# 3 Applied Cryptography

## 3-1 On the Construction of Fast Secure Set-Intersection Protocols

**NOJIMA Ryo**

In this paper, we consider a two-party secure set-intersection protocol. In this protocol, there are two parties, a server and a client, and they have secret-sets $Q_S$ and $Q_C$, respectively, where $|Q_S| = |Q_C| = n$. The goal of the protocol is the client obtaining $S_A \cap S_B$ while preserving the secret-sets secret. We introduce three protocols which implement this functionality in this paper. In the first protocol, we concentrate on the communication complexity and show that the protocol matches the lower bound. In the second protocol, we design the protocol based on the protocol proposed by Freedman et al. Compared to the other efficient protocols, this protocol is more secure and is adequate for the practical use. In the last protocol, we concentrate on reducing the time-complexity. In other existent protocols, at least one party needs to perform $O(n \log n)$ operations. However, in our protocol, the time-complexity is reduced to $O(n)$. The essential idea behind the protocol is the employment of the approximation algorithm for the Jaccard's distance. We examine these three protocols in detail in this paper.

***Keywords***
Secure computation, Additive homomorphic cryptosystems, Set-intersection

### 1 Introduction

People tend to assume that cryptography is principally used to ensure security in communications, as in the case of SSH and SSL technologies. In this paper, on the other hand, we discuss an innovative application of cryptography referred to as Secure Computation. Secure Computation was initially proposed by Andrew Chi-Chih Yao, who won the Turing award in 2000, launched by the problem he formulated below [7].

> Assume that two rich people, Alice and Bob, meet on the road. They want to find out which of them has more money. However, each wants to avoid letting the other know how much they have in their pocket.

Without this restriction, there would be no problem: Alice and Bob would compare amounts, and both would know who was richer. The question in this case is whether there is a solution that answers the same question without revealing to either party the amount of money in the other's pocket. Surprisingly, Yao applied cryptographic techniques to come up with just such a solution. The topic discussed in this article is closely related to this problem and is referred to as the "secure set-intersection protocol". The problem we consider here can be set forth as below, in a manner similar to the presentation of the problem above.

> Alice and Bob have secret sets, SA and SB, respectively. They want to know only the set intersection of SA and SB. Howev-

er, they want to avoid revealing to the other any of the other elements of the secret sets.

For example, let us assume $S_A = \{1, 345, 787, 88\}$ and $S_B = \{9893, 3232, 89, 345\}$. As $S_A \cap S_B = \{345\}$, we want to let Alice and Bob acquire only $S_A \cap S_B = \{345\}$ without Alice's disclosure of {1, 787, 88} or Bob's {9893, 3232, 89}. The solution to this problem was proposed by Freedman et al.[3]. We improved this protocol and designed a resulting system capable of identifying a spear attack[8].

As the title indicates, this article discusses the acceleration of the secure set-intersection protocol. If we are to construct a rapid protocol, it must feature high algorithmic efficiency. To ensure such efficiency, we need to take time complexity and communication complexity into consideration. This article describes three protocols for implementing the secure set-intersection protocol and discusses the time complexity, communication complexity, and security of these protocols. Each of the three protocols has distinct features. The first protocol is most suited to handling communication complexity. The second protocol is based on that proposed by Freedman et al., with a slight modification that significantly improves security. This protocol is the most secure among the three. The third protocol is focused on time complexity and is designed to operate more rapidly than the remaining two protocols.

## 2 Preliminary

### 2.1 Additive homomorphic cryptosystem

First, let us briefly describe a public key cryptosystem. In a public key cryptosystem, the public key, *pk*, for encryption differs from the secret key, *sk*, for decryption. The user, or receiver, possessing the secret key *sk* publishes only the public key *pk*. The sender of the message uses *pk* to encrypt the message and to send it to the receiver. The receiver uses *sk* to decrypt the ciphertext in order to obtain the original message. Here, we denote the ciphertext of the message, $m$, as $\mathrm{Enc}(m)$. An additive homomorphic cryptosystem can produce $\mathrm{Enc}(m_1 + m_2)$ from $\mathrm{Enc}(m_1)$ and $\mathrm{Enc}(m_2)$ without the secret key *sk*. Cryptosystems of this type include Paillier cryptosystem[6] and ElGamal cryptosystem[2].

