An efficient approach regarding two party private set union/intersection cardinality

Nan Cheng

Fudan University, China

Abstract. The two-party private set union/intersection cardinality problems (PSI/PSU-CA) are among the privacy-preserving computing problems, which are also fundamental and essential. They require that after protocol execution, at least one of the two parties gets their intersection/union cardinality securely with neither one leaked any other extra information to the other. Techniques solving these problems have many applications in reality: for example, they afford to preserve privacy for the data owner, promote more business cooperation, and suppress data abuse. However, we found that existing protocols often fall in a lack of applicable scenario, availability, and efficiency. The Egert-Fischlin protocol, which is incapable of solving PSI-CA problem and is only applicable to little or middle scale sets. The Dong-Loukides protocol, which costs too much in offline phase and is also unfit for a dataset that updates frequently. To tackle the issues mentioned above, we propose a new protocol; it adopts an offline-online computing approach that solves the PSU-CA problem and PSI-CA problem with less cost in total. And it is applicable on a broader scenario. Furthermore, simulation-based proof of security is presented. Going through experimental comparison with the cutting-edge Dong-Loukides protocol, we have shown that our protocol cost less in total.

Keywords: Privacy Preserving Computing, Bloom filter, Homomorphic Encryption

1 Introduction

More and more organizations have obtained considerable returns in the current digital age by collecting data and using it based on their personalized needs. For example, internet companies can develop recommendation systems to provide personalized internet products or services from consumption data; advertisers can also target targeted marketing through users' historical browsing data and combined with intelligent algorithms. This return will, in turn, spur more data collection needs.

During this process, user privacy will inevitably become harder to protect, so data privacy computing technology has become more and more necessary: For users, this technology can provide convenience while protecting their privacy to the maximum. For merchants, this technology can help merchants avoid the leakage of core data and maximize the value of their data held. In the context of increasing global emphasis on data security, major countries and organizations have enacted legislation to protect data privacy (such as HIPAA, COPPA, GLBA in the United States, and DPP in the European Union). Under this context, data privacy computing technology has become a current academic research boom.

The two-party PSI (Private Set Intersection) problem is a fundamental one within the secure two-party calculation model. In a protocol to solve the two-party PSI problem, we use Alice and Bob to represent the two parties, assuming they hold a set of characters of any length X and Y. After a series of interactions, we hope that at least one party can get the intersection of the two parties $X \cap Y$, and each participant cannot learn any other information about the elements of the other party's set. We can solve this problem by a series of protocols [17, 19–21] based on (Oblivious Transfer, OT).

The two-party PSU/PSI-CA problem is an extension of the two-party PSI problem. In a protocol to solve the two-party PSU/PSI-CA problem, it requires the output of $|X \cup Y|$ or $|X \cap Y|$, and no extra information is leaking to the other in the process, including any knowledge of the set element and the set size. This problem corresponds to many privacy computing scenarios in reality. For example, in a social network, two users who want to obtain the proportions of mutual friends without revealing any

specific friend information; in the health area, customers who hold private genetic data who want to figure out the probability of infection without leaking privacy by communicating with the public risk genetic databases. This article is to discuss about related protocols about the PSU/PSI-CA problem.

All privacy computing problems can be solved in theory by general secure computing protocols, such as GMW protocol [11], garbled circuit [26]. But these general solutions require many calculation and communication costs, thus being cumbersome and inefficient in real-life scenarios. Generally speaking, for specific security computing problems, specific protocols are usually more practical. Among those protocols solving PSU/PSI-CA problems, we divide them into two categories by the accuracy of the protocol output.

(1) The first one outputs exactly accurate results, and here we call this class a perfectly computing protocol. For example, as in work [7, 8, 12, 13, 15, 16], they both employ a fuzzy polynomial evaluation method, which selects a polynomial to represent the input set and then merges sets intersections through homomorphic encryption technology. However, this class employs many public-key primitives, so they still cost massive resources under a semi-honest model.

(2) The second one is called a imperfect computing protocol, it allows for a certain degree of deviation. And this kind of protocols could make a better trade-off between efficiency and availability even when facing a large-scale data input. For example:

- The Egert-Fischlin protocol [5] uses the ElGamal public key system [9] to encrypt each bit of the Bloom filter of one of the parties when implementing private computation. The advantage of the Egert-Fischlin protocol lies in its imbalance that the main computational overhead of the protocol concentrates on one of them, so it is suitable for server-client type application scenarios [5]. However, because its computational and communication complexity are both $O(n)$, it becomes harder to deploy in reality when the data input gradually becomes bigger.
- Based on the Flajolet-Martin technology [6] and OT($\frac{1}{4}$) technology, the Dong-Loukides protocol [4] can preprocess a set of length n into a Flajolet-Martin sketch [6] of length $\log((n))$, thus greatly reducing the cost of subsequent computation when using OT($\frac{1}{4}$) technology. However, the OT($\frac{1}{4}$) protocol must be executed serially in subsequent computation. Therefore, it has many communication rounds, and when the network environment becomes poor, the bottleneck of network transmission becomes more obvious in total. Besides, to improve the accuracy of the output results, this protocol needs to generate a large number of different Flajolet-Martin sketches [6] for the same set in the offline phase. So its hash cost is relatively high. However, the experimental data [4] shows that the Dong-Loukides protocol [4] is still the most efficient solution to the PSU/PSI-CA problem until now.

