Soft Comput (2018) 22:2517–2526
https://doi.org/10.1007/s00500-017-2506-x

METHODOLOGIES AND APPLICATION

A privacy-preserving fuzzy interest matching protocol for friends finding in social networks

Xu An Wang[1,2] · **Fatos Xhafa**[3] · **Xiaoshuang Luo**[1] · **Shuaiwei Zhang**[1] · **Yong Ding**[2]

Published online: 7 February 2017

Abstract Nowadays, it is very popular to make friends, share photographs, and exchange news throughout social networks. Social networks widely expand the area of people's social connections and make communication much smoother than ever before. In a social network, there are many social groups established based on common interests among persons, such as learning group, family group, and reading group. People often describe their profiles when registering as a user in a social network. Then social networks can organize these users into groups of friends according to their profiles. However, an important issue must be considered, namely many users' sensitive profiles could have been leaked out during this process. Therefore, it is reasonable to design a privacy-preserving friends-finding protocol in social network. Toward this goal, we design a fuzzy interest matching protocol based on private set intersection. Concretely, two candidate users can first organize their profiles into sets, then use Bloom filters to generate new data structures, and finally find the intersection sets to decide whether being friends or not in the social network. The protocol is shown to be secure in the malicious model and can be useful for practical purposes.

Communicated by V. Loia.

✉ Xu An Wang
wangxazjd@163.com

Fatos Xhafa
fatos@cs.upc.edu

1 Key Laboratory of Information and Network Security, Engineering University of Chinese Armed Police Force, Xi'an, People's Republic of China

2 Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, People's Republic of China

3 Department of Computer Science, Universitat Politècnica de Catalunya, Barcelona, Spain

1 Introduction

Social network is a multi-function platform for members to communicate with each other conveniently and establish social relationship. There exist many kinds of social network services, such as instant messaging, photograph sharing, news discussion, and instant financial paying. At present, Facebook, Twitter, Myspace, QQ, WeChat, and many other social network platforms have all become extremely popular around the world. It is estimated that the record number of sharing contents everyday on Facebook is as high as 4 billion and that number for twitter is about 340 million. Furthermore, due to the fast development of mobile social networks, people could publish information about videos, photographs, articles, and so on at any time and any place, which makes communication and sharing with friends very convenient.

Usually social network users tend to build their online social network from real social friends, such as relatives, colleagues, and classmates. (Chen et al. 2013; Hu et al. 2011). But this might not fully satisfy the requirements of online communication. For example, football fans would like to pay attention to news and techniques around football. Thus they would have more preferences on setting up a social group on discussing about football. Therefore, social networks should provide a platform for people to communicate and add someone as friends according to their will on purely on personal knowledge or personal relations. However, some sensitive information, such as personal attributes and locations, could be abused, which can contribute to serious concerns. In order to preserve the privacy of information sharing, we design a

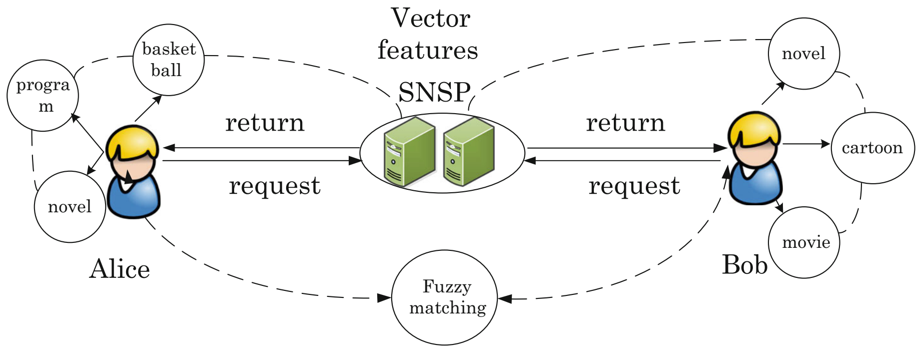

Fig. 1 Interest matching for two social network users

scenario and adapt some measures to deal with the related security problems.

Let us consider the following scenario which is shown in Fig. 1: Alice and Bob are strangers in a social network. Alice finds that Bob's interests are similar with her interests, so she wants to be a friend to Bob. However, Alice would not like to leak her privacy to other people when facing strangers. Therefore, an access for them to enable interaction in the social network is needed.

Let us consider additionally the following scenario: Alice and Bob are strangers in a social network. Alice finds that Bob's interests are similar with himself, so she wants to be a friend to Bob. In some cases, Alice would not like to leak his privacy to other people when facing strangers. So, a secure access for them to interact is a basic requirement.

A way to deal with this problem can be as follows: Actually, both parties cannot interact with each other directly in social networks. They need to get the help of SNSP (social network service provider) to transmit information. Alice and Bob cannot communicate independently, while both of them can communicate with SNSP, according to the following steps.

1. Assume that Alice wants to inquire friendship to someone who has common interests with her and makes a request to SNSP.
2. When SNSP obtains the request of Alice, it will select some users in accordance with the conditions and make a set of interest for Alice. If Alice likes reading novels, watching cartoons and seeing movies, the set will be $V_{Alice} = \{novel, cartoon, movie\}$. Similarly, the interest set for Bob will be $V_{Bob} = \{basketball, novel, program\}$.
3. Alice and Bob will take a fuzzy matching, each with its own set. If succeed, they will be friends. If not, their friendship in the social network would not be established.

