Hindawi
Security and Communication Networks
Volume 2017, Article ID 9717580, 11 pages
https://doi.org/10.1155/2017/9717580

Research Article

Efficient Secure Multiparty Subset Computation

Sufang Zhou,[1] Shundong Li,[1] Jiawei Dou,[2] Yaling Geng,[1] and Xin Liu[1]

[1]*School of Computer Science, Shaanxi Normal University, Xi'an 710062, China*
[2]*School of Mathematic and Information Science, Shaanxi Normal University, Xi'an 710062, China*

Correspondence should be addressed to Shundong Li; shundong@snnu.edu.cn

Received 7 January 2017; Revised 21 April 2017; Accepted 11 June 2017; Published 28 August 2017

Academic Editor: Jimson Mathew

Secure subset problem is important in secure multiparty computation, which is a vital field in cryptography. Most of the existing protocols for this problem can only keep the elements of one set private, while leaking the elements of the other set. In other words, they cannot solve the secure subset problem perfectly. While a few studies have addressed actual secure subsets, these protocols were mainly based on the oblivious polynomial evaluations with inefficient computation. In this study, we first design an efficient secure subset protocol for sets whose elements are drawn from a known set based on a new encoding method and homomorphic encryption scheme. If the elements of the sets are taken from a large domain, the existing protocol is inefficient. Using the Bloom filter and homomorphic encryption scheme, we further present an efficient protocol with linear computational complexity in the cardinality of the large set, and this is considered to be practical for inputs consisting of a large number of data. However, the second protocol that we design may yield a false positive. This probability can be rapidly decreased by reexecuting the protocol with different hash functions. Furthermore, we present the experimental performance analyses of these protocols.

1. Introduction

The prompt development of networks provides a great opportunity for multiparty cooperative computation, and it challenges the privacy of the participants' information. In a complex network environment, parties may not trust each other during computations, and they are required to keep their information private. Secure multiparty computation is a key technology for privacy-preserving in cooperative computations. Thus, secure multiparty computation attracts increasing attention in the international cryptographic community.

Secure multiparty computation was first introduced by Yao [1] as a millionaires' problem in 1982. The millionaires' problem can be described as follows. Two millionaires, Alice and Bob, want to know who is richer, but neither Alice nor Bob wants to disclose her/his own wealth to the other. This is a secure two-party computation problem. After this, Ben-Or et al. [2] gave the first secure multiparty computation protocol. A secure multiparty computation involves any two or more parties who use their own private data to cooperatively compute a function in order to obtain the predetermined output while keeping their input information private. Secure multiparty computation is a general cryptographic protocol. Many cryptographic protocols for cooperative computations that contain two or more parties can be viewed as secure multiparty computation protocols, and these include key exchange protocols [3], digital signature protocols [4], secret sharing protocols [5], zero-knowledge proof protocols [6], and oblivious transfer protocols [7]. Secure multiparty computation is a key technology in network security, and it has been the focus of the international cryptographic community for many years. The Turing Award winner Goldwasser [8] predicted that "the field of multiparty computations is today where public key cryptography was ten years ago, namely, an extremely powerful tool and rich theory whose real-life usage is at this time only beginning but will become in the future an integral part of our computing reality."

Goldreich et al. [9, 10] thoroughly studied the secure multiparty computation problem and established its theoretical foundation. They proved that secure multiparty computation problems are theoretically solvable and proposed a general solution to secure multiparty computation problems. Because the general solution is inefficient and impractical for special problems, they also noted that, to improve efficiency, special solutions should be developed for special problems. This

observation motivates people to study solutions to various secure multiparty computation problems. The problems studied include millionaires' problems [11, 12], secure computational geometry problems [13], comparisons of information without it being leaked [14], private bidding and auction problems [15], and privacy-preserving data mining problems [16]. In addition, there are many other new secure multiparty problems that need to be studied.

Because many problems can be abstracted as set problems, private set operation is a highly important field in secure multiparty computation. These problems include set intersection [18], set union [19], and subsets [17]. The set intersection problem and the set union problem have been widely studied, while there are only few studies of the subset problem. However, there are a variety of applications for the subset problem.

(1) In data mining, there is an important principle (Apriori Principle) about the association rule, which states that if an itemset is frequent, then all of its subsets must also be frequent [20]. Suppose that both Alice and Bob are suppliers of a supermarket W. Alice has a large frequent itemset A that is generated with data mining from the transactions of W. Bob has an itemset B, and he wants to know whether B is also a frequent itemset. However, he cannot perform data mining on the transaction data of W (either he cannot obtain the transaction data or he does not have data mining knowledge). Therefore, he resorts to Apriori Principle, but he does not want to disclose B to Alice. As expected, Alice also wishes to keep A a secret. In this application, they have to privately determine whether $B \subseteq A$.

(2) In secret sharing, a secret is divided into w shares, and they are privately given to w parties who are called the legal shareholders, and any t or more shareholders can reconstruct the secret. During the reconstruction of the secret, some illegal shareholders may take part in the reconstruction. To prevent illegal shareholders from taking part in the reconstruction, the authenticity of the shareholder participants must be privately determined. This is where the secure subset protocol comes into play.

It is generally known that the subset problem is a special case of the set intersection. However, when applied to solve the subset problem, existing set intersection protocols can lead to both insecure and inefficient solutions. For the subset problem, we only need to determine whether $B \subseteq A$. Meanwhile, the intersection protocols have to compute every element where $x \in A \wedge x \in B$. This method will first disclose the same elements between B and A for the subset problem. Furthermore, the subset problem is a decision problem, and it does not need to compute all the elements of $A \cap B$. Thus, the set intersection protocols are not suitable for the subset problem.

If there are two sets A and B, where $|A| \geq |B|$, in most current studies, many private subset operations can be classified into two different cases. First, two parties proved that $B \subseteq A$, leaking the elements of set B [21–23]. Second, two parties proved that $B \subseteq A$ without keeping the privacy of the elements of set A [24–27].

In addition, Kissner and Song [17] proposed a secure solution to the subset problem based on the Paillier additively homomorphic encryption scheme [28], the representation of elements of a set as roots of a polynomial, and the mathematical properties of polynomials. In their proposed solution, both sets A and B can be kept private. Let δ be the encryption of the polynomial $p(x)$ that represents the larger set A. Note that if $B \subseteq A$, then $p(b) = 0$ is true for every element $b \in B$ (or vice versa). That is, $B \subseteq A \Leftrightarrow \forall_{b \in B} p(b) = 0$. The party who has the smaller set B evaluates the encrypted polynomial δ at each element $b \in B$ to obtain $|B|$ ciphertexts, and it multiplies these ciphertexts to obtain β. If β is an encryption of 0, then $B \subseteq A$. However, the computational complexity of this protocol takes $(2mn + 4m + 8) \log N + mn$ ($|A| = m$, $|B| = n$) modular multiplications (mod N^2, details are presented in Section 5.1). This depends on the product of $|A|$ and $|B|$. However, the protocol is inefficient for the computation of a large quantity of data.