In this article, we focused on the design and analysis of the imperfect computing protocol because it can make a balance between efficiency and availability.Inspired by Dong et al. [4], we propose a naive methodology. It solves the PSU-CA problem with an new protocol; furthermore, it solves the PSI-CA problem by combining Pallier encryption with minor modification. The protocol solving the PSU-CA problem is composed of two phases: the setup and online phases. Alice and Bob first interactively generate enough pseudo-random related message pairs in the setup phase by executing Silent-OT protocol [3]. In the online phase, assuming that Bob is the secret transmission sender, Bob masks the original message with the pseudo-random related message pairs and then sends it to Alice, and Alice affords to decrypt the masked message directly associating messages received in the setup phase. Finally, Alice and Bob will get the secret-sharing S_A and S_B seperately about $HW(BF_A \vee BF_B)$ after finishing the above steps. Then:

- In the sub-protocol to solve the PSU-CA problem, let Alice be the output party of the final result, then Bob needs to send his secret share S_B to Alice. Finally, Alice merges the two secret shares S_A and S_B and output the estimated cardinality of the union of the two sets.

– In the sub-protocol to solve the PSI-CA problem, Alice and Bob also need to perform an extra homomorphic calculation protocol. After completing the protocol execution, one of the two parties outputs the estimated cardinality of the intersection of the two sets.

Since we can complete the heavy computation used for secret sharing in advance in offline-phase, and there is only efficient masking computation going on in online-phase. Either for a sub-protocol solving PSU-CA problem or PSI-CA problem, the subsequent overhead is very small. Therefore, compared with most previous protocols based on public-key primitives, our solution brings a more remarkable efficiency improvement.

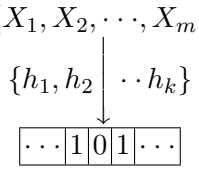
2 Preliminaries

In this article, for a finite set X, we use the symbol $|X|$ to denote its cardinality, and the symbol BF_A denotes the Bloom filter corresponding to the set A. For a natural number n, we use the symbol n^* to denote its integer estimate. The symbol $r \leftarrow_{\$} S$ denotes a random and uniform selection of a variable r from a finite set S. For a positive real number x, the symbol $\lfloor x \rfloor$ means non-negative integer n, where $(n = \max\{n \in \mathbb{Z}^+, 0 \leq n \leq x\})$. For a binary string A , the symbol $HW(A)$ denotes the number of 1 in A.

The symbol $\mathbb{Z}_n^*$ denotes a set, which satisfies $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$.

2.1 Bloom Filter

Bloom first proposed bloom filter in [2]. By mapping the elements in a set one by one through several hash functions to a binary string, we get a lightweight binary string, namely the Bloom filter. It is mainly used to detect the membership of elements in a set. When constructing a Bloom filter based on a large-scale data set X, we first fix the generation parameters, including k independent hash functions, maximum capacity C, error rate r. The bloom filter is first initialized as a all-zero binary string B that k is evenly divided by $|B|$ (where $|B|$ is related to C), then by hashing every element in X using k independent hash functions $h_l : \{0,1\} \mapsto \{1, ..., |B|\}$, in which $l \in (1, ..., k)$ and set the corresponding bit value of $|B|$ to 1, we get the final bloom filter.



Any third party can apply this data structure to detect the membership of an element in a set. For example, to determine if element a in set X, by hashing a with corresponding k hash functions, we get k integers, if every bit value of the corresponding position in X is 1, then possibly a is inside X. Otherwise, it must not. Because innately, there exist occurrences of a false-positive hit in bloom filter use. Intuitively, if the size of X, we denote that d is made large. The hashing collision will decrease, which will lead to a lower error rate. However, this will reduce the efficiency of calculation. According to the literature [25], using the following formula to constrain, we can obtain a trade-off between the bloom filter size and the error rate.

$$k = -\frac{\ln e}{\ln 2}, \frac{d}{m} = \frac{\ln 2}{k}$$

We can use the formula proposed by [24] to estimate the number of elements it contains:

$$n^* = -\frac{m}{k} \ln\left(1 - \frac{HW(BF)}{m}\right) \tag{1}$$

We denote $BF_A \vee BF_B$ as the binary string of the bitwise OR of BF_A and BF_B. Then, the estimation formulas about the union and intersection size when using a bloom filter are:

$$|A \cup B|^* = -\frac{m}{k} \ln\left(1 - \frac{HW(BF_A \vee BF_B)}{m}\right)$$

$$|A \cap B|^* = |A| + |B| + \frac{m}{k} \ln\left(1 - \frac{HW(BF_A \vee BF_B)}{m}\right)$$

In summary, the Bloom filter can be used to estimate the cardinality of the union of two sets, but this implicitly means that they both know the upper bound of the cardinality of the other set.

2.2 Paillier cryptoSystem

This article has employed the Paillier encryption system [18] in the protocol to solve the PSI-CA problem. It is a public key encryption system with homomorphic properties, which divides into key generation algorithm, encryption algorithm, and decryption algorithm, as follows.

Key Generation:

(1) Randomly and independently select two unequal large prime numbers p and q, and compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$.

(2) Select $g \leftarrow_{\$} \mathbb{Z}^*_{n^2}$.

(3) Define $L(x) = \frac{x-1}{n}$, if $\mu = L(g^\lambda \bmod n^2)^{-1} \bmod n$ exists, then output $PK = (n, g), SK = (\lambda, \mu)$, otherwise goes back to step (2) to select a different g.

