Hindawi
IET Information Security
Volume 2024, Article ID 6052651, 12 pages
https://doi.org/10.1049/2024/6052651

Research Article

Low Communication-Cost PSI Protocol for Unbalanced Two-Party Private Sets

Jingyu Ning, Zhenhua Tan, Kaibing Zhang, and Weizhong Ye

Software College, Northeastern University, Shenyang 110819, China

Correspondence should be addressed to Zhenhua Tan; tanzh@mail.neu.edu.cn

Received 28 September 2023; Revised 24 December 2023; Accepted 2 February 2024; Published 17 February 2024

Academic Editor: Helena Rifà-Pous

Two-party private set intersection (PSI) plays a pivotal role in secure two-party computation protocols. The communication cost in a PSI protocol is normally influenced by the sizes of the participating parties. However, for parties with unbalanced sets, the communication costs of existing protocols mainly depend on the size of the larger set, leading to high communication cost. In this paper, we propose a low communication-cost PSI protocol designed specifically for unbalanced two-party private sets, aiming to enhance the efficiency of communication. For each item in the smaller set, the receiver queries whether it belongs to the larger set, such that the communication cost depends solely on the smaller set. The queries are implemented by private information retrieval which is constructed with trapdoor hash function. Our investigation indicates that in each instance of invoking the trapdoor hash function, the receiver is required to transmit both a hash key and an encoding key to the sender, thus incurring significant communication cost. In order to address this concern, we propose the utilization of a seed hash key, a seed encoding key, and a Latin square. By employing these components, the sender can autonomously generate all the necessary hash keys and encoding keys, obviating the multiple transmissions of such keys. The proposed protocol is provably secure against a semihonest adversary under the Decisional Diffie–Hellman assumption. Through implementation demonstration, we showcase that when the sizes of the two sets are 2^8 and 2^{14}, the communication cost of our protocol is only 3.3% of the state-of-the-art protocol and under 100 Kbps bandwidth, we achieve 1.46x speedup compared to the state-of-the-art protocol. Our source code is available on GitHub: https://github.com/TAN-OpenLab/Unbanlanced-PSI.

1. Introduction

Private set intersection (PSI) protocol is a special case of secure two-party computation, which allows the receiver and the sender, holding the input sets X and Y, respectively, to compute $X \cap Y$ without revealing anything else (other than upper bounds on their sizes) [1, 2]. PSI has served many privacy-preserving applications including COVID-19 risk scoring [3], contact tracing [4], advertising conversion rate calculation [5], and mobile privacy contact discovery [6].

The first PSI protocol is based on Diffie–Hellman (DH) key exchange algorithms [7, 8]. After that, PSI has been extensively studied and many protocols have been proposed to improve the performance, consisting of communication performance and computation performance. The current PSI protocols are largely based on two technologies, namely, DH key exchange [2, 7–10] and oblivious transfer (OT) [3, 11–23]. Among OT-based protocols, some are based on cuckoo hashing [11–19], and some are based on oblivious key-value stores (OKVS). In addition, some protocols are based on RSA [3, 20–23] and homomorphic encryption [24, 25].

When measuring the efficiency of a PSI protocol, the communication cost and the computation cost are two major aspects [22]. Recent evidence suggests that communication costs are far more important than computation costs [2]. Existing PSI protocols primarily focus on the intersection of two sets with similar sizes and have obtained low communication costs. However, it is significant to note that the communication costs in these protocols are influenced by the sizes of both sets. This is due to the linear relationship between the communication cost and the size of each party's set. Consequently, for unbalanced set sizes, the larger set size exerts a greater impact on the overall communication costs. Chen et al. [25] considered the effect of unbalanced sets on the communication cost and proposed a PSI protocol based on fully homomorphic encryption (FHE) such that the number of messages sent by each party had a linear relationship with the size of the smaller set. However, FHE requires a large ciphertext space, bringing about a room for improvement.