1.1 Paper's contribution

In this paper, we present a variant of *Private Set Intersection* (PSI) and design a secure protocol of fuzzy interest matching for friends finding in social networks. This variant is shown secure in the malicious model based on Bloom filter and homomorphic encryption. We then present an outsourced computation scheme in which the client outsources his complex computation tasks to a trusted powerful service provider P. This is each a commonplace in the cloud computing where service providers can provide large number of resources and powerful computation ability (Meriem et al. 2014; Wang et al. 2016; Zhu and Yang 2015; Guo and Xu 2015; Cristina et al. 2014; Xia et al. 2015; Fu et al. 2015, 2016; Ren et al. 2016a, b).

Compared with the state of the art, our protocol has some advantages that are drawn by evaluating its security and performance. Our protocol has the following properties:

- *Being more secure*

  - The PSI variant and the outsourced scheme are provably secure in the malicious model. Therefore, our protocols based on PSI are against malicious adversary. The previous works presenting secure schemes in the malicious model are Dachman-Soled et al. (2009), De Cristofaro et al. (2010), Freedman et al. (2004), Hazay and Lindell (2008), Hazay and Nissim (2010), Jarecki and Liu (2009), and Kissner and Song (2005).
  - Our scheme is secure in the standard model (without random oracles). The only cryptographic assumption is the decisional composite residuosity.
  - Our scheme is client set-size-independent. Although we set the upper bound on the size of the client set, this is not related to client set size. We check the server for the client set elements to get the intersection. Therefore, the client does not need to meet the requirements of false positive.

- *Being more efficient*

  - The PSI variant has linear complexity $O(m)$, where m denotes the size of Bloom filter. The outsourced protocol is very efficient and the only expensive cost is hash function, which can achieve linear complexity $O(n)$, where

n represents the number of set elements, whereas previous protocols can achieve linear complexity $O(v + w)$ (De Cristofaro and Tsudik 2010), where v and w also represent the number of elements in the set.

– We encrypted Bloom filters by Paillier cryptosystem with additive homomorphic property. The server only performs modular multiplication rather than expensive operations, such as modular exponentiations. In order to reduce the computation task of the client, we outsource complex computation load to the service provider P.
– We used Bloom filter that is based on hash function to store elements of both sides. Note that the hash functions are not full domain hash functions.

• *Supporting homomorphic computation*

Homomorphic encryption allows specific types of computations to be carried out on ciphertexts and generates an encrypted result. In this paper, we utilize an additive homomorphic public-key cryptosystem—Paillier encryption. What the server operates are ciphertexts, which can guarantee the security of the client. Practically, we want the server to perform additive operation for plaintexts, but it could not be likely to execute on plaintexts. To achieve our goal, the server only performs modular multiplication that can compute what we need to get the intersection. This is a main advantage of our protocol.

1.2 Paper organization

The remainder of this paper is organized as follows: In Sect. 2, we refer to related work on this research topic. Some preliminary concepts, definitions, and terminology are given in Sect. 3. In Sect. 4, we present our proposal constructed by Paillier cryptosystem and its additive homomorphic property, where we also prove its security and analyze its efficiency, and in Sect. 5 we present the protocol. Further in Sect. 6, we present an outsourced computation scheme. We summarize this paper's contributions and give an outlook to future work in Sect. 7.

2 Related work

In *Eurocrypt'04*, Freedman et al. (2004) firstly presented protocols based on homomorphic encryption and balanced hashing for both semi-honest and malicious environments. Since then, there have been proposed a large number of private set intersection protocols. These protocols can be classified into four kinds.

1. *Based on oblivious polynomial evaluation* Oblivious polynomial evaluation is an effective way to construct private set intersection. It does not need to disclose the coefficients of a polynomial. The main idea is considering the elements set as the roots of the polynomial. One can evaluates it on the other party's set elements obliviously. The protocol presented by Freedman et al. (2004) is based on oblivious polynomial evaluation. Cheielewski and Hoepman (2008) considered that the construction proposed by Freedman is incorrect and proved that a client can obtain server's elements on the condition that they do not have the same elements. These protocols are used by generic zero-knowledge proofs and secure in the semi-honest model and malicious model. Dachman-Soled et al. (2009) do not use generic zero-knowledge proofs and presented an improved construction secure against malicious adversaries. Hazay and Nissim (2010) also put forward private set intersection protocols based on random oracle model in secure and malicious environments, respectively.
2. *Based on oblivious pseudo-random functions* The main idea of oblivious pseudo-random functions is that the client can evaluate a keyed and pseudo-random function on its put. But the key is controlled by the server. The goal is to compute the intersection on the pseudo-random functions of the set elements. Then, the client gets the result of the pseudo-random function obliviously. Hazay and Lindell (2008) presented the first protocol, Jarecki and Liu (2010) and De Cristofaro and Tsudik (2010) improved these protocols later.
3. *Based on bloom filters* In 2012, Many et al. (2012) present a secure multiplication protocol based on Bloom filters and each party obtains an intersection. But the intersection Bloom filter leaks out information of other parties. Kerschbaum Kerschbaum (2012) constructs an outsourced private set intersection protocol using Gold wasser–Micali homomorphic encryption. But the protocol has high communication overhead. Dong et al. (2013) proposed two protocols based on the semi-honest and malicious model, which are much faster. Debnath and Dutta (2015) proposed two constructions of PSI-CA, one is secure in the standard model and the other is secure in the random oracle model under the decisional Diffie–Hellman assumption against malicious adversary. However, the ideas of their construction are different with the prior work.
4. *Based on blind signature* The idea of these protocols based on blind signature is to present or aggregate signatures set elements, hash the result of the verification, and compute the intersection on the hashes. The advantage of using blind signatures is that the client could obtain a signature without disclosing it. In 2009, Camenisch and Zaverucha (2009) presented a private set intersection protocol that requires the input set must be signed and certified by a trusted party. Dachman-Soled et al.