Encryption:

(1) Make sure that $m \in [0, n]$, otherwise output $\perp$.

(2) Select $r \leftarrow_{\$} \mathbb{Z}^*_n$.

(3) Output $c = g^m \cdot r^n \bmod n^2$.

Decryption:

(1) Input ciphertext c that to be decrypted, if $c \notin \mathbb{Z}^*_{n^2}$, output $\perp$.

(2) Otherwise output $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$.

The Pallier encryption system is built on the higher residuosity problem modulo n^2. It is believed that, when n is large enough, the higher residuosity problem modulo n^2 is hard to solve if the factorization of n is unknown. The cryptosystem also possesses the properties of homomorphic addition:

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = (m_1 + m_2) \bmod n,$$
$$D(E(m_1, r_1)^{m_2} \bmod n^2) = (m_1 \cdot m_2) \bmod n.$$

2.3 Oblivious Transfer

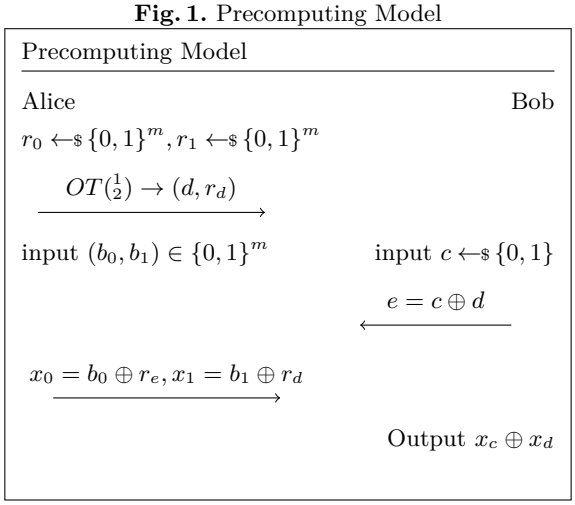
Oblivious Transfer (OT) is a protocol for secretly transferring information. It is a two-party protocol. In this article, we use $OT\binom{t}{n}$ to represent the general t out of n OT protocol, in which the sender is holding a group of messages m with a size of n, the receiver can arbitrarily choose $t(t < n)$ elements from the sender. After executing the agreement, the sender does not know what precisely the receiver has received, and the receiver does not know any other element of the sender except the t elements of its own choice. In terms of the most basic $OT\binom{1}{2}$, the sender holds a pair of message (m_0, m_1), the receiver holds his private bit $b \leftarrow_{\$} \{0, 1\}$, after executing $OT\binom{1}{2}$, the sender will not know the receiver's choice to be m_0 or m_1, and the receiver will not know any knowledge about m_{1-b}. The concept of OT was first proposed by Rabin [22] in the 1980s. The OT protocol constructed at that time was $OT\binom{1}{2}$, which was based on the RSA encryption system [23] construction and has undergone rich development and expansion in the following decades. Ishai et al. [14] proposed an extended version of OT that has

dramatically improved the computational efficiency of OT, making it more attracting to implement protocols based on OT.

Based on the pre-computing oblivious transfer model [1] of a secret transmission, we proposed the protocol in this article.

First, enough pseudo-random message pairs are generated by Silent-OT [3] protocol in the offline stage, and then the protocol performs only efficient mask operations in the subsequent online stage. A primary pre-compute mode is illustrated in figure 1.

Fig. 1. Precomputing Model



2.4 Security Model

At present, the two mainstream security models are Semi-honest model and malicious model [10]. Under the semi-honest model, the adversary will record all the data in the protocol interaction process and try to extract other Secret information of the participants. In the malicious model, the adversary will deviate in any way to destroy or snoop.

All the security protocols involved in this article are built in the semi-honest model [10]. Generally speaking, when designing a privacy computing protocol, firstly, we will try to design in the semi-honest model. Though, the protocols built in the semi-honest model is weaker than what in the malicious model, the actual business deployment still must comply with specific security guidelines authorized by laws and regulations. This implies that the protocols built in the semi-honest model still has a wide range of application prospects. Besides, we could utilize some general conversion methods such as zero-knowledge proof based conversion on protocols designed in the semi-honest model to meet the requirements of the malicious model further.

3 Two-Party PSU/PSI-CA Protocols

In this section, we will give the protocol details and related proofs regarding the PSU-CA problem and the PSI-CA problem. First, we will provide the protocol details regarding the PSU-CA problem and the PSI-CA problem (section 3.1), And then respectively give their correctness proofs (Section 3.2). Finally, we provide simulation-based safety proof (Section 3.3).

