Hindawi
Security and Communication Networks
Volume 2018, Article ID 5841967, 12 pages
https://doi.org/10.1155/2018/5841967

Research Article

Outsourcing Set Intersection Computation Based on Bloom Filter for Privacy Preservation in Multimedia Processing

Hongliang Zhu,[1,2] **Meiqi Chen**,[1,2] **Maohua Sun**,[3] **Xin Liao**,[4] **and Lei Hu**[3]

[1]*Beijing University of Posts and Telecommunications, Beijing, China*
[2]*National Engineering Laboratory for Disaster Backup and Recovery, Beijing, China*
[3]*Capital University of Economics and Business, Beijing, China*
[4]*Hunan University, Hunan, China*

Correspondence should be addressed to Maohua Sun; sunmaohua@cueb.edu.cn

Received 26 September 2017; Accepted 22 February 2018; Published 5 April 2018

Academic Editor: Xinpeng Zhang

With the development of cloud computing, the advantages of low cost and high computation ability meet the demands of complicated computation of multimedia processing. Outsourcing computation of cloud could enable users with limited computing resources to store and process distributed multimedia application data without installing multimedia application software in local computer terminals, but the main problem is how to protect the security of user data in untrusted public cloud services. In recent years, the privacy-preserving outsourcing computation is one of the most common methods to solve the security problems of cloud computing. However, the existing computation cannot meet the needs for the large number of nodes and the dynamic topologies. In this paper, we introduce a novel privacy-preserving outsourcing computation method which combines GM homomorphic encryption scheme and Bloom filter together to solve this problem and propose a new privacy-preserving outsourcing set intersection computation protocol. Results show that the new protocol resolves the privacy-preserving outsourcing set intersection computation problem without increasing the complexity and the false positive probability. Besides, the number of participants, the size of input secret sets, and the online time of participants are not limited.

1. Introduction

Network multimedia comes into fashion in the form of services; there are many methods to protect multimedia data in traditional service mode, such as steganography [1, 2] and data embedding [3]. By providing diversified media services, a new service mode, multimedia computing, has become an attractive technology to generate, edit, process, and search various media contents, like images, videos, audios, graphs, and so on [4]. For purposes of multimedia applications and services based on Internet and mobile Internet, it needs lots of computation resources so as to serve millions of netizens and wireless users, which means a large demand for multimedia cloud computing. Cloud computing is a new computing mode which could provide kinds of data service based on its computational resources. As an important application of cloud computing, outsourcing computation could enable users with narrow computing power to outsource complex function calculations to cloud servers and could guarantee the correctness of outputs and privacy of both inputs and outputs. So in this new multimedia computation mode based on cloud computing, users can store and process distributed multimedia application data without installing multimedia application software in local computer terminals to ease off the load of maintenance and updating. With regard to the large amount of computation of sites, data, and attribute dimensions, we introduce PSI into cloud computing. There is a wide range of applications where Secure Multiparty Computation is introduced into cloud computing considering the privacy-preserving algorithms.

Private Set Intersection (PSI) is an important research branch of Secure Multiparty Computation (SMC), which is a research hotspot in recent years. Privacy-preserving set operation can be described as the situation that multiple

participants wish to complete set intersection computation based on their private secret sets, and they cannot receive additional information other than results after computation. In PSI research model, participants complete secure computation using their private computing resources through mutual communication. Privacy preservation has become a key factor in extending the application of cloud computing, and it is the current research trend. In order to implement PSI in cloud computing successfully to solve the problems mentioned, Privacy-preserving Outsourcing Set Intersection (POSI) is proposed.

1.1. Contributions. The work we have completed in this paper contributes to the study and development of privacy preservation as well as outsourcing computation in several aspects as follows:

(1) We summarize system models of current privacy-preserving technology and propose a system model of privacy-preserving outsourcing computation protocol in cloud computing. It can guarantee the security and correctness of the data.

(2) We study and implement a privacy-preserving set intersection protocol based on GM homomorphic encryption scheme and Bloom filter, and the proposed protocol is proved to be significant.

(3) In detail, the protocol has some characteristics as follows:

(a) The participant encrypts the secret set locally and consigns ciphertexts to the server who completes the outsourcing computation, but the server is unable to know about the participant's secret set because it does not have the private key to decrypt them. So it guarantees security. Participants can check whether one or more items of data are in the intersection.

(b) The protocol does not require sizes of participants' sets being the same as well as public compared to the existing PSI protocols [5–14].

(c) The protocol can implement secure outsourcing computation of more than two participants' secret set intersection without the limitation that participants should be online at the same time, while the existing secure outsourcing computation protocol of set intersection [15] can only solve the situation with two participants online.

(d) The protocol has a lower probability of communication complexity and false positive error verification compared with [15].

(e) The protocol is safe under the semihonest model. We provide a full proof with simulation based security. There are two reasons why we do not design a protocol in the malicious model. (1) The proposed algorithm can be packaged as software. When we use peripheral secure technology to make the software difficult to be tampered with, semihonest model is safe enough. (2) Converting protocol in semihonest model to malicious model is an independent research topic with plenty of achievements currently. If necessary, the algorithm can be converted into one in the malicious model based on the existing research findings.

1.2. Related Work. The following sections describe the research progress of privacy-preserving set intersection and outsourcing computation.

1.2.1. Secure Multiparty Computation. Protocols for Secure Multiparty Computation enable a set of parties to carry out a joint computation on private inputs, without revealing anything but the output. Over the past decade, there has been a major research effort to develop Secure Multiparty Computation. Zhou et al. [16] proposed a secure multiparty subset protocol using the Bloom filter and homomorphic encryption scheme. However, their protocol may yield a false positive. Liu et al. [17] proposed an information-theoretically secure protocol to solve the multiparty millionaires' problem using the vectorization and secret splitting methods; their protocol can resist collusion attacks. Sun et al. [18] proposed a secure outsourcing multiparty computation protocol on lattice-based encrypted data in two-cloud-servers scenario. Their protocol was completely noninteractive between any users, and both of the computation and the communication complexities of each user in our solution were independent of the computing function.

1.2.2. Privacy-Preserving Set Intersection. Privacy-preserving set intersection is a research focus in the field of cryptography. The PSI problem can be described as the situation that multiple participants wish to complete the set intersection computation based on their private secret sets, and they cannot receive additional information other than results after the computation.

According to different implementation principles, we can classify research findings of PSI into the following four types.

(i) The Oblivious Polynomial Evaluation Based Protocols. Oblivious polynomial evaluation is the first method to implement the PSI protocol. Dachman-Soled et al. [5] implemented a PSI protocol in malicious models using Shamir Threshold Secret Sharing technology. The computational complexity of the algorithm is $O(mnk \log n + mk^2 \log^2 n)$, and the communication complexity is $O(nk + mk^2 \log^2 n)$, in which k is the secure parameter, while m and n are the sizes of the participant input sets.

