



Privacy-Preserving Data Mining in Malicious Model

Murat Kantarcioglu
Department of Computer Science
University of Texas at Dallas
Richardson, TX, 75083-0688, USA
muratk@utdallas.edu

Onur Kardes
Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ, 07030, USA
onur@cs.stevens-tech.edu

Department of Computer Science
Technical Report CS-2006-06
October 02, 2006

Department of Computer Science · Stevens Institute of Technology
Castle Point on Hudson · Hoboken, NJ 07030 · USA

Privacy-Preserving Data Mining in Malicious Model

Murat Kantarcioglu

Onur Kardes

Abstract

Most of the previous cryptographic work in privacy-preserving data mining suggest solutions in the semi-honest model. Semi-honest model assumes that participating parties follows the prescribed protocol but try to infer private information using the messages they receive during the protocol. Although semi-honest model is realistic in many settings, there are cases where it may be better to use the “malicious model” which tries to prevent any malicious behavior by using more expensive cryptographic techniques. Clearly, protocols that are secure in the malicious model provide more security compared to the ones in the semi-honest model. At the same time, there is an obvious trade-off: malicious model that provides higher security guarantees with higher computational cost versus more efficient semi-honest model with less security guarantees. In this paper, our goal is to analyze the relative cost of privacy-preserving data mining algorithms in both models to see their difference in performance. In order to make a realistic comparison, we first provide the “malicious” versions of the commonly used semi-honest subprotocols in privacy-preserving data mining and compare the performance of these protocols in both models.

1 Introduction

Data needed for many crucial data mining tasks is distributed among different parties with different security and privacy concerns. For example, different credit card companies may need to combine their data sets to build a better credit card fraud detection system. In many of these distributed data mining settings, the revelation of the original data sets are not acceptable due to privacy concerns. For example, credit card companies may not be willing to share their data sets directly, due to the fact that, those data sets may reveal valuable competitive information. In order to address this problem, many privacy-preserving data mining protocols that use cryptographic techniques have been suggested. (Please read Section 2.1 for the detailed discussion of the previous work) In the semi-honest model, each party follows the protocol without any deviation and adversaries behave in a semi-honest fashion; which may be a valid assumption for many different scenarios. For example, in the case of credit card companies, semi-honest behavior may be reasonable because of several reasons: First of all, companies may not want to take the risk of learning an incorrect result due to malicious behavior (i.e., due to deviating from the prescribed protocol). Also, if they caught cheating during or after the execution of the protocol, this may jeopardize their future relations with the other companies. Nevertheless, semi-honest assumption may not be enough in some certain settings. For example, different nations that cooperate to learn

data mining models for terrorist detection may not trust each other completely. In this case, they may prefer to use the protections of the malicious model.

However, the cost trade-off between the protocols in the malicious model and the semi-honest model is not clear in practice. We believe that giving a detailed analysis of the corresponding costs could be a valuable tool for decision makers. Since there are many different privacy-preserving data mining algorithms in practice, it is not feasible to make every one of them secure in the malicious model for comparison purposes. Instead, we focus on the common subprotocols that are used in privacy-preserving data mining and modify them to be secure in the malicious model. Since many data mining algorithms make use of these subprotocols in different fashions, information about the relative cost of implementing any given subprotocol in the malicious model could be used to give a precise estimate of the total cost of the privacy-preserving data mining algorithm in the malicious model. For the reasons stated above, we analyzed the costs of secure equality, secure dot product and secure set operations protocols in the malicious model. In this paper, we focus on the two-party protocols and leave the extension for the multi-party case as a future work.

1.1 Our Contributions

We provide two-party secure protocols in the malicious model for equality, dot product and full domain set operations. For dot product, we provide two different protocols: the first protocol is a straightforward extension of the semi-honest version and the second protocol is a secure protocol in the malicious that provides a more efficient way than the first one. These initial results indicate that for efficiency purposes, we may want to fully redesign the protocols in the malicious model, instead of directly converting the semi-honest protocols using zero-knowledge proofs. We also provide extensive experimental analysis of the given protocols.

1.2 Organization of the Paper

In section 2, we discuss the necessary background information in privacy-preserving data mining and secure multi-party computations. In section 3, we provide secure subprotocols in the malicious model along with their security analysis. In section 4, we compare the relative efficiency of each protocol both in the semi-honest and the malicious model. Finally, we conclude the paper with the discussions of the current results and the future work.

2 Background

In this section, we first discuss the previous work done in privacy-preserving data mining. Later on, we provide the necessary cryptographic definitions and tools used in this paper.

2.1 Related Work

Many different distributed privacy-preserving data mining algorithms were implemented using cryptographic techniques. Usually two different assumptions about the distribution of the data are used in those protocols. In the horizontally partitioned data, it is assumed that different sites collect the same set of information about different entities. For example, different credit card companies may collect credit card transactions of different individuals. Secure protocols for mining decision trees [1], association rules [2], k-means clusters [3], k-nn classifiers [4] and many other models have been developed for the horizontally partitioned data.

In the vertically partitioned case, it is assumed that different sites collect information about the same set of entities but they collect different feature sets. For example, both a university and a hospital may collect information about a student. Again, secure protocols for the vertically partitioned case have been developed for mining association rules [5], decision trees [6] and k-means clusters [7].

In several recent studies, all of those previous protocols were proven to be secure in the semi-honest model. Instead, in this paper, we focus on the security issues in the malicious model and provide the malicious versions of the subprotocols used in the previous privacy-preserving data mining algorithms.

2.2 Cryptographic Background

Our definition of privacy-preserving data mining implies that nothing other than the final data mining result is revealed during the data mining process. This definition is equivalent to the “security” definition used in secure multiparty computation (SMC) literature.

In this section, we give an overview of the cryptographic definitions and techniques that are used in this paper.

2.2.1 Security in Malicious Model

In order to provide proofs of security, we need to provide an exact definition of security first. Since this is a well studied subject in SMC domain, we directly follow the security definitions given in the literature. In our analysis, we assume that a malicious (i.e., active) adversary can only corrupt the same (and one) party during the entire protocol. In other words, we only deal with active, static adversaries. As mentioned in the introduction section, we focus on only two-party protocols. Therefore, we modify the definitions given in [8] for two-party case.¹

Below, we first introduce the execution of the protocol in the real-life model and then define the execution of the protocol in the ideal model. Finally, we define the security as emulating the real-life execution of a protocol in the ideal model.

¹Although similar security definitions given in [9] and [10] could be used for our purposes, since we use the same zero-knowledge techniques given in [8]. We follow the definitions from [8].