Furthermore, Ye et al. [29] and Sang and Shen [30] separately gave their subset protocols, which are mainly based on the oblivious polynomial evaluations, and which are similar to Kissner's protocol. The subset protocol of [29] was presented in the distributed setting. By using (t, w) Shamir's secret sharing scheme, the polynomial constructed based on the larger set A was distributed to multiple servers. The party who had the smaller set interacted with at least t servers to compute the subset problem based on the standard variant of the ElGamal encryption. The overall cost for the computation is $O(t|A||B|)$, and the communication is $O(t|A||B| \log p)$ bits. In the subset protocol of [30], Sang utilized a nonmalleable NonInteractive Zero-Knowledge (NIZK) argument, which is based on the Boneh-Goh-Nissim (BGN) cryptosystem to protect it against malicious attacks. Without considering the computational complexity of malicious attacks, the computational complexity of this protocol is $O(|A||B|)$ besides the NIZK argument. Meanwhile, our protocols have a linear computational complexity in the cardinality of the large set $O(|A|)$ (details are presented in Section 5.1).

Moreover, Blanton and Aguiar [31] created an efficient subset protocol based on the oblivious algorithms, such as oblivious sorting algorithms and oblivious equality algorithms. Unfortunately, this protocol is constructed using the circuit method and has the drawbacks of the circuit method [32].

Shundong et al. [12] described a secure subset protocol that retains the privacy of both sets A and B, and it is based on symmetric cryptography and has high efficiency. However, the smaller set B can only have one element in the protocol. If set B has more than one element, the parties have to execute the protocol $|B|$ times and choose new pseudorandom sequences on each occasion, which is tedious.

In this study, we mainly propose two secure subset protocols for different situations using homomorphic encryption schemes which can be multiplicative or additive. Because a multiplicatively homomorphic encryption scheme is more efficient than an additive one, we choose a multiplicative one

to build our protocols. To the best of our knowledge, encryption schemes can currently encrypt only integer messages. In addition, the sets to be computed always come from a known set whose elements are not integers for many often-occurring ranges. For this case, we design an efficient protocol, which is based on a new encoding method, and a homomorphic encryption scheme. The computational complexity of this protocol is linear in the size of the large set. For the situation in which the sets are taken from a large domain, we further present an efficient protocol based on Bloom filters and a homomorphic encryption scheme to improve efficiency without compromising accuracy much. Furthermore, we show that, by using the Bloom filter, we can solve the subset problem for sets that are taken from an exponentially large domain.

The rest of this paper is organized as follows. In Section 2, we introduce some preliminaries. In Section 3, we propose an efficient secure subset protocol for sets whose elements are drawn from a known set using a new encoding method and homomorphic encryption schemes. In Section 4, we show the secure subset protocol for sets within a large domain based on the Bloom filter and homomorphic encryption schemes, while in Section 5, we present an analysis of secure subset protocols and the experimental implementation. Finally, in Section 6, we conclude this paper.

2. Preliminaries

2.1. Secure Subset Problem. Alice has a set $A = \{a_1, \ldots, a_m\}$, and Bob has a set $B = \{b_1, \ldots, b_n\}$. Alice and Bob want to determine whether B is a subset of A without disclosing any information about the elements of their sets relative to each other. This can be abstracted as a secure subset problem.

2.2. Homomorphic Encryption Scheme. A homomorphic encryption scheme is an encryption scheme with some special properties that make the homomorphic encryption scheme a building block of many secure multiparty computation protocols. A conventional public key encryption scheme $\mathscr{E}$ consists of three algorithms: $\text{KeyGen}_{\mathscr{E}}$, $\text{Encrypt}_{\mathscr{E}}$, and $\text{Decrypt}_{\mathscr{E}}$.

(i) $\text{KeyGen}_{\mathscr{E}}$. $\text{KeyGen}_{\mathscr{E}}$ takes a security parameter k as the input, and it outputs a secret key sk and the corresponding public key pk with the definition of the plaintext space $\mathscr{P}$ and the ciphertext space $\mathscr{C}$.

$$\left(\text{sk}, \text{pk}, \mathscr{P}, \mathscr{C}\right) \longleftarrow \text{KeyGen}_{\mathscr{E}}(k). \tag{1}$$

(ii) $\text{Encrypt}_{\mathscr{E}}$. Taking pk and a plaintext $M \in \mathscr{P}$ as inputs, $\text{Encrypt}_{\mathscr{E}}$ outputs a ciphertext $C \in \mathscr{C}$.

$$C \longleftarrow \text{Encrypt}_{\mathscr{E}}(\text{pk}, M). \tag{2}$$

(iii) $\text{Decrypt}_{\mathscr{E}}$. Taking a ciphertext $C \in \mathscr{C}$ and the secret key sk as inputs, $\text{Decrypt}_{\mathscr{E}}$ outputs the plaintext $M \in \mathscr{P}$.

$$M \longleftarrow \text{Decrypt}_{\mathscr{E}}(\text{sk}, C). \tag{3}$$

In addition to the three conventional algorithms, a homomorphic encryption scheme $\mathscr{E}$ has an efficient algorithm $\text{Evaluate}_{\mathscr{E}}$, which takes as inputs the public key pk, an operation S, and a tuple of ciphertexts $\mathbf{C} = \langle C_1, \ldots, C_s\rangle$ (C_i is the ciphertext of M_i, $i = 1, \ldots, s$), and it outputs a ciphertext of $S(M_1, \ldots, M_s)$.

$$\begin{aligned}&\text{Encrypt}_{\mathscr{E}}\left(\text{pk}, S\left(M_1, \ldots, M_s\right)\right)\\ &\quad = \text{Evaluate}_{\mathscr{E}}\left(\text{pk}, S, \mathbf{C}\right).\end{aligned} \tag{4}$$

Our construction uses semantically secure public key encryption schemes that preserve the group homomorphism under some computational complexity assumptions. This property is obtained by the Paillier encryption scheme [28] and the ElGamal encryption scheme [33] under the Composite Residuosity Class (CRC) assumption and the Computational Diffie-Hellman (CDH) assumption, respectively. Details are presented as follows.

Pailler Encryption Scheme

(i) KeyGen. On inputting a security parameter k, this algorithm generates two large primes p, q, sets $N = pq$, and $\lambda = \text{lcm}(p-1, q-1)$ and computes g such that $\gcd(L(g^{\lambda} \bmod N^2), N) = 1$, where $L(x)$ is defined as

$$L(x) = \frac{x-1}{N}. \tag{5}$$

$Z^*_{N^2}$ is the ciphertext space, and Z^*_N is the plaintext space. The public key is (g, N), and the private key is λ.

(ii) Encrypt. To encrypt plaintext $M \in Z^*_N$, the *Encrypt* algorithm selects a random number $r < N$ and computes

$$E(M) = g^M \cdot r^N \bmod N^2. \tag{6}$$

(iii) Decrypt. To decrypt the ciphertext $C = E(M) \in Z^*_{N^2}$, the *Decrypt* algorithm computes

$$M = \frac{L\left(C^{\lambda} \bmod N^2\right)}{L\left(g^{\lambda} \bmod N^2\right)} \bmod N. \tag{7}$$

(iv) Evaluate. For ciphertexts $C_1 = E(M_1)$, $C_2 = E(M_2)$, and $C_3 = E(M)$ and a constant c, we have

$$\begin{aligned}E(M_1) \cdot E(M_2) &= \left(g^{M_1} r_1^N \bmod N^2\right)\\ &\quad \cdot \left(g^{M_2} r_2^N \bmod N^2\right)\\ &= g^{M_1+M_2} \cdot (r_1 r_2)^N \bmod N^2\\ &= E(M_1 + M_2),\\ E(M)^c &= \left(g^M r_1^N \bmod N^2\right)^c\\ &= g^{cM} (r_1^c)^N \bmod N^2 = E(cM).\end{aligned} \tag{8}$$

In this encryption scheme, if $M_1 = 0$, then $E(M_1) \cdot E(M_2) = E(M_1 + M_2) = E(M_2)$.