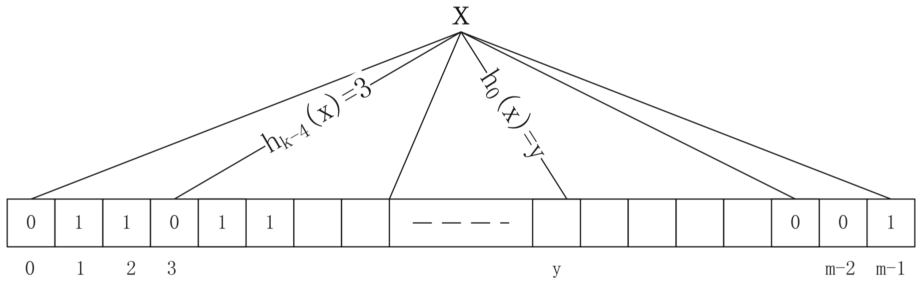

Fig. 2 Store the hashes of x by Bloom filter

(2009), presented protocols secure against semi-honest adversaries and have linear complexity, which is the most efficient protocol at present. Along this line, De Cristofaro et al. (2010) extended the protocols to the malicious model.

3 Preliminaries and notations

3.1 Fuzzy private matching

In *Eurocrypt04*, Freedman, Nissim, and Pinkas first introduced the private fuzzy matching problem. The problem is defined for two parties, and each of them owns a set, respectively. Every set has T elements. The one party computes the fuzzy set intersection of two sets. If there exist at least t similar elements in the intersection, then the two sets match successfully. The process to compute the intersection should guarantee the security of the other party's set and that would not leak out any information. At the same time, the other party would not learn anything about the content.

Let us suppose that the vectors of client's set are $C = \{a_1, a_2, \ldots, a_T\}$ and the server's set is $S = \{s_1, s_2, \ldots, s_T\}$. When there are at least t common elements between C and S, we denote $C \approx_t S$.

3.2 Bloom filters

A Bloom filter (Bloom 1970) is a compact data structure supporting for data storage and membership querying, as shown in Fig. 2. It is an array of m bits that can represent a set $S = \{s_1, s_2, \ldots, s_n)$ with at most n elements. A Bloom filter couples with a set of k independent uniform hash functions $H = (h_0, h_1, \ldots, h_{k-1}\}$ such that each h_i maps elements to index numbers over the range $[0, m-1]$ uniformly. We give a *Create Algorithm* for client in Fig. 3. Further, we use BF_s to denote a Bloom filter that encodes the set S , and use $BF_s(i)$ to denote the bit at index i in BF_s. For example, in Fig. 2, when initializing, all bits in the array are set to 0. To insert an element $x \in S$ into the filter, the element is hashed using the k hash functions to get k index numbers. The bits at all these indexes in the bit array are set to 1, set $BF_s[h_i(x)] = 1$ for $0 \leq i \leq k-1$. To check whether an element y is in S, y

```
Create Algorithm(n, m, C, $BF_C$)
Input: n, m, a set C
Output: a Bloom filter $BF_C$
1  for all $x \in$ C
2      for $i$ = 0 to m-1
3          $BF_C[i]$=0
4      End for
5      for $i$ =0 to k-1
6          j = hash($x$)
7          if $BF_C[j]$==0 then
8              $BF_C[j]$=1
9          End if
10     End for
11 End for
```

Fig. 3 Create Algorithm

is hashed by the k hash functions and all locations y hashes are checked. If any of the bits at the locations is 0, y is not in S; otherwise, y is probably in the set S.

However, a Bloom filter could have false positive in practice. It is possible that y is not in the set S, but all locations of $BF_S[h_i(y)]$ are all equal to 1. A particular bit in the Bloom filter is set to 1, the probability of which is $p = 1-(1-1/m)^{kn}$. Bose et al. (2008) proved the upper bound of the false-positive probability is as follows:

$$\epsilon = p^k \cdot \left(1 + O\left(\frac{k}{p}\sqrt{\frac{\ln(m) - k\ln(p)}{m}}\right)\right)$$

which is negligible in k. Given T elements added into Bloom filter and the maximum false-positive rate 2^{-k}, the necessary size of Bloom filter m can be set to $\frac{Tk}{\ln^2 2}$.

3.3 Paillier encryption scheme

In this section, we briefly introduce the Paillier encryption scheme. The Paillier encryption scheme (Paillier 1999) is a probabilistic public-key algorithm, which is composed of key generation, encryption, and decryption as follows:

1. Key generation: Choose two large prime numbers p and q randomly such that

   $$gcd(pq,(p-1)(q-1))=1$$

   compute

   $$n=pq,\lambda=lcm(p-1,q-1)$$

   where lcm stands for the least common multiple. Select random integer g by checking the existence of the following modular multiplicative inverse:

   $$\mu=(L(g^{\lambda}(mod n^2)))^{-1}(\mathrm{mod} n)$$

   where function L is defined as $L(u)=\frac{\mu-1}{n}$. Note that the notation a/b denote the quotient of a divided by b. Finally, the public (encryption) key is (n,g) and the private (decryption) key is (λ,μ).
2. Encryption: Let m be a message to be encrypted and $m\in Z_n$. Select random r where $r\in Z_n^*$, compute the ciphertext $c=g^m\cdot r^n(\mathrm{mod} n^2)$.
3. Decryption: Let c be the ciphertext to decrypt, where $c\in Z_{n^2}^*$. Compute the plaintext messages as $m=L(c^{\lambda}(mod n^2))\cdot\mu \bmod n$.