Definition 2.1. The Real-Life Model[8]

Let π be a two-party protocol where each party P_i has a secret input x_i^s and a public input x_i^p . Also each P_i returns a private output y_i^s and a public output y_i^p after the execution of the protocol. Let A be the adversary that can corrupt any one (and only one) party during the execution of the protocol. Let $\vec{x} = (x_1^s, x_1^p, x_2^s, x_2^p)$ be the participating parties input, let $\vec{r} = (r_1, r_2, r_A)$ be the random inputs of the parties and the adversary, let $C \in \{1, 2\}$ be the index of the corrupted party, and let $z \in \{0, 1\}^*$ be the auxiliary input. Also let k be the security parameter. We denote the output of the adversary after the execution of the protocol as $ADVR_{\pi,A}(k, \vec{x}, C, z, \vec{r})$ and similarly the output of the party P_i as

$$EXEC_{\pi,A}(k, \vec{x}, C, z, \vec{r})_i$$

Let

$$\begin{aligned} EXEC_{\pi,A}(k, \vec{x}, C, z, \vec{r}) &= (ADVR_{\pi,A}(k, \vec{x}, C, z, \vec{r}), \\ &EXEC_{\pi,A}(k, \vec{x}, C, z, \vec{r})_1, \\ &EXEC_{\pi,A}(k, \vec{x}, C, z, \vec{r})_2) \end{aligned}$$

Also we denote the $EXEC_{\pi,A}(k, \vec{x}, C, z)$ be the random variable for a uniformly chosen \vec{r} . Finally, we define a distribution ensemble $EXEC_{\pi,A}$ with security parameter k which is indexed by (\vec{x}, C, z) as : $EXEC_{\pi,A}(k, \vec{x}, C, z)_{k \in N, \vec{x} \in (\{0,1\}^*)^2, C \in \{1,2\}, z \in \{0,1\}^*}$

Definition 2.2. Ideal Model[8]

Let $f : N \times (\{0, 1\}^*)^4 \times \{0, 1\}^* \mapsto (\{0, 1\}^*)^4$ be a probabilistic two-party function computable in probabilistic polynomial time. We define the output of f as $f(k, x_1^s, x_1^p, x_2^s, x_2^p, r) = (y_1^s, y_1^p, y_2^s, y_2^p)$ where k is the security parameter and r is the random input. In the ideal model, parties send their inputs to an incorruptible trusted party which draws r uniformly random, computes f and returns the party P_i its output value (y_i^s, y_i^p) . At the beginning of the execution the ideal model adversary S sees the x_1^p and x_2^p values and the secret x_C^s value for the corrupted adversary. After this point, S replaces x_C^s, x_C^p with the \bar{x}_C^s, \bar{x}_C^p values of its choice. f is then evaluated by the trusted party using the modified inputs. After the evaluation, P_i receives its output value (y_i^s, y_i^p) . Again adversary sees the y_1^p, y_2^p values and the y_C^s value for the corrupted party. Similar to the real-life mode, let

$$\begin{aligned} IDEAL_{f,S}(k, \vec{x}, C, z, \vec{r}) &= (ADVR_{f,S}(k, \vec{x}, C, z, \vec{r}) \\ &IDEAL_{f,S}(k, \vec{x}, C, z, \vec{r})_1 \\ &IDEAL_{f,S}(k, \vec{x}, C, z, \vec{r})_2) \end{aligned}$$

denote the collection of the the outputs and $IDEAL_{f,S}$ is the distribution ensemble indexed by (\vec{x}, C, z) .

Definition 2.3. Security in the Static Malicious Adversary Setting[8]

Let f be a two-party function and let π be a protocol for two parties. We say that π securely evaluate f in the static setting if for any probabilistic polynomial time adversary A , there

exists an ideal-model adversary S whose running time is polynomial in the running time of A , and such that

$$IDEAL_{f,S} \stackrel{C}{\approx} EXEC_{\pi,A} \quad (1)$$

where $\stackrel{C}{\approx}$ denotes the computational indistinguishability between two ensembles.

Basically, security in this model implies that given any adversary in the real-life model we can emulate an adversary in the ideal model that has computationally indistinguishable outputs. Therefore, in the security proofs, we will define an ideal model adversary S that runs *any* given real-life adversary A as a subroutine in a black-box fashion and we will show that their views are computationally indistinguishable.

We also need to combine several secure function evaluations to create new protocols. In order to prove that the composed protocol is secure, we first show that the composed protocol is secure given oracle access to the needed functions. Later on, using the theorems stated in [10], we can replace the oracle calls with the real-life protocols without violating security. In order to formalize the above intuition, we first define the hybrid model:

Definition 2.4. [8] *The Hybrid Model: two-party case*

The execution of the protocol π in the (h_1, h_2, \dots, h_m) -hybrid model proceeds as in the real-life model, except that the parties have oracle access to a trusted party T for evaluating the two-party functions h_1, \dots, h_m . These function evaluations proceed as in the ideal-model. Similar to the previous definitions, we denote the output of the protocol with the following distribution ensemble

$$EXEC_{\pi,A}^{h_1, \dots, h_m}$$

Similar to the security definition given above, we define the security in the hybrid model by requiring that for any adversary operating in the hybrid model, there exists an adversary S in the ideal model such that

$$IDEAL_{f,S} \stackrel{C}{\approx} EXEC_{\pi,A}^{h_1, \dots, h_m} \quad (2)$$

As mentioned before, we can replace the oracle calls to function h_i with its secure implementation in the real-life model without sacrificing security. Please refer to [11] for details.

2.2.2 Homomorphic Encryption

Many subprotocols used in privacy-preserving data mining algorithms such as secure $\log(x)$ [12], set operations [13], secure dot product [7] protocols use homomorphic encryption. In order to provide a fair comparison, we also utilize secure homomorphic public key cryptosystem in our solutions against malicious adversaries.

Let $E_{pk}(\cdot)$ denote the encryption function with public key pk and $D_{pr}(\cdot)$ denote the decryption function with private key pr . A secure public key cryptosystem is called homomorphic if it satisfies the following requirements: (1) Given the encryption of m_1 and m_2 , $E_{pk}(m_1)$ and $E_{pk}(m_2)$, there exists an efficient algorithm to compute the public key encryption of $m_1 + m_2$, denoted $E_{pk}(m_1 + m_2) := E_{pk}(m_1) +_h E_{pk}(m_2)$. (2) Given a constant k and

the encryption of m_1 , $E_{pk}(m_1)$, there exists an efficient algorithm to compute the public key encryption of km_1 , denoted $E_{pk}(km_1) := k \times_h E_{pk}(m_1)$.