ElGamal Encryption Scheme

(i) KeyGen. On inputting a security parameter γ, the *KeyGen* algorithm generates a large prime p and a generator α, and it randomly chooses a number z as a private key. The public key is $y = \alpha^z \bmod p$.

(ii) Encrypt. Taking M and y as inputs, the *Encrypt* algorithm selects a random number r and computes

$$E(M) = (c_1, c_2) = (\alpha^r \bmod p, My^r \bmod p). \quad (9)$$

(iii) Decrypt. This algorithm takes $E(M)$ and z as inputs and computes

$$c_2 \cdot c_1^{-z} \bmod p = My^r \cdot (\alpha^r)^{-z} \bmod p = M \bmod p. \quad (10)$$

(iv) Evaluate. Given ciphertexts $E(M_1)$, $E(M_2)$, and $E(M)$ and a constant c, we can compute that

$$\begin{aligned} &E(M_1) \cdot E(M_2) \\ &\quad = (\alpha^{r_1} \bmod p, M_1 y^{r_1} \bmod p) \\ &\qquad \cdot (\alpha^{r_2} \bmod p, M_2 y^{r_2} \bmod p) \\ &\quad = (\alpha^{r_1+r_2} \bmod p, M_1 \cdot M_2 y^{r_1+r_2} \bmod p) \\ &\quad = E(M_1 \cdot M_2), \\ &E(M)^c = (\alpha^r \bmod p, My^r \bmod p)^c \\ &\quad = (\alpha^{rc} \bmod p, M^c y^{rc} \bmod p) = E(M^c). \end{aligned} \quad (11)$$

In this encryption scheme, if $M_1 = 1$, then $E(M_1) \cdot E(M_2) = E(M_1 \cdot M_2) = E(M_2)$.

These two schemes are semantically secure under the CRC assumption or the CDH assumption. That is, given two messages M_0 and M_1, as well as a ciphertext $E(M_t)$ ($t \in \{0, 1\}$) encrypted by these encryption schemes, no probabilistic polynomial-time algorithm can determine whether the ciphertext $E(M_t)$ is a ciphertext of M_1 or M_0 with nonnegligible advantages.

2.3. Security of Secure Multiparty Computation. We assume that all parties are semihonest. In general, a semihonest party follows the prescribed protocol correctly, except that it keeps a record of all its intermediate computations and may try to derive the other party's private inputs from the record. Goldreich [10] also designed a compiler that can force each party to either behave in a semihonest manner or be detected. Given a protocol π, which privately computes function f in the semihonest model, this compiler can produce a new protocol Π, which privately computes f in the malicious model. This work demonstrates that the study based on the semihonest model is very important. Therefore, our work focuses on solutions to the subset problem in the semihonest model.

Different methods are used to prove the security in different cryptographic fields. The proof method, which reduces the security to a difficult assumption in the standard model or the random oracle model, is suitable for verifying encryption schemes and signature schemes. The simulation paradigm is widely accepted and is used to prove the security of secure multiparty computation protocols. The basic idea behind the simulation paradigm is to compare a real secure multiparty computation protocol with an ideal one. The real protocol is considered as secure if the real secure multiparty computation protocol does not leak more information than the ideal one. The ideal secure multiparty computing protocol can be described as follows.

Assume that there is an absolute trusted third party, denoted by Trent, who will neither lie nor leak any information that should not be revealed. Alice has a number x_1, Bob has a number x_2, and they want to securely compute a function $f(x_1, x_2)$. They can do as follows: (a) Alice and Bob, respectively, send x_1 and x_2 to Trent, (b) Trent computes the function $f(x_1, x_2)$, and (c) Trent tells Alice and Bob the result.

Because most secure multiparty computation protocols are constructed using public key encryption schemes, the security proof for a secure multiparty computation protocol is to reduce the security of the protocol to the security of the public key encryption scheme on which the protocol is based. That is, to prove that a multiparty computation protocol is secure, we must prove that the real secure multiparty computation protocol does not leak more information than the ideal protocol with the assumption that the public key encryption scheme used in the real protocol is secure. In other words, the information that a party obtains in a real secure multiparty computation protocol can be simulated by a simulator that only obtains the result and one party's input, and if the sets of information obtained from both methods are computationally indistinguishable, the real protocol is secure.

Intuitively, a protocol that computes f is secure if whatever a set of semihonest parties can obtain after participating in the protocol could be obtained from the inputs and outputs of these same parties. In the simulation paradigm, this means that the VIEW (this will be discussed later) of a set of semihonest parties during a protocol execution can be simulated by their inputs and outputs.

Suppose that there are two parties Alice and Bob who have sets A and B, respectively. They want to privately compute $f(A, B)$, which is a polynomial-time function. Further, suppose that π is a protocol-computing function $f(A, B)$. The VIEW of Alice, who has the set A, during the execution of π on the input (A, B), is denoted by $\text{VIEW}_1^\pi(A, B) = (A, r^1, m_1^1, \ldots, m_t^1)$, where r^1 is the result of Alice's internal coin tosses, and m_i^1 ($i = 1, \ldots, t$) is the i-th message that Alice received. The output of Alice after the execution of π is denoted as $\text{OUTPUT}_1^\pi(A, B)$, which is implicit in Alice's VIEW. Similarly, Bob's VIEW and output during the execution of π are $\text{VIEW}_2^\pi(A, B) = (B, r^2, m_1^2, \ldots, m_t^2)$ and $\text{OUTPUT}_2^\pi(A, B)$.

Definition 1 (security in the semihonest model [10]). For a function f, we say that π privately computes f if there exist two probabilistic polynomial-time simulators, denoted by S_1 and S_2, such that

$$\begin{aligned}&\{(S_1(A, f_1(A,B)), f_2(A,B))\}_{A,B\in\{0,1\}^*}\\&\quad\stackrel{c}{\equiv}\{(\mathrm{VIEW}_1^{\pi}(A,B), \mathrm{OUTPUT}_2^{\pi}(A,B))\}_{A,B\in\{0,1\}^*},\\&\{(f_1(A,B), S_2(B, f_2(A,B)))\}_{A,B\in\{0,1\}^*}\\&\quad\stackrel{c}{\equiv}\{(\mathrm{OUTPUT}_1^{\pi}(A,B), \mathrm{VIEW}_2^{\pi}(A,B))\}_{A,B\in\{0,1\}^*},\end{aligned} \tag{12}$$

where $\stackrel{c}{\equiv}$ denotes *computational indistinguishability*.

3. Protocol for Sets Whose Elements Are Drawn from a Known Set

Suppose that Alice has a set A and Bob has a set B. A straightforward way to compute the subset problem between A and B, without worrying about the privacy, is as follows: Alice sends her set A to Bob; Bob computes whether $B \subseteq A$; then tells the result to Alice. Thus, Alice and Bob obtain the subset relation between A and B.

By the definition of subset, if $B \subseteq A$, then for any element $x \in B \Rightarrow x \in A$. Thus, we can reduce the subset problem to checking whether all the elements of set B are in set A. If all the elements of B are the elements of A, then $B \subseteq A$; otherwise, $B \nsubseteq A$.

Suppose Alice and Bob have sets $A = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_n\}$ $(A, B \subseteq U = \{u_1, \ldots, u_l\})$, respectively. They want to determine whether or not $B \subseteq A$ without disclosing either A or B.

3.1. Foundations of This Protocol. Before we describe the idea of our protocol, we first present the building blocks— a 1-r encoding method and a 1-0 encoding method—based on the definition of the characteristic vector of mathematics.

