國立交通大學

資訊科學與工程研究所

碩士論文

計算兩集合交集與否的公平協定

Fairness of Secure Computation Protocols for Disjointness of Two Sets

研 究 生: 官振傑

指導教授: 曾文貴 教授

中華民國 九十九 年 六 月

E
S
A
1896

計算兩集合交集與否的公平協定

Fairness of Secure Computation Protocols for Disjointness of Two Sets

研 究 生: 官振傑　　Student: Albert Guan

指導教授: 曾文貴 教授　　Advisor: Wen-Guey Tzeng

國立交通大學

資訊科學與工程研究所

碩士論文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國 九十九 年 六 月

E
S
A
1896

Abstract

Given two finite nonempty sets A and B. Assume that at least one of the set contains more than one elements. We present fairness secure computation protocols for determining whether the given two sets have intersection or not.

Each party has a set as the input to the protocol. At the end of the protocol both parties know whether the two sets have intersection or not, but noting else. In particular, each party does not know the set the other party has. Furthermore, if the intersection of the two sets is not empty, then none of the parties know the elements in the intersection, nor do they know the cardinality of the intersection.

Our first protocol is complete fair with respect to both parties. At the end of the protocol, both parties knows the final result or none of them know it, even if there is a malicious party who may violate the protocol. Our second protocol is almost complete fair. The probability that "one party learns the result but the other does not" is negligible. This protocol needs less computational time, and it also needs less proofs of knowledge in the protocol.

Our protocols need only $O(mn)$ comparisons of the obfuscated elements, where m and n are the number of elements in A and B, respectively. The number of comparisons is independent of the size of the domain from which the elements of the sets A and B are selected.

It is known that the millionaires' problem can be reduced to the set intersection problem. Therefore, the millionaires' problem can be solved efficiently by our protocol even if the input domain is very large.

摘　　要

給定兩個有限集合 A 和 B. 假設沒有一個集合是空的, 而且至少有一個集合包含多於一個元素. 我們設計一個決定此二集合是否有交集的公平且安全的計算協定.

在此計算協定開始時, 每位參與者各自擁有一個集合當作此協定的輸入. 在此協定結束時, 每位參與者都知到這兩個集合是否有交集, 除此之外, 雙方都不知道其他的資訊. 例如: 他們都不知道別人的集合是甚麼. 假設此二集合的交集非空集合, 也沒有一方會知道這些交集的元素會是甚麼, 或是 A 和 B 的交集包含多少元素.

我們的第一個協定對雙方而言是完全公平的. 這意思是指, 即使有一方是惡意者, 他不忠實遵守協定來執行, 甚至可以提前結束此協定, 他也占不到便宜. 協定結束之後一定是雙方都知道答案或雙方都不知道答案. 我們的第二個協定對雙方而言幾乎是完全公平的. 第二個協定所需要的計算量比第一個少, 而且其安全性所需要的假設條件也比較少.

我們的協定先將兩集合的元素轉換成模糊的形式, 使得雙方都無知道照轉換後的元素與原來的元素對應關係. 我們的協定只需要 $O(mn)$ 的比較經過轉換之後集合的元素是否相等就可讓雙方知道答案, 其中 m 和 n 分別爲集合 A 和 B 集合元素的個數. 其所需的比較次數和 A 與 B 集合元素的定義範圍無關.

我們已知百萬富翁問題可以化約爲兩集合是否有交集的問題來解. 所以, 即使是定議域非常大的百萬富翁問題, 利用我們的協定來解百萬富翁問題也會非常有效率.

Contents

List of Figures

E S A
1896

List of Tables

E S A
1896

Chapter 1

Introduction

Secure computation is fundamental in modern cryptography. Many applications are based on secure two-party computation protocols. In this thesis, we present secure and fair protocols for determining whether the intersection of two sets is empty or not.

In cryptography, a *secure two-party computation protocol* is a protocol for computing a function $f(x, y)$ jointly by two parties. Let the two parties be Alice and Bob. Initially, Alice has an input x and Bob has an input y. After the execution of the protocol, both Alice and Bob knows the output $f(x, y)$, but nothing else. This is a security requirement of the protocol. In particular, Alice does not know the input y of Bob, and Bob does not know the input x of Alice, unless x or y can be deduced from the value of $f(x, y)$ and the information they already have. Furthermore, if the intersection of the two sets is not empty, then none of the parties know the elements in the intersection, nor do they know the cardinality of the intersection of the two sets.

In the design of the two-party computation protocols, two models are usually considered. The first model is the *semi-honest* model. In the semi-honest model, both parties perform computations according to the specification of the protocol.

However, they may be curious to learn more information from the information they already have and the information transmitted in the protocol. The second model considers the case in which one of the party may be *malicious*. The malicious party may deviate from the protocol in any way, or aborting the protocol. In this thesis, we design protocols under the malicious model.

One of the important property of a two-party computation protocol is *fairness*, especially in the malicious model. A secure two-party computation is *complete fairness* if one party learns the value of $f(x, y)$, then so does the other party.

In some cases, a somewhat relaxed condition may also be useful. A secure two-party computation is *almost complete fairness* if the probability of the event "one party learns the value of $f(x, y)$ and the other party does not" is negligible.

In this thesis, we propose two secure two-party computation protocols for deciding whether the intersection of two sets is empty or not, assuming that there may be a malicious party. The first protocol has the property of complete fairness, and the second one is almost complete fairness. The first one needs more computation and more proofs of knowledge, while the second one needs less. Specifically, the first one needs $O(mn \log p)$ computations, while the second one needs only $O(mn)$ computations.

Cleve has shown that complete fairness secure computation for *exclusive-or problem* is impossible [2]. Based on this results, we show that if each of the two sets A and B contains one element from $\{0, 1\}$, or any two-element set, then there are no protocols with complete fairness to determine the intersection of the two sets is empty or not. Therefore, one of the contributions of our work is that we completely classify the disjointness problem of two sets by their cardinality. If $m = n = 1$, then there are no complete fairness protocols. Otherwise, if $n > 1$ or

$m > 1$, then there exist such a protocol.

Our protocol first obfuscates the elements in each set so that no party knows the mapping of the original elements and the obfuscated elements. The two parties then compares the obfuscated elements in each set in a way described latter. Let the number of the elements of the two sets be m and n, respectively. Our protocol needs only $O(mn)$ comparison of the the obfuscated elements. The number of comparisons is independent of the size of the input domain. Hence, it is more efficient than currently known protocols, especially when the input domain is very large.

Gordon, Hazay, Katz and Lindell showed that the millionaires' problem has a complete fairness secure computation protocol [3]. However, the round complexity of their protocol is proportional to the size of the input domain. Thus, only problems with polynomial-size domain is feasible if their protocol is to be used. It is known that the millionaires' problem can be reduced to the set intersection problem [4]. Therefore, by using our protocols the millionaires' problem can be solved more efficiently.