PSI protocols for unbalanced sets have considerable application scenarios. One significant application scenario is private contact discovery [6], where a user of a mobile application wishes to identify which of his friends in his address book are also users of the same application. However, the server is not allowed to reveal its users' information and the user not want to submit his or her address book. In this case, the server has a large set with all the users, while the mobile side has a relatively small set. Another well-known application scenario of unbalanced PSI protocol is advertising conversion rate calculation [5] where the ad supplier knows the users who have seen a particular ad, and the company knows who made a purchase. The two parties are unwilling to expose the underlying data, but both parties would still like to compute how many users both saw an ad and made a corresponding purchase. According to our observation, unbalanced PSI protocol would be utilized in authentication schemes [26–28] to match authentication information between two users who do not trust each other.

Therefore, we aim to construct a PSI protocol specifically tailored for unbalanced sets, eliminating the effect of the larger set size on the communication cost. By taking this approach, the communication cost is solely determined by the smaller set. The rationale for our research is derived from this motivation.

Focusing on the above motivation, we propose a PSI protocol based on trapdoor hash (TDH) function and Latin square for the intersection of a larger set (sender) and a smaller set (receiver).

Inspired by Döttling et al. [29] and Chase et al. [30], we construct private information retrieval (PIR) by TDH to realize low communication cost. Following this way, the number of invoking of PIR equals to the size of the smaller set. However, the hash key and the encoding key should be sent from the receiver to the sender for each PIR, resulting in large communication cost. To address this issue, we design a seed hash key, a seed encoding key, and a Latin square which are sent by the receiver. The sender can generate all the hash keys and encoding keys by himself from the seed keys. Intuitively, the seed hash key, the seed encoding key, the hash keys, and the encoding keys are vectors with the same dimension. We permute the items in the seed hash key to obtain each hash key and permute the items in the seed encoding key to obtain each encoding key, where the permutation rule is defined in the Latin square.

The main contributions of this paper are as follows:

(1) We design a permutation rule according to the Latin square, by which the sender can generate a range of hash keys and encoding keys for all rounds of TDH function from only one seed hash key and one seed encoding key. The communication cost of transmitting multiple keys is reduced to transmitting two seed keys. The seed keys and the Latin square design do not involve the items of the two sets, thus it can be performed in the offline phase.

(2) In the process of PSI, the smaller set is taken as the ergodic source, and the larger set is taken as the verification source. Every time an item of the smaller set is calculated, the TDH function is called once to verify whether it belongs to the intersection by retrieving it from the larger set. Consequently, the communication cost depends only on the smaller set, and this work is valid for unbalanced sets.

(3) We implement out protocol and public it on GitHub: https://github.com/TAN-OpenLab/Unbanlanced-PSI .

The proposed protocol is provably secure against a semihonest adversary under the Decisional Diffie–Hellman (DDH) assumption. Performance analysis demonstrates that the proposed protocol enhances the communication performance in PSI protocols in terms of unbalanced sets where the size ratio of two sets exceeds 8.

2. Related Work

Since its introduction, many techniques have been proposed to improve PSI's performance. In this section, we discuss the state-of-the-art PSI protocols and focus on the communication cost of them. From here on, we assume that the receiver's set has n_1 items, and the sender's set has n_2 items $(n_1 \le n_2)$, where each item has σ-bit length. We let λ and κ denote the statistical and computational security parameters, respectively.

Early PSI protocols based on DH have been around since 1986 [7, 8] and proven secure against semihonest adversaries. Current PSI protocol can be divided into two categories. The first category is semihonest PSI protocols [11–13, 18–20, 22, 25], and the second category is malicious PSI protocols [2, 3, 14–16, 21, 23].

In semihonest protocols, the parties have to follow the exact prespecified protocol, which implies that they cannot change their inputs or outputs. PSZ14 [11] is based on private equality test, where the receiver and the sender, respectively, insert all their items in the bins by cuckoo hashing and all hash functions. The using of cuckoo hashing reduces the comparisons of equality test from $O(n_1 n_2)$ to $O(n_1 \log n_2 / \log\log n_2)$. The receiver compares σ bits for each comparison. PSSZ15 [13] is based on PSZ14 [11] and permutation-based hashing technique, which splits each item into left part with $\log n_1$ bits length and right part with $(\sigma - \log n_1)$ bits length. Only the right part is compared, so $(\sigma - \log n_1)$ bits are compared for each comparison. PSSZ15 [13] and PSZ14 [11] are based on the OT extension protocol proposed by KK13 [31]. KKRT [12] improved KK13 [31] by extending 1-out-of-256 OT to 1-out-of-∞ OT and proposed batch, related-key oblivious pseudorandom function (OPRF). Their PSI protocol is 3.1–3.6× faster than PSSZ15 [13]. CLR17 [25] used FHE and improved it by batching, windowing, and modulus switching. They constructed a PSI protocol for unbalanced sets and achieved a communication overhead of $O(n_1 \log n_2)$. PRTY19 [20] improved the OT extension