1-r Encoding. A 1-r encoding is used to encode a set to a 1-r vector, where every component is either 1 or r, where r is a random number and $r \in Z_p^* \wedge r \neq 1$. The principle for encoding a set $A = \{a_1, \ldots, a_m\} \subseteq U = \{u_1, \ldots, u_l\}$ to a 1-r vector $V_A^r = (v_{a1}, \ldots, v_{al})$ is as follows: if $u_i \in A$ $(i = 1, \ldots, l)$, then $v_{ai} = 1$; otherwise, $v_{ai} = r$ $(r \in Z_p^* \wedge r \neq 1)$. This can also be described by the following pseudocodes:

```
For i = 1 to l
  If u_i ∈ A
    v_ai ← 1
  Else v_ai ← r   (r ∈ Z_p^* ∧ r ≠ 1)
End
```

1-0 Encoding. This method is similar to a 1-r encoding, but with a small difference. Encoding a set $B = \{b_1, \ldots, b_n\} \subseteq U = \{u_1, \ldots, u_l\}$ to a 1-0 vector $V_B^1 = (v_{b1}, \ldots, v_{bl})$ is as follows: if $u_i \in B$ $(i = 1, \ldots, l)$, then $v_{bi} = 1$; otherwise, $v_{bi} = 0$. This can also be described by the pseudocodes as follows:

```
For i = 1 to l
  If u_i ∈ B
    v_bi ← 1
  Else v_bi ← 0
End
```

From a high-level perspective, the 1-r encoding (1-0 encoding) encodes an $x \in A$ $(x \in B)$ with a one component and an $x \notin A$ $(x \notin B)$ with a random (zero) component. Alice and Bob can use the above encoding methods to compute the subset problem.

Alice encodes set A to a 1-r vector V_A^r, and Bob encodes set B to a 1-0 vector V_B^1. Alice sends her vector V_A^r to Bob. Bob chooses the components of V_A^r corresponding to the one components of V_B^1 and computes their product v, $v = \prod_{v_{bi}=1} v_{ai}$. If $v = 1$, then $B \subseteq A$; otherwise $B \nsubseteq A$. This is the principle of deciding the subset relation between sets A and B. For simplicity, we give a simple example in Table 1. V' is the vector that is chosen from vector V_A^r according to the one components of V_B^1.

Alice and Bob can also compute the subset using another method. Alice and Bob encode A to a 0-r vector $V_A^{r\,*}$ and B to a 1-0 vector V_B^1 based on a 0-r encoding and the 1-0 encoding, respectively. The 0-r encoding is similar to the 1-r encoding and requires only that we change one component to zero components and $r \in Z_p^* \wedge r \neq 0$. Bob computes $v^* = \sum_{v_{bi}=1} v_{ai}$. If $v^* = 0$, then $B \subseteq A$; otherwise, $B \nsubseteq A$.

However, we can use the above approaches to easily determine whether $B \subseteq A$ easily, but it is not secure. We use semantically secure and homomorphic encryption schemes to privately compute v or v^* in order to privately determine whether $B \subseteq A$.

3.2. Protocol. We give a solution to the secure subset problem in Protocol 2 based on the above foundations. Because a multiplicatively homomorphic encryption scheme is more efficient than an additive one, we choose a multiplicative one and encode A to V_A^r to present this protocol. The ElGamal encryption scheme is semantically secure if the CDH assumption holds, which can make the ciphertexts of the same plaintext indistinguishable. Therefore, we can have different ciphertexts of plaintext 1. In addition, the ElGamal encryption scheme is multiplicatively homomorphic, and we can therefore obtain $E(M_1 \cdot M_2)$ using ciphertexts $E(M_1)$ and $E(M_2)$. Furthermore, $E(M)^1 = E(M)$; $E(M)^0 = 1$. Thus, we present Protocol 2 based on the ElGamal encryption scheme. For ease of explanation, we define $P(A, B)$ as follows: if $B \subseteq A$, $P(A, B) = 1$; otherwise, $P(A, B) = 0$.

Protocol 2. Secure subset protocol for sets whose elements are drawn from a known set.

Inputs. Alice and Bob's input sets $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ $(A, B \subseteq U = \{u_1, \ldots, u_l\})$.

Table 1: Principle of the subset problem for sets whose elements are drawn from a known set.

Set/vector	1	2	3	4	5	6
U	11	12	13	14	15	16
A	11	12		14	15	
V_A^r	1	1	r_{a3}	1	1	r_{a6}
B	11			14	15	
V_B^1	1	0	0	1	1	0
V'	1			1	1	

Output. $P(A, B)$.

(1) Alice generates both her private key sk and its corresponding public key pk. She publishes pk while keeping sk private.

(2) Alice encodes set A as vector $V_A^r = (v_{a1}, \ldots, v_{al})$. She further encrypts V_A^r as

$$E\left(V_A^r\right) = \left(E\left(v_{a1}\right), \ldots, E\left(v_{al}\right)\right) \tag{13}$$

with pk. She sends $E(V_A^r)$ to Bob.

(3) Bob encodes B as $V_B^1 = (v_{b1}, \ldots, v_{bl})$ using 1-0 encoding. He computes

$$E(v) = \prod_{i=1}^{l} \left(E\left(v_{ai}\right)^{v_{bi}}\right) \bmod p = E\left(\prod_{v_{bi}=1} v_{ai}\right). \tag{14}$$

Furthermore, he randomly chooses a number $r_b \in Z_p^*$ $(r_b \neq 0)$ and computes

$$\begin{aligned} E(V) &= (E(v))^{r_b} \bmod p = \left(\alpha^r \bmod p, v \cdot y^r \bmod p\right)^{r_b} \\ &= \left(\alpha^{r \cdot r_b} \bmod p, v^{r_b} \cdot y^{r \cdot r_b} \bmod p\right) = E\left(v^{r_b}\right). \end{aligned} \tag{15}$$

Then, he sends $E(V)$ to Alice.

(4) Alice decrypts $E(V)$ to obtain V. If $V = 1$, then Alice tells Bob that $B \subseteq A$; otherwise, Alice tells Bob that $B \nsubseteq A$.

Because the ciphertexts of random numbers are also random, Alice needs only to encrypt the one components of V_A^r in step (2). That is, Alice needs only to encrypt her own m elements. This reduces the computational complexity.

If $v = 1$, then $V = v^{r_b} = 1$; otherwise, if $v \neq 1$, then $V = v^{r_b}$ is a random number. Thus, the random number r_b does not change the result. In this protocol, all the parties are semihonest and may try to derive information based on the message sequences that they obtained. The random number r_b can randomize the computation of Bob. In step (3), if Bob does not insert the random number r_b, Alice may deduce useful information from B. If $|B| = n$ is small, Alice can obtain the ciphertexts that Bob used to compute the product ciphertext $E(v)$. Thus, Alice obtains the 1-0 vector V_B^1. Furthermore, she obtains set B. Even if $|B| = n$ is sufficiently large, Alice cannot derive Bob's set from $E(v)$, but if B is not a subset of A, Alice may derive which elements are not in set A based on v and V_A^r.

3.3. Security of Protocol 2. In this manuscript, we prove the security of Protocol 2 using the simulation paradigm.

Theorem 3. *Protocol 2, denoted by π, for computing the subset problem is private.*

Proof. To prove this theorem, we show that there exist two simulators S_1 and S_2 such that (12) holds. We first show the construction of S_1.