Homomorphic properties: Given two ciphertexts $E(m_1,PK)=g^{m_1}r_1^n(\mathrm{mod} n^2)$ and $E(m_2,PK)=g^{m_2}r_1^n(\mathrm{mod}\ n^2)$, where r_1 and r_2 are randomly chosen from Z_n^*, we have

$$\begin{aligned}&E(m_1,pk)\cdot E(m_2,pk)\\&=(g^{m_1}r_1^n)(g^{m_2}r_2^n)(\mathrm{mod} n^2)=g^{m_1+m_2}(r_1r_2)^n(\mathrm{mod} n^2)\\&=E(m_1+m_2,pk)\end{aligned}$$

Paillier security: The Paillier encryption scheme was proved semantic secure against chosen-plaintext attacks (IND-CPA) under the decisional composite residuosity (DCR) assumption. In our scheme, we mainly encrypt using 0 or 1. For the same 0 or 1, choosing different random r could be encrypted into different numbers, which benefit our construction to some extent.

4 Security model

Before introducing our new protocols, we briefly discuss the security models of adversaries for two-party protocols (Oded 2009). Security of protocols in the real model is evaluated by comparison to an ideal model. In the ideal model, client and server submit their input to a trusted third party that can execute PSI protocols and returns the final result to the client. Goldreich gives definitions of the semi-honest model and the malicious model.

In the malicious model, a malicious adversary can behave arbitrary feasible deviated from the specified program. We consider the real model in which a real protocol is executed. A malicious party may follow an arbitrary feasible strategy which gets an auxiliary input. Particularly, the malicious party may refuse to participate or abort the execution at any point in time, which is different from the semi-honest party. But we can simulate the same behavior of every adversary in the ideal model.

5 Fuzzy matching protocol based on PSI

We exploit the properties of Paillier encryption to construct our scheme, which includes five stages. Our fuzzy matching protocol based on PSI is introduced in the following. The similar elements between two sets are obtained by PSI protocol and then the result of fuzzy matching would be achieved.

5.1 Proposed scheme based on the malicious model

1. Encryption: First, the client will generate private key and public key of Paillier encryption scheme. The client encrypts BF_C with public key.

   $$E(BF_C)=[E(BF_C(0)),\ldots,E(BF_C(m-1))]$$

   The client will transfer $E(BF_C)$ to the server directly.
2. Computation: The server receives $E(BF_C)$ from client and computes the following formulas according to Paillier encryption's homomorphic properties, that is:

   $$\begin{aligned}&E(BF_C)\cdot g^{BF_s}\\&=E([BF_C[0]+BF_S[0],\ldots,\\&BF_C[m-1]+BF_S[m-1]])\\&=E(BF_C+BF_S)=E(BF_{C\cup S})\end{aligned}$$

Therefore, we can find that the client and the server's two Bloom filters are added together (see Fig. 4). Then, the server generates $r=[r_0,\ldots,r_{m-1}]\in Z_q^m$ randomly and computes $E(r(BF_{C\cup S}/2))$. That is:

$$\begin{aligned}&E(r(BF_{C\cup S}/2))\\&=E[(r_0[BF_C[0]+BF_S[0]-2],\ldots,\\&r_{m-1}[BF_C[m-1]+BF_S[m-1]-2])]\\&=E[BF_{C\cup S}]\cdot E(-2)^r\\&=E(BF_C)\cdot g^{BF_S}\cdot E(-2)^r\end{aligned}$$

Next, the server will transfer $E(r(BF_{C\cup S}/2))$ to the client. In the real execution, the server could compute the final result of $E(r(BF_{C\cup S}/2))$ rather than store the intermediate results, such as $E(BF_{C\cup S})$ or $E(BF_C)\cdot g^{BF_s}$.

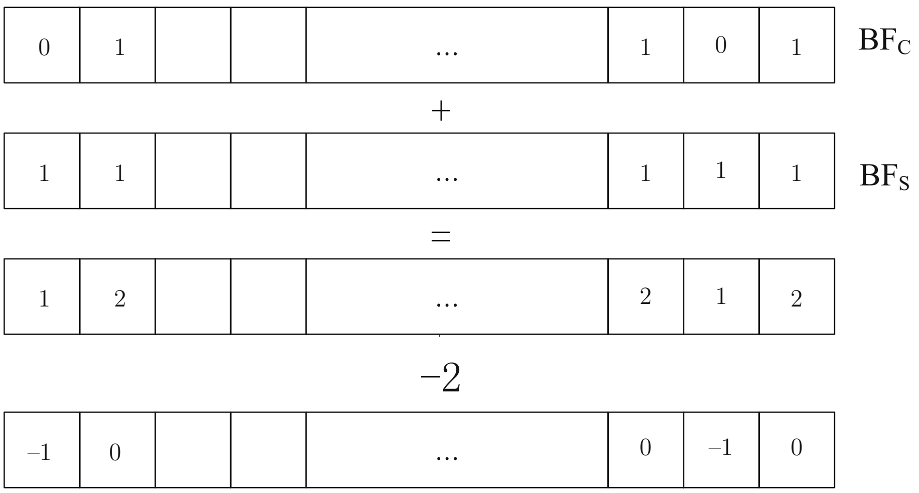

Fig. 4 Two Bloom filters added together and subtracted by 2

3. Recover: From the outcome of $E(r(BF_{C\cup S}/2))$, the client will decrypt $E(r(BF_{C\cup S}/2))$ with private keys. The client can calculate the value of $r(BF_C[i]+BF_S[i]-2)$. If $r(BF_C[i]+BF_S[i]-2)$ equals 0, we can know $BF_C[i]=BF_S[i]=1$ and then $BF_{C\cup S}[i]=1$. Otherwise, $BF_C[i]\neq BF_S[i]$ and then $BF_{C\cup S}[i]=0$.
4. Check: For any $x\in C$, if the locations in $BF_{C\cup S}$ mapped by $hash(x)$ are all 1, then $x\in C\cup S$ as can be seen from Alg. 1. In our algorithm, we can compute the elements of $C\cup S$ and thus get the number of similar elements of C and S. We record this number as t.