### 2.2 Problem establishment

Let us denote the sets that Alice and Bob secretly hold as $S_A$ and $S_B$, respectively. Here, $S_A, S_B \subseteq U = \{1, 2, \ldots, N\}$, where $U$ is a universal set. For simplicity, we assume $|S_A| = |S_B| = n$.

A range of different secure set-intersection protocols is available. This article concentrates on the particular problems below.

**Problem 1:**
Input: Alice's input is $S_A$, and Bob's input is $S_B$.
Bob's output: Set intersection of $S_A$ and $S_B$
**Problem 2:**
Input: Alice's input is $S_A$, and Bob's input is $S_B$.
Bob's output: Number of set intersections of $S_A$ and $S_B$

When assessing the "efficiency" of the protocol, we need to consider the two factors below.

1. Time complexity
   — Sum of the time required for all users to end the protocol
2. Communication complexity
   — Sum of the size of the data transmitted in the communication channel

To construct an efficient protocol, the above two factors must be minimized to the full extent possible. Obviously, if there is a protocol that solves Problem 1 for time $O(t)$ and communication $O(c)$, then there is a protocol that solves Problem 2 with time $O(t + n \log n)$ and communication $O(c)$. However, the inverse does not generally hold. Thus, we can presume that it will be easier to construct a protocol that solves Problem 2.

## 3 Exact secure set-intersection protocol

In this article, we first consider the construction of a non-secure set-intersection protocol, followed by a method to make the protocol secure.

Let us consider the vector representation of a set. In other words, let $S$ denote a set and $V$ denote a vector of length $N$. Then, we define $V[x - 1] = 1$ if $x \in S$, and $V[x - 1] = 0$ if $x \notin S$. For example, if $U = \{1, 2, 3, 4, 5\}$ and $S = \{1, 3, 5\}$, the vector representation, $V$, of $S$ is $V = [1, 0, 1, 0, 1]$.

**Protocol 1**

Input: Alice's input is $S_A$, and Bob's input is $S_B$.

Step 1: Alice converts $S_A$ to vector $V_A$, and sends the vector to Bob.

Step 2: Bob outputs the set intersection from $V_A$ and $S_B$.

Here, the communication complexity (in other words, the number of bits transmitted in the communication channel) of this simple protocol is $N$. The problem we now consider is whether any other protocol exists featuring a smaller communication complexity. Unfortunately, there is no such protocol. This conclusion can be obtained by slightly modifying the result in [4].

Theorem 1: There is no set-intersection protocol with a communication complexity smaller than $N$.

Proof: This theorem can be proved by the contradiction.

Let us assume that there is a protocol that can produce a set intersection with a communication complexity of $N - 1$ bits. Based on this assumption, the number of different patterns of information that flows between the two parties is $2^{N-1}$ at most. Let us consider two pairs of different input patterns: $(A, A')$ and $(B, B')$. Here, let us express $X' = U - X$ for set $X$. As $A, B \subseteq \{1, \ldots, N\}$, there are $2^N$ input patterns. According to the pigeonhole principle, there must be input pairs $(A, A')$ and $(B, B')$ that share the same communication sequence. Here, for sets $S_1$ and $S_2$, let us denote $DJ(S_1, S_2) = 0$ if $S_1 \cap S_2 = \phi$, and $DJ(S_1, S_2) = 1$ in other cases. Using this notation, we can make $DJ(A, A') = DJ(B, B') = 0$ and $DJ(B, A') = 1$. Here, we consider the case in which the input pairs are $(A, A')$ and $(B, A')$. When Alice sends the first message, she sends the same message to Bob whether the input is $A$ or $B$. As Bob's input is $A'$, he sends the same message whether Alice's input is $A$ or $B$. This situation continues until the end of the protocol. Consequently, the input pairs $(A, A')$ and $(B, A')$ feature the same communication pattern throughout. Thus, for inputs $(A, A')$ and $(B, A')$, Bob produces the same output. However, as $DJ(A, A') \neq DJ(B, A')$, the protocol produces erroneous output. Therefore, the communication complexity must be $\Omega(N)$.

Thus, no protocol exists with communication complexity or time complexity smaller than $N$. Here, we are discussing the lower bound of communication complexity. This lower bound is closely related to the lower bound of communication complexity in the stream algorithm. For example, this lower bound of the set intersection can reveal that no space-efficient algorithm can be constructed to detect a DoS attack and port scan [5].