(ii) The Oblivious Pseudorandom Function Based Protocols. At the TCC Conference in 2008, Hazay and Lindell [6] proposed a privacy-preserving set intersection protocol based on the oblivious pseudorandom function. The scheme is safe in the weakly malicious model, which means participants' malicious behavior will be found with a high probability. Later, Hazay and Nissim [7] used zero-knowledge proof and perfectly hiding commitment scheme to implement a privacy-preserving set intersection protocol in malicious

model. The communication complexity of the algorithm is $O(m + n(\log\log m + \sigma))$, and the computational complexity is $O(m + n\sigma)$, in which m and n are the sizes of the two sets. σ is elements' largest binary number of bits in the set. Jarecki and Liu [8] proposed a privacy-preserving intersection protocol under the CRS model based on the Decisional-q-Diffie-Hellman Inversion hypothesis. De Cristofaro and Tsudik [9, 10] proposed a privacy-preserving intersection operation protocol with linear complexity under the semihonest model based on the One-More-Gap-DH hypothesis. Later, De Cristofaro et al. [11] proposed an efficient privacy-preserving intersection operation scheme against malicious attackers based on the DDH hypothesis.

(iii) The Bloom Filter Based Protocols. Bloom filter is a new data structure introduced in recent years, of which the structure is similar to bit-map. Compared to bit-map, Bloom filter saves more space and can quickly judge whether an element is in a set. But there is a certain rate of error recognition in this method. In 2012, Many et al. [12] introduced Bloom filter into the privacy-preserving intersection operations. They used the secure multipart multiplication protocol to get the Bloom filter vector corresponding to the intersection of participants and then get the set intersection. However, the algorithm is insecure because the intersection Bloom filter vector leaked information of each participant's set. In 2013, Dong et al. [13] designed a more efficient privacy-preserving intersection protocol based on Bloom filter, using secret sharing and oblivious transfer. Take the privacy-preserving intersection operation protocol under semihonest model as an example; the scheme Dong et al. [13] proposed requires $2(k + k\log_2 e)n$ times of hash operations and hundreds of public key operations. In 2014, Pinkas and Schneider [14] designed a random confusion Bloom filter to optimize efficiency of the protocol of Dong et al. [13], using oblivious extension protocol.

(iv) The Garbled-Circuit Technology Based Protocols. Using garbled-circuit technology to solve privacy-preserving problems is a common method of Secure Multiparty Computation, but many references in the past suggest that the method is less efficient. In 2012, Huang et al. [19] designed the intersection-specific circuit based on the idea of *"Sort-Compare-Shuffle"* and implemented the privacy-preserving intersection operation protocol using Yao's generic garbled-circuit method. The experimental results of Huang et al. [19] show that the scheme of De Cristofaro and Tsudik [9, 10] is more efficient when the security level is low, and as the security level increases, the scheme of Huang et al. [19] is significantly better than that of De Cristofaro and Tsudik [9, 10] considering efficiency of the program. In 2014, Pinkas and Schneider [14] optimized the GMW scheme using oblivious extension protocol, used the optimized GMW scheme to evaluate the intersecting circuit designed by Huang et al. [19], and implemented a more efficient privacy-preserving intersection operation protocol on Boolean circuits. The computational complexity is $18n\sigma\log n$ times of symmetric encryption operations, while the communication complexity is $O(6nk\sigma\log n)$, in which k is the secure parameter.

1.2.3. Privacy-Preserving Outsourcing Computation. Outsourcing computation in multimedia processing is an emerging technology in recent years. Although the study of privacy-preserving outsourcing computation has just started, it is the current research hot spot.

At the CRYPTO conference in 2010, Gennaro et al. [20] proposed privacy-preserving issues in verifiable computations and designed a privacy-preserving outsourcing computation protocol that can achieve verifiable efficiency based on the homomorphic encryption technology. In 2011, Mohassel [21] designed a noninteractive security outsourcing computation protocol on linear algebraic operations based on homomorphic encryption. In 2013, Parno et al. [22] designed the Pinocchio system which implemented efficient outsourcing computation, but the system did not take into account the privacy-preserving issues of the information input by participants; Schoenmakers et al. [23] designed the Trinocchio system to solve the leakage of Pinocchio system, enabling efficient verifiable secure outsourcing computation. In the same year, Peter et al. [24] designed a secure outsourcing computation protocol for common functional functions, using a dual decryption mechanism scheme with additive homology, and implemented an efficient face recognition system in cloud computing environment based on this protocol. In 2013, Xing et al. [25] constructed a verifiable secure outsourcing computation protocol using the blind product as a matrix product, matrix determinant, and matrix inverse. The security does not depend on any cryptographic assumptions. In 2014, Hu and Tang [26] implemented the secure outsourcing protocol of multiplication on the elliptic curve in the cloud computing environment, which could effectively accelerate the efficiency of signature verification.

Although the PSI protocol has implemented plenty of achievements, they cannot be converted to be used in privacy-preserving set intersection outsourcing computation directly. At present, the research on the privacy-preserving issues in set intersection outsourcing computation has just started, while the findings are still not enough. According to our searching results, Kerschbaum [15] proposed a set intersection secure outsourcing protocol based on SYY homomorphic encryption scheme and Bloom filter. However, the protocol has the following problems: (1) the protocol only solves the secure outsourcing computation of two participants' set intersections, while one of the participants needs to be both common participant and server at the same time; (2) during the process of the protocol, all the participants are required to be online at the same time; (3) there is a high probability of false positive error judgement in the protocol.

1.3. Organizational Structure. In the second session, we introduce secure definition in the scheme and the underlying cryptographic tools. We show the system model in Section 3 and present the privacy-preserving set intersection computation protocol which can be applied into cloud computing in Section 4. In Section 5, we give the correct proof of the protocol, error probability analysis, and security proof as well as efficiency analysis and comparison. Finally, we summarize

prospects of our protocol's application in multimedia processing based on cloud computing in the Conclusion.

2. Background

2.1. Secure Model and Secure Definition. Since the protocol proposed in this paper belongs to one kind of the Secure Multiparty Computation protocols, we use secure models and secure definitions of Secure Multiparty Computation protocols.

Participants of Secure Multiparty Computation are classified into honest participants, semihonest participants, and malicious participants. During the implementation of the protocol, honest participants completely comply with the protocol, with no provision of false data, leakage, eavesdropping, and suspension of the protocol; semihonest participants will finish each step following the requirements of the implementation without behaviors mentioned earlier, but they will keep all the information they collected in order to judge secret messages of other participants; malicious participants completely ignore the requirements of the protocol. They may provide false data, leak all the information they collect, eavesdrop, or even suspend protocols.

The semihonest model is safe and widely used in Secure Multiparty Computation. The model can be intuitively understood as the situation that if a semihonest participant can directly use their input and output of protocols to obtain any information he can reach in the implementation of the protocol by a separate simulation of the entire protocol implementation process, it can be guaranteed in the protocol that the input is private. If a computation protocol can be simulated like this, participants cannot obtain valuable information from the execution of the protocol, and such protocol is safe.

Definition 1 (private computation under semihonest model). In the implementation of protocol Π, the information that participants P_1 and P_2 obtain is recorded as

$$\begin{aligned} \mathrm{VIEW}_1^{\Pi}(x, y) &= \left(x, r^1, m_1^1, m_2^1, \ldots, m_t^1\right) \\ \mathrm{VIEW}_2^{\Pi}(x, y) &= \left(x, r^2, m_1^2, m_2^2, \ldots, m_t^2\right). \end{aligned} \tag{1}$$

In the equations, r^i represents the random number P_i generates and m_j^i represents the jth message P_i receives. After the protocol ends, the output of participant P_i is recorded as $\mathrm{OUTPUT}_i^{\Pi}(x, y)$. We can see that in fact $\mathrm{OUTPUT}_i^{\Pi}(x, y)$ is a part of $\mathrm{VIEW}_i^{\Pi}(x, y)$.