For the sake of completeness, we provide a brief definition of the homomorphic cryptosystem that we use in our experiments. Please refer to [14] for details. Paillier cryptosystem, which is based on composite residuosity assumption satisfies the above properties and can be defined as follows:

- **Key Generation:** Let p and q are prime numbers where $p < q$ and p does not divide $q - 1$. For the Paillier encryption scheme, we set the public key pk to n where $n = pq$ and private key pr to (λ, n) where λ is the lowest common multiplier of $p - 1, q - 1$.
- **Encryption with the Public Key:** Given n , the message m , and a random number r from 1 to $n - 1$, encryption of the message m can be calculated as follows: $E_{pk}(m) = (1 + n)^m r^n \bmod n^2$. Also note that given any encrypted message, we can get a different encryption by multiplying it with some random r^n .
- **Decryption with the Private Key:** Given n , the cipher text $c = E_{pk}(m)$, we can calculate the $D_{pr}(c)$ as follows: $m = \frac{(c^\lambda \bmod n^2) - 1}{n} \lambda^{-1} \bmod n$ where λ^{-1} is the inverse of λ in modulo n .
- **Adding Two Ciphertexts ($+_h$):** Given the encryption of m_1 and m_2 , $E_{pk}(m_1)$ and $E_{pk}(m_2)$, we can calculate the $E_{pk}(m_1 + m_2)$ as follows:

$$\begin{aligned}
 E_{pk}(m_1) * E_{pk}(m_2) \bmod n^2 &= ((1 + n)^{m_1} r_1^n) * ((1 + n)^{m_2} r_2^n) \bmod n^2 \\
 &= ((1 + n)^{m_1 + m_2} (r_1 * r_2)^n) \bmod n^2 \\
 &= E_{pk}(m_1 + m_2)
 \end{aligned}$$

We would like to emphasize that this addition will actually return $E_{pk}(m_1 + m_2 \bmod n)$.

- **Multiplying a Ciphertext with a Constant ($k \times_h E_{pk}(m_1)$):** Given a constant k and the encryption of $m_1, E_{pk}(m_1)$, we can calculate $k \times_h E_{pk}(m_1)$ as follows:

$$\begin{aligned}
 k \times_h E_{pk}(m_1) &:= E_{pk}(m_1)^k \bmod n^2 \\
 &= ((1 + n)^{m_1} r_1^n)^k \bmod n^2 \\
 &= (1 + n)^{km_1} r_1^{kn} \bmod n^2 \\
 &= E_{pk}(km_1)
 \end{aligned}$$

In our solutions, which are secure against malicious adversaries, we use a slightly different version of the homomorphic encryption described above. Also, we use efficient zero-knowledge protocols in random oracle model to prove that the actions taken by the parties are correct without revealing any other information. We briefly summarize those protocols below. The implementation details of those protocols for Paillier encryption can be found in [14, 8].

- **Threshold Decryption (two-party case):** Given the common public key pk , the private key pr corresponding to pk has been divided into two pieces pr_0 and pr_1 . There exists an efficient, secure protocol $D_{pr_i}(E_{pk}(a))$ such that it outputs the random share of the decryption result s_i along with the non-interactive zero knowledge proof $POD(pr_i, E_{pk}(a), s_i)$ showing that pr_i used correctly. Those shares can be combined to calculate the decryption result. Also any single share of the private key pr_i cannot be used to decrypt the ciphertext alone. In other words s_i does not reveal anything about the final decryption result. We also use the special version of the threshold decryption such that only one party learns the decryption result. Such protocol could be easily implemented by using the fact that for any given $E_{pk}(a)$, the party that needs to learn the decryption result could generate $E_{pk}(r1)$ and then both parties jointly decrypt the $E_{pk}(a) +_h E_{pk}(r1)$. Since only one party knows the $r1$, that party can learn the correct decryption result.
- **Proving that you know a plaintext:** A party P_i can compute the zero knowledge proof $POK(e_a)$ where he shows that he knows an element a in the domain of valid plaintexts such that $D_{pr}(e_a) = a$.
- **Proving that multiplication is correct:** Assume that party P_i is given an encryption $E_{pk}(a)$ and chooses constant c and calculates $E_{pk}(a.c)$. Later on, P_i can give zero knowledge proof $POMC(e_a, e_c, e_{a.c})$ such that $D_{pr}(e_a) = D_{pr}(a)$ and $D_{pr}(e_{a.c}) = D_{pr}(e_c).D_{pr}(e_a)$.

In our security proofs, we use the simulators for the above zero-knowledge proofs guaranteed due to their security properties. As discussed in [8], those simulators return a state of the adversary that is statistically indistinguishable from the state of the adversary in the real-life execution. Also, those simulators return the secret inputs used by the adversary for valid zero-knowledge proofs with overwhelming probability. We use both of these properties in our security proofs.

3 Secure Protocols in Malicious Model

All the protocols mentioned in this section are implemented in semi-honest model as primitives for different privacy-preserving data mining algorithms. Here we summarize these basic sub-protocols and provide ways to make them secure in malicious model by using efficient zero-knowledge proofs given above.

3.1 Secure Equality Protocol

One of the most common tools needed in many privacy-preserving data mining algorithms is to calculate whether two items are equal or not without revealing these items. Such secure equality protocol could easily be implemented using homomorphic encryption in semi-honest model without using threshold decryption. The idea is simple. P_0 generates a homomorphic

key pair and send the pk to P_1 along with the item $E_{pk}(x_0)$. Given pk , P_1 calculates $E_{pk}(-1.(x_1))$ where x_1 is the P_1 's item. After that P_1 calculates $e = (E_{pk}(-1.(x_1)) +_h E_{pk}(x_0)) \times_h r = E_{pk}((x_0 - x_1)r)$ where r is a random number and send e to P_0 . Clearly if $x_0 = x_1$ then $D_{pr}(e) = 0$ else it is a random number. Although the above protocol is correct in semi-honest model, it does not work in malicious model where P_1 uses a $r = 0$ to make the decryption 0 any time he wants.

We can develop an equality protocol that is secure in the malicious model using the threshold version of the homomorphic encryption for two-party case. Basic idea in the malicious version is that two parties jointly calculate $E_{pk}((x_0 - x_1)(r_0 + r_1))$ where r_0, r_1 are random numbers chosen by the respective parties. Here at each step, using the zero knowledge protocols described above, P_0 and P_1 prove that the actions they have taken are consistent with the protocol without revealing anything.

In this protocol, each party sends the encrypted x_i value to the other party. Since each party only has the share of the private key, they cannot decrypt the other party's encrypted x_i value. Later on using the homomorphic encryption, each party calculates $E_{pk}((x_0 - x_1)r_i)$ and proves to the other party that the calculation is correct. After that each party calculates $E_{pk}((x_0 - x_1)(r_0 + r_1))$ and jointly decrypts $E_{pk}((x_0 - x_1)(r_0 + r_1))$ to learn $(x_0 - x_1)(r_0 + r_1)$. Clearly if the decrypted value is zero with very high probability $x_0 = x_1$ else it means that $x_0 \neq x_1$. The details of the secure equality protocol in malicious model is given in Protocol 1. Please note that we use an oracle call to do threshold decryption, instead of actual implementation. As stated before, such oracle calls could be replaced with actual secure implementation.