The rest of the thesis is organized as follows. In Chapter 2, we modify the protocols for equality test of two integers proposed by Boudot, Schoenmakers, and Traoré [1]. The modified protocols will be used by our protocol for comparing the elements. In Chapter 3, we show that some special case of set disjointness problem is impossible. In Chapter 4 we present our first protocol, which is complete fairness. In Chapter 5, we present our second protocol, which is almost complete fairness. We also analyze the security and the performance of each protocol. In Chapter 6, we make conclusions and present future works.

E S A
1896

Chapter 2

Preliminaries

Our protocol carefully selects a pair of elements, one from each set, to be compared in each round. They proposed two versions of the protocol for equality test [1]. One with complete fairness and the other one does not. We modify the complete fairness version of the equality test protocol proposed by Boudot, Schoenmakers, and Traoré to compare whether these two elements are equal or not. The modified version of their protocols can be briefly described as follows.

2.1 Equality test without Fairness

2.1.1 Cryptographic Assumptions

Let q be a large prime, G_q be a group of order q. The cryptographic assumptions required in this version of Boudot et al.'s protocol are:

1. The Discrete Logarithm assumption (DL)

   On input G_q, g, y, it is infeasible to compute $\log_g y$.

2. The Diffie-Hellman assumption (DH)

   It is infeasible to compute g^{ab} given g, g^a, g^b for some random $a, b \in \mathbf{Z}_q$.

3. The Decision Diffie-Hellman assumption (DDH)

   It is infeasible to decide deterministically whether $c = ab$ given g, g^a, g^b, g^c for some random $a, b, c \in \mathbf{Z}_q$.

2.1.2 Proofs of knowledge

The proofs of knowledge required in this version of their protocols are:

1. Proof of the knowledge of a discrete logarithm

   The protocol allows Alice to prove to Bob that she knows an element $x \in \mathbf{Z}_q$ satisfying $y = g^x$, where y is Alice's public key. Alice randomly selects an integer $r \in \mathbf{Z}_q$, computes $W = g^r$, $c = h(W)$, and $D = r - xc \pmod{q}$. Then Alice sends the proof (c, D) to Bob. Bob is convinced if $c = h(g^D y^c)$.

2. Proof of the equality of two discrete logarithms

   The protocol allows Alice to prove to Bob that she knows an element $x \in \mathbf{Z}_q$ satisfying $y_1 = g_1^x$ and $y_2 = g_2^x$, where y_1 and y_2 are Alice's public keys. Alice randomly selects $r \in \mathbf{Z}_q$, computes $W_1 = g_1^r$, $W_2 = g_2^r$, $c = h(W_1, W_2)$, and $D = r - xc \pmod{q}$. Then Alice sends the proof (c, D) to Bob. Bob is convinced if $c = h(g_1^D y_1^c, g_2^D y_2^c)$.

2.1.3 The protocol

Boudot et al.'s secure two-party computation protocol for equality test without fairness can be summarize as follows.

1. Alice and Bob generate a group G_q of a large prime order q, by taking G_q as a subgroup of $\mathbf{Z}_p^*$ for a large prime p such that $q \mid (p-1)$.

2. They decide on generators g_0, g_1, g_2 of G_q, for which they do not know the value of $\log_{g_i} g_j$ for $i \neq j$, $0 \leq i, j < 3$.

3. Generate g_3.

   (a) Alice generates $g_a = g_1^{x_a}$ for a random $x_a \in \mathbf{Z}_q^*$, and sends it to Bob.

   (b) Bob generates $g_b = g_1^{x_b}$ for a random $x_b \in \mathbf{Z}_q^*$ and sends it to Alice.

   (c) They prove the knowledge of x_a and x_b to each other respectively, by using Schnorr's protocol [8]. They also check whether $g_a = 1$ and $g_b = 1$ or not. If any of g_a or g_b is 1, they abort the protocol.

   Let $g_3 = g_1^{x_a x_b} = g_a^{x_b} = g_b^{x_a}$, which can be computed independently by Alice and Bob.

4. Alice selects a random element $a \in \mathbf{Z}_q$ and computes

$$(P_a, Q_a) = (g_3^a, g_1^a g_2^x). \tag{2.1}$$

   Then she shows that there indeed exists an $a \in \mathbf{Z}_q$ for which she knows $x \in \mathbf{Z}_q$ satisfying 2.1.

5. Alice shows that she knows $a \in \mathbf{Z}_q$ by computing $a_i \in \mathbf{Z}_q$, $i = 0, \ldots, k-1$ subject to the condition that $a = \sum_{i=0}^{k-1} a_i 2^i \pmod{q}$ and, and setting $B_i = g_3^{a_i}$, $i = 0, \ldots, k-1$.

   Since Alice can only know one a satisfying $P_a = g_3^a$, it follows that (P_a, Q_a) is correctly computed by Alice.

6. Alice sends (P_a, Q_a) and the proofs over to Bob.

7. Bob verifies the proofs and also checks that $P_a = \prod_{i=0}^{k-1} B_i^{2^i}$.

8. By symmetry, Bob does the same as Alice, computing P_b, Q_b satisfying

$$(P_b, Q_b) = (g_3^b, g_1^b g_2^y). \tag{2.2}$$

where $b \in \mathbf{Z}_q$ are randomly chosen.

9. Alice and Bob both compute $(P_a/P_b, Q_a/Q_b)$, which will be of the form:

$$(P_a/P_b, Q_a/Q_b) = (g_3^{a-b}, g_1^{a-b} g_2^{x-y}). \tag{2.3}$$

10. Then Alice produces

$$R_a = (Q_a/Q_b)^{x_a}$$

and a proof that $\log_{g_1} g_a = \log_{Q_a/Q_b} R_a$ to show that R_a is correctly formed. Alice then sends R_a, as well as the proof, to Bob.

11. Similarly, Bob produces

$$R_b = (Q_a/Q_b)^{x_b}$$

and a corresponding proof, and send them to Alice.

12. Since both Alice and Bob know equation 2.3 and the definition of g_3, they can compute R_{ab}.

$$R_{ab} = R_a^{x_b} = R_b^{x_a} = (Q_a/Q_b)^{x_a x_b} = g_3^{a-b} g_2^{(x-y)x_a x_b}. \tag{2.4}$$

At the end of the protocol, Alice and Bob test if $P_a/P_b = R_{ab}$. If it is equal, then $x = y$, otherwise, $x \neq y$.

2.2 Equality test with Complete Fairness

2.2.1 Cryptographic Assumptions

The cryptographic assumptions required by the complete fairness version of Boudot, Schoenmakers, and Traoré's protocol are:

1. The Discrete Logarithm assumption (DL)

   On input G_q, g, y, it is infeasible to compute $\log_g y$.

2. The Diffie-Hellman assumption (DH)

   It is infeasible to compute g^{ab} given g, g^a, g^b for some random $a, b \in \mathbf{Z}_q$.

3. The Decisional Diffie-Hellman assumption (DDH)

   It is infeasible to decide whether $c = ab$ given g, g^a, g^b, g^c for some random $a, b, c \in \mathbf{Z}_q$.