As for the deterministic function f, we can say that protocol Π computes f under the semihonest model privately if and only if probability polynomial time algorithms S_1 and S_2 exist, and it conforms to the equations:

$$\begin{aligned} \left\{S_1(x, f_1(x, y))\right\}_{x,y\in\{0,1\}^*} &\overset{c}{\equiv} \left\{\mathrm{VIEW}_1^{\Pi}(x, y)\right\}_{x,y\in\{0,1\}^*} \\ \left\{S_2(y, f_2(x, y))\right\}_{x,y\in\{0,1\}^*} &\overset{c}{\equiv} \left\{\mathrm{VIEW}_2^{\Pi}(x, y)\right\}_{x,y\in\{0,1\}^*} \end{aligned} \tag{2}$$

for $|x| = |y|$.

2.2. GM Homomorphic Encryption. A high-level description of Gentry's scheme is as follows. The scheme is based on identifying ideals I in polynomial quotient rings $Z[x]/(f(x))$ (with $^\circ(f) = n$) with euclidean lattices $L_I \subseteq R^n$ by mapping eachresidue polynomial $r(x) = a_0 + \cdots + a_{n-1}x^{n-1}$ to its vector of coefficients $(a_0, \ldots, a_{n-1})$. Gentry calls these objects ideal lattices. Ideal lattices provide additive and multiplicative homomorphisms modulo a public key ideal. We obtain an encryption procedure Encrypt such that $\mathrm{Encrypt}(x_1) + \mathrm{Encrypt}(x_2) = \mathrm{Encrypt}(x_1 + x_2)$ and $\mathrm{Encrypt}(x_1) \cdot \mathrm{Encrypt}(x_2) = \mathrm{Encrypt}(x_1 \cdot x_2)$. Therefore, any circuit C with efficient description can be evaluated homomorphically. However, this somewhat fully homomorphic scheme (SWHE) is not perfect. Due to the noisy nature of the scheme, with each homomorphic gate evaluation the noise term in the partial result grows. After the evaluation of only a logarithmic depth circuit, the decryption fails to recover the correct result. To make the scheme work, Gentry uses a number of tricks. He introduces a reencryption procedure called Recrypt that takes a noisy ciphertext and returns a noise-reduced version. In a brilliant move, Gentry manages to obtain Recrypt again from the SWHE scheme by simply homomorphically evaluating the decryption circuit using encrypted secret key bits on the noisy ciphertext. To make this work, the SWHE needs to be able to handle circuits that are deeper than its own decryption circuit before the level of noise becomes too large. SWHE schemes with this property are called bootstrappable.

2.3. XOR Secret Sharing. The secret publisher converts his secret s into n subsecrets and sends them to other participants. The secret sharing scheme is called a (t, n) threshold secret sharing scheme when they can recover the secret s if and only if at least t participants contribute their specific subsecrets.

When the threshold $t = n$, the XOR secret sharing scheme proposed by Ishai et al. [27] is widely used. The details are as follows.

Participants. The participants are secret publisher D and n participants $P_1, P_2, \ldots, P_n$.

Input. The input is secret s that secret publisher D inputs.

Secret Sharing. (1) Secret publisher generates $n - 1$ random numbers $r_1, r_2, \ldots, r_{n-1}$, and the length of each is $|s|$.

(2) Secret publisher calculates the nth secret:

$$r_n = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus s. \tag{3}$$

$\{r_1, r_2, \ldots, r_n\}$ compose subsecrets of s.

(3) As for $i = 1, 2, \ldots, n$, the secret publisher sends subsecret r_i to P_i.

Secret Recovery. When it is necessary to recover the secret s, n participants $P_1, P_2, \ldots, P_n$ contribute their own subsecrets and do the following operation:

$$s = r_1 \oplus r_2 \oplus \cdots \oplus r_n. \tag{4}$$

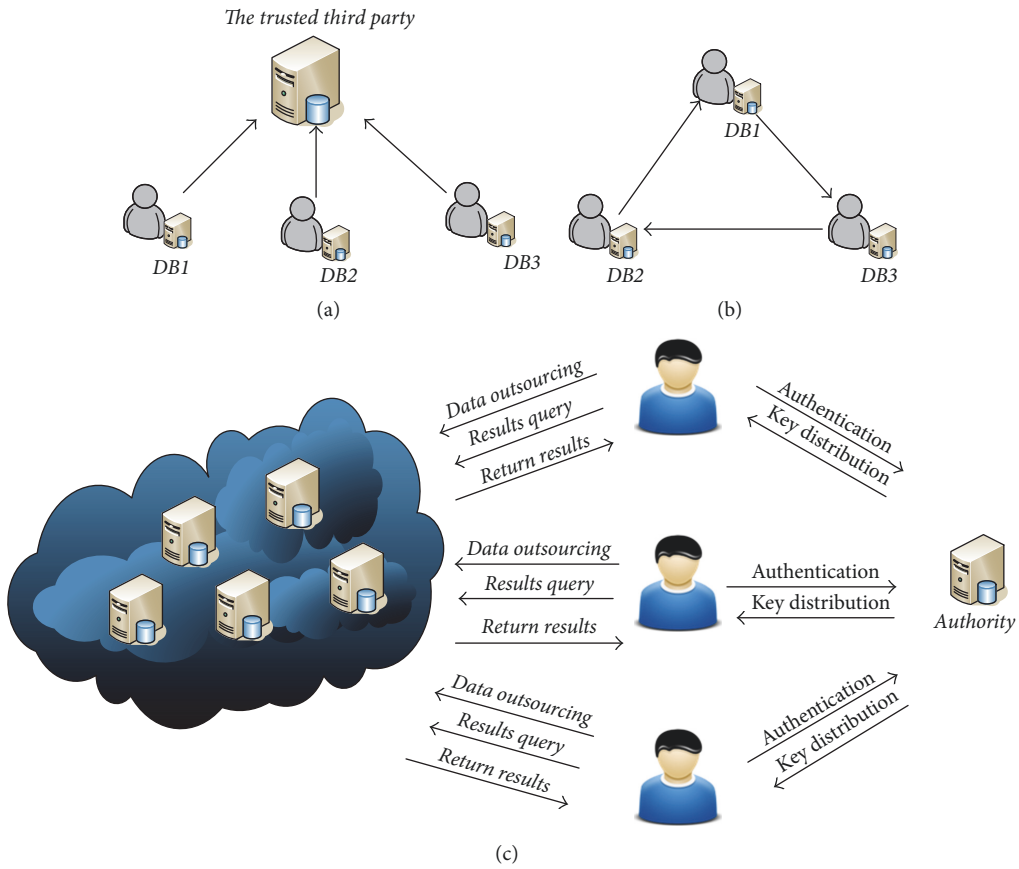

Figure 1: Secure Multiparty Computation model.