Algorithm 1 Check Algorithm ($BF_{C\cup S}$, a set C and a set $C\cup S$.

```
Require: A bloom filter $BF_{C\cup S}$, a set $C$ and a set $C\cup S$.
Ensure: True if $x \in C$, false else.
1: for all $x \in C$,
2:     for $i = 0$ to $k-1$.
3:     $i = hash(x)$
4:     End For
5:     If all $BF_{C\cup S}[i] = 1$ then
6:     $x \in C\cup S$
7:     End if
8: End For
```

5. Match: The client computes $\eta=\frac{t}{n}$. If η meets the requirements of system, they would be friends. Else, the client would reject the request of the server.

5.2 Analysis of our scheme

Correctness. As we know, all locations of Bloom filters are 0 or 1. Figure 4 shows the details of two Bloom filters added together and subtracted by 2. If we do not consider the encryption of them, both Bloom filters added together will be $BF_{C\cup S}$ that each location is 0, 1, and 2. When the client receives $E(r(BF_{C\cup S}/2))$ from the server, what we encrypt is -1, 0, and -2. Let us consider $BF[i]+BF[i]-2$, namely $E(r(BF_C[i]+BF_S[i]-2))=E(r\cdot 0)=E(0)$. We can find that the outcome decrypted by the client is 0, the result will be $BF_C[i]=BF_S[i]=1$. Therefore, the intersection of both Bloom filters $BF_{C\cup S}$ can be achieved, and the client executes the Check Algorithm to compute the similar elements with server.

Security proof. The security of our scheme is based on the security of private set intersection protocol. In order to prove the security of our scheme, we only need to prove the security of PSI protocol. We give security proof by comparison between the real model and an ideal model. The real model is the execution of our PSI protocol. The ideal model is the execution of the set intersection protocol implemented by a trusted server. Furthermore, the client and the server may behave arbitrarily during protocol execution except protocol abortion.

Theorem 1 *If the decisional composite residuosity (DCR) assumption holds, then the protocol PSI implements private set intersection in the malicious model securely.*

Proof 1. Confidentiality of the client: All inputs of the client are encrypted by Paillier encryption. Although what we encrypt is 0 or 1, the results of encryption are different numbers. In other words, the server cannot identify the distribution of 0s and 1s. Besides, security in PSI is based on IND-CPA secure encryption that can guarantee the security of client.
2. Confidentiality of the server: The server only computes the final results according to the algorithm and cannot decrypt it to get BF_C without private keys. To prove the security of our scheme against malicious adversary, it must be shown that for any possible client (server) behavior in the real model, there is an input

that the client (server) provides to the trusted third party (TTP) in the ideal model, such that his view in the real protocol is efficiently distinguishable from his view in the ideal model. Therefore, we give two constructions of simulator SIM_S and SIM_C from a malicious real world. We first give the simulator SIM_S.

(a) *Constructions of a simulator SIM_S from a malicious real-world server S'*:

(i) The simulator SIM_S encodes server's all elements by BF_S.

(ii) The simulator SIM_S receives $E(BF_C)$ from the client and simulates $E(r(BF_{C\cup S}/2))$.

(iii) The simulator SIM_S now plays the role of the ideal server interacting with the ideal client.
Since Paillier encryption scheme is IND-CPA secure under the decisional composite residuosity (DCR) assumption, the view of the malicious server S' in the simulation by SIM_S and in the real protocol is indistinguishable.

(b) *Constructions of a simulator SIM_C from a malicious real-world client C'*:

(i) The simulator SIM_C encodes the client's all elements by BF_C and receives the encrypted results $E(BF_C)$ from malicious client C'.

(ii) The simulator SIM_C receives the input $E(r(BF_{C\cup S}/2))$ from the ideal server and records it.

(iii) The simulator SIM_C plays the role of the ideal client and simulates $r(BF_{C\cup S}/2)$.

Since the server cannot modify the computing results without private key in the real model, what the client receives could be secure and confidential. Therefore, the view of the malicious client C' in the simulation by SIM_C and in the real protocol is indistinguishable. □

5.3 Complexity analysis

The complexity of our protocol is $O(m)$, m represents the size of Bloom filter. We used hash function, Paillier encryption, and modular multiplication. We analyze the efficiency of our protocol in terms of computation, communication, and storage.

- *Computational complexity:* To build BF_C or BF_S, each party needs $n \cdot k$ hash operations. For the client, it needs to encrypt BF_C by Paillier encryption with pubic keys and decrypt m ciphertexts. For the server, it only needs to compute m times modular multiplication.
- *Memory complexity:* The client needs to keep a copy of two Bloom filters, one is BF_C and the other is $BF_{C\cup S}$. Meanwhile, it also needs to store m ciphertexts. The server needs to keep a copy of one Bloom filter and m ciphertexts.