(1) S_1 receives $(A, P(A, B))$ as the input and randomly chooses a set $B' \subseteq U$ such that $P(A, B') = P(A, B)$. S_1 simulates the execution of Protocol 2 based on A, B'. S_1 encodes sets A and B' to $V_A^r = (v_{a1}, \ldots, v_{al})$ and $V_B^{1\prime} = (v'_{b1}, \ldots, v'_{bl})$, respectively.

(2) S_1 encrypts the vector V_A^r using the public key pk to obtain ciphertexts $E(V_A^r) = (E(v_{a1}), \ldots, E(v_{al}))$.

(3) S_1 first computes

$$E\left(v'\right) = \prod_{i=1}^{l} \left(E\left(v_{ai}\right)^{v'_{bi}}\right) \bmod p = E\left(\prod_{v_{bi}=1} v_{ai}\right). \tag{16}$$

S_1 further chooses a random number r'_b $(r'_b \neq 0)$ and computes $E(V') = (E(v'))^{r'_b} = E(v'^{r'_b})$.

(4) S_1 decrypts $E(V')$ and obtains V'.

Let $S_1(A, P(A, B)) = \{A, V_A^r, E(V_A^r), E(V'), V'\}$ $(V' = P(A, B'))$. In this protocol, $\mathrm{VIEW}_1^\pi(A, B) = \{A, V_A^r, E(V_A^r), E(V), V\}$ $(V = P(A, B))$. Because the ElGamal encryption scheme is semantically secure with the CDH assumption, messages that are encrypted based on this scheme are computationally indistinguishable. This means that the message sequences that Alice obtained in Protocol 2 and the message sequences that S_1 simulated are computationally indistinguishable. As $P(A, B) = P(A, B')$, it follows that

$$\begin{aligned} &\left\{\left(S_1\left(A, f_1\left(A, B\right)\right), f_2\left(A, B\right)\right)\right\}_{A,B \in \{0,1\}^*} \\ &\quad \stackrel{c}{\equiv} \left\{\left(\mathrm{VIEW}_1^\pi\left(A, B\right), \mathrm{OUTPUT}_2^\pi\left(A, B\right)\right)\right\}_{A,B \in \{0,1\}^*}. \end{aligned} \tag{17}$$

Now, let us examine the construction of S_2. Based on the inputs $(B, P(A, B))$, S_2 proceeds as follows:

(1) S_2 chooses a set $A' \subseteq U$ such that $P(A', B) = P(A, B)$, and it simulates the execution of Protocol 2 with sets A' and B. Based on the 1-r encoding and 1-0 encoding, S_2 encodes A' to $V_A^{r\prime} = (v'_{a1}, \ldots, v'_{al})$ and B to $V_B^1 = (v_{b1}, \ldots, v_{bl})$, respectively.

(2) S_2 encrypts the vector $V_A^{r\prime}$ to obtain

$$E\left(V_A^{r\prime}\right) = \left(E\left(v'_{a1}\right), \ldots, E\left(v'_{al}\right)\right). \quad (18)$$

(3) S_2 computes

$$E\left(v'\right) = \prod_{i=1}^{l}\left(E\left(v'_{ai}\right)^{v_{bi}}\right) \bmod p = E\left(\prod_{v_{bi}=1} v'_{ai}\right). \quad (19)$$

Furthermore, S_2 chooses a random number $r_b \neq 0$ and computes $E(V') = E(v')^{r_b} = E(v'^{r_b})$.

(4) S_2 obtains V' from $E(V')$.

Let $S_2(B, P(A,B)) = \{B, V_B, E(V_A), E(V'), V'\}$ ($V' = P(A', B)$). In this protocol, $\text{VIEW}_2^{\pi}(A,B) = \{B, U_B, E(V_A), E(V), V\}$ ($V = P(A,B)$). Because messages that are encrypted using the ElGamal encryption scheme are computationally indistinguishable under the CDH assumption, the message sequences that Alice obtains in Protocol 2 and the message sequences that S_2 is simulating are computationally indistinguishable. As $P(A,B) = P(A',B)$, we have

$$\left\{\left(f_1(A,B), S_2\left(B, f_2(A,B)\right)\right)\right\}_{A,B\in\{0,1\}^*} \overset{c}{\equiv} \left\{\left(\text{OUTPUT}_1^{\pi}(A,B), \text{VIEW}_2^{\pi}(A,B)\right)\right\}_{A,B\in\{0,1\}^*}. \quad (20)$$

□

4. Protocol for Sets with Large Domains

In Protocol 2, we present a subset protocol for sets whose elements are drawn from a known set. Because the communication complexity is linear in $|U|$, this is awkward if $|U|$ is large. Therefore, we construct a secure subset protocol for sets taken from a large domain. Suppose there are two sets A and B ($|A| \geq |B|$). This protocol is efficient with a linear computational complexity in $|A|$, whereas the computational complexity of Kissner's protocol [17] is linear in the product of $|A|$ and $|B|$. If $|A| = |B| = m$, the protocol of Kissner has an $O(m^2)$ computational complexity that is quadratic. Thus, this protocol cannot generally be considered practical for inputs consisting of a large number of data [34]. However, the protocol that we construct reduces the computational cost at the cost of degraded accuracy. That is, our protocol has a negligible false positive, and in Section 4.2, we show how to decrease the false positive.

4.1. Foundations of This Protocol. The following secure subset protocol is based on the Bloom filter [35] and a variant Bloom filter. We present the building blocks of the protocol before giving the idea behind the protocol.

Bloom Filter. A Bloom filter $\text{BF}_B = (w, n, k, H)$ is a vector of w bits that can represent a set B of at most n elements. There are k independent uniform hash functions $H = (h_1, \ldots, h_k)$, and each h_j ($j = 1, \ldots, k$) maps the elements of B to $[1, w]$ uniformly. In this paper, we use $\text{BF}_B[i]$ to denote the bit at index i in BF_B. Initially, all bits in the array are set to 0. To insert an element $b \in B$ into the filter, we compute each hash function $h_j(b)$ and set $\text{BF}_B[h_j(b)] = 1$. After all the elements of B are inserted in BF_B, the Bloom filter BF_B represents set B.

To check if an item b' is in B, we check all components of BF_B that are hashed by b'. If any bit at the components is 0, then $b' \notin B$; otherwise, $b' \in B$ with high probability. However, while a Bloom filter may yield a false positive, it never yields a false negative. That is, if $b \in B$, it must be that $\text{BF}_B[h_j(b)] = 1$ ($j = 1, \ldots, k$); if $b \notin B$, it may be that $\text{BF}_B[h_j(b)] = 1$. The probability of the false positive is

$$P' = \frac{w!}{w^{k(n+1)}} \sum_{i=1}^{w}\sum_{j=1}^{i} (-1)^{i-j} \frac{j^{kn} i^k}{(w-i)!\, j!\, (i-j)!}. \quad (21)$$

According to [36], it is about 2^{-k}. We can choose k based on our practical applications. If the size of BF_B is $w = nk \log_2 e$, the probability that a specific component is one is 1/2 [37].

Suppose Alice has a set A, and Bob has a set B. Sets A and B are taken from an exponentially large domain, and they can compute whether $B \subseteq A$ using the Bloom filter as follows.

Protocol 4. Secure subset protocol for sets taken from an exponentially large domain.

Inputs. Alice and Bob input sets = $\{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_n\}$.

Output. Whether $B \subseteq A$.

(1) Alice and Bob negotiate the parameters w, k, H for their Bloom filters.

(2) Alice and Bob represent sets A and B to Bloom filters BF_A and BF_B, respectively.