2.4. Bloom Filter. The Bloom filter [28] set is a data structure used to judge whether an element is in a set. A Bloom filter contains several hash functions hash_i $(i = 1, 2, \ldots, k)$ and a Bloom filter set BF. When building a Bloom filter set, use the hash function first to map the data x which is to be inserted to the kth position of BF, and then set the data on those positions to 1. When all the data is inserted, the Bloom filter set is completed. When verifying whether a data y is in a set, use the hash function first to map y to the kth position of BF. If the values of these k data bits are all 1, there is a great possibility that y is in the set; otherwise it is not in for sure.

3. System Model

A trusted third party is a model that solves the privacy-preserving problem in distributed computation, as shown in Figure 1(a). However, it is difficult to find a completely credible third party in real life, so this system model is rarely used at present. Currently in the field of Secure Multiparty Computation, a widely used system model is shown in Figure 1(b). It needs a number of participants to complete the secure computation of a certain function through information interaction instead of a trusted third party. To achieve the privacy-preserving outsourcing computation, we can not use the model of Figure 1(a) directly because a completely trusted third party does not exist; nor can we use Figure 1(b) model directly, because a lot of computation is consigned to the server.

The system model we use is shown in Figure 1(c). Although a completely trusted third party does not exist, the authority (for example, an authoritative digital certificate authority) does exist. Before the protocol is formally conducted, the participant will be authenticated by the authority first. If the audit passes, the authority sends the system key to participants. In the process of the protocol, participants use the public key to encrypt their own secret sets and consign the ciphertexts to the server. The server computes all the ciphertexts it takes over and saves them. Then every participant may request to verify whether one or some of data is in the intersection of the sets at any time.

Then we describe the behavior pattern of all participants and the server after the authentication in the system model applied in cloud computing shown in Figure 1(c). In this system model, the problem to be solved can be described as follows: m participants $P_1, P_2, \ldots, P_m$ hold secret messages separately, and the participant completes the operation $f(s_1, s_2, \ldots, s_m)$ by leasing a server with powerful computing resources. In terms of security, the participant wishes others not to be informed of other useful information except the results after completing the computation; the server is unable to know the participants' secret messages $s_1, s_2, \ldots, s_m$, and the server can not know the result $f(s_1, s_2, \ldots, s_m)$.

We divide the information interaction between participants and servers into three stages: preprocessing, outsourcing computation, and results query. In the preprocessing stage, behavior of participants and servers is as follows:

$$\begin{aligned} P_i : S_i &\xrightarrow{\pi} \pi(S_i) \\ P_i &\xrightarrow{\pi(S_i)} \text{Server}. \end{aligned} \tag{5}$$

As for each participant P_i, the first step is converting S_i to $\pi(S_i)$ through a certain operation π locally and then sending $\pi(S_i)$ to the server. The operation π should be unidirectional; otherwise the server will be informed of the participant's secret message.

In the outsourcing computation stage, the server converts all the outsourced data of participants to data sources $F(\pi(S_1), \pi(S_2), \ldots)$ of the results query stage through a certain operation F. We can use the following equation to represent the server's behavior pattern:

$$\begin{aligned} \text{Server} : &\{\pi(S_1), \pi(S_2), \ldots\} \\ &\xrightarrow{F} F(\pi(S_1), \pi(S_2), \ldots). \end{aligned} \tag{6}$$

In the results query stage, the behavior pattern of the inquirer P_i and the server is as follows:

$$\begin{aligned} P_i &\xrightarrow{Q_i} \text{Server} \\ \text{Server} : \{F(\pi(S_1), \pi(S_2), \ldots), Q_i\} &\xrightarrow{\Delta} R_i \\ \text{Server} &\xrightarrow{R_i} P_i \\ P_i : R_i &\xrightarrow{\Phi} \Phi(R_i). \end{aligned} \tag{7}$$

It means that the participant constructs query data Q_i and sends it to the server first. The server conducts operation Δ using the result $F(\pi(S_1), \pi(S_2), \ldots)$ of the previous stage and Q_i as input and then gets the result R_i and sends it to P_i. Participants P_i conduct a certain decryption Φ to R_i and gets the final result $\Phi(R_i)$. The correctness requirement of this model is $\Phi(R_i) = f(s_1, s_2, \ldots, s_m)$.

4. Privacy-Preserving Set Intersection Outsourcing Computation Protocol

In this section, we design the set intersection secure outsourcing protocol in accordance with three stages of preprocessing, outsourcing computation, and results query. We state in this section that participants and authorities have completed authentication and key distribution in Figure 1(c).

The protocol uses the following symbols: P represents all participants, P_i represents the ith participant, and m represents the number of participants. The secret set of participant P_i is S_i, and its size is represented by $|S_i|$. GF_i represents the Bloom filter set of participant P_i and $\text{GF}_i(j)$ represents the jth element in the Bloom filter set. The number of elements in Bloom filter is t while the number of hash functions used in the process of forming Bloom filters is k. CGF_i represents ciphertexts corresponding to Bloom filter set of P_i. The length of ciphertexts in XOR secret sharing is n, and the length of ciphertexts in GM encryption algorithm is l.

4.1. Preprocessing. In the preprocessing stage, the participant generates Bloom filter set corresponding to his private secret set. In order to reduce the probability of false positives, participants share data of secret sets to the k elements of Bloom filter, using XOR secret sharing. We can get the positions of the k elements by hashing. In order to achieve privacy preservation, participants use the GM algorithm to do encryption operations on their respective Bloom filter sets and send them to server. The preprocessing protocol process is as shown in Algorithm 1.

After the previous computation, participant P_i gets encrypted Bloom filter set CBF_i, and P_i needs to send CBF_i to server to complete the data outsourcing.

4.2. Outsourcing Computation. After the previous stage ends, server receives the encrypted Bloom filter sets CBF_i ($i = 0, 1, \ldots, m$) that all the participants send. Server does the following operations in the outsourcing computation stage:

```
for (j = 0; j < tn; j + +)
{
    CBF(j) = ∏_{i=0}^{m} CBF_i(j)
}
```

4.3. Results Query. In the results query stage as shown in Algorithm 2, participants query whether one of more data is in the intersection.

5. Theoretical Analysis

In this section, we analyze the correctness, error probability, security, and performance of the protocol and compare the results with the existing ones.

