x'-1)^1 \\ \vdots \\ \binom{k}{k} (x'-1)^k \end{bmatrix}, \quad \text{and } \mathbf{w}_k^b = \begin{bmatrix} (y')^k \\ (y')^{k-1} \\ \vdots \\ (y')^0 \end{bmatrix}.$$

It is clear that we have transformed the secure computation of $(x' + y') \ln(x' + y')$ to secure computations of a series of secure scalar products of vectors \mathbf{v}_k^a and \mathbf{v}_k^b of length $k + 2$, and vectors \mathbf{w}_k^a and \mathbf{w}_k^b of length $k + 1$ for $1 \leq k \leq n$. The computational complexity and communication costs for $\mathbf{v}_k^a \cdot \mathbf{v}_k^b$ and $\mathbf{w}_k^a \cdot \mathbf{w}_k^b$ using Protocol 1 are $O(k + 2)$ and $O(k + 1)$ respectively. Overall, the computational complexity and communication costs are both $\sum_{k=1}^n (O(k + 2) + O(k + 1)) = O(n^2)$.

Method 2: Here we propose a more efficient way to securely split $(x' + y') \ln(x' + y')$.

From Eq. 4.5, we observe that if two parties securely split every k th term $(x' + y' - 1)^k$ into two private portions for $1 \leq k \leq n + 1$, then the two parties may easily obtain two portions respectively for terms in Eq. 4.5 (i.e., $(x' + y') \ln(x' + y')$) as $(-1)^{(k-1)}/k$ is a common constant factor known by two parties.

Therefore, we propose the following theorem to securely split all terms $(x' + y' - 1)^k$ for $1 \leq k \leq n + 1$:

THEOREM 4.1. *Every term of a total of n separate terms: $(x + y)^1, (x + y)^2, \dots, (x + y)^n$ can be splitted into two portions in the overall $n - 1$ executions of secure scalar product protocols for vectors of length 2. Hence, the overall computational complexity and communication costs are both $nO(2) = O(n)$.*

Proof. We prove the theorem above by mathematical induction. When $n = 1$, the private portions of the term $(x + y)^1$ are $r_a = x$ and $r_b = y$ which does not require any execution of secure scalar product protocols.

Assume the theorem is true for $n = \ell$; i.e., splitting every term of the total ℓ separate terms requires $\ell - 1$ executions of secure scalar product protocols into two

portions. Let the private portions of ℓ term be r_a and r_b . When $n = \ell + 1$, the new $(\ell + 1)$ th term

$$(x + y)^{\ell+1} = (r_a + r_b)(x + y)$$

can be splitted into two portions based in the manner described in Section 4.2. To securely split the new $(\ell + 1)$ th term, one more execution of secure scalar product protocols for vector of length 2 is required. Hence, $\ell - 1 + 1 = \ell$ executions of secure scalar product protocols are required to split the $\ell + 1$ terms. Thus, Theorem 4.1 is proved. \square

Based on Eq. 4.5, there are a total of $n + 1$ terms $((x' + 1) + y')^1, ((x' + 1) + y')^2, \dots$, and $((x' + 1) + y')^{n+1}$ to be securely splitted. Hence, the computational complexity and communication costs to securely split $(x' + y') \ln(x' + y')$ are both $((n + 1) - 1)O(2) = O(n)$ where n depends on the error predefined by the user when the Taylor series is used to approximate $\ln(x)$ in Eq. 4.3.

Summary: Compared with the first approach, the second approach is preferred as it has lower computational complexity $O(n)$ and communication cost $O(n)$.

The existing approach is proposed by Lindell and Pinkas [14] that runs the Yao's circuit evaluation protocol [26] on a circuit. The circuit is linear in the size of x and y . The oblivious transfers are required at every bit of circuit input. The overall computational complexity and communication costs are $O(\max\{\log |S|, n\}) = O(\log |S|)$ and $O(n \log |S|)$ respectively [14] where S is usually large (e.g., $\log |S| = 20$). These are both more than $O(n)$ of the proposed approach.

4.4 Other Secure Computations We can also securely compute kernel functions in various machine/statistical learning techniques without disclosing the private portions from secure building blocks f that have been applied in privacy-preserving SVM [27]:

- Linear kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b})$;
- Gaussian kernel $k(\mathbf{a}, \mathbf{b}) = \exp(-\|\mathbf{a} - \mathbf{b}\|^2 / 2\sigma^2)$;
- Polynomial kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + d)^p$; and
- Sigmoidal kernel $k(\mathbf{a}, \mathbf{b}) = \tanh(d_1(\mathbf{a} \cdot \mathbf{b}) + d_2)$

where \mathbf{a} and \mathbf{b} are private vectors held by Party A and Party B respectively. For kernel function g , the secure building blocks f are (1) the secure building blocks in linear kernel, polynomial kernel, and Sigmoidal kernel or (2) square of Euclidean distance in Gaussian kernel. In existing approaches [27], two private portions from secure building blocks f have to be assumed to be known by two parties to compute kernel functions g , such as the

polynomial kernel and Sigmoidal kernel. As shown in Section 3, it may breach the data privacy of the honest party if the private portion from secure scalar product protocols f is known by the other party.