2.2.2 Proofs of knowledge

The proofs of knowledge's required by the complete fairness version of their protocol are:

1. Proof of the knowledge of a discrete logarithm

   There exists protocol allows Alice to prove to Bob that she knows an element $x \in \mathbf{Z}_q$ satisfying $y = g^x$, where y is Alice's public key. Alice randomly selects an integer $r \in \mathbf{Z}_q$, computes $W = g^r$, $c = h(W)$, and $D = r - xc \pmod{q}$. Then Alice sends the proof (c, D) to Bob. Bob is convinced if $c = h(g^D y^c)$.

2. Proof of the knowledge of discrete coordinate

   There exists protocol allows Alice to prove to Bob that she knows $x_1, x_2 \in \mathbf{Z}_q$

satisfying $y = g_1^{x_1} g_2^{x_2}$, where y is Alice's public key. Alice randomly selects two integers $r_1,\ r_2 \in \mathbf{Z}_q$, computes $W = g_1^{r_1} g_2^{r_2}$, $c = h(W)$, $D_1 = r_1 - x_1 c \pmod{q}$, and $D_2 = r_2 - x_2 c \pmod{q}$. Then Alice sends the proof (c, D_1, D_2) to Bob. Bob is convinced if $c = h(g_1^{D_1} g_2^{D_2} y^c)$.

3. Proof of the equality of two discrete logarithms

   There exists protocol allows Alice to prove to Bob that she knows an element $x \in \mathbf{Z}_q$ satisfying $y_1 = g_1^x$ and $y_2 = g_2^x$, where y_1 and y_2 are Alice's public keys. Alice randomly selects $r \in \mathbf{Z}_q$, computes $W_1 = g_1^r$, $W_2 = g_2^r$, $c = h(W_1, W_2)$, and $D = r - xc \pmod{q}$. Then Alice sends the proof (c, D) to Bob. Bob is convinced if $c = h(g_1^D y_1^c, g_2^D y_2^c)$.

4. Proof of the equality of two discrete coordinates

   There exists protocol allows Alice to prove to Bob that she knows x_1, x_{21}, x_{22} satisfying $y_1 = g_1^{x_1} g_2^{x_{21}}$ and $y_2 = g_1^{x_1} g_2^{x_{22}}$, where y_1 and y_2 are Alice's public keys. Alice randomly selects $r_1, r_{21}, r_{22} \in \mathbf{Z}_q$, computes $W_1 = g_1^{r_1} g_2^{r_{21}}$, $W_2 = g_1^{r_1} g_2^{r_{22}}$, $c = h(W_1, W_2)$, $D_1 = r_1 - x_1 c \pmod{q}$, $D_{21} = r_{21} - x_{21} c \pmod{q}$, and $D_{22} = r_{22} - x_{22} c \pmod{q}$. Alice sends the proof (c, D_1, D_{21}, D_{22}) to Bob. Bob is convinced if $c = h(g_1^{D_1} g_2^{D_{21}} y_1^c, g_1^{D_1} g_2^{D_{22}} y_2^c)$.

5. Proof that a coordinate is equal to 0 or to 1

   There exists protocol allows Alice to prove to Bob that she knows x_1, x_2 with $x_1 \in \mathbf{Z}_q$ and $x_2 \in \{0, 1\}$ satisfying $y = g_1^{x_1} g_2^{x_2}$, where y is Alice's public key. Suppose $x_2 = v$ with $v = 0$ or $v = 1$. Alice randomly selects $r, c_{1-v}, D_{1-v} \in \mathbf{Z}_q$, computes $W_v = g_1^r$, $W_{1-v} = g_1^{D_{1-v}} (y/g_2^{1-v})^{c_{1-v}}$, $c = h(W_0, W_1)$, $c_v = c - c_{1-v} \pmod{q}$, and $D_v = r - x_1 c_v \pmod{q}$. Alice sends the proof (c_0, c_1, D_0, D_1) to Bob. Bob is convinced if $c_0 + c_1 = h(g_1^{D_0} y^{c_0}, g^{D_1} (y/g_2)^{c_1})$

(mod q).

2.2.3 The protocol

Boudot et al.'s fair protocol for equality test can be summarize as follows:

1. Alice and Bob generate a group G_q of a large prime order q, by taking G_q as a subgroup of $\mathbf{Z}_p^*$ for a large prime p such that $q \mid (p-1)$.

2. They decide on generators g_0, g_1, g_2 of G_q, for which they do not know the value of $\log_{g_i} g_j$ for $i \neq j$, $0 \leq i, j < 3$.

3. Generate g_3:

   (a) Alice generates $g_a = g_1^{x_a}$ for a random $x_a \in \mathbf{Z}_q^*$.

   (b) Bob generates $g_b = g_1^{x_b}$ for a random $x_b \in \mathbf{Z}_q^*$.

   (c) They prove the knowledge of x_a and x_b to each other, using Schnorr's protocol.

   (d) They also check that $g_a \neq 1$ and $g_b \neq 1$.

   Let $g_3 = g_1^{x_a x_b} = g_a^{x_b} = g_b^{x_a}$, which can be computed independently by Alice and Bob.

4. Alice selects a random element $a \in \mathbf{Z}_q$ and a random number e, $0 \leq e < 2^k$, and computes

$$(P_a, Q_a) = (g_3^a g_0^e, g_1^a g_2^x). \tag{2.5}$$

5. Alice shows that she knows $a, e \in \mathbf{Z}_q$ with $0 \leq e < 2^k$ by computing $a_i \in \mathbf{Z}_q$ and $e_i \in \{0, 1\}$, $i = 0, \ldots, k-1$ subject to the condition that $a = \sum_{i=0}^{k-1} a_i 2^i$ (mod q) and $e = \sum_{i=0}^{k-1} a_i 2^i$, and setting $B_i = g_3^{a_i} g_0^{e_i}$, $i = 0, \ldots, k-1$.

6. Alice proves that each e_i is in $\{0, 1\}$. Since Alice can only know one pair a, e satisfying $P_a = g_3^a g_0^e$, it follows that (P_a, Q_a) is correctly computed by Alice. Alice sends (P_a, Q_a) and the proofs over to Bob.

7. Bob verifies the proofs and also checks that $P_a = \prod_{i=0}^{k-1} B_i^{2^i}$.

8. Alice sends g_3^a to Bob.

9. By symmetry, Bob does the same as Alice, computing P_b, Q_b satisfying

$$(P_b, Q_b) = (g_3^b g_0^f, g_1^b g_2^y). \tag{2.6}$$

where $b \in \mathbf{Z}_q$ and $f, 0 \leq f < 2^k$ are randomly chosen.

10. Alice and Bob both compute $(P_a/P_b, Q_a/Q_b)$, which will be of the form:

$$(P_a/P_b, Q_a/Q_b) = (g_3^{a-b} g_0^{e-f}, g_1^{a-b} g_2^{x-y}). \tag{2.7}$$

11. Then Alice produces

$$R_a = (Q_a/Q_b)^{x_a}$$

and a proof that $\log_{g_1} g_a = \log_{Q_a/Q_b} R_a$ to show that R_a is correctly formed.