We can easily prove that the above protocol is secure in malicious model. More specifically,

Theorem 3.1. *Protocol 1 is secure in the (decryption)-hybrid model assuming that the non-interactive zero-knowledge protocols used are secure in the malicious model.*

Proof. (Sketch) Here we give an outline of the proof by skipping a few technical details. In order to prove the security of the protocol, for any adversary A operating in the hybrid model, we need to define an adversary S_A operating in the real-life model such that the views of the both adversaries are computationally indistinguishable. In order to define such S_A , we will use A as a subroutine. Before the simulation starts, S_A will be given the description of the A , private input of the corrupted party x_0 ,² the final result of the equality test b , public key pk and the private key of the P_1 . Now we can define the S_A as follows:

1. Run A to get $E_{pk}(x_0)$ along with the $POK(e_{x_0})$
2. Run the simulator S_{POK} by giving the current state of the A and $E_{pk}(x_0)$ as an input to S_{POK} . If the simulator of the proof fails terminate the protocol, else set the state of A returned by the S_{POK}

²Since the equality protocol defined here is symmetric, we assume that P_0 is corrupted without loss of generality

Protocol 1 Secure Equality in Malicious Model Using Threshold Decryption

Require: Two parties P_0 and P_1 with the shares pr_0 and pr_1 of the private key and private inputs x_0 and x_1 .

Ensure: Return 1 if $x_0 = x_1$ else return 0

for all P_i **do**

 Calculate $e_{x_i} = E_{pk}(x_i)$

 Create $POK(e_{x_i})$

 Send $(e_{x_i}, POK(e_{x_i}))$ to P_{1-i}

end for

for all P_i **do**

 Check $POK(e_{x_{1-i}})$ is valid else **ABORT**

 Calculate $e_{x_0-x_1} = e_{x_0} +_h (-1 \times_h e_{x_1})$

 Choose non-zero random r_i and calculate $e_{r_i} = E_{pk}(r_i)$, $e_{(x_0-x_1)r_i} = e_{x_0-x_1} \times_h r_i$

 Create $POMC(e_{x_0-x_1}, e_{r_i}, e_{(x_0-x_1)r_i})$

 Send $(e_{x_0-x_1}, e_{r_i}, e_{(x_0-x_1)r_i})$, $POK(e_{r_i})$, $POMC(e_{x_0-x_1}, e_{r_i}, e_{(x_0-x_1)r_i})$ to P_{1-i}

end for

for all P_i **do**

 Check whether the $e_{x_0-x_1}$ send by P_{1-i} is correct else **ABORT**

 Check whether $POK(e_{r_{1-i}})$ is valid else **ABORT**

 Check whether $POMC(e_{x_0-x_1}, e_{r_{1-i}}, e_{(x_0-x_1)r_{1-i}})$ valid else **ABORT**

 Calculate $e_{(x_0-x_1)(r_0+r_1)} = e_{(x_0-x_1)r_1} +_h e_{(x_0-x_1)r_0}$

end for

for all P_i **do**

 Jointly use the trusted party T to get $D_{pr}(e_{(x_0-x_1)(r_0+r_1)})$

if $(x_0 - x_1)(r_0 + r_1) = 0$ **then**

 return 1

else

 return 0

end if

end for

3. If b is 1 then feed A with $E_{pk}(x_0)$ else feed A with $E_{pk}(r_a)$ for some random $r_a \neq x_0$ along with the correct zero-knowledge proof. Let x_{S_A} be the plaintext value given to A in this step.
4. Run S_{POCM} to simulate the zero-knowledge proof and set the state of A by the state returned by the S_{POCM} . If the proof fails terminate. Also feed the A with the correct zero-knowledge proof for the encrypted value given to A in the previous state.
5. Get the $e_{(x_0-x_{S_A})(r_0+r_1)}$ for the oracle call to *Decrypt* function and gives A a random number if $b = 0$ else gives 0
6. S_A outputs whatever A outputs.

We now need to prove that the view of S_A is computationally indistinguishable from the execution in the hybrid-model. First note that until step 2, the view of the A in the simulation is statistically indistinguishable to the view of A in the hybrid-model. Security of the zero-knowledge guarantees that on computationally indistinguishable inputs, the output state of the zero-knowledge proof simulator is identical to state of A in the hybrid protocol. With the similar arguments, we can say that the state of A before the step 5 is identical to the state in the hybrid model. Now we need to show that the result returned by the decryption call in the simulation is statistically indistinguishable than the one seen by the A in the actual implementation. If $b = 1$, then in the both executions, A will be given 0, if $b = 0$ than in the simulation, A can see any value with equal probability, in the hybrid-model execution A will get $(x_0 - x_1)(r_0 + r_1)$. Since r_1 is random, the probability that $(x_0 - x_1)(r_0 + r_1)$ equals to zero is negligibly small in terms of security parameter. Also $(x_0 - x_1)(r_0 + r_1)$ is distributed uniformly distributed in Z_n^* assuming that all operation are modulo n . Therefore the state of A after the step 5 is the same in both executions. This concludes our proof. \square

3.2 Secure Dot Product

Secure Dot product is an another useful subprotocol that is commonly used in many privacy-preserving data mining algorithms. Using homomorphic encryption, it is straightforward to develop a secure dot product protocol in the semi-honest model. Similarly, P_0 generates a homomorphic key pair and send the pk to P_1 along with the encrypted vector $E_{pk}(\bar{x}_0) = (E_{pk}(x_{00}), E_{pk}(x_{01}), \dots, E_{pk}(x_{0n}))$. Given pk , P_1 calculates $e_{\bar{x}_0.\bar{x}_1} = (E_{pk}(x_{00}) \times_h x_{10}) +_h (E_{pk}(x_{01}) \times_h x_{11}) +_h \dots +_h (E_{pk}(x_{0n}) \times_h x_{1n})$ where x_{1i} is the P_1 's vector's i^{th} component and sends $e_{\bar{x}_0.\bar{x}_1} +_h E_{pk}(r_1)$ to P_0 . By decrypting the P_1 's message, P_0 learns the random share of the dot product result $\bar{x}_0.\bar{x}_1 + r_1$. Clearly P_0 and P_1 can combine their shares to learn $\bar{x}_0.\bar{x}_1$.

Now we show how to evaluate the dot product in malicious model for two-party case securely using threshold homomorphic encryption. We provide two different secure dot product protocols that can be used in the malicious model. The first protocol given in Section 3.2.1 is a generic extension of the semi-honest protocol described above using appropriate zero-knowledge proofs. Later on, in Section 3.2.2, we show ways to provide a solution that is more efficient than the straightforward extension version. Our second dot product protocol can

also be seen as an example where more efficient protocols can be developed instead of using generic transformation techniques to convert semi-honest protocols to malicious protocols.