Fig. 2. Secret Sharing protocol

Secret Sharing protocol

Alice	Bob
BF_A of size m	BF_B of size m
$\{(S_1^0, S_1^1), (S_2^0, S_2^1), ..., (S_m^0, S_m^1)\}$	$\{c_1, c_2, ..., c_m\}, \{S_1^{c_1}, S_2^{c_2}, ..., S_m^{c_m}\}$
	set $\ddot{r} = \{c_1 \oplus (1 - BF_B[1]), ..., c_m \oplus (1 - BF_B[m])\}$
	$\xleftarrow{\ddot{r}}$
for i=1 to m do:	
pick $r_i \leftarrow\$ \{0,1\}^\lambda$	
set $M_i = (S_i^{\ddot{r}[i]} \oplus r_i, S_i^{1-\ddot{r}[i]} \oplus ((1 - BF_A[i] + r^i) \bmod 2^\lambda))$	
end	
$\xrightarrow{\{M_1, ..., M_m\}}$	
Output $S_1 = 2^\lambda - (\sum_{i=1}^{m} r_i) \bmod 2^\lambda$	
	for i=1 to m do:
	set $x_i = M_i^{1-BF_B[i]}$
	compute $x_i = x_i \oplus S_i^{c_i}$
	end
	Output $S_2 = (\sum_{i=1}^{m} x_i) \bmod 2^\lambda$

Fig. 3. Homomorphic Computing Protocol

Homomorphic Computing Protocol

Alice	Bob
$\{pk = (n, g), sk\}, \hat{S_1} = 2^n - 2^\lambda + S_1, p, k, m,$ Set A of size $\lvert A\rvert$	$S_2, p, k, m,$ Set B of size $\lvert B\rvert$
$\xrightarrow{pk, Enc_{pk}(\hat{S_1})}$	
	Pick $\tau \leftarrow\$ (\exp^{\frac{-k}{m}}, 1)$
	Set $s = Enc_{pk}(S_1) \cdot Enc_{pk}(S_2)$
	Compute $\hat{s} = s^{floor(p \cdot \tau \exp^{\frac{k \cdot \lvert B\rvert}{m}})} \bmod n^2$
	$\xleftarrow{\hat{s}}$
Compute $T = \lfloor \lvert A\rvert - (\frac{m}{k}) \cdot \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} \rfloor$	
$\xrightarrow{T}$	
	Output T

3.1 Protocol Construction

In the execution of the two protocols, we may as well set the two participants as Alice and Bob who hold the sets A and B respectively. Our protocols require both parties to input two Bloom filters with the same generating parameters, and the generation process is as follows:

(1) Alice and Bob agree on a sufficiently large capacity C, an error rate r, from equation (1), compute the length (m) of the bloom filter to be generated and the number of hash functions (k) used.
(2) Alice and Bob use the same k hash functions to process their input sets A, B, and finally receive their respective bloom filter sequences BF_A, BF_B independently.

After that, in the protocol description, we will directly use BF_A and BF_B as input, and we will not repeat the generation process. Also, when Alice and Bob agree on a large enough capacity which is C, it means that both Alice and Bob have got to know the upper bound of the size of the other party. Whether in the PSU-CA protocol or the PSI-CA protocol, Alice and Bob perform the same operation during the offline phase as follows.

Offline Phase Alice is the sender and Bob is the receiver. The SilentOT protocol [3] is called to generate enough pseudo-random related sequences, where Alice gets $\{(S_1^0, S_1^1), (S_2^0, S_2^1), ..., (S_N^0, S_N^1)\}$, that $\forall i \in N^+, \forall j \in (0,1)$, there exists $S_i^j \in \{0,1\}^\lambda$. Bob gets $\{c_1, c_2, ..., c_N\}, \{S_1^{c_1}, S_2^{c_2}, ..., S_N^{c_N}\}$, where $\forall i \in N^+, c_i \in \{0,1\}$. Then they continue the online phase as follows.

Online Phase Whether in a PSU-CA protocol or a PSI-CA protocol, Alice and Bob must first execute the following secret sharing protocol, as illustrated in figure 2:

(i) That is, Alice and Bob perform two rounds of interaction, the specific steps are as follows:
  (1) Bob selects $\ddot{r} \leftarrow_\$ \{0,1\}^m$ randomly, then send it to Alice.
  (2) After Alice receiving $\ddot{r}$, prepare m message pairs $\{M_1, ..., M_m\}$, where $\forall M_i (1 \le i \le m)$ that randomly select $r_i \leftarrow_\$ \{0,1\}^\lambda$. Set $M_i = (S_i^{\ddot{r}[i]} \oplus r_i, S_i^{1-\ddot{r}[i]} \oplus (BF_A[i] + r^i) \bmod 2^\lambda))$, meanwhile Alice compute $s_1 = 2^\lambda - (\sum_{i=1}^m r_i \bmod 2^\lambda)$. Then alice send $\{M_1, ..., M_m\}$ to Bob.
  (3) After Bob receiving $(\{M_1, ..., M_m\}, s_1)$, $\forall i \in m$, select $x_i = M_i^{1-BF_B[i]}$, and compute $x_i = x_i \oplus R_i$. Compute $s_2 = (\sum_{i=1}^m x_i) \bmod 2^\lambda$.
(ii) After executing the secret sharing protocol, Bob sends S_2 to Alice, and Alice calculates $S = (S_1 + S_2) \bmod 2^\lambda$, and outputs $|A \cup B|^* = -\frac{m}{k} \ln \frac{s}{m}$. We denote the protocol with this additional round of transmission as #1, #1 can be used to solve the **PSU-CA** problem in the semi-honest model.
(iii) To compute the (PSI-CA) problem, after executing the above secret sharing protocol, Alice needs to select the parameters of the Pallier encryption system $pk = (n, g)$, $sk = (\omega, \mu)$, also negotiate the amplification factor p with Bob, and then perform the following homomorphic calculation protocol. We denote this interaction process that mixes the two protocols as the protocol #2, #2 can solve the **PSI-CA** problem in the semi-honest model.
  (1) Alice generate $(pk = (n, g), sk = (\omega, \mu))$ of the Paillier crypto system, and send $pk = (n, g), Enc_{pk}(\hat{s_1})$ to Bob.
  (2) Bob compute $s = Enc_{pk}(\hat{s_1}) \cdot Enc_{pk}(s_2)$, and select $\tau \leftarrow_\$ (\exp^{\frac{-k}{m}}, 1)$, let $p = 10000$ be the precision control parameter, and compute $\hat{s} = s^{p \cdot \tau \exp \frac{k \cdot |B|}{m}} \bmod n^2$. Finally he send $\hat{s}$ to Alice.
  (3) Alice compute $T = |A| - \lfloor (\frac{m}{k}) \cdot \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} \rfloor$, and send T to Bob.