Parameters: Receiver's input set size n_1, sender's input set size n_2.
Inputs: Receiver inputs a set of items $X = \{x_0, \ldots, x_{n_1-1}\}$, where $x_i \in \mathbb{Z}_n$.
Sender inputs a set of items $Y = \{y_0, \ldots, y_{n_2-1}\}$, where $y_i \in \mathbb{Z}_n$.
Output: Receiver receives the set intersection $X \cap Y$.

Figure 1: PSI ideal functionality.

protocol proposed by IKNP03 [32], and proposed lightweight PSI based on sparse OT extension. The sender generates a polynomial P using its set and sends P to the receiver. The receiver computes the corresponding values of his items in P. For each item in the intersection, the results computed by the two parties will be the same. The communication cost of PRTY19 [20] is 40%–50% lower than that of KKRT [12], so it is targeted at low-bandwidth situations. However, the computation of PRTY19 [20] is not as efficient as KKRT [12], so KKRT is faster at high bandwidth. CM20 [22] achieves a better balance between KKRT [12] and PRTY19 [20]. Single-point OPRF of KKRT [12] is extended to multipoint OPRF, where the key of PRF is a matrix and a single PRF will compare all the items.

In malicious protocols, the parties may not follow the exact prespecified protocol, thus the inputs from both parties need to be verified. On the basis of KKRT [12], PRTY20 [21] added homomorphic function as linear error correcting code [16] and proposed a malicious security PSI protocol PaXoS, which was almost as fast as KKRT [12]. RT21 [2] presented a construction for a batched OPRF based on vector-OLE and the PaXoS data structure. GPR21 [23] considered that cuckoo hashing will lead to a failure probability p of OKVS structure. They therefore showed novel techniques to improve OKVS such that the failure probability was reduced to p^c for a constant c. RT21 [2] pointed out that OT-based PSI protocols required a certain number of base OTs first, which applies to large sets. On small sets, DH-PSI protocols are less costly, so they proposed a DH-PSI-based PSI for small sets, and reduced the communication cost by interpolating polynomials.

3. Preliminaries

3.1. PSI Functionality. We use the PSI functionality described in Chase and Miao's [22] study. PSI allows two parties to compute the intersection of their data sets without revealing any additional information, as shown in Figure 1.

3.2. Security Model. We use the security model described in David et al.'s [1] and Goldreich's [33] studies. PSI is a cryptographic protocol of secure two-party computation. There are two adversarial models which are usually considered, namely semihonest model and malicious model. A semihonest party is one who follows the protocol properly with the exception that it keeps a record of all its intermediate computations and may try to learn as much as possible from the messages they receive from other parties. A malicious adversary may deviate arbitrarily from the prescribed protocol in an attempt to violate security. The semihonest model and the malicious model are designed for different application scenarios, thus both of them have practical value and research value. Our protocol is designed under the semihonest model in this paper.

We say the protocol is secure if we can construct simulators who can generate the outputs without the information of the private sets. The outputs should be indistinguishable in probabilistic polynomial time from those generated by the real sender and receiver, respectively. This means that even if a semihonest adversary corrupts the sender or the receiver, it cannot obtain any meaningful information about the private sets.

3.3. Decisional Diffie–Hellman Assumption. Our protocol relies on the DDH assumption [34], which we state in the following.