5.1. Correctness

Theorem 2. *When the participant is able to construct the Bloom filter successfully, the proposed set intersection secure outsourcing protocol is correct.*

Proof. $\forall q \in I$, then, for $i = 1, 2, \ldots, m$, there is

$$q \in S_i \text{ as well as } \bigoplus_{j=0}^{k-1} \text{BF}_i\left(\text{hash}_j(q)\right) = q. \tag{8}$$

Because GM algorithm has a characteristic of XOR homomorphies, for $j = 0, 1, \ldots, k-1$, there is

$$\text{CBF}\left(\text{hash}_j(q)\right) = \text{Enc}\left(\bigoplus_{i=1}^{m} \text{BF}_i\left(\text{hash}_j(q)\right)\right). \tag{9}$$

```
Participants: P_i
Input: the input sets of P_i: S_i
System parameters: k hash functions hash_i (i = 1, 2, ..., k)
for (j = 0; j < t; j + +)
    BF_i(j) = NULL;
    //In the initial state, the Bloom filter set
    //of participants is empty
for (j = 0; j < |S_i|; j + +)
{
    π = NULL;
    for (δ = 0; δ < k; δ + +)
    {
        if (BF_i(hash_δ(S_i(j))) == NULL)
        {
            π = hash_δ(S_i(j));
BF_i(hash_δ(S_i(j))) = Random(0, 1)^n;
            //generate random numbers
            //with the length of n;
        }
        if (π == NULL)
            Return error1; //error
        else
BF_i(π) = S_i(j) ⊕ BF_i(hash_1(S_i(j))) ⊕
                BF_i(hash_2(S_i(j))) ⊕
                ··· ⊕ BF_i(hash_k(S_i(j)))
    }
}
ε = 0;
for (j = 0; j < t; j + +)
{
    if (BF_i(j) == NULL)
        BF_i(j) = Random(0, 1)^n;
    for (μ = 0; μ < n; μ + +)
    {
        ε = ε + 1;
        CBF_i(ε) = Enc(BF_i(j)_μ);
        //BF_i(j)_μ represents the μth
        //bit in BF_i(j);
    }
}
```

Algorithm 1: Preprocessing protocol.

When participants query whether q is in the set intersection, for $j = 0, 1, \ldots, k-1$, there is

$$\mathrm{QBF}\left(\mathrm{hash}_j\left(q\right)\right) = 1. \tag{10}$$

So participants use extended oblivious transfer protocol to get the set ABF, and there is

$$\mathrm{ABF}\left(\mathrm{hash}_j\left(q\right)\right) = \mathrm{CBF}\left(\mathrm{hash}_j\left(q\right)\right). \tag{11}$$

We can know from (9) and (11) that

$$\mathrm{Dec}\left(\mathrm{ABF}\left(\mathrm{hash}_j\left(q\right)\right)\right) = \bigoplus_{i=1}^{m} \mathrm{BF}_i\left(\mathrm{hash}_j\left(q\right)\right). \tag{12}$$

So

$$\begin{aligned}\pi &= \bigoplus_{\delta=0}^{k-1} \mathrm{Dec}\left(\mathrm{ABF}\left(\mathrm{hash}_\delta\left(q\right)\right)\right)\\ &= \bigoplus_{\delta=0}^{k-1}\left(\bigoplus_{i=1}^{m} \mathrm{BF}_i\left(\mathrm{hash}_\delta\left(q\right)\right)\right)\\ &= \bigoplus_{i=1}^{m}\left(\bigoplus_{\delta=0}^{k-1} \mathrm{BF}_i\left(\mathrm{hash}_\delta\left(q\right)\right)\right) = \bigoplus_{1}^{m} q.\end{aligned} \tag{13}$$

So when m is even, $\pi = \{0\}^n$; when m is odd, $\pi = q$.

Similarly, if $q \notin I$, when m is even, $\pi \neq \{0\}^n$; when m is odd, $\pi \neq q$.

The proof is finished. □

Participants: P_i, Server
Input: P_i inputs query set $Q = \{q_1, q_2, \ldots, q_\tau\}$; Server inputs the Bloom filter set CBF of intersection I.
Output: P_i gets query results $R = \{r_1, r_2, \ldots, r_\tau\}$, if $r_i = 1$, then data $q_i \in I$; otherwise $q_i \notin I$.
Step 1. Server generates random Bloom filter set RBF and Bloom filter set pair PBF following the steps below.

```
for (j = 0; j < tn; j++)
{
    RBF(j) = Random(0, 1)^l;
    PBF(i) = (RBF(j), CBF(j));
}
```

Step 2. P_i generates query Bloom filter set QBF according to query set Q.

```
for (j = 0; j < tn; j++)
    QBF(j) = 0;
for (j = 0; j < Q; j++)
{
    for (δ = 0; δ < k; δ++)
    {
        QBF(hash_δ(q_j)) = 1;
    }
}
```

Step 3. P_i and Server implement oblivious transfer protocol OT_l^t. P_i is set to be a Receiver and Server is set to be a Sender. The input of P_i is QBF while the input of Server is PBF. After the oblivious transfer protocol is completed, P_i gets the set ABF. When $\mathrm{QBF}(j) = 0$, $\mathrm{ABF}(j) = \mathrm{RBF}(j)$; when $\mathrm{QBF}(j) = 1$, $\mathrm{ABF}(j) = \mathrm{CBF}(j)$.
Step 4. P_i checks whether each element of Q is in the intersection following the steps below.

```
for (j = 0; j < τ; j++)
{
    π = {0}^n;
    for (δ = 0; δ < k; δ++)
    {
        π = π ⊕ Dec(ABF(hash_δ(q_j)))
    }
    if (m(mod2) == 0)
    {
        if (π == {0}^n)
            r_j = 1;
        else
            r_j = 0;
    }
    else
    {
        if (π == q_j)
            r_j = 1;
        else
            r_j = 0;
    }
}
```

Algorithm 2: Results query stage.

5.2. Error Probability

Theorem 3. *The probability that participant P_i constructs Bloom filter set based on XOR secret sharing successfully is*

$$P = 1 - p_1^h \times \left(1 + O\left(\frac{h}{p_1}\sqrt{\frac{\ln t - h \ln p_1}{t}}\right)\right) \tag{14}$$

in which $p_1 = 1 - (1 - 1/t)^{h(|S_i|-1)}$.

Proof. The necessary and sufficient condition that participants P_i are unable to map their data x of their secret sets to the Bloom filter set when building a Bloom filter set based on XOR secret sharing is that the k positions in Bloom filter data x gets after being mapped by k hash functions are occupied. And the necessary and sufficient condition of general Bloom filters with false positive authentication is that all the k positions data y gets after being mapped by k hash functions are all set to one. Thus, the probability of participant P_i being unable to construct a Bloom filter set based on XOR secret sharing is the same as the probability of a generic Bloom filter set with false positive. From [29] we can see that the probability is $p' = p_1^h \times (1 + O((h/p_1)\sqrt{(\ln t - h \ln p_1)/t}))$, in which $p_1 = 1 - (1 - 1/t)^{h(|S_i|-1)}$. Thus, the probability of participant P_i successfully constructing a Bloom filter set based on XOR secret sharing is $P = 1 - p_1^h \times (1 + O((h/p_1)\sqrt{(\ln t - h \ln p_1)/t}))$.

The proof is finished. □

Theorem 4. *After the participant constructs Bloom filter successfully, the false positive error probability is* $(1/2)^n$.

Proof. $\forall x \notin I$, when the result is $x \in I$, then there will be false positive verification. Consider the following matrix:

$$Z = \begin{bmatrix} \mathrm{BF}\_x_1^1 & \mathrm{BF}\_x_1^2 & \cdots & \mathrm{BF}\_x_1^k \\ \mathrm{BF}\_x_2^1 & \mathrm{BF}\_x_2^2 & \cdots & \mathrm{BF}\_x_2^k \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{BF}\_x_m^1 & \mathrm{BF}\_x_m^2 & \cdots & \mathrm{BF}\_x_m^k \end{bmatrix} \tag{15}$$

in which $\mathrm{BF}\_x_i^j = \mathrm{BF}_i(\mathrm{hash}_j(x))$.