Let us now make Protocol 1 secure.

**Secure set-intersection protocol (Protocol 1)**

Input: Alice's input is $S_A(V_A)$, $pk$, and Bob's input is $S_B$ ($V_B$), $pk$, $sk$.

Step 1: Bob sends $Enc(V_B[0])$, $Enc(V_B[1])$, …, and $Enc(V_B[N - 1])$ to Alice.

Step 2: Alice calculates $c_i = Enc(r_i(V_B[i] - V_A[i] + i + 1]))$ for each $i$ and sends $\{(i, c_i)\}_i$ to Bob. Here, $r_i$ is a random number selected for each $i$.

Step 3: Bob decrypts the ciphertext that he has received and if the elements are included in $S_B$, he outputs them as included in the set intersection.

Theorem 1 states that when the size of the

universal set is $N$, the cost of the communication is $\Omega(N)$. Particularly when the size $n$ of the set satisfies $n \log N < N$, the protocol below is more efficient.

**Protocol 2:**

Input: Alice's input is $S_A$, and Bob's input is $S_B$.

Step 1: Alice sends each element of $S_A$ to Bob.

Step 2: Bob outputs the set intersection of $S_A$ and $S_B$.

The communication complexity of this protocol is $n \log N$. In other words, if $N > n \log N$, this protocol is more efficient than Protocol 1 in both time complexity and communication complexity.

We can consider the protocol below to make the protocol above secure.

**Secure set-intersection protocol (Protocol 2)**

Input: Alice's input is $S_A = \{a_1, \ldots, a_n\}$, $pk$, and Bob's input is $S_B = \{b_1, \ldots, b_n\}$, $pk$, $sk$.

Step 1: Bob sends $\mathrm{Enc}(b_1)$, $\mathrm{Enc}(b_2)$, …, and $\mathrm{Enc}(b_n)$ to Alice.

Step 2: Alice sends $\mathrm{Enc}(r_{ij}(b_i - \mathrm{aj}) + \mathrm{aj})$ for each $i$ and $j$. Here, $r_{ij}$ is a random number.

Step 3: Bob decrypts the ciphertext that he has received, and if the plaintext is included in $S_B$, he outputs it as included in the set intersection.

The communication complexity of this protocol is $O(n^2)$ and may not always be efficient. The solution to this problem was proposed by Freedman et al.[3]. However, we needed to modify Freedman's protocol slightly to make it secure. Thus, here we indicate the modified protocol.

**Secure set-intersection protocol (Modified Protocol 2)**

Input: Alice's input is $S_A = \{a_1, \ldots, a_n\}$, $pk$, and Bob's input is $S_B = \{b_1, \ldots, b_n\}$, $pk$, $sk$.

Step 1: Bob encrypts each $c_i$ in $f(X) = X^n + c_{n-1}X^{n-1} + \ldots + c_0 = (X - b_1)(X - b_2)\ldots(X - b_n)$ and sends them to Alice.

Step 2: Alice sends $\mathrm{Enc}(r_j f(a_j) + a_j)$ for each $j$ to Bob.

Step 3: Bob decrypts the ciphertext that he has received, and if the plaintext is included in $S_B$, he outputs it as included in the set intersection.

The communication complexity of this protocol is $O(n)$, a significant improvement compared to $O(n^2)$ of Protocol 2.

When we implement this protocol using the bucket allocation technique, we can sometimes speed up the process as much as 20-fold.

We used this protocol when we constructed the spear-attack identification system[8].

## 4 Approximate protocol

### 4.1 Approximate secure set-intersection protocol

In the previous sections, we considered protocols that output a set intersection. In this section, we consider the construction of an approximate protocol to acquire a fast-operating secure set-intersection protocol. The approximate protocol here calculates the approximate value of $|S_A \cap S_B|$ instead of calculating $S_A \cap S_B$.

First, we introduce the family of min-wise independent functions as a tool for acquiring the approximate protocol.

Definition

[Family of min-wise independent functions[1]]

If a family of functions, $H \subset [N] \rightarrow [u]$, satisfies

$$\Pr_{h \leftarrow H}[h(x) = \min\{h(y) | y \in X\}] = 1/|X|$$

for arbitrary $X \subset [N]$ and $x \in X$, we say that the family of functions satisfies min-wise independence.