3.2.1 Converting Secure Dot protocol in Semi-Honest Model to Malicious Model

If we look at the semi-honest dot product protocol carefully, we need to make sure that the P_1 does the multiplications correctly and all the encryptions sent are valid. These could be easily achieved using the zero knowledge protocols described in Section 2.2.2. P_0 sends the encrypted values along with the associated proof of correct encryption to P_1 . For each multiplication, P_1 generates the zero knowledge proof of correct multiplication and send those to P_0 . P_0 can check those proofs to make sure that dot product calculated correctly. The details are described in Protocol 2.

Protocol 2 Secure Dot Product in Malicious Model Using Threshold Decryption: Extension of the semi-honest version

Require: Two parties P_0 and P_1 with the shares pr_0 and pr_1 of the private key pr and n bit vectors \bar{x}_i where \bar{x}_i belongs to P_i .

Ensure: Return $r_0 = \sum_i^n (x_{0i} \cdot x_{1i}) + r_1$ to P_0 and r_1 to P_1

for P_0 **do**

$\forall i$, set $e_{x_{0i}} = E_{pk}(x_{0i})$ and create $POK(e_{x_{0i}})$

Send encryptions and non-interactive zero knowledge proofs to P_1

end for

for P_1 **do**

$\forall i$, check whether $POK(e_{x_{0i}})$ is correct else **ABORT**

$\forall i$, calculate $e_{x_{1i}} = E_{pk}(x_{1i})$, $e_{x_{0i} \cdot x_{1i}} = e_{x_{0i}} \times_h x_{1i}$

Calculate $e_{r_1} = E_{pk}(r_1)$ and $POK(e_{r_1})$

Calculate $E_{pk}(\sum_i^n (x_{0i} \cdot x_{1i}) + r_1) = e_{x_{00} \cdot x_{10}} +_h e_{x_{01} \cdot x_{11}} +_h \dots +_h e_{x_{0n} \cdot x_{1n}} +_h e_{r_1}$

Calculate $e_s = E_{pk}(\sum_i^n (x_{0i} \cdot x_{1i}) + r_1)$

$\forall i$, send $e_{x_{1i}}$, $e_{x_{0i} \cdot x_{1i}}$, $POK(e_{x_{1i}})$, $POMC(e_{x_{0i}}, e_{x_{1i}}, e_{x_{0i} \cdot x_{1i}})$, e_{r_1} , $POK(e_{r_1})$, and e_s to P_0

end for

for P_0 **do**

$\forall i$, check whether the $POK(e_{x_{1i}})$ is correct else **ABORT**

$\forall i$, check whether the $POMC(e_{x_{0i}}, e_{x_{1i}}, e_{x_{0i} \cdot x_{1i}})$ is correct else **ABORT**

Calculate $e_s = E_{pk}(\sum_i^n (x_{0i} \cdot x_{1i}) + r_1)$

end for

Jointly, call private decrypt function such that only P_0 learns the decryption of e_s

As before we prove the security of the protocol in malicious model.

Theorem 3.2. *Protocol 2 is secure in (private decrypt)-hybrid model assuming that the non-interactive zero-knowledge protocols used are secure in the malicious model.*

Proof. (Sketch) Again, for any adversary A operating in the hybrid model, we need to find an adversary S_A operating in the ideal model. In order to simplify our simulator, let us describe S_A for two different cases where either P_0 or P_1 is corrupted. First let us assume that A controls P_0 , we can define S_A as follows:

1. S_A gets the final result $\sum_i^n(x_{0i}.x_{1i}) + r_1$ as input
2. S_A uses the simulator S_{POK} for each x_{0i} input sequentially. ³ If any one of the proofs terminate then S_A terminates also.
3. S_A sets the state of A returned by the last run of S_{POK}
4. S_A simulates the honest P_1 by constructing the required correct zero knowledge proofs. Feeds A with those proofs.
5. S_A simulates the joint call to private decrypt function by returning $\sum_i^n(x_{0i}.x_{1i}) + r_1$ to A
6. S_A outputs whatever A outputs

Please note that the state of the A after the last execution of S_{POK} is identical in both worlds. Since zero knowledge proofs seen by the A that are given by S_A for simulating the correct behavior of P_1 are encrypted using a semantically secure encryption, the view of A in both worlds should be computationally indistinguishable. Therefore, the state of A before the oracle call to private decrypt function should be identical. Since S_A gives the correct result to A , the outputs in both worlds should be computationally indistinguishable.

Now let us assume that P_1 is corrupted, the construction of S_A is very similar to the case where P_0 is corrupted. Only the order of execution of simulators S_{POK} and S_{POCM} are different. We omit the further details here. \square

3.2.2 Secure and Efficient Dot Product for Malicious Model

In the previous secure dot protocol, P_1 is required to prove that each multiplication that it computes is correct. Actually, all we need to do is to check whether the final result is correct (i.e., whether r_o is calculated correctly). Also if both parties are malicious, we do not care whether the privacy of any party is protected or parties get correct results. Assuming at least one party will behave in a semi-honest fashion (other party can do any malicious act), we can give a more efficient protocol. Please note that the assumption that at least one party is semi-honest is given in the definitions of the malicious model. Therefore, using this fact do not reduce the security guarantees provided by the malicious model. Assuming at least one party is semi-honest, it can be assured that $r_0 = \sum_{i=0}^n(x_{0i}.x_{1i}) + r_1$ is evaluated correctly by at least one party. Please note that both P_0 and P_1 has enough information to calculate r_0 . If both P_0 and P_1 calculate the same r_0 value then calculations must be

³The zero-knowledge proofs used in our protocols can be run in parallel. Since simulating the sequential execution is conceptually simpler, we assumed that the zero-knowledge proofs are executed sequentially.

correct, because at least one of them is semi-honest and calculates correct r_0 . Therefore, if we securely make sure that both parties calculate the same value, then either of the local calculations could be decrypted to reveal r_0 to P_0 . In our second protocol, firstly each party sends the encrypted inputs along with the knowledge of plaintext proofs to each other, then each party P_i locally computes its respective $e_{r_0^i} = E_{pk}(r_0^i)$. After that point, they use a slightly modified version of the equality protocol to check whether $D_{pr}(e_{r_0^0}) = D_{pr}(e_{r_0^1})$ or not. If they tend to be equal, both parties jointly decrypt one of it to reveal r_{0i} to reveal r_0 to P_0 . Clearly, in this version, we do not need to send expensive zero-knowledge proof of correct multiplications for every multiplication. Since both sides can calculate the $e_{r_{0i}}$ values in parallel, the following protocol can offer huge savings where the vectors used for the dot product has many components.

We provide the details of the efficient secure dot product function in Protocol 3.