(3) Alice sends BF_A to Bob.

(4) Bob checks BF_A using BF_B. If $\text{BF}_B[i] = 1$ ($i = 1, \ldots, w$), then $\text{BF}_A[i] = 1$, $B \subseteq A$; otherwise, $B \nsubseteq A$. He sends the result to Alice.

The above protocol has a low computational complexity based on hash functions. However, when the sets of parties are not taken from an exponentially large domain, Bob can obtain the set A from BF_A using an exhaustive search. Thus, we designed a solution for sets taken from a large, but not exponentially large domain based on the Bloom filter and a variant of the Bloom filter. Before presenting the principles behind this solution, we show the variant of the Bloom filter.

Variant Bloom Filter. The variant Bloom filter is similar to the Bloom filter with a small difference. In the Bloom filter, each component is either 0 or 1 bit, while the component of the variant Bloom filter is either r ($r \in Z_p^* \wedge r \neq 1$) or 1. Similarly, to insert an element $a \in A$ into a variant Bloom filter $\text{VBF}_A = (w, m, k, H)$ of a set A, we compute $h_j(a)$ ($j = 1, \ldots, k$) and set $\text{VBF}_A[h_j(a)] = 1$. After all the elements of A are inserted, we let the remaining components of VBF_A be random numbers other than 1. Because the

Table 2: Idea of the subset problem for sets with a large domain.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
VBF_A	1	1	1	1	r_5	1	1	1	1	1	1	1	r_{13}	r_{14}
BF_B	1	1	0	1	0	0	1	0	1	0	0	1	0	0
BF	1	1		1			1		1			1		

Table 3: Hash table.

	h_1	h_2	h_3
134	9	2	12
189	8	10	4
258	7	1	4
393	6	11	3

variant Bloom filter just changes 0 components to random numbers compared to the Bloom filter, the false-positive probability of the variant Bloom filter is the same as that of the Bloom filter.

Suppose that Alice and Bob have sets A and B, respectively, which are taken from a large domain, and they want to decide whether $B \subseteq A$. Alice represents set A to a variant Bloom filter VBF_A and sends to Bob. Bob represents set B to a Bloom filter BF_B. He computes

$$v = \prod_{\mathrm{BF}_B[i]=1} \mathrm{VBF}_A[i] \quad (i = 1, \ldots, w). \tag{22}$$

If $v = 1$, then $B \subseteq A$; otherwise, $B \nsubseteq A$. This is the idea behind deciding whether $B \subseteq A$. However, Alice and Bob can also solve the subset problem to represent set A to another variant Bloom filter VBF_A^*. Besides VBF_A^* represents the 1 component of VBF_A to 0 components, and it is similar to VBF_A. Bob computes

$$v^* = \sum_{\mathrm{BF}_B[i]=0} \mathrm{VBF}_A^*[i] \tag{23}$$

instead of v. If $v^* = 0$, then $B \subseteq A$; otherwise, $B \nsubseteq A$.

For simplicity, we give a simple example in Table 2 with the variant Bloom filter VBF_A. Suppose that Alice has a set $A = \{134, 189, 258, 393\}$ and Bob has a set $B = \{134, 258\}$. Alice and Bob represent A and B to a variant Bloom filter VBF_A and a Bloom filter BF_B, respectively. Let the length $w = 14$ and hash functions $H = (h_1, h_2, h_3)$ for both VBF_A and BF_B. The hash functions $h_j(x)$ $(j = 1, 2, 3)$ map any value to $[1, 14]$ uniformly, as in Table 3. Alice sets as 1 the 134, 189, 258, and 393 corresponding components in VBF_A and as random numbers other components that are not mapped. Bob sets the components corresponding to 134 and 258 as 1 within BF_B and other components as 0. BF is the vector that is chosen from VBF_A according to the 1 component of BF_B. Thus, if the product v of all the components of BF is 1, then $B \subseteq A$; otherwise, $B \nsubseteq A$.

Because the domain of sets is not exponentially large, these ideas are insufficiently secure for strict applications. Fortunately, we can obtain a secure scheme using homomorphic encryption. The VBF_A method can be implemented with a multiplicatively homomorphic encryption scheme, and the VBF_A^* method can be implemented with an additively homomorphic encryption scheme. Because multiplicatively homomorphic encryption schemes are usually more efficient than additive ones, we represent our protocol in the next subsection with the VBF_A method and the ElGamal encryption scheme that has multiplicative homomorphism.

4.2. The Protocol

Protocol 5. Secure subset protocol for sets within a large domain.

Inputs. Alice inputs set $A = \{a_1, \ldots, a_m\}$, and Bob inputs set $B = \{b_1, \ldots, b_n\}$.

Output. Whether or not $B \subseteq A$.

(1) Alice and Bob negotiate the parameters $w, k, H = \{h_1, h_2, \ldots, h_k\}$ to construct their Bloom filters.

(2) Alice performs the following:

(i) generating her private key sk and its corresponding public key pk;

(ii) representing her set A to a variant Bloom filter VBF_A and encrypts VBF_A,

$$E(\mathrm{VBF}_A) = (E(\mathrm{VBF}_A[1]), E(\mathrm{VBF}_A[2]), \ldots, E(\mathrm{VBF}_A[w])); \tag{24}$$

(iii) sends $E(\mathrm{VBF}_A)$ and pk to Bob.

(3) Bob computes the following:

(i) $E(v) = 1$

For $i = 1$ to w

If $\mathrm{BF}_B[i] = 1$

$$E(v) \longleftarrow E(v) \cdot E(\mathrm{VBF}_A[i]) \tag{25}$$

Return $E(v)$

(ii) Bob randomly chooses $r_b \in Z_p^*$ $(r_b \neq 0, 1)$ and evaluates

$$E(V) = (E(v))^{r_b} \bmod p = (\alpha^r \bmod p, v \cdot y^r \bmod p)^{r_b} = (\alpha^{r \cdot r_b} \bmod p, v^{r_b} \cdot y^{r \cdot r_b} \bmod p) = E(v^{r_b}) \tag{26}$$

He sends $E(V)$ to Alice.

(4) Alice decrypts $E(V)$ and obtains V. If $V = 1$, then $B \subseteq A$; otherwise, $B \subseteq A$.

The ElGamal encryption scheme is multiplicatively homomorphic, and multiplying the ciphertexts is the same as multiplying the corresponding plaintexts. Thus,

$$E\left(\mathrm{VBF}_A[1]\right)^{\mathrm{BF}_B[1]} \cdot E\left(\mathrm{VBF}_A[2]\right)^{\mathrm{BF}_B[2]} \cdots E\left(\mathrm{VBF}_A[w]\right)^{\mathrm{BF}_B[w]} = E\left(\prod_{\mathrm{BF}_B[i]=1} \mathrm{VBF}_A[i]\right) = E(v). \tag{27}$$

In step (3), if $v = 1$, then $v^{r_b} = 1$; otherwise, $v^{r_b} \neq 1$. Thus, r_b does not change the result. However, if Bob does not insert the random number r_b, Alice may obtain more information than she should. This is similar to Protocol 2 (we have omitted details in this paper). To lower the computational complexity, Alice can only encrypt the 1 component in VBF_A as Protocol 2.