If the number of participants m is even, $\bigoplus_{i=1}^m (\bigoplus_{j=1}^k \mathrm{BF}\_x_i^j) = \{0\}^n$. We can know from the process of construction that the probability is $(1/2)^n$; if it is odd, $\bigoplus_{i=1}^m (\bigoplus_{j=1}^k \mathrm{BF}\_x_i^j) = x$; the probability is also $(1/2)^n$.

In conclusion, the probability of false positive error is $(1/2)^n$ after the participant constructs the Bloom filter successfully in this scheme.

The proof is finished. □

5.3. Security

Theorem 5. *Assuming that the underlying GM homomorphic encryption scheme and the OT protocol are secure under the semihonest model, the proposed set intersection security outsourcing protocol safely implements the outsourcing computation of the participant's secret set under the semihonest model.*

Proof. The protocol proposed in this paper is asymmetric, which means only the participant is informed of the result. So

$$\begin{aligned} & f(S_1, S_2, \ldots, S_m, Q) \\ & \quad \stackrel{\text{def}}{=} \left(f_P(S_1, S_2, \ldots, S_m, Q), f_S(S_1, S_2, \ldots, S_m)\right) \\ & \quad \stackrel{\text{def}}{=} \left(f_P(S_1, S_2, \ldots, S_m, Q), \Lambda\right) \end{aligned} \tag{16}$$

in which Λ means empty strings and π means the proposed security outsourcing protocol. The security analysis is performed from the server view and the participant view, respectively, as follows.

Server View. First analyze the situation where the serve is attacked. During the execution of the protocol π, the server's view is

$$\begin{aligned} & \text{view}_s^{\pi}(S_1, S_2, \ldots, S_m, Q) \\ & = \left\{\Lambda, r^s, \text{CBF}_1, \text{CBF}_2, \ldots, \text{CBF}_m, \text{CBF}, \text{PBF}, \text{view}_s^{\text{OT}}\right\} \end{aligned} \tag{17}$$

in which Λ means output of the server and $\{\text{CBF}_1, \text{CBF}_2, \ldots, \text{CBF}_m, \text{CBF}, \text{PBF}, \text{view}_s^{\text{OT}}\}$ means the view of the server in the protocol.

Create the simulator Sim_S as follows. Sim_S receives the output Λ of the server and simulates behavior of the server in the protocol. First, Sim_S generates even-distributed random toss r^{Sim} and generates $\text{CBF}_1', \text{CBF}_2', \ldots, \text{CBF}_m'$ in accordance with the following rules:

$$\begin{aligned} & \text{for } (i = 0; j < m; i++) \\ & \quad \text{for } (j = 0; j < t; j++) \\ & \quad\quad \text{CBF}_i'(j) \leftarrow \text{Enc}(\text{Random}(0,1)^n) \end{aligned}$$

Then Sim_S calculates CBF' according to the following rules:

$$\begin{aligned} & \text{for } (j = 0; j < t; j++) \\ & \quad \text{CBF}'(j) = \textstyle\prod_{i=0}^{m} \text{CBF}_i'(j) \end{aligned}$$

Then Sim_S generates intermediate information PBF' of the results query stage:

$$\begin{aligned} & \text{for } (j = 0; j < t; j++) \\ & \quad \text{PBF}'(i) = (\text{Random}(0,1)^l, \text{CBF}'(j)) \end{aligned}$$

Finally, Sim_S simulates the oblivious transfer protocol of results query stage, using PBF' as input and Λ as output, and generates the view $\text{view}_{\text{Sim}}^{\text{OT}}$.

After the whole simulation completes, Sim_S outputs the simulation view:

$$\begin{aligned} & \text{view}_{\text{sim}_S}^{\pi} = \left\{\Lambda, r^{\text{Sim}}, \text{CBF}_1', \text{CBF}_2', \ldots, \text{CBF}_m', \text{CBF}', \text{PBF}', \right. \\ & \quad \left. \text{view}_{\text{Sim}}^{\text{OT}}\right\}. \end{aligned} \tag{18}$$

r^{Sim} and r^s are distributed uniformly, so

$$r^{\text{Sim}} \stackrel{c}{\equiv} r^s. \tag{19}$$

It is assumed that the GM encryption scheme is safe under the semihonest model, and the introduction of random numbers in the GM scheme makes ciphertexts of the GM encryption scheme indistinguishable, so

$$\begin{aligned} & \left\{\text{CBF}_1', \text{CBF}_2', \ldots, \text{CBF}_m', \text{CBF}', \text{PBF}'\right\} \\ & \quad \stackrel{c}{\equiv} \left\{\text{CBF}_1, \text{CBF}_2, \ldots, \text{CBF}_m, \text{CBF}, \text{PBF}\right\}. \end{aligned} \tag{20}$$

In the results query stage, as for the oblivious transfer protocol, the input information PBF' of Sim_S and the server's input information PBF have indistinguishability, and we assume that the underlying OT protocol in the semihonest model is safe, so

$$\text{view}_{\text{Sim}}^{\text{OT}} \stackrel{c}{\equiv} \text{view}_s^{\text{OT}}. \tag{21}$$

In conclusion,

$$\text{view}_{\text{sim}_S}^{\pi}(S_s, S_c) \stackrel{c}{\equiv} \text{view}_s^{\pi}. \tag{22}$$

Participant View. Now we analyze the situation where participant P_1 is attacked. The participant view in protocol π is

$$\begin{aligned} & \text{view}_{P_1}^{\pi}(S_1, S_2, \ldots, S_m, Q) \\ & \quad = \left\{S_1, Q, R, r^P, \text{CBF}_1, \text{QBF}, \text{ABF}, \text{view}_P^{\text{OT}}\right\} \end{aligned} \tag{23}$$

in which S_1 and Q are the input information of P_1, while R is the output information of P_1.

And $\{r^P, \text{CBF}_1, \text{QBF}, \text{ABF}, \text{view}_P^{\text{OT}}\}$ is the information view generated by P_1 in the protocol.

We describe construction of simulator Sim_P as follows. Sim_P receives the input information S_1 and the output R of P_1 and simulates the behavior of the protocol P_1 in the protocol. First, Sim_P generates a uniform distribution of random toss r^{Sim} and generates the encrypted Bloom filter set CBF_1' following steps of the protocol according to inputs. In the results query stage, Sim_P generates the query Bloom filter QBF' following steps of the protocol using Q as input. Sim_P simulates and generates ABF' according to output R (see Algorithm 3).

Finally, Sim_P simulates the oblivious transfer protocol in the results query stage using QBF' as input and ABF' as output and generates the view $\text{view}_{\text{Sim}}^{\text{OT}}$.