12. Similarly, Bob produces

$$R_b = (Q_a/Q_b)^{x_b}$$

and a corresponding proof. Now Alice and Bob both know on account of equation 2.7 and the definition of g_3 that

$$R_{ab} = R_a^{x_b} = R_b^{x_a} = (Q_a/Q_b)^{x_a x_b} = g_3^{a-b} g_2^{(x-y)x_a x_b}. \tag{2.8}$$

13. Finally, Alice and Bob fairly disclose the values of e and f. Once these values are released, both Alice and Bob may determine whether $x = y$ by testing whether

$$P_a/P_b = R_{ab} g_0^{e-f}. \tag{2.9}$$

By equation 2.7 and equation 2.8 this equality holds if and only if $x = y$.

To disclose e and f without revealing the values of a and b, Alice and Bob execute the following step for $i = k-1, \ldots, 1$.

1. They send each other the values of a_i, e_i and b_i, f_i, respectively.
2. Bob checks that $B_i = g_3^{a_i} g_0^{e_i}$ and Alice does a similar check for b_i, f_i.
3. At the last step, they release only e_0 and f_0, respectively.
4. Finally, Alice proves that she knows $\log_{g_3} B_0/g_0^{e_0}$ and Bob gives a similar proof for f_0. This convinces the other party that the bits e_0 and f_0 are correct.

The main modification is in the step 8 and step 9 of the fairness version. Alice needs to send g_3^a to Bob, and Bob also needs to send g_3^b to Alice. These modifications allow Alice and Bob to compute the final results deterministically when one party aborts the protocol. Therefore, it guarantees the fairness of the protocol for both parties. For example, if Bob deliberately aborts the protocol at $i = l$, he will only be at most one bit ahead of Alice to test the equality of the two elements. At the time Bob aborts the protocol, Alice has $A_i = g_3^{b_i} g_0^{f_i}$ and g_3^b. By using g_3^b she can search for the combinations of the missing bits $f_0, f_1, \ldots, f_l$. Thus, she needs no more than twice as much time compared to Bob to compute the same result.

E S A
1896

Chapter 3

Impossibility of Complete Fairness for Single Element Sets

Before we present our secure and fair protocols for determining the disjointness of two sets, we show that some special cases of the set disjointness problem cannot be solved with complete fairness.

In 1986, Cleve showed that complete fairness secure two-party computation of the *exclusive-or problem* is impossible [2]. In any two-party computation of the exclusive-or, a malicious party can always bias the output of the other party. Our proof of the impossibility for this special case is based on Cleve's result. This implies that the general version of the disjointness of two sets cannot be computed with complete fairness.

Theorem 1 *The general version of the disjointness of two sets cannot be computed with complete fairness.*

Proof It is sufficient to show that some special case cannot be computed with complete fairness. Consider two sets A and B which are subsets of $\{0, 1\}$. Assume that each set contains only one elements, then the value of $f(A, B)$ can be illustrated in Table 3.1.

Table 3.1: The value of $f(A, B)$ when A and B contains exactly one element from $\{0, 1\}$.

	0	1
0	1	0
1	0	1

It is clear that this is a negation of the exclusive-or problem. If the disjointness of the sets A and B can be computed securely with complete fairness then the exclusive-or problem can also be computed securely with complete fairness. This is a contradiction to Cleve's result. ∎

Therefore, we requires that at least one of the set A or B contains at least two elements in our protocols.

We note that when the universal set U from which the two sets A and B are drawn are very large and each set contains only one element from U, then the set disjointness problem is equivalent to the equality test problem. Thus, the problem has complete fairness secure two party computation when U is large, even if each set A or B contains only one element from U.

Chapter 4

Protocol with Complete Fairness

Assume that Alice has a set

$$A = \{a_1, a_2, \ldots, a_m\},$$

and Bob has a set

$$B = \{b_1, b_2, \ldots, b_n\}.$$

They want to know whether $A \cap B$ is empty or not. We assume that both sets are nonempty and the set B contains more than 1 element, that is, $n > 1$, $m \geq 1$. If B contains only 1 element, we switch the roles of A and B in the following protocol.

Let λ be the security parameter. Assume that both the sets A and B are finite subset of integers whose value can be represented by λ bits.

For any two sets A and B, define a function f:

$$f(A, B) = \begin{cases} 0 & \text{if } A \cap B \text{ is empty,} \\ 1 & \text{if } A \cap B \text{ is not empty.} \end{cases}$$

The two-party protocol for computing $f(A, B)$ consists of two phases.

In the first phase, Alice and Bob exchange A and B, but in the obfuscated form. It is required that the same elements are mapped into the same integer to allow correct comparison in the second phase. This can be done as follows. Each element

$a \in A$ is obfuscated by taking $h(a)$ to a random power $\alpha\beta$, modulo a large prime p, where $h(x)$ is a cryptographic hash function known to both Alice and Bob. The elements of B can be obfuscated in a similar way. The random powers α and β are chosen randomly by Alice and Bob, respectively.

In the second phase, Alice and Bob compare the elements, one from each set, to determine whether they are equal or not. For the complete fairness version, the way the two elements are chosen is simple and straight forward. This is due to the complete fairness version of equality test of the elements is used. In the almost complete fairness version, a secure but may not be fair version of equality test is used to compare the elements. It requires less computation and less proofs of knowledge in the protocol. However, elements to be compared must be carefully chosen so that almost complete fairness can be achieved.

In this chapter, we describe our first protocol for two parties to determine whether the intersection of two sets is empty or not. The protocol is secure and complete fairness.

4.1 Description of the Protocol

First, Alice and Bob choose a large prime p, $p > 2^\lambda$, and the *discrete logarithm problem* over $\mathbf{Z}_p^*$ is intractable. Since each element in A and B can be represented by at most λ bits, p is larger than any element in A and B. Finally, we let h denote a cryptography hash function.

Then Alice and Bob run the following protocol.

1. Alice randomly chooses a positive integer $\alpha \in \mathbf{Z}_{p-1}$, computes the set

$$A_1 = \{h(a)^\alpha \bmod p \,|\, \forall a \in A\},$$

and sends the elements $a_1^{(1)}, a_2^{(1)}, \ldots, a_m^{(1)}$ of A_1, in random order to Bob.

2. Bob randomly chooses a positive integer $\beta \in \mathbf{Z}_{p-1}$, computes the set

$$B_1 = \{h(b)^\beta \bmod p \,|\, \forall b \in B\},$$

and sends the elements $b_1^{(1)}, b_2^{(1)}, \ldots, b_n^{(1)}$ of B_1, in random order to Alice.

3. After receiving $a_1^{(1)}, a_2^{(1)}, \ldots, a_m^{(1)}$ from Alice, Bob computes the set

$$A_2 = \{a^\beta \bmod p \,|\, \forall a \in A_1\},$$

and stores the elements $a_1^{(2)}, a_2^{(2)}, \ldots, a_m^{(2)}$ of A_2, in random order.

4. After receiving $b_1^{(1)}, b_2^{(1)}, \ldots, b_n^{(1)}$ from Bob, Alice computes the set

$$B_2 = \{b^\alpha \bmod p \,|\, \forall b \in B_1\},$$

and stores the elements $b_1^{(2)}, b_2^{(2)}, \ldots, b_n^{(2)}$ of B_2, in random order.