Protocol 3 Secure Dot Product in Malicious Model Using Threshold Decryption: Efficient dot product Specific Version

Require: Two parties P_0 and P_1 with the shares pr_0 and pr_1 of the private key pr and n bit vectors \bar{x}_i where \bar{x}_i belongs to P_i .

Ensure: Return $r_0 = \sum_i^n (x_{0i} \cdot x_{1i}) + r_1$ to P_0 and r_1 to P_1

for all P_i **do**

$\forall j$, set $e_{x_{ij}} = E_{pk}(x_{ij})$ and create $POK(e_{x_{ij}})$

Send encryptions and non-interactive zero knowledge proofs to P_{1-i}

end for

for P_0 **do**

$\forall i$, check whether the $POK(e_{x_{1i}})$ is correct else **ABORT**

Check whether $POK(e_{r_1})$ is correct else **ABORT**

Set $e_{r_0^0}$ to $E_{pk}(\sum_i^n (x_{0i} \cdot x_{1i}) + r_1) = e_{x_{00} \cdot x_{10}} +_h e_{x_{01} \cdot x_{11}} +_h \dots +_h e_{x_{0n} \cdot x_{1n}} +_h e_{r_1}$

end for

for P_1 **do**

$\forall i$, check whether the $POK(e_{x_{1i}})$ is correct else **ABORT**

Check whether $POK(e_{r_1})$ is correct else **ABORT**

Set $e_{r_0^1}$ to $E_{pk}(\sum_i^n (x_{0i} \cdot x_{1i}) + r_1) = e_{x_{00} \cdot x_{10}} +_h e_{x_{01} \cdot x_{11}} +_h \dots +_h e_{x_{0n} \cdot x_{1n}} +_h e_{r_1}$

end for

Jointly call decrypt equality protocol to check whether $D_{pr}(e_{r_0^1}) = D_{pr}(e_{r_0^0})$

for all P_i **do**

if Secure equality protocol returns true for $D_{pr}(e_{r_0^1}) = D_{pr}(e_{r_0^0})$ **then**

Jointly call private decrypt protocol such that P_0 learns the $D_{pr}(e_{r_0^1})$

end if

end for

As before, we prove the security of the protocol in malicious model.

Theorem 3.3. *The Protocol 3 is secure in (secure equality, private decrypt)-hybrid model.*

Proof. (Sketch) Without loss of generality, let us assume that real-world adversary A controls P_0 . Again for any real-world adversary A , we define a simulator S_A such that output of the adversary is computationally indistinguishable in both worlds. Again, let us assume that S_A is given the output of the protocol. Now we can define the S_A as follows:

1. S_A runs S_{POK} to verify the zero knowledge proofs and set A to the state returned by S_{POK} .
2. Use the properties of S_{POK} to learn the x_{0i} values⁴ with overwhelming probability. If S_{POK} does not return the x_{0i} values then abort.
3. Generate random x_{1i} and r_1 values such that the dot product result is consistent with the one given to S_A .
4. Feed A with the correct zero knowledge proofs for all $E_{pk}(x_{1i})$ and $E_{pk}(r_1)$.
5. Calculate the correct encrypted value $e_{r_0^1}$.
6. Get the input from the A for secure equality protocol.
7. Run the simulator for the secure equality protocol with the correct inputs.
8. Set the state of A to the state returned by the secure equality protocol simulator
9. If the both inputs are equal then return the correct result to A after the private decrypt call else abort.
10. Output whatever A outputs.

We need to show that the output of the A in the both worlds should be computationally indistinguishable. First note that, since A only sees the encrypted x_{1i} values, the state of A after Step 5 is identical, otherwise A could be used as distinguisher for homomorphic encryption. Since the inputs to the simulator for secure equality is computationally indistinguishable in both worlds, the state of A after the Step 8 is identical. Finally before the last step, the input given to A in the both worlds are the same and states are identical. Therefore, the output of A in both worlds should be computationally indistinguishable. Again the simulator for the case where A controls P_1 is similar. Therefore, we omit the discussion of that case. \square

3.3 Secure Set Operations

Secure set operations using homomorphic encryption have been proposed in the literature earlier in [15]. Recently Dawn et. al. also showed how to use homomorphic encryption and the zero knowledge proofs for constructing secure set operation protocols in the malicious

⁴This is possible due to the special properties of the zero-knowledge proofs we have used. Please see [8] for details.

model [13]. Given that P_0 has n items and P_1 has m items, the protocols described in [13] require $O(nm)$ homomorphic encryptions and zero knowledge proof of correct multiplications. Although those protocols are quite efficient if the total item domain size D of the items is much bigger than the number of items possessed by P_0 and P_1 (i.e., $D > nm$). For the cases where both m and n are bigger than $O(\sqrt{D})$ or where n or m equal to D , we suggest using simple secure set intersection and set union protocols that are secure in malicious model. Our algorithms require $O(D)$ homomorphic encryptions and zero knowledge proof of correct multiplications. The main idea is that we can represent the sets owned by each party as a bit vector size D and use secure multiplication property of the homomorphic encryption and associated zero knowledge proof to give secure set protocols in malicious model. Let us assume that x_{0i} is set to 1 if P_0 has item i in its private set else it is set to 0. (similarly for x_{1i} for P_1) Clearly for calculating set intersection, we need to calculate $x_{0i} \wedge x_{1i}$ for each i . Similarly, for set union, we need to calculate $x_{0i} \vee x_{1i}$ for all i . Note that \wedge operation is just a multiplication. For set union, we can rewrite $x_{0i} \vee x_{1i}$ as $\neg(\neg x_{0i} \wedge \neg x_{1i})$. This implies that if $(\neg x_{0i} \wedge \neg x_{1i})$ is equal to zero then item i is in the set union. Therefore, we can use the multiplication protocol for set union too. The details of the set intersection protocol is given in Protocol 4. The same protocol can be used for two-party set union using $\neg x_{0i}$ and $\neg x_{1i}$ as the input values and negating the output bits.