After the whole protocol simulation is completed, Sim_P outputs the simulation view

$$\begin{aligned} & \text{view}_{\text{sim}_P}^{\pi} \\ & \quad = \left\{S_1, Q, R, r^{\text{Sim}}, \text{CBF}_1', \text{QBF}', \text{ABF}', \text{view}_{\text{Sim}}^{\text{OT}}\right\}. \end{aligned} \tag{24}$$

r^{Sim} and r^P are distributed uniformly, so

$$r^{\text{Sim}} \stackrel{c}{\equiv} r^P. \tag{25}$$

```
for (j = 0; j < t; j + +)
    PABF(j) = NULL;
for (j = 0; j < |Q|; j + +)
{
    π = NULL;
    if (R(j) == 1)
    {
        for (δ = 0; δ < k; δ + +)
        {
            if (PABF_i(hash_δ(Q(j))) == NULL)
            {
                π = hash_δ(Q(j));
                PABF_i(hash_δ(Q(j))) = Random(0, 1)^n;
            }
        }
PABF_i(π) =
    S_i(j) ⊕ (⊕_{μ=1}^{k} PABF_i(hash_μ(Q(j))))
    }
}
for (j = 0; j < t; j + +)
{
    if (PABF_i(j) == NULL)
        PABF_i(j) = Random(0, 1)^n;
        ABF'(j) = Enc(PABF_i(j));
}
```

Algorithm 3

It is assumed that the GM encryption scheme is safe under the semihonest model, and the introduction of random numbers in the GM scheme makes ciphertexts of the GM encryption scheme indistinguishable, so

$$\left\{\mathrm{CBF}_1', \mathrm{QBF}', \mathrm{ABF}'\right\} \stackrel{c}{\equiv} \left\{\mathrm{CBF}_1, \mathrm{QBF}, \mathrm{ABF}\right\}. \quad (26)$$

In the process of generating the query Bloom filter, according to steps of the protocol, when inputs are the same, there will be identical query Bloom filter sets, so $\mathrm{QBF} = \mathrm{QBF}'$.

In the results query stage, as for the oblivious transfer protocol, the input information QBF' of Sim_S and the server's input information QBF are the same. The output ABF' of Sim_P and the input ABF of participants are indistinguishable. We assume that the underlying OT protocol under the semihonest model is safe, so

$$\mathrm{view}_{\mathrm{Sim}}^{\mathrm{OT}} \stackrel{c}{\equiv} \mathrm{view}_{P}^{\mathrm{OT}}. \quad (27)$$

In conclusion,

$$\mathrm{view}_{\mathrm{sim}_P}^{\pi} \stackrel{c}{\equiv} \mathrm{view}_{P_1}^{\pi}. \quad (28)$$

So we can say that the proposed protocol under semihonest model is safe.

The proof is finished. □

5.4. Performance Analysis. Now we analyze the efficiency of the protocol from two aspects: computational complexity and communication complexity.

5.4.1. Computational Complexity. As for each participant P_i, the hash operation is performed $k|S_i|$ times during the preprocessing stage, and the GM encryption operates kn times; during the results query stage, it is hashed $k|Q_i|$ times and does OT_l^t operation once, while the GM decryption operation is performed at most $kn|Q_i|$ times. As for the server, the ciphertext multiplication operation is performed tmn times in the outsourcing computation stage in all; OT_l^t is performed once in the results query stage.

When implementing OT_l^t using extended OT technology [27], Receiver needs to perform 2λ times of public key operations and $1.44hs$ times of hash operations. Sender needs to perform λ times of public key operations and $1.44hs$ times of hash operations, in which λ represents the security parameter of extended OT protocol. When using the GM algorithm, the encryption operation needs to perform one modular multiplication while the decryption operation needs to perform one modular multiplication, and the multiplication of ciphertexts requires one modular multiplication. Therefore, the participant in this scheme needs to implement the public key algorithm $kn + kn|Q_i| + 2\lambda$ times and the hash algorithm $1.44ks$ times; the server needs to implement the public key algorithm $tmn + \lambda$ times and the hash algorithm $k|S_i| + k|Q_i| + 1.44ks$ times.

5.4.2. Communication Complexity. At the end of preprocessing stage, each participant sends tl bits data to the server, and the server receives tlm bits data in all. In the results query stage, the participant and the server transfer $2\lambda t$ bits of data, respectively.

6. Comparison

There are a number of different parameters due to the fact that existing privacy-preserving set intersection outsourcing protocols are different from privacy-preserving set intersection protocols in principle. Parameters are instantiated in order to compare efficiency of protocols. Common parameters: the sizes of the participant sets are all s. $k = 8$, $t = sk/\ln^2 2$, and $l = 100$. In the proposed protocol, the query set $Q = S$; the Kerschbaum scheme [15] can only achieve security outsourcing computation of two participants, so $m = 2$ in this scheme; the length of ciphertexts in XOR secret sharing is $n = 8$. And, in the Kerschbaum scheme, $\beta = 8$. Construction and query of Bloom filter are based on Dong's open source experimental model [13], which uses SHA-1 to instantiate hash functions; OT protocol uses classical Naor-Pinkas scheme [30].

After summarizing and comparing the existing algorithms in Figure 2 and Table 1, we can see the following. (1) The computational complexity and the communication complexity are lower than that of Huang's scheme and similar to that of Dong's. Also it is slightly lower than Kerschbaum's. (2) The false positive probability is higher than that of Huang's, but the same as Dong's and Kerschbaum's scheme. (3) The proposed algorithm solves the problem of privacy preservation in outsourcing computation considering the cloud computing environment; in the Kerschbaum scheme, a participant is needed to be the server, so it is a traditional

Table 1: Comparison of protocols in applicability.

Comparison of algorithms	Number of participants	System model	Online necessary or not
Dong et al. [13]	2	Traditional Secure Multiparty Computation model	Yes
Huang et al. [19]	2	Traditional Secure Multiparty Computation model	Yes
Kerschbaum [15]	2	Traditional Secure Multiparty Computation model	Yes
The proposed algorithm	≥2	Outsourcing computation model	No

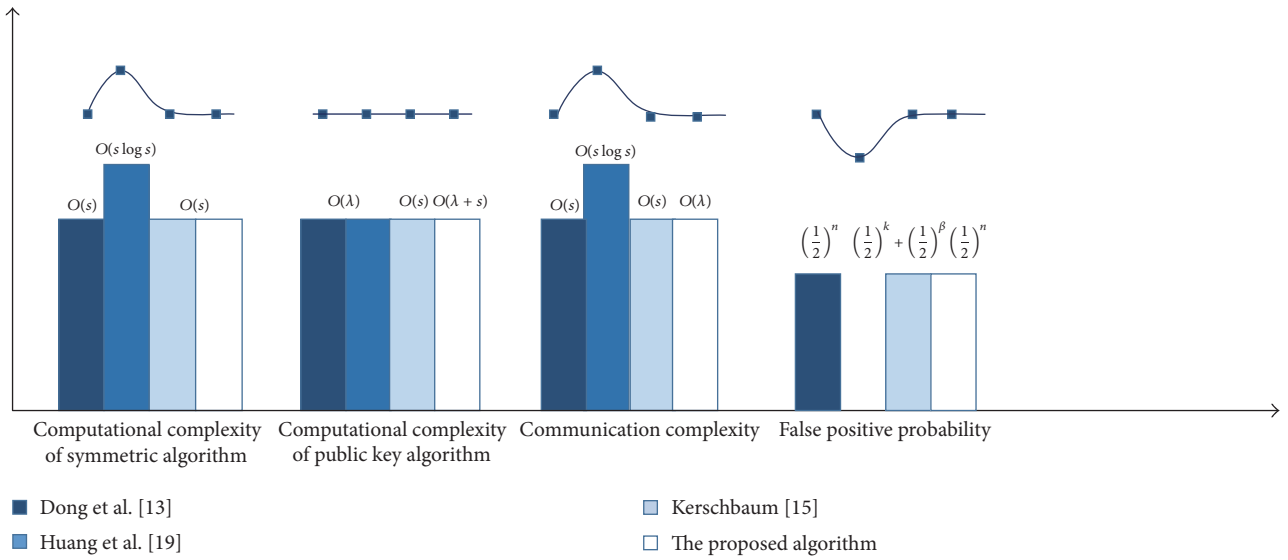

Figure 2: Comparison of protocols in complexity and false positive probability.

secure computation model; in Huang's and Dong's scheme, traditional secure computation model is used to solve PSI problem. (4) The proposed algorithm can solve the secure outsourcing computation with two or more participants, while the others can only deal with the situation of two. (5) It does not need all the participants being online at real time in the proposed algorithm, while the others need them to be online in order to complete the computation at the same time.