In the above protocol, each element is first hashed and then obfuscated by raising a random power $\alpha\beta$ modulo p, where α is chosen by Alice and β is chosen by Bob. Since the values of α and β are chosen at random, each element is mapped to a random element in the subgroup generated by $h(a_i)$ or $h(b_j)$ in $\mathbf{Z}_p^*$. Therefore, neither Alice or Bob can predict the final transformed value of each element.

We also need a random permutation so that the mapping from the original elements and the obfuscated elements cannot be known by Alice and Bob. This is done by sending the element in a random order and storing the resulting elements in a random order.

At the end of the first phase, Alice has the set B_2; while Bob has the set A_2. Note that $B_2 \cap A_2 \neq \emptyset$ if and only if $B \cap A \neq \emptyset$.

In the second phase, each element b in B_2 is checked for whether it is in A_2. Alice and Bob run a modified Boudot, Schoenmakers, and Traoré's protocol to determine whether b is in A_2 or not. Bob gets the comparison result one bit ahead of Alice. However, Alice can validate the result if she gets all the bits from Bob. Otherwise, she can compute the rest of the bits without the help of Bob.

The second phase of our first protocol can be described as follows. Alice and Bob compare the elements by running the protocol given in Figure 4.1.

for $i = 1, 2, \ldots, n$ **do**
  for $j = 1, 2, \ldots, m$ **do**
    Alice and Bob compare the equality of $b_i^{(2)}$ and $a_j^{(2)}$;
    if $b_i^{(2)} = a_j^{(2)}$ **then**
      Alice and Bob output 1;
      stop;
    end if
  end for
end for
Alice and Bob output 0;
stop;

Figure 4.1: Phase 2 of Complete Fairness Secure Computation of Set Disjointness

We must specify what needs to be done if one party aborts the protocol. We assume that both party can learn the value of $f(A, B)$ only through the comparison of the elements. If any party aborts the protocol in phase 1, then both parties output nothing. They do not know any information about the $f(A, B)$ yet. On the other hand, if any party aborts the protocol in phase 2, we assume that this party knows that $f(A, B) = 1$. If this is the case, then both party outputs 1.

The assumption of the party knows $f(A, B) = 1$ is the only case that he/she

aborts the protocol is reasonable, since without knowing $f(A, B) = 1$ and aborts the protocol would have no advantage.

4.2 Security Analysis of the Protocol

A two-party computation is secure and complete fairness if the view of the adversary in the real protocol is computationally indistinguishable from the view in the ideal model of computation. The view consists of the outputs of both parties.

This is formalized by first considering an ideal model of computing the same function f. In the ideal model, there is a trusted third party $\mathcal{TP}$, which is trusted by both party and is incorruptible. Two parties send their inputs x and y to the trusted third party in a secure way. The trusted third party computes the function f on their inputs x and y. Finally, the trusted party sends to each party the value of the functions f on input x and y.

A protocol π is said to securely compute f with complete fairness if for every non-uniform probabilistic polynomial adversary $\mathcal{A}$ in a real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that the view of the adversary in the real execution of the protocol is computationally indistinguishable from the view in the ideal implementation.

$$\{\mathbf{IDEAL}_{f,\mathcal{S}(z)}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*} \stackrel{c}{\equiv} \{\mathbf{REAL}_{\pi,\mathcal{A}(z)}(x,y)\}_{(x,y)\in X\times Y, z\in\{0,1\}^*}$$

In set disjointness problem, phase 1 is the same as phase 1 of our protocol. Phase 2 of the ideal model can be described in Figure 4.2.

It is reasonable to assume that the ideal model computes the set disjointness problem securely with complete fairness.

for $i = 1, 2, \ldots, n$ **do**
  for $j = 1, 2, \ldots, m$ **do**
    Alice securely sends $b_i^{(2)}$ to $\mathcal{TP}$;
    Bob securely sends $a_j^{(2)}$ to $\mathcal{TP}$;
    $\mathcal{TP}$ computes $c = \begin{cases} 1 & \text{if } b_i^{(2)} = a_j^{(2)} \\ 0 & \text{otherwise.} \end{cases}$, and sends c to Alice and Bob;
    if $c = 1$ **then**
      Alice and Bob output 1;
      stop;
    end if
  end for
end for
Alice and Bob output 0;
stop;

Figure 4.2: Phase 2 of the Ideal Model for Complete Fairness Secure Computation of Set Disjointness

As in the real world model, we require that one party output 1 whenever the other party aborts the protocol in any iteration.

Theorem 2 *Our first protocol securely computes the set disjointness problem with complete fairness.*

Proof Let $\mathcal{A}$ be an adversary in our protocol who can learn the value of $f(A, B)$ and, at the same time, can prevent the other party from learning it, with non-negligible probability. We show that our protocol computes the set disjointness problem with complete fairness by showing that if there is an adversary $\mathcal{A}$ in our protocol, then there is an adversary $\mathcal{S}$ corrupting the same party in the ideal model. In the ideal model, the adversary $\mathcal{S}$ can also learn the value of $f(A, B)$ and, at the same time, can prevent the other party from learning it, with non-negligible probability.

First, let Alice be the adversary $\mathcal{A}$ in our protocol. The adversary $\mathcal{A}$ is also called the real world adversary. We construct an ideal world adversary $\mathcal{S}$ given black-box access to $\mathcal{A}$. A sketch diagram of the construction is given in Figure 4.3.

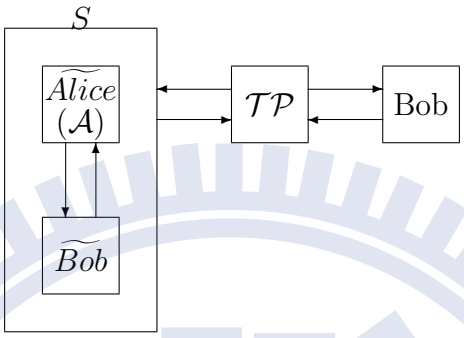


Figure 4.3: Construct an adversary S from adversary A corrupting Alice

Note that, in Figure 4.3, we use "$\widetilde{Bob}$" and "Bob" to distinguish the two different parties in different models. "$\widetilde{Bob}$" is in the real world model, and "Bob" is in the ideal Model.

The detailed construction of $\mathcal{S}$ from $\mathcal{A}$ are given as follows.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input the set A.

2. If $\mathcal{A}$ aborts in Phase 1, then $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs, and halts. Otherwise, $\mathcal{S}$ proceeds the phase 2 as below.

3. To simulate each iteration $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$,

   (a) If $\mathcal{A}$ aborts, then $\mathcal{S}$ sends $b_j^{(2)}$ to $\mathcal{TP}$, outputs whatever $\mathcal{A}$ outputs, and halts.

   (b) If $\mathcal{A}$ does not abort and uses x_i to compare $\widetilde{Bob}$'s element in the protocol. $\mathcal{S}$ sends x_i to $\mathcal{TP}$ and obtains c_i from $\mathcal{TP}$.