3.2 Analysis of correctness

Proposition 1. *In the semi-honest model, after executing protocol #1,* $|A \cup B|^* = -\frac{m}{k} \ln \frac{(s_1+s_2) \bmod 2^\lambda}{m}$.

Proof. If Alice and Bob strictly follow the steps of the protocols above. Then in protocol #1, $\forall i \in (1, m)$, if $BF_B[i] = 0$, then $x_i = S_i^{1-(c_i \oplus (1-BF_B[i]))} \oplus ((1 - BF_A[i] + r^i) \bmod 2^\lambda) \oplus S_i^{c_i} = (1 - BF_A[i] + r^i) \bmod 2^\lambda$, if $BF_A[i] = 0$, $x_i = (1 + r^i) \bmod 2^\lambda$, otherwise $x_i = r^i \bmod 2^\lambda$.

If $BF_B[i] = 1$, then $x_i = S_i^{c_i} \oplus r^i \oplus S_i^{c_i} = r^i$.

So $s_2 = z + \sum_{i=1}^{m} r_i$, where $z = m - HW(BF_A \vee BF_B)$, then $(s_1 + s_2) \bmod 2^\lambda = z$. From equation (1), there is $|A \cup B|^* = -\frac{m}{k} \ln \frac{z}{m}$.

Thus proving the previous proposition.

Proposition 2. *In the semi-honest model, after executing protocol #2,* $|A \cap B|^* = \lfloor |A| - (\frac{m}{k}) \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} \rfloor$.

Proof. From equation (2), there is:

$$\begin{aligned}
(|A \cup B| - |B|)^* &= |A \cup B|^* - |B|^* \\
&= (-\frac{m}{k})(\ln \frac{m - HW(BF_A \vee BF_B)}{m} - \ln \frac{m - HW(BF_A)}{m}) \\
&= \frac{m}{k} \ln \frac{m - HW(BF_B)}{m - HW(BF_A \vee BF_B)} \\
&= \frac{m}{k} \ln \frac{m}{\exp^{\frac{k \cdot |B|}{m}} \cdot (m - HW(BF_A \vee BF_B))} \\
&= \frac{m}{k} \ln \frac{p \cdot m}{p \cdot \exp^{\frac{k \cdot |B|}{m}} \cdot (m - HW(BF_A \vee BF_B))}
\end{aligned}$$

If Alice and Bob strictly follow the steps of the protocols above. Then in the above homomorphic protocol, because $s = Enc_{pk}(\hat{s_1} + s_2) = Enc_{pk}(2^n - 2^\lambda + s_1 + s_2) = Enc_{pk}(2^n + HW(m - HW(BF_A \vee BF_B)))$, and the plaintex space of the Paillier system is $\{0, 1\}^n$, then the term 2^n will disapear in computation, thus leading to $s = Enc_{pk}(HW(m - HW(BF_A \vee BF_B)))$, so $Dec_{sk}(\hat{s}) = p \cdot \tau \cdot \exp^{\frac{k \cdot |B|}{m}} \cdot (m - HW(BF_A \vee BF_B))$. And so $\frac{Dec_{sk}(\hat{s})}{\tau} = p \cdot \exp^{\frac{k \cdot |B|}{m}} \cdot (m - HW(BF_A \vee BF_B))$, also $(|A \cup B| - |B|)^* = \frac{m}{k} \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} + \frac{m}{k} \ln \tau \approx \frac{m}{k} \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} (because \frac{m}{k} \ln \tau \in (0, 1))$. So,

$$\begin{aligned}
|A \cap B|^* &= |A|^* + |B|^* - |A \cup B|^* \\
&= |A|^* - (|A \cup B|^* - |B|^*) \\
&\approx |A|^* - \frac{m}{k} \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})}
\end{aligned}$$

Thus proving the previous proposition.

3.3 Analysis of Security

Theorem 1. *There exists a PPT* SIM_1 *such that for all security parameters* λ *(*$2^\lambda > |BF_A|$*) and inputs* BF_A, BF_B,

$$REAL^{1,\lambda}(BF_A, BF_B, S(\cdot), R(\cdot)) = SIM_1(1^\lambda, BF_A, HW(BF_A \vee BF_B), S(\cdot))$$

Where $S(\cdot), R(\cdot)$ *are pseduoranom pairs of Alice and Bob, and* $REAL^{1,\lambda}(BF_A, BF_B)$ *is the interacting transcript of protocol #1.*