Lemma: When $H$ is a family of min-wise independent functions,

$$\Pr_{h \leftarrow H}[\min\{h(A)\} = \min\{h(B)\}] = |A \cap B|/|A \cup B|$$

holds for arbitrary $A, B \subseteq [N]$.

Let us define match$(A, B) = |A \cap B|/|A \cup B|$ and sum$(A, B) = |A| + |B|$. When we express $|A \cap B|$ with match and sum, we have the expression below.

$$|A \cap B| = \text{match}(A, B) \bullet \text{sum}(A, B) \bullet (1 + \text{match}(A, B))^{-1}$$

In other words, acquiring the approximate value for $|A \cap B|$ is equivalent to acquiring the approximate value for $|A \cap B|/|A \cup B|$. Thus, in this section, we consider the protocol for calculating the approximate value of $|A \cap B|/|A \cup B|$. First, we consider the non-secure protocol as before.

**Protocol 3:**

Input: Alice's input is $S_A$, Bob's input is $S_B$, and the common input is $l$.

Output: Alice's output is none, and Bob's output is the approximate value of $|S_A \cap S_B|$.

Step 1: Alice randomly selects $l$ items of min-wise independent functions and sends them to Bob.

Step 2: Alice calculates$(a_1, \ldots, a_l) = (\min\{h_1(S_A)\}, \ldots, \min\{h_l(S_A)\})$. Bob similarly calculates $(b_1, \ldots, b_l) = (\min\{h_1(S_B)\}, \ldots, \min\{h_l(S_B)\})$.

Step 3: Alice sends $(a_1, \ldots, a_l)$ to Bob.

Step 4: Bob calculates $|\{1 \le i \le l \mid a_i, = b_i\}|/l$ and outputs the result.

## 4.2 Analysis

Here, we analyze the degree of difference between $|\{1 \le i \le l \mid a_i, = b_i\}|/l$ output by Bob and $|A \cap B|/|A \cup B|$. The probability that the $i$-th values match is $|A \cap B|/|A \cup B|$ due to the nature of the min-wise independent functions. Let us consider a random variable, $X_i$, which takes the value 1 when the $i$-th values match and 0 when the $i$-th values do not match. Let us calculate the expectation and the variance of this random variable. The expectation is $E[X_i] = 1 \bullet \Pr[X_i = 1] + 0 \bullet \Pr[X_i = 0]$ x$= |A \cap B|/|A \cup B|$. The variance is $\text{Var}[X_i] = E[X_i^2] - E[X_i]^2$ by definition. Noting that $E[X_i^2] = 1 \bullet \Pr[X_i = 1] + 0 \bullet \Pr[X_i = 0] = |A \cap B|/|A \cup B|$, we have $|A \cap B|/|A \cup B| - (|A \cap B|/|A \cup B|)^2$. Here, we consider a new random variable $X$ defined as $X = (X_1 + X_2 + \ldots + X_l)/l$. Due to the linearity of the expectation value, we obtain the expression below.

$$\begin{aligned} E[X] &= E[(X_1 + X_2 + \ldots + X_l)/l] \\ &= E[(X_1 + X_2 + \ldots + X_l)]/l \\ &= (E[X_1] + E[X_2] + \ldots + E[X_l])/l \\ &= |A \cap B|/|A \cup B| \end{aligned}$$

For the variance, we obtain the expression below.

$$\begin{aligned} \text{Var}[X] &= \text{Var}[(X_1 + X_2 + \ldots + X_l)/l] \\ &= \text{Var}[X_1/l] + \text{Var}[X_2/l] + \ldots + \text{Var}[X_l/l] \\ &= (|A \cap B|/|A \cup B| - (|A \cap B|/|A \cup B|)^2)/l^2 \end{aligned}$$

Thus, from Chebyshev's inequality, the relationship below holds for an arbitrary positive value of $\varepsilon$.

$$\Pr[|X - |A \cap B|/|A \cup B| \ge \varepsilon] \le (|A \cap B|/|A \cup B| - (|A \cap B|/|A \cup B|)2)/l^2 \varepsilon^2 \le 1/l^2 \varepsilon^2$$

Thus, we have acquired the probability that the output of the algorithm is not $|A \cap B|/|A \cup B|$.