(c) If $c_i = 1$, then $\mathcal{S}$ uses x_i to compare $\widetilde{Bob}$'s element. Otherwise, $c = 0$, $\mathcal{S}$ chooses a random integer $x'_j \neq x_i$ and uses x'_j to compare $\widetilde{Bob}$'s element.

We analyze the adversary $\mathcal{S}$ described above. Let A denote the input of Alice. The view of $\mathcal{A}$ in an execution with $\mathcal{S}$ is identical to its view in a real world execution with $\widetilde{Bob}$. The only difference is that the elements sent by $\mathcal{S}$ is a random integer x'_j, instead of $b_j^{(2)}$. This does not affect the view of $\mathcal{A}$, since the equality test used in the protocol is secure, and the comparison result is the same for x'_j or $a_j^{(2)}$ is used. Hence, what is left to proof is that the joint distribution of $\mathcal{A}$'s view and Alice's output is identical in the real world and the ideal world. We show this by separately considering different cases:

1. $\mathcal{S}$ sends nothing to the trusted third party because $\mathcal{A}$ aborted the protocol in Phase 1. There will be no output for both party in the real world or in the ideal world models.

2. $\mathcal{A}$ aborts the protocol for some i, $1 \leq i \leq n$. In the real world, Bob would assume that Alice knows the intersection is not empty and then output 1. In the ideal world, Bob would also output 1, because $\mathcal{S}$ sends $b_j^{(2)}$ to $\mathcal{TP}$ at this round, or he has already know the value of $f(A, B)$ in the previous iteration.

3. If $\mathcal{A}$ does not abort the protocol, and the protocol ended normally, then both party out puts $f(A, B)$, in the real world model and in the ideal world model.

Based on the above argument, we conclude that the joint distribution of $\mathcal{A}$'s view and Alice's output is identical in the real world and the ideal world.

Now, let Bob be a real world adversary $\mathcal{A}$. We construct an adversary $\mathcal{S}$ given black-box access to $\mathcal{A}$. A sketch diagram of the construction is given in Figure 4.4.

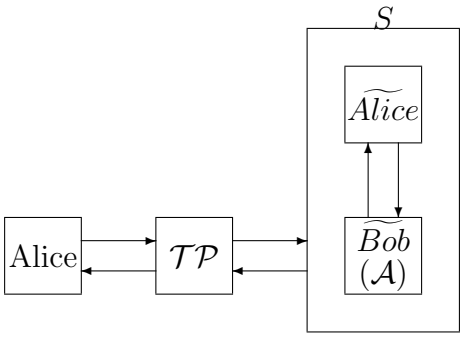


Figure 4.4: Construct an adversary S from adversary A corrupting Bob

Note that, in Figure 4.4, we use "$\widetilde{Alice}$" and "Alice" to distinguish the two different parties in different models. "$\widetilde{Alice}$" is in the real world model, and "Alice" is in the ideal Model.

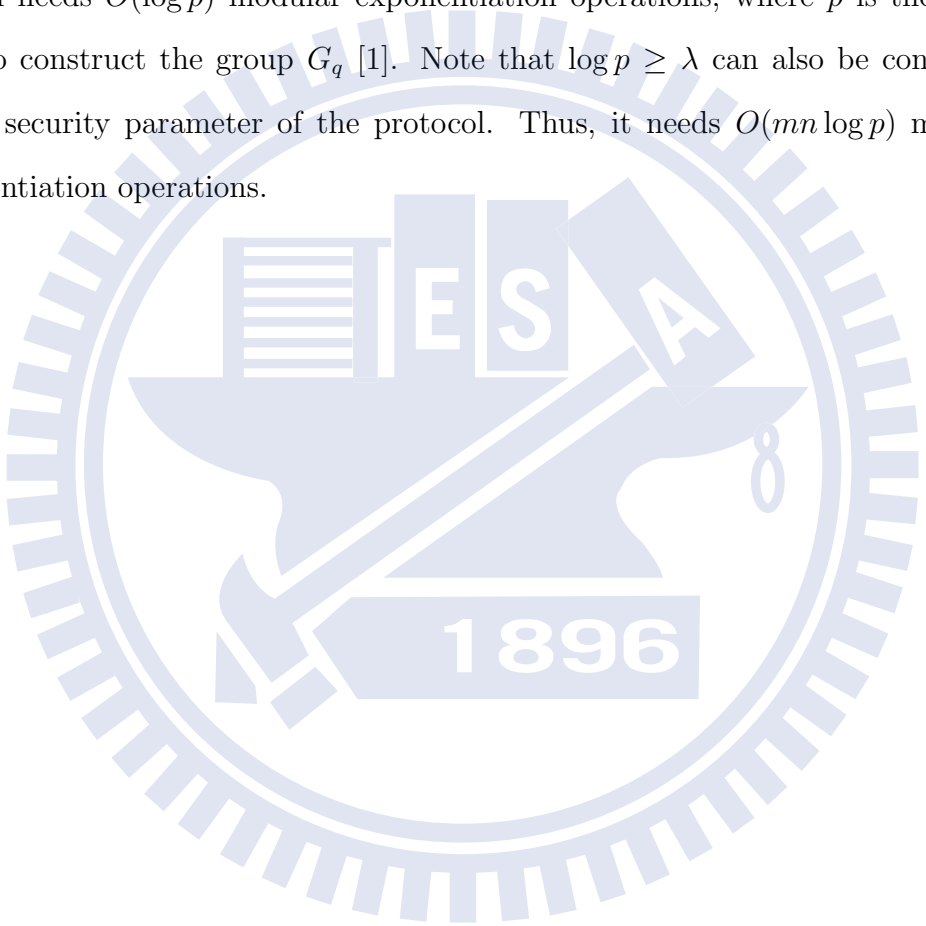
The details of the construction are given as follows.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input the set B.
2. If $\mathcal{A}$ sends abort in Phase 1, then $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs, and halts. Otherwise, $\mathcal{S}$ proceeds phase 2 as below.
3. To simulate each iteration $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.
   (a) If $\mathcal{A}$ aborts, then $\mathcal{S}$ sends $a_j^{(2)}$ to $\mathcal{TP}$, output whatever $\mathcal{A}$ outputs, and halts.
   (b) If $\mathcal{A}$ does not aborts and sends y_j to $\widetilde{Alice}$, then $\mathcal{S}$ sends y_j to $\mathcal{TP}$, obtains c_{ij} from $\mathcal{TP}$.
   (c) If $c_{ij} = 1$ then $\mathcal{S}$ sends y_j to $\widetilde{Alice}$. Otherwise $\mathcal{S}$ chooses a random integer $y'_j \neq y_j$ and sends y'_j to $\mathcal{A}$.

A proof for the case when Alice is real world adversary is similar to the previous case, and it is omitted. ∎

4.3 Performance Analysis of the Protocol

Recall that m and n are the cardinality of the sets A and B, respectively. The protocol needs at most mn comparison of the elements. Since the Boudot, Schoenmakers, and Traoré's protocol is used in the comparison of the element, each comparison needs $O(\log p)$ modular exponentiation operations, where p is the prime used to construct the group G_q [1]. Note that $\log p \geq \lambda$ can also be considered as the security parameter of the protocol. Thus, it needs $O(mn \log p)$ modular exponentiation operations.