Table 1 Efficiency analysis

Complexity	Client	Server
Computation	$nkt_h + mt_p$	$nkt_h + mt_m$
Communication	m group elements	m group elements
Storage	$2m$bit+m group elements	$m$bit+m group elements

Table 2 Cost of Paillier cryptosystem

Algorithm (Liu et al. 2016)	Enc	Dec
PC run time	7.660ms	8.221ms
Smart phone run time	44.727ms	45.904ms

- *Communication complexity:* The data transferred in this protocol are m ciphertexts.

In Table 1, t_p , t_h, and t_m represent the computational cost of one-time Paillier encryption, hash function, and modular multiplication.

In order to demonstrate our scheme's efficiency, we evaluate its performance. The computation cost of the proposed scheme is roughly evaluated on a personal computer with 3.6 GHz eight-core and 12GB RAM memory (Liu et al. 2016). We currently use SHA-1 to build Bloom filters, and let N be 1024 bits to achieve 80-bits security. The reference running time of Paillier encryption and decryption is shown in Table 2. Compared with Paillier encryption, the computation cost of hash function and modular multiplication can be neglected. Therefore, the computation cost of client depends on the cost of Paillier encryption. Furthermore, the computation cost depends on the size of Bloom filter. Now, we let $k = 80$ and give different values of m and n to acquire the implementation. When we compute the cost of client, we neglect the cost of hash function and only take Paillier encryption and decryption into consideration.

As Table 1 and Fig. 5 show, the computation cost of our protocol has linear complexity $O(m)$. In other words, the performance of our protocol depends on the size of Bloom filter. However, it is different from De Cristofaro et al. (2010); De Cristofaro and Tsudik (2010), whose complexity is $O(n)$. Compared with the server, the client has large computational overhead. If this protocol is implemented in the smart phone, the overhead of the client would be much larger. Therefore, an outsourced fuzzy matching protocol is presented to solve this problem.

6 Outsourced fuzzy matching protocol

In our scheme, the Paillier encryption is the most expensive operation executed by the client. In many cases, the

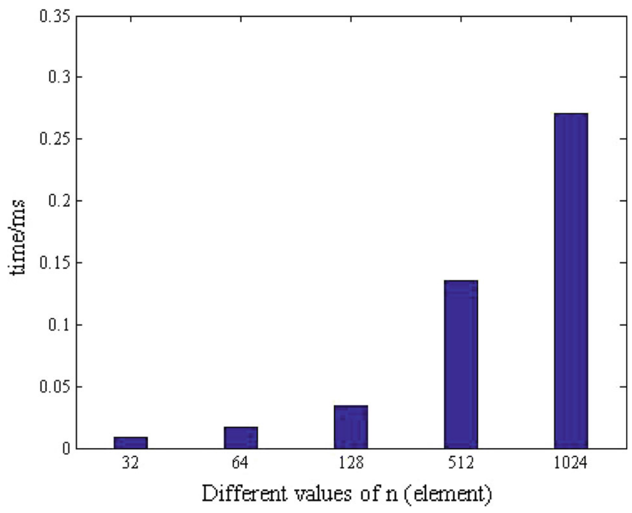

Fig. 5 Computation cost of the client

client maybe mobile phones, PDA, and other small devices, with limited computation resources. Outsourcing computation allows resource-constrained clients to outsource their complex computation workloads to a server which has powerful computation ability and larger computation resources.

6.1 Outsourced Paillier cryptosystem

Now, we give an efficient and secure outsourced algorithm for Paillier cryptosystem.

- The protocol for encryption algorithm is as follows:

1. The client runs *Rand* algorithm (*Rand* can be easily implemented in mobile devices) to generate random pairs $(\alpha, g_0^{\alpha} \bmod N^2)$ and $(\beta, g_0^{\beta} \bmod N^2)$, which can be completed during the off-line phase such as charging for mobile devices.
2. The client computes $gg_0^{\alpha}, rg_0^{\beta}, m\alpha + N\beta, g, g_0$ and outsources them to the cloud.
3. The cloud computes $P = (gg_0^{\alpha})^m \bmod N^2, Q = (rg_0^{\beta})^N \bmod N^2, R = g_0^{m\alpha+N\beta} \bmod N^2$ and returns them to the client.
4. The client computes $\frac{PQ}{R} \bmod N^2$.

- The protocol for decryption algorithm is as follows:

1. The client runs *Rand* algorithm to generate random pairs $(\alpha', g_0^{\alpha'} \bmod N^2)$ and $(\beta', g_0^{\beta'} \bmod N^2)$.
2. Suppose the ciphertext is c, the client computes $cg_0^{\alpha'}, \lambda\alpha' - \beta', g_0, g$ and outsources them to the cloud.
3. The cloud computes $P = (cg_0^{\alpha'})^{\lambda} \bmod N^2, Q = g_0^{\lambda\alpha'-\beta'} \bmod N^2$ and returns them to the client.
4. The client computes $L(\frac{P}{Qg_0^{\beta'}} \bmod N^2)\mu \bmod N$, which is the outcome of decryption.

6.2 Outsourced protocol

This outsourced protocol mainly aims at reducing the cost of public key encryption for client computation. The difference from the above protocol can be seen in the following.

1. *Outsourced encryption:* The client outsources the encryption of BF_C to the cloud and returns the encrypted $E(BF_C)$ to the server.
2. *Homomorphic computation:* The server also computes the result of $E(r(BF_{C\cup S}/2))$ and returns it to the client.
3. *Outsourced decryption:* The client outsources $E(r(BF_{C\cup S}/2))$ to the cloud and decrypts it for the result of $r(BF_{C\cup S}/2)$.
4. *Recover, check, and match:* This step is the same as the Sect. 5. Finally, the client obtains the intersection and judges whether the two sets match successfully or not.