We can easily prove that the above protocol is secure in the malicious model. More specifically,

Theorem 3.4. *The Protocol 4 is secure in (threshold decryption)-hybrid model assuming that the non-interactive zero-knowledge protocols used are secure in the malicious model.*

Proof. (Sketch) Again, we need to prove that for any given adversary A operating in the hybrid model, we can simulate its actions in the ideal world. The S_A operating in the ideal world, is very similar to the one given in the proof sketch of the Theorem 3.2. First let us assume that A controls P_0 , we can define S_A as follows:

1. S_A gets the final results for all I_i values
2. S_A uses the simulator S_{POK} for each x_{0i} input sequentially. If any one of the proofs terminate then S_A terminates also.
3. S_A sets the state of A returned by the last run of S_{POK}
4. S_A simulates the honest P_1 by constructing the correct zero knowledge proofs that are required. Feeds A with those proofs.
5. S_A simulates the joint call to private decrypt function by returning the correct I_i values for all i .
6. S_A outputs whatever A outputs

Please note that the state of the A after the last execution of S_{POK} is identical in both worlds. Since zero knowledge proofs seen by the A that are given by S_A for simulating the

Protocol 4 Secure Set Intersection in Malicious Model Using Threshold Decryption

Require: Two parties P_0 and P_1 with the shares pr_0 and pr_1 of the private key pr and input bit vectors size D where x_{ij} is set to one if P_i has item j

Ensure: Return D bit vector I that represents the set intersection where I_i is set to one if item i is in the set intersection.

for P_0 **do**

$\forall i$, set $e_{x_{i0}} = E_{pk}(x_{0i})$ and create $POK(e_{x_{0i}})$ to prove that each x_{0i} is either zero or one
Send encryptions and non-interactive zero knowledge proofs to P_1

end for

for P_1 **do**

$\forall i$, check whether $POK(e_{x_{0i}})$ is correct else **ABORT**

$\forall i, x_{1i}$ calculate $e_{x_{1i}} = E_{pk}(x_{1i})$, $e_{x_{0i}.x_{1i}} = e_{x_{0i}} \times_h x_{1i}$

$\forall i$, send $e_{x_{1i}}, e_{x_{0i}.x_{1i}}$, $POK(e_{x_{1i}})$ (again proving x_{1i} is either zero or one), and $POMC(e_{x_{0i}}, e_{x_{1i}}, e_{x_{0i}.x_{1i}})$ to P_0

end for

for P_0 **do**

$\forall i$, check whether the $POK(e_{x_{1i}})$ is correct else **ABORT**

$\forall i$, check whether the $POMC(e_{x_{0i}}, e_{x_{1i}}, e_{x_{0i}.x_{1i}})$ is correct else **ABORT**

end for

$\forall i$, jointly call threshold decryption function to learn $D_{pr}(e_{x_{0i}.x_{1i}})$.

set I_i to $D_{pr}(e_{x_{0i}.x_{1i}})$.

correct behavior of P_1 are encrypted using a semantically secure encryption, the view of A in both worlds should be computationally indistinguishable. Therefore, the state of A before the oracle calls to private decrypt function should be identical. Since S_A gives the correct result to A , the outputs in both worlds should be computationally indistinguishable.

Since the case where A controls the P_1 is similar. We omit the simulator for that case. \square

3.4 Secure Comparison

Secure comparison is another important subprotocol that is commonly used in many different privacy-preserving data mining protocols. Since the most efficient protocols follow the generic circuit evaluation methods, generic conversion techniques could be used for creating secure comparison in the malicious model. Please refer to [11] for more details.

4 Performance Comparison

In this section, we analyze the efficiency loss of privacy-preserving data mining algorithms caused by malicious model. As stated before, the efficiency of a distributed data mining algorithm can be estimated in terms of the primitives it utilizes; i.e. number of secure dot products, secure comparisons, etc. Therefore, in the performance evaluation section we explore the efficiency trade-offs in these basic sub-protocols and use these results to estimate the possible overall slow down in privacy-preserving distributed data mining algorithms in malicious model.

4.1 Secure Dot Product

Efficiency of the secure dot product protocol is highly dependent on the size of the input dataset. On the other hand, it is not mainly effected by the length of the key that is used in the protocol [16]. Here, in order to evaluate the effect of using malicious mode on the secure dot product protocol we implement the model that is introduced in Section III-B and compare it with secure dot product protocol in semi-honest model in Figure 1. We also implement the extended version of this model which provides an efficient way of computing secure dot product in malicious model.

As it can be observed from Figure 1, there is a significant difference between running times of secure dot product protocol in semi-honest and malicious (straightforward) models. This does not lead to a conclusion that evaluating the secure dot product protocol in malicious model is totally unusable and inapplicable; in Figure 1 we also show that the overall running time in malicious model can be reduced to a half by slightly modifying the protocol in favor of the malicious model. It is possible that more sophisticated methods and modifications may lead to bigger increases in efficiency. Moreover, implementing the protocol by using generic circuit evaluation methods [11] is not feasible at all: The generated circuit should be duplicated as many as the size of the dataset, because the secure dot product protocol

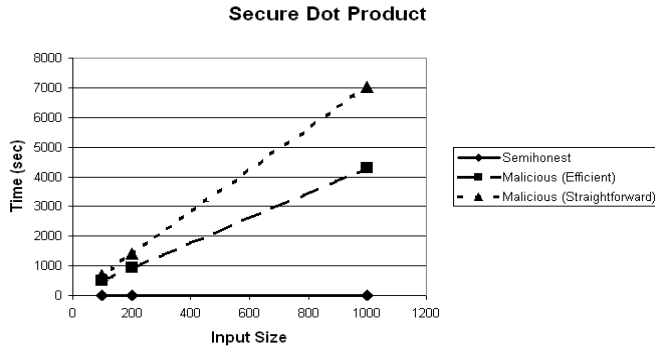


Figure 1: Performance evaluation results for the secure dot product protocol

involves looping. This requirement makes the circuit almost impossible to evaluate even with only 1000 elements.

4.2 Secure Comparison and Equality Check

Referring to famous Millionaires problem [17] secure comparison is an important and commonly used primitive in privacy-preserving data mining applications. In the basic secure comparison protocol two values are compared to output $<$, $>$ or $=$ without revealing the values. It is mainly used as a primitive in more complex algorithms such as sorting, finding median or mean among a group of elements. Meanwhile, secure equality check is a special case of the secure comparison protocol.

For both primitives, as the number of inputs is fixed (2) the bit-length of the inputs play more important role on the efficiency. Especially in generic circuit evaluation techniques, input length is a deterministic factor. In this section, we implement and then compare 3 different versions of the secure equality check protocol. The results are displayed in Figure 2; from which it can be observed that there is no significant difference between semi-honest and malicious models and our solution using the malicious model is superior to circuit evaluation.

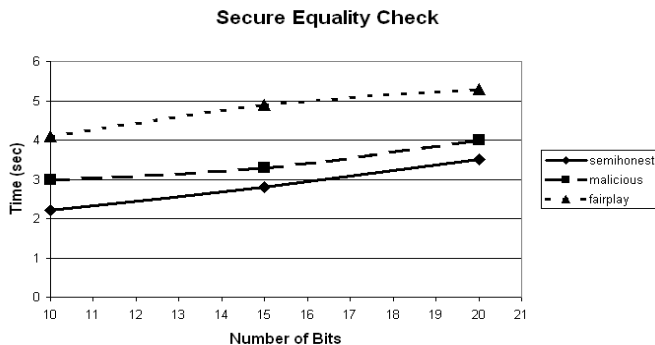


Figure 2: Performance evaluation results for the secure equality check protocol

4.3 Set Union and Intersection

As it was explained in Section III-C the set intersection protocol can easily be converted into and implemented as a secure dot product. In addition to this, in the set intersection protocol the inputs for secure dot product are specially bit vectors which allow parties to skip the multiplications of 0s in their shares (i.e. the elements in the domain that they do not have). This property provides more efficient results than the regular secure dot product especially when the bit vectors are sparse.