Chapter 5

Protocol with Almost Complete Fairness

In this section, we describe our second protocol for two parties to determine whether the intersection of two sets is empty or not. The protocol is secure and almost complete fairness.

5.1 Description of the Protocol

The protocol for almost complete fairness also consists of two phases. The first phase of the protocol involves the obfuscation of the elements in each set, and it is the same as the first phase of the previous protocol with complete fairness.

In the second phase, Alice has the set B_2 and Bob has the set A_2, and each element b in B_2 is checked for whether it is in A_2 or not. Alice and Bob run a secure two-party protocol to determine whether b is in A_2 or not.

In order to achieve almost complete fairness, the way of selecting elements in B_2 to check its membership in A_2 should be carefully designed. We now describe how Alice chooses the element in B_2 to be compared in each round.

An element could be checked multiple number of times at different rounds. The

selection order of elements in B_2 and the number of rounds of checking membership are determined by the following random process.

Alice randomly selects an integer w as the number of of rounds, $n < w \leq 2n$, and sets a table $T[1 \ldots w]$ of w entries. Each entry $T[i]$ stores the element in B_2 to be compared in the i-th round. She randomly assigns the n elements of B_2 to any subset of n entries of the table T. Since $w > n$, not all entries in T are assigned. Let z be the maximum index of the entry which was assigned in the random assignment of the table T, and $b = T[z]$. The unassigned entries of the table T are filled in by the following rules:

1. $z = w = 2n$. For each unassigned entry of T, Alice assigns b to the entry with probability $\frac{1}{2}$ and each $b' \in B_2 \backslash \{b\}$ to the entry with probability $\frac{1}{2(n-1)}$.
2. otherwise $w \neq 2n$ or $z \neq w$. For each unassigned entry of T, Alice assigns each $b' \in B_2$ to the empty entry with probability $\frac{1}{n}$.

This arrangement is necessary to make sure that the probability of "the final round is $2n$" is negligible. By *final round* we mean after this round, all elements in B_2 has been compared. That is, k is the final round, if

$$B_2 = \cup_{i=1}^{k} T[i].$$

Therefore, Bob can only guess which round is the final round. In other words, he can make sure that this is a final round only at the $2n$-th round, and the probability that this occurs is negligible.

Alice and Bob then compare the elements by running the protocol as shown in Figure 5.1.

In this phase, Alice and Bob repeatedly compare the elements of B_2 and A_2. The more efficient but may not be fair method of equality test proposed by Boudot,

```
for i = 1, 2, ..., w do
    for j = 1, 2, ..., m do
        Alice and Bob compare the equality of T[i] and a_j^(2);
        if T[i] = a_j^(2) then
            Alice and Bob output 1;
            stop;
        end if
    end for
end for
Alice and Bob output 0;
stop;
```

Figure 5.1: Phase 2 of Almost Complete Fairness Secure Computation of Set Disjointness

Schoenmakers, and Traoré can be used in the comparisons of the elements [1]. The comparison result will be known for Bob first, and known for Alice later. In their protocol, if Bob refuses to send the value or sending the wrong value to Alice, Alice can notice this fact, and she then react in the same way as Bob abort the protocol.

The protocol for secure computation of set intersection can be summarized in Figure 5.2 and Figure 5.3.

To design a secure protocol with complete fairness even when there is a malicious party, we need to define what should be done when the malicious party aborts the protocol abnormally. The abnormally termination of the protocol includes Alice or Bob terminate or sending wrong information to the other party in the comparison of the equality of the two elements. Note that sending wrong information can be detected easily in the Boudot, Schoenmakers, and Traoré's protocol.

It is simple for our protocol to deal with abnormal abortion of the protocol. If Alice aborts the protocol, then both parties output 0, and if Bob aborts the

Phase 1

Input: Alice has $A = \{a_1, a_2, \ldots, a_m\}$, Bob has $B = \{b_1, b_2, \ldots, b_n\}$.

Output: Alice has $B_2 = \{b_1^{(2)}, b_2^{(2)}, \ldots, b_n^{(2)}\}$, Bob has $A_2 = \{a_1^{(2)}, a_2^{(2)}, \ldots, a_m^{(2)}\}$.

1. Alice and Bob choose a hash function h and a large prime p such that the discrete logarithm over $\mathbf{Z}_p^*$ is intractable, and p is greater than the security parameter λ.
2. Alice randomly chooses $\alpha \in \mathbf{N}$, computes the set
$$A_1 = \{h(a)^\alpha \bmod p \,|\, \forall a \in A\},$$
and sends the elements of A_1, $a_1^{(1)}, a_2^{(1)}, \ldots, a_m^{(1)}$, in random order to Bob.
3. Bob randomly choose $\beta \in \mathbf{N}$, computes the set
$$B_1 = \{h(b)^\beta \bmod p \,|\, \forall b \in B\},$$
and sends the elements of B_1, $b_1^{(1)}, b_2^{(1)}, \ldots, b_n^{(1)}$, in random order to Alice.
4. After receiving A_1 from Alice, Bob computes the set
$$A_2 = \{a^\beta \bmod p \,|\, \forall a \in A_1\},$$
and stores the elements of A_2, $a_1^{(2)}, a_2^{(2)}, \ldots, a_m^{(2)}$, in random order.
5. After receiving B_1 from Bob, Alice computes the set
$$B_2 = \{b^\alpha \bmod p \,|\, \forall b \in B_1\},$$
and stores the elements of B_2, $b_1^{(2)}, b_2^{(2)}, \ldots, b_n^{(2)}$, in random order.

Figure 5.2: Summary of the Set Disjointness Protocol, Phase 1

protocol, then both party output 1.

This is because if Alice aborts the protocol, then Alice must know that all elements have been compared. If this is not the case, then she gains no advantage to abort the protocol. Therefore, Bob should assume that $A \cap B = \emptyset$.

On the other hand, if Bob aborts the protocol, then Bob must have found out that some element in his set is equal to some element in Alice's. Otherwise he gains no advantage to abort the protocol. Therefore, Alice should assume that

Phase 2

Input: Alice has $B_2 = \{b_1^{(2)}, b_2^{(2)}, \ldots, b_n^{(2)}\}$, Bob has $A_2 = \{a_1^{(2)}, a_2^{(2)}, \ldots, a_m^{(2)}\}$.

Output: Alice and Bob both learns $f(A, B)$.