6.3 Security analysis

The security of this protocol depends on the security of public-key encryption.

Theorem 2 *If the decisional composite residuosity (DCR) assumption and discrete logarithm holds, then the outsourced protocol PSI implements private set intersection in the malicious model.*

Proof The client outsources its input to the cloud and computes ciphertexts. The cloud receives inputs from the client and the cloud cannot decrypt it to get the plaintexts or key data. Therefore, the client can obtain the true output from the cloud. In the protocol execution, the server only receives encrypted messages. These are all secure due to IND-CPA security of our encryption scheme. □

6.4 Performance analysis

Throughout outsourcing, the client reduces the heavy computation task. From Table 3, the client only needs to do hash functions and does not need to execute complex public-key encryption. In Sect. 3, the notion of m has been explained in detail. For the security of our constructions, m is at least of $nk/ln_2^2 \approx 0.48$nk. Therefore, the computation time of Sect. 5 will be $nk(t_h + 0.48t_p)$. As shown in Tables 3 and 4 (see also DAI 2009), and from the Paillier encryption, it has much more expensive cost than hash function even though they are in the different platforms. From the perspective of the order of magnitude, the time cost of hash function can be

Table 3 Comparison of client computation

Protocol	Client Computation
Our non-outsourced proposal	$nkt_h + mt_p$
Our outsourced proposal	nkt_h

Table 4 Cost for running SHA-1 one-time

Windows Vista Intel Core2 1.83 GHZ 32-bit mode setting	Time cycles = 1/1.83 GHZ
Algorithm	cycles/Byte
SHA-1	11.4

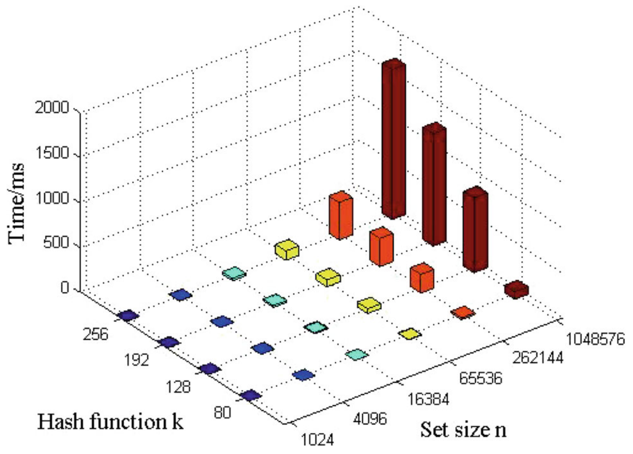

Fig. 6 Cost of the client in space diagram form

neglected, whereas in the outsourced protocol, the only operation is hash function and the efficiency is improved greatly. In conclusion, this outsourced protocol is much more efficient than the former version of the protocol.

We also roughly evaluate its performance using the software Crypto++ 5.6.0 running on Windows Vista Intel Core2 1.83 GHZ 32-bit mode (DAI 2009). SHA-1 is used for hash functions. Given a number k of different hash functions and set size n, such that $k = 80, 128, 192, 256$ and $n = 2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}, 2^{20}$, the time cost of the client is shown in Fig. 6 and Fig. 7. It can be seen from the figures that the computation time of the client has the relation to the multiplication of nk and the hash operation. When the values of n is fixed, the computation time increases linearly with the increase in k. When the values of k are fixed, the same characteristic is presented. Therefore, this protocol is very efficient and can support large-scale data sets.

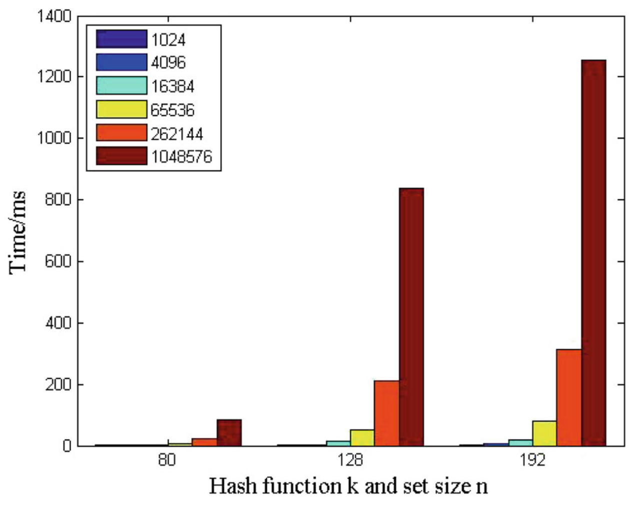

Fig. 7 Cost of the client in histogram form

7 Conclusion

This paper presented two fuzzy matching protocols based on private set intersection protocol. They are all against malicious adversaries in the standard model and can be used in social network for many applications like finding friends. The overhead of the existing protocol in the literature is very high due to the large computation of the client. To solve this problem, we proposed an outsourced protocol that can reduce the computation overhead of the client significantly. Compared with the prior work, the securities of two protocols can be achieved in malicious model. The efficiency of the former protocol can achieve linear complexity and the latter protocol is much more efficient than the former and can support friends finding in large social networks. However, we also note our work can only deal with fixed sets, it is better to support fuzzy matching on the sets with variable size. This will result in expanding the user base from which the related sets of interests can be formed, which is very interesting and our future work.