Algorithm 1: The simulator for Alice

Input: $1^\lambda, BF_A, HW(BF_A \vee BF_B)$
Output: $SimView(Alice)$
(1) Select $\ddot{r} \leftarrow_\$ \{0,1\}^{|BF_A|}$ as the simulation output to Bob in first round.
(2) Compute $z = |BF_A| - HW(BF_A \vee BF_B)$, $c_i \leftarrow_\$ |BF_A|^+$, where $i = \{1, 2, ..., z\}$. Select$r_j \leftarrow_\$ \{0,1\}^\lambda$, where $j = \{1, 2, ..., |BF_A|\}$, compute $s_2 = 2^\lambda - (\sum_{i=1}^{|BF_A|} r_j)$ as Alice's simulation output. And for $\forall i \in z^+$, let $r_{c_i} = (r_{c_i} + 1) \bmod \{0,1\}^\lambda$. Then $\forall i \in |BF_A|^+$, set $M_i = (S_i^0 \oplus r_i, S_i^1 \oplus r_i)$ as Alice's simulation output in second round.
(3) Compute $s_2 = (\sum_{i=1}^{|BF_A|} r_i) \bmod 2^\lambda$ as Bob's simulation output in third round.

Proof. We describe the simulator algorithm SIM_1 in Algorithm 1.

In step (1) of algorithm 1, we select $\ddot{r}$ in $\{0,1\}^{|BF_A|}$ randomly and uniformly, likely in the real scenario, $\ddot{r} = \{c_1 \oplus (1 - BF_B[1]), ..., c_m \oplus (1 - BF_B[m])\}$, because c_i is selected randomly, thus the two $\ddot{r}$ in simulation and in real execution are computational indistinguishable. And in step (2), because S_i^0, S_i^1, r_i is unifromly distributed in $\{0,1\}^\lambda$ that $S_i^0 \sim \mathcal{U}(0, 2^\lambda)$, $S_i^1 \sim \mathcal{U}(0, 2^\lambda)$, $r_i \sim \mathcal{U}(0, 2^\lambda)$, so M_i in step (2) is also uniformly distributed in $\{0,1\}^{2\cdot\lambda}$ as M_i in a real execution, thus the two M_i are computational indistinguishable. For the same reason, the output in step (3) is also computational indistinguishable with the real output in a real execution.

In summary, the above simulation algorithm can simulate the entire transcript of protocol #1. So the theorem is proved.

Theorem 2. *There exists a PPT SIM_2 such that for all security parameters λ ($2^\lambda > |BF_A|$) and inputs BF_A, BF_B,*

$$REAL^{1,\lambda}(BF_A, BF_B, S(\cdot), R(\cdot)) = SIM_2(1^\lambda, BF_B, HW(BF_A \vee BF_B), R(\cdot))$$

Where $S(\cdot), R(\cdot)$ are pseduoranom pairs of Alice and Bob, and $REAL^{1,\lambda}(BF_A, BF_B)$ is the interacting transcript of protocol #1.

Proof. We describe the simulator algorithm SIM_1 in Algorithm 2.

Algorithm 2: The simulator for Bob

Input: $1^\lambda, BF_B, HW(BF_A \vee BF_B), R(\cdot)$
Output: $SimView(Bob)$
(1) Simulate Bob's output from input $R(\cdot), BF_B$ in first round as required in protocol #1.
(2) Compute $z = |BF_B| - HW(BF_A \vee BF_B)$, $c_i \leftarrow_\$ |BF_A|^+$ where $i = \{1, 2, ..., z\}$. Select $r_j \leftarrow_\$ \{0,1\}^\lambda$ where $j = \{1, 2, ..., |BF_B|\}$, compute $s_1 = 2^\lambda - (\sum_{i=1}^{|BF_A|} r_j)$ as Alice's output simulation. Then $\forall i \in z^+$, let $r_{c_i} = (r_{c_i} + 1) \bmod \{0,1\}^\lambda$, $\forall i \in |BF_A|^+$, set $M_i^{1-BF_B[i]} = S_i^{c_i} \oplus r_j$, and set $M_i^{BF_B[i]} \leftarrow_\$ \{0,1\}^\lambda$ as Alice's simulation output in second round.
(3) Compute $s_2 = (\sum_{i=1}^{|BF_B|} r_i) \bmod 2^\lambda$ as Bob's simulation output in third round.

The simulation in step (1) of Algorithm 2 is the same as in the real execution. Meanwhile, the output to Bob in step (2), namely M and $R(\cdot), s_2$, meet well the constraints in a real protocol execution. Furthermore, because $S_i^{BF_B[i]}$ is uniformly distributed and independent with $S_i^{1-BF_B[i]}$. Thus the union distribution (M_i^0, M_i^1) is computational indistinguishable with the output in step (2) of Algorithm 2. Similar to the proof of Theorem 1, the simulation output of step (3) is computational indistinguishable from the real output in a real protocol execution.

In summary, the above simulation algorithm can simulate the entire interaction of the protocol #1. So the theorem is proved.

Theorem 3. *There exists a PPT SIM_1 such that for all security parameters λ ($2^\lambda > |BF_A|$) and inputs $(s_1, s_2, |A|, |B|, m, k)$,*

$$REAL^{1,\lambda}(s_1, s_2, |A|, |B|, m, k) = SIM_1(1^\lambda, s_1, |A|, f(\cdot))$$

Where $f(\cdot) = f(s_1, s_2, |A|, |B|) = \lfloor |A| + |B| - (\frac{m}{k}) \cdot \ln \frac{(s_1+s_2) \bmod 2^\lambda}{m} \rfloor$, m and k are constants, and $REAL^{1,\lambda}(s_1, s_2, |A|, |B|)$ is the interacting transcript of the homomorphic protocol 2.