In the comparison, we can know from Figure 2 that our algorithm can deal with privacy preservation in outsourcing computation without increasing computational complexity, communication complexity, and false positive probability. In addition, as shown in Table 1, it has great advantages considering the limit of some factors, such as the number of participants, sizes of inputs, and requirement of being online. So, to a large extent, the proposed algorithm improves the solution of privacy preservation in cloud computing.

7. Conclusion

In this paper, we propose a privacy-preserving outsourcing computation system model which can be used in multimedia processing based on cloud computing to solve security and correctness problems. Based on this model, we design a privacy-preserving set intersection outsourcing computation protocol based on GM homomorphic encryption scheme and Bloom filter. The results show that the proposed protocol achieves privacy preservation in the outsourcing computation without increasing computational complexity, the communication complexity, and the false positive probability. And the protocol does not limit the number of participants, the input secret sizes, and whether participant is online in real time. Obviously, not only is the method proposed suitable for multimedia processing, but also it can be used for cloud computing, distributed computing, Internet of things, virtual property transactions, and so on.

In the next few years, we will continue designing the privacy-preserving set intersection outsourcing computation protocol and extending its application in cloud computing. We will focus on the further improvement of efficiency of the algorithm, as well as the design of algorithms against malicious attackers.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant no. 61402162), Hunan Provincial Natural Science Foundation of China (Grant no. 2017JJ3040),

Applied Sci-Tech R&D Special Fund Program of Guangdong Province (no. 2015B010131007), and National High Technology Research and Development Program of China (863 Program) (nos. 2015AA016005, 2015AA017201).

References

[1] X. Liao, G. Chen, and J. Yin, "Content-adaptive steganalysis for color images," *Security and Communication Networks*, vol. 9, no. 18, pp. 5756–5763, 2016.

[2] X. Liao, J. Yin, S. Guo, X. Li, and A. K. Sangaiah, "Medical JPEG image steganography based on preserving inter-block dependencies," *Computers and Electrical Engineering*, 2017.

[3] X. Liao, Z. Qin, and L. Ding, "Data embedding in digital images using critical functions," *Signal Processing: Image Communication*, vol. 58, pp. 146–156, 2017.

[4] G. Han, W. Que, G. Jia, and L. Shu, "An efficient virtual machine consolidation scheme for multimedia cloud computing," *Sensors*, vol. 16, no. 2, article 246, 2016.

[5] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," in *Proceedings of the International Conference on Applied Cryptography and Network Security. ACNS 2009*, vol. 5536, pp. 125–142, 2009.

[6] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Proceedings of the Theory of Cryptography Conference. TCC 2008*, vol. 4948, pp. 155–175, 2008.

[7] C. Hazay and K. Nissim, "Efficient set operations in the presence of malicious adversaries," in *Proceedings of the International Workshop on Public Key Cryptography – PKC 2010*, vol. 6056, pp. 312–331, 2010.

[8] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," in *Proceedings of the TCC 2009: Theory of Cryptography*, vol. 5444, pp. 577–594, 2009.

[9] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Proceedings of the International Conference on Financial Cryptography and Data Security: FC 2010*, vol. 6052, pp. 143–159, 2010.

[10] E. De Cristofaro and G. Tsudik, "Experimenting with fast private set intersection," in *Proceedings of the International Conference on Trust and Trustworthy Computing: TRUST 2012*, vol. 7344, pp. 55–73, 2012.

[11] E. De Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in malicious model," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security: ASIACRYPT 2010*, vol. 6477, pp. 213–231, 2010.

[12] D. Many, M. Burkhart, and X. Dimitropoulos, "Fast private set operations with sepia," Tech. Rep. 345, Mar 2012.

[13] 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.

[14] B. Pinkas and T. Schneider, "Faster private set intersection based on OT Extension," in *Proceedings of the 23rd UNSENIX Security Symposium*, 2014.

[15] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2012*, pp. 85-86, Republic of Korea, May 2012.

[16] S. Zhou, S. Li, J. Dou, Y. Geng, and X. Liu, "Efficient secure multiparty subset computation," *Security and Communication Networks*, vol. 2017, Article ID 9717580, 2017.

[17] X. Liu, S. Li, X. Chen, G. Xu, X. Zhang, and Y. Zhou, "Efficient Solutions to Two-Party and Multiparty Millionaires' Problem," *Security and Communication Networks*, vol. 2017, pp. 1–11, 2017.

[18] Y. Sun, Q. Wen, Y. Zhang, H. Zhang, Z. Jin, and W. Li, "Two-cloud-servers-assisted secure outsourcing multiparty computation," *The Scientific World Journal*, vol. 2014, Article ID 413265, 2014.

[19] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *Proceedings of the NDSS Symposium 2012*, 2012.

[20] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: outsourcing computation to untrusted workers," in *Advances in cryptology—CRYPTO*, vol. 6223 of *Lecture Notes in Computer Science*, pp. 465–482, Springer, 2010.

[21] P. Mohassel, "Efficient and secure delegation of linear algebra," Cryptology ePrint Archive, Report 2011/605, 2011.

[22] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proceedings of the 34th IEEE Symposium on Security and Privacy, SP 2013*, pp. 238–252, usa, May 2013.

[23] B. Schoenmakers, M. Veeningen, and N. de Vreede, "Trinocchio, privacy-preserving outsourcing by distributed verifiable computation," Cryptology ePrint Archive, Report 2015/480, 2015.

[24] A. Peter, E. Tews, and S. Katzenbeisser, "Efficient outsourcing multiparty computation under multiple keys," Cryptology ePrint Archive, Report 2013/013, 2013.

[25] H. Xing, P. DingYi, T. ChunMing, and D. S. Wong, "Verifiable and secure outsourcing of matrix calculation and its application," *SCIENTIA SINICA Informationis*, vol. 43, pp. 842–852, 2013.

[26] X. Hu and C. M. Tang, "Securely outsourcing computation of point multiplication on elliptic curves in cloud computing," *Journal of Hunan University of Science and Technology*, vol. 29, pp. 119–123, 2014.

[27] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Proceedings of the Annual International Cryptology Conference: CRYPTO 2003*, vol. 2729 of LNCS, pp. 145–161, Springer.

[28] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[29] 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.

[30] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of the SODA '01 Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pp. 448–457, SIAM, Washington, DC, USA.