Acknowledgements This work was supported by the National Natural Science Foundation of China (61272492, 61572521), the Natural Science Foundation of Shaanxi Province (2014JM8300), and Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201610).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Bloom B (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426

Bose P, Guo H, Kranakis E, Maheshwari A, Morin P, Morrison J, Smid MHM, Tang Y (2008) On the false-positive rate of bloom filters. Inf Process Lett 108(4):210–213

Camenisch J, Zaverucha GM (2009) Private intersection of certified sets. In: Dingledine R, Golle P (eds) FC 2009. LNCS, vol 5628. Springer, Berlin, pp 108–127

Cheielewski L, Hoepman J (2008) Fuzzy private matching (extended abstract). In: Third international conference on IEEE availability, reliability and security

Chen C, Pai P, Hung W (2013) A new decision making process for selecting project leader based on social network and knowledge map. Int J Fuzzy Syst 15(1):36–46

Cristina D, Elena A, Catalin L, Valentin C (2014) A solution for the management of multimedia sessions in hybrid clouds. Int J Space-Based Situated Comput 4(2):77–87

Dachman-Soled D, Malkin T, Raykova M, Yung M (2009) Efficient robust private set intersection. In: Abdalla M, Pointcheval D, Fouque PA, Vergnaud D (eds) ACNS 09. LNCS, vol 5536. Springer, Berlin, pp 125–142

Dai W (2009) Crypto++ library: 5.6.0 benchmarks. http://www.cryptopp.com

De Cristofaro E, Kim J, Tsudik G (2010) Linear-complexity private set intersection protocols secure in malicious model. In: Abe M (ed) ASIACRYPT 2010. LNCS, vol 6477. Springer, Berlin, pp 213–231 (2010)

De Cristofaro E, Tsudik G (2010) Practical private set intersection protocols with linear complexity. In: Sion R (ed) FC 2010. LNCS, vol 6052. Springer, Berlin, pp 143–159

Debnath SK, Dutta R (2015) Secure and efficient private set intersection cardinality using bloom filter. In: ISC 2015. LNCS, Springer, Berlin, pp 209–226

Dong C, Chen L, Wen Z (2013) When private set intersection meets big data: an efficient and scalable protocol. In: Sadeghi AR, Gligor VD, Yung M (eds) ACM CCS 13. ACM Press, pp 789–800

Freedman MJ, Nissim K, Pinkas B (2004) Efficient private matching and set intersection. In: Cachin C, Camenisch J (eds) EUROCRYPT 2004. LNCS, vol 3027. Springer, Berlin, pp 1–19

Fu Z, Ren K, Shu J, Sun X, Huang F (2015) Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE Trans Parallel Distrib Syst. doi:10.1109/TPDS.2015.2506573

Fu Z, Wu X, Guan C, Sun X, Ren K (2016) Towards efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans Inf Forensics Secur. doi:10.1109/TIFS.2016.2596138

Guo S, Xu H (2015) A secure delegation scheme of large polynomial computation in multi-party cloud. Int J Grid Util Comput 6(2):1–7

Hazay C, Lindell Y (2008) Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti R (ed) TCC 2008. LNCS, vol 4948. Springer, Berlin, pp 155–175

Hazay C, Nissim K (2010) Efficient set operations in the presence of malicious adversaries. In: Nguyen PQ, Pointcheval D (eds) PKC 2010. LNCS, vol 6056. Springer, Berlin, pp 312–331

Hu J, Hu Y, Bein H (2011) Constructing a corporate social responsibility fund using fuzzy multiple criteria decision making. Int J Fuzzy Syst 13(3):195–205

Jarecki S, Liu X (2009) Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold O (ed) TCC 2009. LNCS, vol 5444. Springer, Berlin, pp 577–594

Jarecki S, Liu X (2010) Fast secure computation of set intersection. In: Garay JA, Prisco RD (eds) SCN 10. LNCS, vol 6280. Springer, Berlin, pp 418–435

Kerschbaum F (2012) Outsourced private set intersection using homomorphic encryption. In: Youm HY, Won Y (eds) ASIACCS 12. ACM Press, pp 85–86

Kissner L, Song DX (2005) Privacy-preserving set operations. In: Shoup V (ed) CRYPTO 2005. LNCS, vol 3621. Springer, Berlin, pp 241–257

Liu X, Deng R, Ding W, Lu R, Qin B (2016) Privacy-preserving outsourced calculation of floating point numbers. IEEE Trans Inf Forensics Secur 11(11):2513–2527

Many D, Burkhart M, Dimitropoulos X (2012) Fast private set operations with sepia. Technical Report 345

Meriem T, Mahmoud B, Fabrice K (2014) An approach for developing an interoperability mechanism between cloud providers. Int J Space-Based Situated Comput 4(2):88–99

Oded G (2009) The foundations of cryptography-vol 2, basic applications. Cambridge University Press, Cambridge

Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed) EUROCRYPT'99. LNCS, vol 1592. Springer, Berlin, pp 223–238

Ren W, Huang S, Ren Y, Choo KR (2016a) LiPISC: a lightweight and flexible method for privacy-aware intersection set computation. PLOS One. http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0157752

Ren W, Liu R, Lei M, Choo KR (2016b) SeGoAC: a tree-based model for self-defined and group-oriented access control in mobile cloud computing. Comput Stand Interfaces. doi:10.1016/j.csi.2016.09.001

Wang Y, Du J, Cheng X, Liu Z, Lin K (2016) Degradation and encryption for outsourced png images in cloud storage. Int J Grid Util Comput 7(1):22–28

Xia Z, Wang X, Sun X, Wang Q (2015) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst 27(2):340–352

Zhu S, Yang X (2015) Protecting data in cloud environment with attribute-based encryption. Int J Grid Util Comput 6(2):91–97