Proof. We describe the simulator algorithm SIM_1 in Algorithm 3.

Algorithm 3: The simulator for Alice

Input: $1^\lambda, s_1, |A|, f(s_1, s_2, |A|, |B|)$
Output: $SimView(Alice)$
(1) Generate (pk, sk) as in the Paillier crypto system, encrypt $Enc_{pk}(s_1)$, and simulate the output of Alice $(pk, Enc_{pk}(s_1))$ as in the first round.
(2) Select $\beta \leftarrow_\$ (0, 1)$, compute $\phi = |A| - (f(s_1, s_2, |A|, |B|) + \beta)$, then compute $\hat{\phi} = \frac{p \cdot m}{\exp^{\frac{\phi \cdot k}{m}}}$. Let $\hat{s} = Enc_{pk}(\hat{\phi})$, simulate what Bob outputs as in the second round.
(3) Output $f(\cdot)$ as the output of Alice in the third round, also output $f(\cdot)$ as Bob's final output.

In step (1) of Algorithm 3, becuase the resistance of chosen plaintext attack of Paillier Crypto System, it's simulation output is computational indistinguishable with what in a real protocol execution. In step (2), because in a real protocol execution, $\tau \leftarrow_\$ (\exp^{\frac{-k}{m}}, 1)$, so $\frac{m}{k} \ln \tau \in (0, 1)$, let $\delta = |A| - (\frac{m}{k}) \cdot \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} - \lfloor |A| - (\frac{m}{k}) \cdot \ln \frac{p \cdot m}{Dec_{sk}(\hat{s})} \rfloor$, then $\delta \in (0, 1)$, select $\beta \leftarrow_\$ (0, 1)$, and finally the result $\hat{\phi}$ could perfectly simulate $Dec_{sk}(\hat{s})$ in a real protocol execution.

In summary, the above simulation algorithm can simulate the entire interaction of the homormorphic protocol 2. So the theorem is proved.

Theorem 4. *There exists a PPT SIM_2 such that for all security parameters λ ($2^\lambda > |BF_A|$) and inputs $(s_1, s_2, |A|, |B|, m, k)$,*

$$REAL^{1,\lambda}(s_1, s_2, |A|, |B|, m, k) = SIM_2(1^\lambda, s_1, |A|, f(\cdot))$$

Where $f(\cdot) = f(s_1, s_2, |A|, |B|) = \lfloor |A| + |B| - (\frac{m}{k}) \cdot \ln \frac{(s_1+s_2) \bmod 2^\lambda}{m} \rfloor$, m and k are constants, and $REAL^{1,\lambda}(s_1, s_2, |A|, |B|)$is the interacting transcript of the homomorphic protocol 2.

Proof. Bob receives only the message $Enc_(pk)(s_1)$ from Alice in the first round of the protocol, we encrypt $Enc_(pk)(0)$ by using public key pk, becuase the resistance of chosen plaintext attack of Paillier Crypto System, it's simulation output is computational indistinguishable with what in a real protocol execution. From input $1^\lambda, s_2, |B|, f(\cdot), m, k$, we could simulate Bob's view in the homomorphic protocol 2. So the theorem is proved.

In summary, combining Theorem 1 and Theorem 2, by using simulation, we have proved the security of the protocol #1 in the semi-honest model. Combining Theorem 1, Theorem 2, Theorem 3, Theorem 4, we have proved the protocol #2, that is, the protocol to solve the PSI-CA problem is also safe in the semi-honest model.

4 Performance Evaluation

In this section, from theoretical analysis and experimental data, we have compared our protocol with the Dong-Loukides protocol [4] on different indicators. Thus showing that our protocol has better performance in general, and suitable for a broader range of application scenarios.

4.1 Theoretical Evaluation

Table 1 is a qualitative performance comparison. Among which N is the size of the collection, k_1 is the number of BF hash functions generated, and k_2 is the number of the hash functions required to generate the Flajolet-Matin sketch where $k_1 \ll k_2$. The hash cost column refers to the hash cost in the preprocessing stage. In Dong-Loukides protocol [4], it refers to the hash cost of the conversion of the set into Flajolet-Martin sketch [6] (we usually need to assign k_2 a relatively big constant to improve the accuracy of the computation result of the protocol). In protocol #1 and protocol #2, it refers to the hashing needed to finsh bloom filter generation ($k_1 = -\frac{\ln r}{\ln 2}$, r is the error rate).

Protocol	PSU-CA	PSI-CA	Hashing Cost	Communication Round	Exponential Cost	Communication Cost
Dong-Loukides protocol	Yes	Yes	$2k_2N$	$k_2 \cdot \log N$	$O(\log N)$	$O(\log N)$
Protocol #1	Yes	No	$2k_1N$	2	0	$O(N)$
Protocol #2	No	Yes	$2k_1N$	5	$O(1)$	$O(N)$

Table 1. Theoretical Comparison

Table 1 shows that the number of communication rounds and the exponential computation cost of the protocol #1 and protocol #2 are all constants. Besides, it shows that the protocol #1 and protocol #2 behave much better than Dong-Loukides protocol [4] on every indicator except for communication cost amount. Furthermore, even if the dataset updates slightly in the Dong-Loukides protocol [4], it is required to regenerate the whole Flajolet-Matin sketch. This defect greatly limits the possible application scenarios adapted for the Dong-Loukides protocol [4]. Besides, the Dong-Linkdes protocol [4] depends on specific dataset when performing offline computation while protocol #1 and protocol #2 are not.