Based on the techniques in Section 4.3, we are able to securely compute kernel functions without disclosing any private portions to the other party. For instance, the Binomial formula or Theorem 4.1 can be used to securely split polynomial kernel. Taylor series and Theorem 4.1 can be used to securely split Sigmoidal kernel in the similar manner of Section 4.3. Due to the lack of space, we will not describe the details to securely split the four kernel functions.

Besides, we are also able to transform many other common functions into the expression of g' if they can be converted to a convergent series by Taylor series. The common functions include the square root ($\sqrt{x+y}$), trigonometric functions ($\sin(x+y)$, $\cos(x+y)$, $\tan(x+y)$, etc.), and hyperbolic functions ($\sinh(x+y)$, $\cosh(x+y)$, $\tanh(x+y)$, etc.). Based on the propose model, we can securely split them without disclosing the private portions x or y to the other party in the similar manner of Section 4.3. We believe that secure computations of the common functions have broad potential applications.

4.5 Summary This section discussed a general model based on the random shares technique. We proposed Protocol 2 based on the general model. Then we showed efficient ways to compute $(x_1+y_1)(x_2+y_2)$ and $(x+y)\log_2(x+y)$. Besides, we showed how three common kernel functions and other common functions can be securely computed.

The random shares technique has been applied to preempt the vulnerabilities of the malicious model in PPDm. However, the measure of random shares cannot restrict and restraint the scope of activity of the malicious party. To do that, we propose two necessary conditions and other two measures in the following section.

5 Other Preemptive Measures against Privacy Vulnerabilities

Of the two current models (semi-honest model and malicious model) of party interaction regularly adopted in PPDm algorithms, the malicious model has a higher risk of privacy breaches as a party in the malicious model may behave (and deviate) arbitrarily from the protocol. The random shares technique was proposed to prevent the probing attack in Section 4. However, it cannot restrict and restraint the scope of activity of the malicious party. Therefore, in this section we propose two conditions in Section 5.1 and then two measures in

Section 5.2.

We can directly check whether the supposedly same data supplied by each party change in the course of execution of the PPDm algorithm. We can also check whether any critical property of the PPDm algorithm is preserved across the same or different iterations of the algorithm, as any violation of such property is definite indication of malicious activity. Although it is remotely possible for certain algorithmic properties to be preserved across iterations even when a malicious party supplied false input, the checks serve to restrict and restraint the scope of activity of the malicious party.

For instance, in decision tree induction, attribute values are repeatedly used to determine the best attribute for splitting the current tree node. Any inconsistency of data values across different iterations from a malicious party can be detected and actions can be taken to expose the malicious party or to prevent further privacy breaches.

In association rule mining, there are properties that hold across all iterations, such as the property that the confidence of a rule must be larger or equal to the support of the same rule. In iterative algorithms such as the Apriori algorithm, larger frequent itemsets are discovered in each progressive iteration. The rules derived from the frequent itemsets across all iterations must preserve the aforementioned property. If the property is not preserved across iterations, it is possible that a malicious party is using false data inputs and steps can be taken to detect and expose the party.

5.1 Two Conditions for Secure Protocols Using the formal definition framework of malicious model by Kantarcioglu and Kardes [12], we formally present the two conditions below with respect to a data mining algorithm Γ :

(1) Given a protocol Π of Γ , the relationships \mathcal{R} between input I_i of Π in iteration i and input I_j of Π in iteration j in real-life model (EXEC) are equivalent to ones in the ideal model (IDEAL).

$$\text{IDEAL}_{\mathcal{R}(I_i, I_j)} \equiv \text{EXEC}_{\mathcal{R}(I_i, I_j)}$$

where $\mathcal{R}(I_i, I_j) \in \{=, nil\}$ is the relationship of inputs I_i and I_j of Π between iterations i and j and nil refers to two objects do not have any relationships.

(2) Given a protocol Π of Γ , the relationship \mathcal{R} of output O_i of Π in iteration i and output O_j of Π in iteration j in real-life model (EXEC) are equivalent to ones in the ideal model (IDEAL).

$$\text{IDEAL}_{\mathcal{R}(O_i, O_j)} \equiv \text{EXEC}_{\mathcal{R}(O_i, O_j)}$$

where $\mathcal{R}(O_i, O_j) \in \{>, =, <, nil\}$ is the relationship of

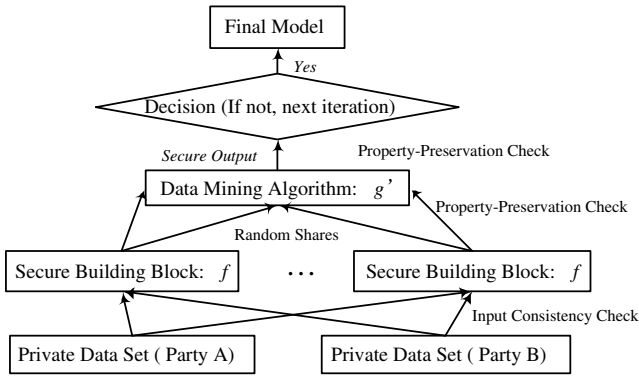


Figure 2: Modified model of privacy-preserving data mining algorithms for two parties.