1. Alice randomly selects a number w of rounds, $n < w \leq 2n$, and sets a table $T[1 \ldots w]$ of w entries. She randomly assigns the n elements of B_2 to some subset of n entries of the table. Let z be the maximum index of the random assignment in the table and $b = T[z]$. The rest entries of the table T are filled in by the following rules:
   (a) $z = w = 2n$. For each unassigned entry of T, Alice assigns b to the entry with probability $\frac{1}{2}$ and each $b' \in B_2 \backslash \{b\}$ to the entry with probability $\frac{1}{2(n-1)}$.
   (b) otherwise. For each unassigned entry of T, Alice assigns each $b' \in B_2$ to the entry with probability $\frac{1}{n}$.
2. For $i = 1, 2, \ldots, w$, Alice and Bob compare element $b = T[i]$ with each elements in A_2 as shown in Figure 5.1.
3. At the end of the protocol, both Alice and Bob know the value of $f(A, B)$.
4. If Alice aborts the protocol, then Bob assumes $f(A, B) = 0$.
5. If Bob aborts the protocol, then Alice assumes $f(A, B) = 1$.

Figure 5.3: Summary of the Set Disjointness Protocol, Phase 2

$A \cap B \neq \emptyset$.

5.2 Analysis of the Protocols

Since $a_i = b_j$ if and only if $h(a_i)^{\alpha\beta} = h(b_j)^{\alpha\beta}$, $B_2 \cap A_2 \neq \emptyset$ if and only if $B \cap A \neq \emptyset$. Therefore, at the end of the protocol, both parties will learn the correct value of $f(A, B)$, if both parties follow the protocol exactly.

Now, we argue that our second protocol achieves almost complete fairness. If both Alice and Bob honestly execute the protocol, both of them can certainly obtain the correct result. However, Alice or Bob may abort the protocol during execution after she (or he) is 100% sure about the result. This is analyzed in the following two conditions.

1. At the moment that Alice is 100% sure about $f(A, B)$, she immediately aborts protocol. This happens when the last element of B_2 has been checked for its membership in A_2. Let $b = T[k]$ be the last element in B_2 for membership checking. That is, each element in $B_2 \backslash \{b\}$ appears in $T[l]$ for some $l < k$. When Alice aborts the protocol after knowing whether $b = T[k]$ is in A_2, Bob already knows that $f(A, B)$ is the result of this round. Thus, Alice does not have any advantage over Bob.

2. Bob is 100% sure about $f(A, B)$. This happens either when Bob knows $f(A, B) = 1$ and refuses to tell Alice the correct result or when Bob is 100% sure that all elements in B_2 have been compared with A_2. In the first case, Alice should assume that $A \cap B \neq \emptyset$, otherwise, Bob gains no advantage to abort the protocol. In the second case, Bob cannot be sure that all elements in B_2 have been compared until the last round, which is not known to him except for $w = 2n$. Thus, when $w = 2n$ and $z = 2n$, Bob gains advantage by abort the protocol before Alice knows that $f(A, B) = 0$. However, The probability that $w = z = 2n$ is negligible, when n is large enough.

Note that Bob can refuse to tell Alice the correct result by abnormally terminate the protocol or sending wrong information to Alice in the equality testing protocol, as noted above.

In our protocol, Bob would gain advantage of knowing $f(A, B) = 0$ before Alice with 100% certainty if and only if $w = 2n$ and $b = T[w]$ never appears in the other entries of T.

Let p be the probability for this event. The value of p can be computed as follows. The probability for $w = 2n$ is $\frac{1}{n}$. The probability for the last position is

selected is $\frac{\binom{w-1}{n-1}}{\binom{w}{n}}$. The probability for the element $b = T[2n]$ is not selected in the remaining $w - n$ rounds is $\left(\frac{1}{2}\right)^{w-n}$. Thus,

$$p = \frac{1}{n} \cdot \frac{\binom{w-1}{n-1}}{\binom{w}{n}} \cdot \left(\frac{1}{2}\right)^{w-n} = \frac{1}{n} \cdot \frac{n}{w} \cdot \left(\frac{1}{2}\right)^{n} = \frac{1}{w} \cdot \left(\frac{1}{2}\right)^{n} < \frac{1}{n \cdot 2^n}.$$

Therefore, the value of p is negligible, when n is large.

If the value of n is small, we may need to increase the value of w to reduce the value of p. Let $w = kn$ for some $k > 2$. Then,

$$p = \frac{1}{n} \cdot \frac{\binom{w-1}{n-1}}{\binom{w}{n}} \cdot \left(\frac{1}{2}\right)^{w-n} = \frac{1}{n} \cdot \frac{n}{w} \cdot \left(\frac{1}{2}\right)^{(k-1)n} = \frac{1}{w} \cdot \left(\frac{1}{2}\right)^{(k-1)n} = \frac{1}{kn \cdot 2^{(k-1)n}}.$$

Therefore, if k is large then the value of p is negligible.

5.3 Performance Analysis of the Protocol

Recall that m and n are the cardinality of the sets A and B, respectively. Since $n < w \leq 2n$, the protocol needs at most $2mn$ comparison of the elements. Each comparison needs $O(1)$ modular exponentiation operations [1]. Thus, it needs $O(mn)$ modular exponentiation operations, which is more efficient than the first protocol.

E
S
A
1896

Chapter 6

Conclusions and Future Works

We have presented two secure two-party computation protocols for determining whether the intersection of two sets is empty or not. The first one has the property of complete fairness, and the second one is almost complete fairness. The first protocol need more computation; while the second one needs less.

Both our protocols need only $O(mn)$ comparisons of the elements. It is independent to the size of the domain from which the set A and B are drawn.

It is known that the millionaires' problem can be reduced to the set intersection problem. Our protocols can be used to solve the millionaires' problem more efficiently, especially when the size of the domain is large.

We also show that when A and B each contains only 1 element of two-element set U, then the problem cannot be computed securely with complete fairness. However, when the cardinality of U is very large, then this case of the set disjointness problem is equivalent to the equality test problem, which can be computed with complete fairness.

The following are plausible subjects for the future research.

1. Design a protocol for computing set intersection,

2. Design a protocol to computing the cardinality of the intersection of sets,

3. Extend our two-party protocol to three-party and any k-party computation protocols for $k > 3$,

4. Find other functions which has complete fair secure computation protocols.

Bibliography

[1] Fabrice Boudot, Berry Schoenmakers, and Jacques Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111:23–36, 2001.

[2] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of 18-th Annual ACM Symposium on Theory of Computing*, pages 364–369. ACM, 1986.

[3] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. In *Proceedings of 40-th Annual ACM Symposium on Theory of Computing*, pages 413–422. ACM, May 2008.

[4] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *In ACNS 2005, volume 3531 of Lecture*, pages 456–466, 2005.

[5] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptography*, 16(3):143–184, 2003.

[6] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22:161–188, 2009.

[7] Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM journal on computing*, 35(5):1245–1281, 2006.

[8] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[9] Andrew C. Yao. Protocols for secure computations. In *Proceedings of 23-rd Annual Symposium on Foundation of Computer Science*, pages 160–164. IEEE, 1982.

[10] Andrew C. Yao. How to generate and exchange secrets. In *Proceedings of 27-th Annual Symposium on Foundation of Computer Science*, pages 162–167. IEEE, 1986.

[11] Qingsong Ye, Huaxiong Wang, and Christophe Tartary. Privacy-preserving distributed set intersection. In *Proceedings of 3-rd International Conference on Availability and Security*, pages 1332–1339. IEEE, 2008.