Analogously, Alice can also represent A as VBF_A^*, and it can encrypt VBF_A^* with an additively homomorphic encryption scheme, such as the Paillier encryption scheme. She sends $E(\mathrm{VBF}_A^*)$ to Bob. Bob computes

$$E(v^*) = E\left(\sum_{\mathrm{BF}_B[i]=0} \mathrm{VBF}_A^*[i]\right) \tag{28}$$

based on the additive homomorphism and his Bloom filter BF_B. He chooses a random number r_b^* ($r_b^* \neq 0$) to randomize v^* and obtains

$$E(V^*) = E(v^*)^{r_b^*} = \left(g^{v^*} r^N \bmod N^2\right)^{r_b^*} = g^{r_b \cdot v^*} r_b^{*N} = E(r_b^* v). \tag{29}$$

Bob sends $E(V^*)$ to Alice. Alice decrypts $E(V^*)$. If $V^* = 0$, then $B \subseteq A$; otherwise, $B \nsubseteq A$. Thus, they obtain the subset relation.

The successful probability of Protocol 5 is stated in Theorem 6.

Theorem 6. *Protocol 5 will succeed with probability* $P = 1 - 2^{-nk}$.

Proof. According to [38], the probability that a particular component in the Bloom filter is 1 is 1/2. Because the variant Bloom filter is similar to the Bloom filter, the probability is also 1/2. In Protocol 5, Bob chooses nk components of $E(\mathrm{VBF}_A)$ to compute the product $E(v)$ based on the 1 component of BF_B. The product $E(v)$ is $E(1)$ only if all of the nk components that Bob chose from $E(\mathrm{VBF}_A)$ encrypt 1. This shows that $B \subseteq A$; otherwise, $B \nsubseteq A$. Thus, the successful probability of Protocol 5 is $P = 1 - 2^{-nk}$.

The successful probability of Protocol 5 can be increased for important applications. If Alice and Bob choose another set of k different hash functions $H' = \{h_1', h_2', \ldots, h_k'\}$ to reexecute Protocol 5 for the same set A and B, the probability of false positive is also 2^{-nk}. In addition, these two executions are in series. Therefore, the successful probability is $P = 1 - 2^{-nk} \cdot 2^{-nk} = 1 - 2^{-2nk}$. Thus, the successful probability to execute Protocol 5 t times is $P = 1 - 2^{-tnk}$ for the same sets with different hash functions on each occasion. □

Corollary 7. *Secure subset protocol for sets within a large domain is private.*

Based on the Theorem 3, it is easy to prove Corollary 7, and we omit the proof here.

5. Analysis of above Protocols

5.1. Efficiency Analysis. Because the subset protocols of [29, 30] have foundations that are similar to Kissner's protocol [17] and Kissner's protocol is more efficient, we give the efficiency comparisons of computational complexity and communication complexity among the protocol of Kissner, Protocols 2 and 5 in this analysis.

Computational Complexity. In Protocol 2, Alice needs m encryptions and one decryption. While the messages to be encrypted are 1, each ElGamal encryption takes $2 \log p$ modular multiplications. For the ElGamal encryption scheme, each decryption takes $\log p$ modular multiplications. Thus, Alice needs $(2m + 1) \log p$ modular multiplications. Bob computes $E(v)$ using $2n$ modular multiplications, and it requires $2 \log p$ to compute $E(V)$ for Bob. The computational cost of Bob is $2 \log p + 2n$ modular multiplications. Therefore, the computational overhead of Protocol 2 is $(2m+3) \log p + 2n$ modular multiplications (mod p).

Alice encrypts her variant Bloom filter using mk encryptions during the execution of Protocol 5. Because the components are 1, each encryption takes $2 \log p$ modular multiplications. Alice also needs to decrypt $E(V)$. Thus, Alice takes $(2mk + 1) \log p$ modular multiplications. Bob evaluates $E(v)$ using $2nk$ modular multiplications. He computes $E(V)$ based on $E(v)$ taking $2 \log p$ modular multiplications, and $2nk + 2 \log p$ modular multiplications are required during Protocol 5. The total computational cost is $(2mk + 3) \log p + 2nk$ modular multiplications (mod p) in Protocol 5. Because k is a constant, the computational cost is linear in m.

In the protocol proposed by Kissner and Song [17], suppose that Alice has a set A and Bob has a set B and $|A| = m$ and $|B| = n$, where $m \geq n$. Alice needs $m + 1$ encryptions to encrypt her polynomial $p(x)$ in order to obtain the encrypted polynomial δ and 1 decryption to decrypt the ciphertext β. Bob needs m modular exponentiations and m modular multiplications to evaluate the encrypted polynomial δ for every element $b_j \in B$ ($j = 1, \ldots, n$). There are n elements within B, and this takes Bob mn modular exponentiations and mn modular multiplications. For the Paillier encryption scheme, every encryption and decryption require two modular exponentiations, and every modular exponentiation requires $2 \log N$ modular multiplications. This protocol takes $(2mn+4m+8) \log N + mn$ modular multiplications (mod N^2).

Communication Complexity. We can measure the communication complexity using the exchanged bits or the

Table 4: Comparison of secure subset protocols.

	Computation	Communication
Kissner's protocol [17]	$(2mn + 4m + 8) \log N + mn$	3
Protocol 2	$(2m + 3) \log p + 2n$	3
Protocol 5	$(2mk + 3) \log p + 2nk$	3

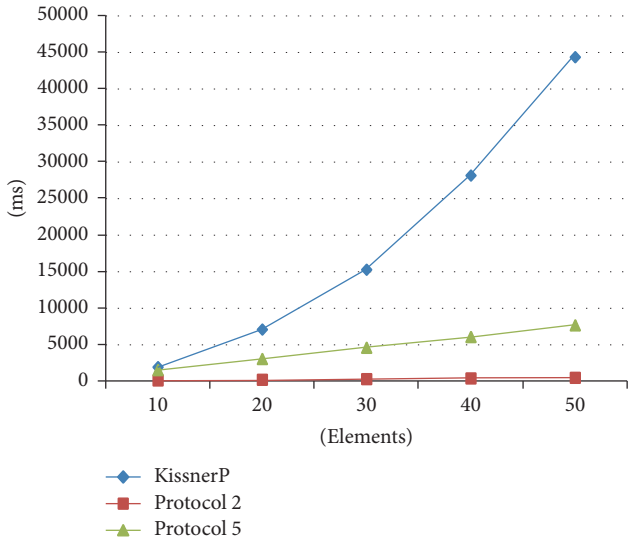

Figure 1: Comparison of the implementation of subset protocols.

communication rounds. In secure multiparty computation, the communicating round is widely used. For Protocols 2 and 5 and Kissner's protocol, each of them involves three communicating rounds.

Based on the above discussion, we summarize the comparison in Table 4. In this table, the modular multiplication for Kissner's protocol is mod N^2, and for our proposed protocols, it is mod p. In order to achieve the same security, $\log N = \log p$.

5.2. Performance Evaluation. Based on the efficiency analysis described above, the experimental setting and the performance evaluation are shown. Our experiment includes the Kissner protocol, Protocols 2 and 5.

In our implementation, we used the Java programming language to implement these protocols, and the experimental environment was as follows: Windows 10 64-bit operating system, with an Intel(R) Core(TM) i3-2100 CPU @ 3.10 GHz processor, and 4 GB of memory. We set both the Paillier scheme modular N and the ElGamal scheme modular p to be 1024 bits.

The experimental results of the subset protocols are shown in Figure 1. "KissnerP" is the protocol proposed by Kissner. Both Protocols 2 and 5 are based on the ElGamal encryption scheme, while Kissner's protocol is based on the Paillier encryption scheme. Because the successful probability of Protocol 5 is $1 - 2^{-nk}$, we instantiate $k = 16$ in the following implementation.