## 4.3 Protocol for making Protocol 3 secure

In this section, we make Protocol 3, which we evaluated in the previous section, secure. To do so, we need to create an equivalence checker using additive homomorphic cryptosystems. In this protocol, Alice has $x$, and Bob has $y$. Bob outputs 1 if $x = y$ and outputs 0 in other cases.

**Protocol for checking security equivalence**

Input: Alice's input is $x$, and Bob's input is $y$.

Bob's output: 1 if $x = y$, and 0 in other cases.

Step 1: Bob sends Enc(y) to Alice.

Step 2: Alice calculates Enc$(r(x - y) + 1)$ and sends the result to Bob.

Step 3: Bob decrypts the ciphertext that he has received and outputs 1 if the result is 1

and 0 in other cases.

Now, we apply the protocol for checking security equivalence, and make Protocol 3 secure.

**Secure set-intersection protocol (Protocol 3)**
Input: Alice's input is $S_A$, Bob's input is $S_B$, and the common input is $l$.
Output: Alice's output is none, and Bob's output is the approximate value of $|S_A \cap S_B|$.
Step 1: Alice randomly selects $l$ items of min-wise independent functions and sends them to Bob.
Step 2: Alice calculates $(a_1, \ldots, a_l) = (\min\{h_1(S_A)\}, \ldots, \min\{h_l(S_A)\})$. Bob similarly calculates $(b_1, \ldots, b_l) = (\min\{h_1(S_B)\}, \ldots, \min\{h_l(S_B)\})$.
Step 3: Bob sends $(\mathrm{Enc}(b_1), \ldots, \mathrm{Enc}(b_l))$ to Alice.
Step 4: Alice sends $(\mathrm{Enc}(r_1(a_1 - b_1) + 1), \ldots, \mathrm{Enc}(r_l(a_l - b_l) + 1))$ to Bob.
Step 5: Bob decrypts the ciphertext and counts the number of "1" responses. He outputs sum/$l$, where sum is the total number of "1".

## 5 Conclusions

As cryptographic protocols have been mainly developed in theoretical studies, few existing protocols can be put to practical societal application[8]. The secure set-intersection protocol discussed in this article is a rare example. We can nevertheless conclude that cryptographic protocols developed in the course of theoretical study retain the possibility of spurring an enormous field of research if we remain sharply aware of the potential practical applications of these protocols.

Considering Internet applications alone, we see a wide range of potential applications of cryptographic protocols. For example, traceback technology is aimed at tracking users that have committed fraud. However, this technology may also be used to invade the privacy of authorized users who have not committed fraud. Cryptographic technologies can be used both to protect the privacy of authorized users and to track unauthorized users. The author's next major research challenge is to design such a cryptographic protocol.

### *References*


1 Andrei. Z. Broder, M. Charikar, Alan. M. Frieze, and Michael Mitzenmacher, "Min-Wise Independent Permutations", J. Compute. Syst. Sci. 60 (3), pp.630-659.
2 Taher ElGamal, "A Public-Key Cryptosystem and a Signature Based on Discrete Logarithms, IEEE Transactions on Information Theory", V.IT-31, N.4, 1985.
3 Michael J. Freedman, Kobbi Nissim, and Benny Pinkas, "Efficient Private Matching and Set Intersection", EUROCRYPT 2004, pp.1-19.
4 Eyal Kushilevitz, and Noam Nissan, "Communication Complexity", Cambridge University Press, 1997.
5 Kirill Levchenko, Ramamohan Paturi, and George Varghese, "On the Difficulty of Scalably Detecting Network Attacks", ACM Conference on Computer and Communication Security 2004, pp.12-20.
6 Pascal Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Class", EUROCRYPT 1999, pp.223-238.
7 Andrew Chi-Chih Yao, "Protocols for Secure Computations", FOCS 1982, pp.160-164.
8 Starting collaborative demonstration experiment to realize a systemfor detecting targeted cyber attacks-Toward realizing a system detecting the cyber attacks against a particular organization-, http://www.nict.go.jp/news/h19-press.html, 2008

***NOJIMA Ryo**, Dr. Eng.*

*Researcher, Traceable Secure Network Group, Information Security Research Center*

*Algorithm, Cryptography*