On the other hand, secure set union protocol is a combination of secure dot product and secure equality. Both set union and intersection protocols are dependent on the input size which is the size of the input domain in this case.

4.4 Privacy-Preserving K-means Clustering Algorithm using Arbitrarily Partitioned Data

The secure dot product and secure comparison protocols are very frequently used primitives in privacy-preserving k-means clustering algorithm that is defined in [7]. According to [7] secure dot products are mainly used for calculating cluster centers where the input size for the secure dot products is proportional to the number of attributes (l) in the partitioned data. Also the total number of secure dot products can be estimated in terms of number of clusters (k) which is a relatively small number (usually <20). Being independent of the size of the database (n) the overall cost of secure dot products in privacy-preserving k-means clustering algorithm can be estimated as $O(k \cdot l)$ [7]. In contrast to some other applications such as Association Rule Mining where the input size for secure dot products are a factor of the size of the data, in privacy-preserving k-means clustering with such small k and l secure dot products in malicious model is efficient enough to be usable.

5 Conclusions

Most of the privacy-preserving data mining algorithms in the literature were developed using the semi-honest model; which relies on the presumption that all participating parties would follow the protocol without producing any harm or threat for others. Although this assumption may be accurate for some cases, there are many real-life examples where more strict statements should be made. The malicious model introduces solid rules for security with the cost of loss of efficiency. However, the trade-off between semi-honest and malicious models is not clear in practice.

In this study, using the standard primitive algorithms in the semi-honest model (secure dot product, comparison, and set intersection) we first provide ways for developing them in malicious model with the help of threshold decryption and zero knowledge proofs. Then we showed that these algorithms can further be improved in terms of efficiency by specializing them in malicious model. As an example we provide an efficient algorithm for secure dot product in malicious model. In the end we evaluated the performance of the algorithms in

malicious model by comparing them with the ones in semi-honest model. We also included the performance analysis results of the circuit evaluation versions as a benchmark; thus showing the superiority of our methods. Finally, we discussed usability and applicability of the primitive algorithms in malicious model when they are used within larger applications such as privacy-preserving k-means clustering and association rule mining.

References

- [1] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” in *Advances in Cryptology – CRYPTO 2000*. Springer-Verlag, Aug. 20-24 2000, pp. 36–54. [Online]. Available: <http://link.springer.de/link/service/series/0558/bibs/1880/18800036.htm>
- [2] M. Kantarcioğlu and C. Clifton, “Privacy-preserving distributed mining of association rules on horizontally partitioned data,” *IEEE TKDE*, vol. 16, no. 9, pp. 1026–1037, Sept. 2004. [Online]. Available: <http://ieeexplore.ieee.org/iel5/69/29187/01316832.pdf?isnumber=29187&prod=JNL&arnumber=1316832&arnumber=1316832&arSt=+1026&ared=+1037&arAuthor=Kantarcio%2C+M.%3B+Clifton%2C+C.>
- [3] X. Lin, C. Clifton, and M. Zhu, “Privacy preserving clustering with distributed EM mixture modeling,” *Knowledge and Information Systems*, vol. 8, no. 1, pp. 68–81, July 2005. [Online]. Available: <http://dx.doi.org/10.1007/s10115-004-0148-7>
- [4] M. Kantarcioğlu and C. Clifton, “Privately computing a distributed k -nn classifier,” in *PKDD2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds., Pisa, Italy, Sept. 20-24 2004, pp. 279–290. [Online]. Available: <http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=3202&spage=279>
- [5] J. Vaidya and C. Clifton, “Privacy preserving association rule mining in vertically partitioned data,” in *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 23-26 2002, pp. 639–644. [Online]. Available: <http://doi.acm.org/10.1145/775047.775142>
- [6] W. Du and Z. Zhan, “Building decision tree classifier on private data,” in *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, C. Clifton and V. Estivill-Castro, Eds., vol. 14. Maebashi City, Japan: Australian Computer Society, Dec. 9 2002, pp. 1–8. [Online]. Available: <http://crpit.com/Vol14.html>
- [7] G. Jagannathan and R. N. Wright, “Privacy-preserving distributed k -means clustering over arbitrarily partitioned data,” in *Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, IL, Aug. 21-24 2005, pp. 593–599.

- [8] R. Cramer, I. Damgård, and J. B. Nielsen, “Multipart computation from threshold homomorphic encryption,” *Lecture Notes in Computer Science*, vol. 2045, pp. 280–??, 2001. [Online]. Available: citeseer.ist.psu.edu/cramer00multipart.html
- [9] O. Goldreich, *The Foundations of Cryptography*. Cambridge University Press, 2004, vol. 2, ch. General Cryptographic Protocols. [Online]. Available: <http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/prot.ps>
- [10] R. Canetti, “Security and composition of multipart cryptographic protocols,” *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, vol. 13, no. 1, pp. 143–202, 2000. [Online]. Available: citeseer.ist.psu.edu/canetti98security.html
- [11] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, “Fairplay - a secure two-party computation system,” in *Usenix Security - 2004*, 2004. [Online]. Available: <http://www.pinkas.net/PAPERS/MNPS.pdf>
- [12] Y. Lindell and B. Pinkas, “Privacy preserving data mining,” *Journal of Cryptology*, vol. 15, no. 3, pp. 177–206, 2002.
- [13] L. Kissner and D. Song, “Privacy-preserving set operations,” in *Advances in Cryptology - CRYPTO 2005*, 2005. [Online]. Available: citeseer.ist.psu.edu/739924.html
- [14] I. Damgard, M. Jurik, and J. Nielsen, “A generalization of paillier’s public-key system with applications to electronic voting,” 2003. [Online]. Available: citeseer.ist.psu.edu/damgard03generalization.html
- [15] M. J. Freedman, K. Nissim, and B. Pinkas, “Efficient private matching and set intersection,” in *Eurocrypt 2004*. Interlaken, Switzerland: International Association for Cryptologic Research (IACR), May 2-6 2004.
- [16] O. Kardes, R. S. Ryger, R. N. Wright, and J. Feigenbaum, “Implementing privacy-preserving bayesian-net discovery for vertically partitioned data,” in *ICDM Workshop on Privacy and Security Aspects of Data Mining*, 2005. [Online]. Available: <http://www.cs.stevens.edu/~rwright/Publications/psdm05.pdf>
- [17] I. Ioannidis and A. Grama, “An efficient protocol for yao’s millionaires’ problem.” [Online]. Available: citeseer.ist.psu.edu/634712.html