In Figure 1, we showed that both Protocol 2 and Protocol 5 have a linear computational complexity, while the computational complexity of Kissner's protocol is quadratic. Thus, our protocols are efficient and practical for large inputs. However, the probability of a false positive of Protocol 5 is $1 - 2^{-nk}$. Then, n, k can be chosen to be sufficiently large to make the probability negligible.

6. Conclusion

The subset problem is an important building block in secure multiparty computation, and it has many applications in privacy-preserving problems. In this study, we first presented an efficient subset protocol for sets whose elements are drawn from a known set. For sets whose elements are obtained from a large domain, we further designed an approximated and efficient subset protocol. These protocols have a linear computational complexity in the size of the large set. However, all parties of our protocols are semihonest. In future work, it is necessary to solve similar problems that fall under the malicious model.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 61272435 and 61373020), Fundamental Research Funds for the Central Universities (Grant no. 2016TS061), the National Foundation Fund of China (201706870028), Natural Science Foundation of Inner Mongolia (Grant no. 2017MS0602), and University Scientific Research Project of Inner Mongolia (Grant no. NJZY17164).

References

[1] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, 1982.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pp. 1–10, USA, May 1988.

[3] C. Bader, D. Hofheinz, T. Jager, E. Kiltz, and Y. Li, "Tightly-secure authenticated key exchange," in *Theory of cryptography, Part I*, pp. 629–658, Springer Berlin Heidelberg, 2015.

[4] A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," *Journal of Cryptology. The Journal of the International Association for Cryptologic Research*, vol. 25, no. 1, pp. 57–115, 2012.

[5] Y. Wu, L. Huang, X. Wang, and N. Yu, "An extensible cheat-proofing multi-secret sharing scheme with low computation complexity," *Security and Communication Networks*, vol. 7, no. 6, pp. 1042–1048, 2014.

[6] R. Canetti, H. Lin, and O. Paneth, "Public-coin concurrent zero-knowledge in the global hash model," in *Theory of Cryptography*, pp. 80–99, Springer Berlin Heidelberg, 2013.

[7] Y. Lindell and H. Zarosim, "On the feasibility of extending oblivious transfer," in *Theory of Cryptography*, pp. 519–538, Springer Berlin Heidelberg, 2013.

[8] S. Goldwasser, "Multi-party computations: past and present," in *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–6, ACM Press, New york, NY, USA, 1997.

[9] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Proceeding of the nineteenth annual ACM conference on Theory of Computing*, pp. 218–229, Piscataway, NJ, USA, 1987.

[10] O. Goldreich, *Foundations of Cryptography*, vol. 2, Cambridge University Press, 2004.

[11] Y. Zhang and S. Zhong, "An efficient solution to generalized Yao's millionaires problem," *Bulletin of the Belgian Mathematical Society. Simon Stevin*, vol. 20, no. 3, pp. 425–433, 2013.

[12] S. Li, D. Wang, Y. Dai, and P. Luo, "Symmetric cryptographic solution to Yao's millionaires' problem and an evaluation of secure multiparty computations," *Information Sciences. An International Journal*, vol. 178, no. 1, pp. 244–255, 2008.

[13] S. Li, C. Wu, D. Wang, and Y. Dai, "Secure multiparty computation of solid geometric problems and their applications," *Information Sciences. An International Journal*, vol. 282, pp. 401–413, 2014.

[14] R. Fagin, M. Naor, and P. Winkler, "Comparing Information Without Leaking It," *Communications of the ACM*, vol. 39, no. 5, pp. 77–85, 1996.

[15] T. Mitsunaga, Y. Manabe, and T. Okamoto, "Efficient secure auction protocols based on the boneh-goh-nissim encryption," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96-A, no. 1, pp. 68–75, 2013.

[16] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," *International Journal of Information Security*, vol. 11, no. 6, pp. 403–418, 2012.

[17] L. Kissner and D. Song, "Privacy-preserving set operations," in *Advances in cryptology (CRYPTO '05)*, pp. 241–257, Springer, Berlin, Germany, 2005.

[18] M. J. Freedman, C. Hazay, K. Nissim, and B. Pinkas, "Efficient set intersection with simulation-based security," *Journal of Cryptology. The Journal of the International Association for Cryptologic Research*, vol. 29, no. 1, pp. 115–155, 2016.

[19] J. Hong, J. W. Kim, J. Kim, K. Park, and J. H. Cheon, "Constant-round privacy preserving multiset union," *Bulletin of the Korean Mathematical Society*, vol. 50, no. 6, pp. 1799–1816, 2013.

[20] G. Sheng, H. Hou, X. Jiang, and Y. Chen, "A novel association rule mining method of big data for power transformers state parameters based on probabilistic graph model," *IEEE Transactions on Smart Grid*, 2016.

[21] J. Camenisch and R. Chaabouni, "Efficient protocols for set membership and range proofs," in *Advances in cryptology (ASIACRYPT '08)*, pp. 234–252, Springer, Berlin, Germany, 2008.

[22] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology (CRYPTO '02)*, pp. 61–76, Springer, Berlin, Germany, 2002.

[23] R. Cramer, I. Damgard, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology (CRYPTO '94)*, pp. 174–187, Springer, Berlin, Germany, 1994.

[24] J. Camenisch and R. Chaabouni, "Efficient protocols for set membership and range proofs," in *Advances in Cryptology-ASIACRYPT*, pp. 234–252, Springer Berlin Heidelberg, 2008.

[25] M. H. Au, P. P. Tsang, W. Susilo, and Y. Mu, "Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems," in *Topics in Cryptology-CT-RSA*, pp. 295–308, Springer Berlin Heidelberg, 2009.

[26] F. Guo, Y. Mu, W. Susilo, and V. Varadharajan, "Membership encryption and its applications," in *Information Security and Privacy*, pp. 219–234, Springer Berlin Heidelberg, 2013.

[27] F. Guo, Y. Mu, and W. Susilo, "Subset membership encryption and its applications to oblivious transfer," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 7, pp. 1098–1107, 2014.

[28] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advance in Cryptology-EUROCRYPT '99*, pp. 223–238, Springer, LNCS, 1999.

[29] Q. Ye, H. Wang, and J. Pieprzyk, "Distributed private matching and set operations," in *International Conference on Information security practice and experience*, pp. 347–360, Springer Berlin Heidelberg, 2008.

[30] Y. Sang and H. Shen, "Efficient and secure protocols for privacy-preserving set operations," *ACM Transactions on Information and System Security*, vol. 13, no. 1, article 9, 2009.

[31] M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations," *International Journal of Information Security*, vol. 15, no. 5, pp. 493–518, 2016.

[32] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, pp. 784–796, USA, October 2012.

[33] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Workshop on the Theory and Application of Cryptographic Techniques*, pp. 10–18, Springer Berlin Heidelberg, 1984.

[34] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright, "Secure multiparty computation of approximations," *ACM Transactions on Algorithms*, vol. 2, no. 3, pp. 435–472, 2006.

[35] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: An efficient and scalable protocol," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*, pp. 789–800, 2013.

[36] K. Christensen, A. Roginsky, and M. Jimeno, "A new analysis of the false positive rate of a Bloom filter," *Information Processing Letters*, vol. 110, no. 21, pp. 944–949, 2010.

[37] P. Bose, H. Guo, E. Kranakis et al., "On the false-positive rate of Bloom filters," *Information Processing Letters*, vol. 108, no. 4, pp. 210–213, 2008.

[38] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: a survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.