O_i and O_j .

The first condition says that for any secure protocol used as part of a PPDM algorithm’s execution, the input data for the same attributes to the protocol must be consistent. The second condition says that the output of the protocol must preserve the property of the data mining algorithm.

5.2 Two Measures for Secure Protocols To fulfill two conditions above, we present input consistency check and property-preservation check respectively.

Input Consistency Check: In a multi-party setting, each party monitors and check the consistency of data supplied by all the other parties in the course of execution of the PPDM algorithm. Using the *collision resistance* property of public key encryption schemes where encryptions are the same only if the original values are the same, data consistency can be checked without revealing the actual data value. Based on the schemes such as Naccache-Stern cryptosystem [15], Paillier Cryptosystem [18], Pohlig-Hellman [19], and so on, encryptions are computed using some private random values. To demonstrate consistency in data supplied by a party, the party is required to use the same random value to encrypt the same input data. Even if the random value is used several times, it is impossible to be disclosed.

Property-Preservation Check: To restraint the behavior of any potential malicious party, properties of the PPDM algorithm can be used. If any of these properties are not preserved in the course of execution of the algorithm, we may assert that a malicious party has provided some false data input. Specific to each PPDM algorithm is a set of idiosyncratic properties that must hold throughout the algorithm.

Any party in the multi-party setting may choose which of these properties to monitor and when for any other party that it suspects. For instance, when two parties performed computations using secure building blocks, an honest party may check a particular algorithmic property when it suspects that there is malicious behavior. That substantially reduces the complexity of the property-preservation check as the check is only performed only when the honest party thinks it is necessary.

In summary, Fig. 2 shows the modified model by incorporating the three measures; namely, random shares, input consistency check and property-preservation check, to the general model with the secure output for decision making in Fig.1 to enhance data security of the participating parties in the current malicious model.

6 Conclusions

In the paper, we discussed the problem of security violations for PPDM when a malicious party gives false data. To demonstrate the severity of the problem, we identified four possible privacy vulnerabilities of secure scalar product protocols. To fix these vulnerabilities, we proposed a general model for two participating parties together with the protocol. To illustrate the applicability of the proposed model, we showed two efficient approaches to securely split $(x_1 + y_1)(x_2 + y_2)$ and $(x + y) \log_2(x + y)$ respectively. We also showed that four kernel functions and other common functions can be securely computed using the proposed model. We proposed two necessary conditions and then two basic measures to restrict and restraint the scope of activity of the malicious party.

We are interested in exploring other applications of secure computations of the common functions, eventually to enhance more algorithms with privacy-preserving features. As the general model based on random shares technique only works for two parties, extending to multiple parties is another part of the future work. We would also like to explore the second problem that is a malicious party aborts the protocol prematurely.

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] 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.

- [3] 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.
- [4] 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.
- [5] O. Goldreich. Secure multi-party computation. manuscript, 2002.
- [6] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
- [7] 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.
- [8] 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.
- [9] Health insurance portability and accountability act of 1996. *U.S. Congress*, 1996.
- [10] 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.
- [11] W. Jiang and C. Clifton. Ac-framework for privacy-preserving collaboration. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, page To Appear, Minneapolis, Minnesota, April 26–28 2007.
- [12] M. Kantarcioglu and O. Kardes. Privacy-preserving data mining in malicious model. Technical report, Department of Computer Science, Stevens Institute of Technology, October 2006.
- [13] M. Kantarcioglu and O. Kardes. Privacy-preserving data mining applications in the malicious model. In *Proceedings of the 6th International Workshop on Privacy Aspects of Data Mining held in conjunction with the IEEE International Conference on Data Mining (ICDM 2007)*, Omaha, Nebraska, United States, October 28 2007.
- [14] 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.
- [15] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 59–66, San Francisco, California, United States, 1998. ACM Press.
- [16] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457, Washington, D.C., United States, 2001.
- [17] M. Naor and B. Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, 2005.
- [18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology - EUROCRYPT '99*, pages 223–238, 1999.
- [19] S. Pohlig and M. Hellman. An improved algorithm for computing logarithm over $gf(p)$ and its cryptographic significance. *IEEE Transaction on Information Theory*, 24:106–110, 1978.
- [20] Standards for privacy of individually identifiable health information. *Federal Register*, pages 53181 – 53273, August 14, 2002.
- [21] 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.
- [22] J. Vaidya and C. Clifton. Privacy preserving naive bayes classifier for vertically partitioned data. In *Proceedings of the SIAM International Conference on Data Mining*, pages 522–526, Lake Buena Vista, Florida, 2004.
- [23] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), 2005.
- [24] 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.
- [25] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining*, pages 713–718, Seattle, WA, USA, 2004.
- [26] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.
- [27] 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.
- [28] J. Zhan, S. Matwin, and L. Chang. Privacy-preserving collaborative association rule mining. *Journal of Network and Computer Applications*, 30(3):1216–1227, 2007.
- [29] S. Zhong. Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sciences*, 177(2):490–503, 2007.