4.2 Experimental evaluation

We have implemented the three protocols (https://github.com/athenKing/PSU-PSI-CA) to compare their specific performance. All three protocols are implemented in C++. The test platform is equipped with a 2.9GHz double core Intel-core I5 processor and 16GB 1867 MHz DDR3 memory. The test completes in a LAN network environment with low network latency. In the implementation of the Dong-Loukides protocol, we set the security parameters to be $\lambda = 128$, the number of independent hash functions k_2 to 4096, and the output Flajolet-Martin summary [6] length to be 32 (theoretically, it can count up to $2^{32} - 1$ Elements). In the implementation of protocol #1 and protocol #2, the security parameters are also set to be $\lambda = 128$, the error rate of the Bloom filter is set to $r = 0.05$, three sets of data (From small to large) The maximum capacity of the corresponding Bloom filters are set to be 5000, 50000, and 400000 respectively. Table 2 shows their actual performance under different scales of input. In the table, N_x and N_y denotes the size of the two input sets, and the PSU/PSI-CA error denotes the relative error of the output result, that is, the relative error of the output result to the real result of the union cardinality and intersection cardinality.

The experimental results show that the preprocessing stage of the Dong-Loukides protocol [4] is relatively time-consuming. So it is not suitable for scenarios where the dataset is updated frequently and is only ideal for privacy computing among scenarios that hold permanent or semi-permanent data. Compared the Dong-Loukides protocol [4] with protocol #1 and protocol #2, the experiment also shows that the total time consumption is smaller in the latter no matter the size of the dataset. When the size of the input dataset increases, this performance advantage becomes more and more apparent. Therefore, our protocol has a higher computational efficiency than the Dong-Loukides protocol [4] in computing scenarios where the data set is updated frequently.

(N_x, N_y)	Protocol	PSU/PSI-CA error	Communication(MB)	Preprocessing(s)	Online(s)	Total Time Cost(s)
(645,3010)	Dong-Loukides	3.6%,24.8%	15.83	0.59	0.63	1.22
	Protocol #1	0.9% , -	0.98	0.20	0.05	0.25
	Protocol #2	- , 6.6%	0.98	0.20	0.50	0.70
(6280,30099)	Dong-Loukides	0.1% , 1.1%	15.83	6.53	0.63	7.32
	Protocol #1	0.1% , -	9.81	0.51	0.40	0.91
	Protocol #2	- , 1.0%	9.81	0.51	1.07	1.58
(62936,300783)	Dong-Loukides	0.1%,0.9%	15.83	63.27	0.63	63.90
	Protocol #1	0.1% , -	78.49	2.84	3.16	6.00
	Protocol #2	- , 0.9%	78.49	2.84	5.32	8.16

Table 2. Experimental Comparision

5 Summary and Outlook

This article proposes a new imperfect computing protocol to solve the PSU/PSI-CA problem, and it proves the security in the semi-honest model. It uses Bloom filters to preprocess the set. Meanwhile it mixes OT preprocessing [1] and Paillier homomorphic encryption technology [18]. Compared with the most advanced Dong-Loukides protocol [4], this protocol takes less time totally and thus possess a wider range of application scenarios.

Our protocol cannot resist malicious adversaries, so in the next step, by combining the current scheme and the techniques of zero-knowledge proof [10], we will try to build an efficient provably safe protocol in the malicious model [10].

References

1. D. Beaver. Precomputing oblivious transfer. In D. Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.
2. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
3. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518. Springer, 2019.
4. C. Dong and G. Loukides. Approximating private set union/intersection cardinality with logarithmic complexity. *IEEE Transactions on Information Forensics and Security*, 12(11):2792–2806, 2017.
5. R. Egert, M. Fischlin, D. Gens, S. Jacob, M. Senker, and J. Tillmanns. Privately computing set-union and set-intersection cardinality via bloom filters. In *Australasian Conference on Information Security and Privacy*, pages 413–430. Springer, 2015.
6. P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.
7. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.
8. K. Frikken. Privacy-preserving set union. In *International Conference on Applied Cryptography and Network Security*, pages 237–252. Springer, 2007.
9. T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.
10. O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
11. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.
12. C. Hazay and K. Nissim. Efficient set operations in the presence of malicious adversaries. In *International Workshop on Public Key Cryptography*, pages 312–331. Springer, 2010.
13. S. Hohenberger and S. A. Weis. Honest-verifier private disjointness testing without random oracles. In *International Workshop on Privacy Enhancing Technologies*, pages 277–294. Springer, 2006.
14. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

15. A. Kiayias and A. Mitrofanova. Testing disjointness of private datasets. In *International Conference on Financial Cryptography and Data Security*, pages 109–124. Springer, 2005.
16. L. Kissner and D. Song. Privacy-preserving set operations. In *Annual International Cryptology Conference*, pages 241–257. Springer, 2005.
17. V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829. ACM, 2016.
18. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
19. B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. 2019.
20. B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 515–530, 2015.
21. B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on {OT} extension. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 797–812, 2014.
22. M. O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2005:187, 2005.
23. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
24. S. J. Swamidass and P. Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *J. Chem. Inf. Model.*, 47(3):952–964, 2007.
25. S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Commun. Surv. Tutorials*, 14(1):131–155, 2012.
26. A. C. Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.