Definition 1. Decisional Diffie–Hellman (DDH) assumption. A (prime-order) group generator is an algorithm $\mathcal{G}$ that takes as an input a security parameter 1^λ and outputs $(\mathbb{G}, p, g)$. $\mathbb{G}$ is a multiplicative cyclic group of order p, and with generator g, where p is always a prime number. We say that $\mathcal{G}$ satisfies the DDH assumption (or is DDH hard) if for any PPT adversary $\mathscr{A}$, it holds that:

$$\begin{aligned} |\Pr \quad & [\mathscr{A}((\mathbb{G}, p, g), (g^{a_1}, g^{a_2}, g^{a_1 a_2})) = 1] \\ -\Pr \quad & [\mathscr{A}((\mathbb{G}, p, g), (g^{a_1}, g^{a_2}, g^{a_3})) = 1]| = \mathrm{negl}(\lambda), \end{aligned} \tag{1}$$

where $(\mathbb{G}, p, g) \xleftarrow{\$} \mathcal{G}$ and $a_1, a_2, a_3 \xleftarrow{\$} \mathbb{Z}_p$.

3.4. Latin Square. The definition of Latin square is similar to [35].

Definition 2. Latin square. Let n be a positive integer, and let $\mathbb{Z}_n$ be the set of n distinct elements. A Latin square S of order n on $\mathbb{Z}_n$ is an n-by-n matrix, where each element of S belongs to $\mathbb{Z}_n$ and each element of $\mathbb{Z}_n$ occurs once in each row and once in each column of S.

In this paper, the following Latin square design method is used to generate an n-order Latin square S. Let the element of the a^{th} row and the b^{th} column be $S_{a,b}$.

Step 1. Randomly shuffle the n elements in $\{0, \ldots, n-1\}$ and let the result be row 0 of S as $S_{0,\cdot} = (S_{0,0}, \ldots, S_{0,n-1})$.

Step 2. Generate other elements of S. For each element in row a and column b:

$$S_{a,b} = (S_{0,b} + a) \bmod n, \tag{2}$$

where $a \in \{1, 2, \ldots, n-1\}$, $b \in \{0, 1, \ldots, n-1\}$.

According to the Latin square design above, we can obtain a Latin square of order n. Each row $S_{a,\cdot}$ can be seen as an arrangement of the elements of $S_{0,\cdot}$. Therefore, we regard $S_{0,\cdot}$ and $S_{a,\cdot}$ as the permutation rule by which we can permute a matrix G to a new matrix G'. Let G be an

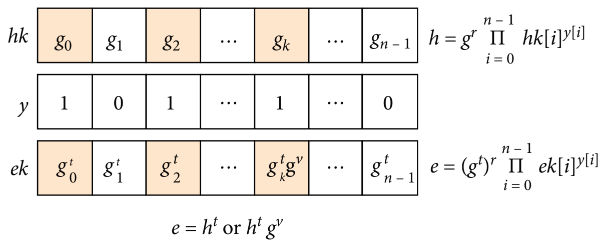
hk
g_0 g_1 g_2 ⋯ g_k ⋯ g_{n-1}
$h = g^r \prod_{i=0}^{n-1} hk[i]^{y[i]}$
y
1 0 1 ⋯ 1 ⋯ 0
ek
g_0^t g_1^t g_2^t ⋯ $g_k^t g^v$ ⋯ g_{n-1}^t
$e = (g^t)^r \prod_{i=0}^{n-1} ek[i]^{y[i]}$
$e = h^t$ or $h^t g^v$

Figure 2: PIR from trapdoor hash function.

arbitrary matrix with n columns. For each $m \in \mathbb{Z}_n$, let the column m of G be the column m' of G' if $S_{0,m} = S_{a,m'}$. Then, G' is a new matrix with n columns

3.5. Private Information Retrieval from Trapdoor Hash Function. TDH function was proposed by Döttling et al. [29]. In this section, we introduce a PIR [36] scheme using TDH.

In PIR, the sender has a private bitstring $y = \{0, 1\}^n$ of length n, and the receiver wants to know the k^{th} bit $y[k]$. The sender will not reveal any information except $y[k]$. Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p, and g is a generator of the group.

Receiver samples the trapdoor $t, v \overset{\$}{\leftarrow} \mathbb{Z}_p$ and samples an n-dimensional vector of random group elements $hk = \{g_0, \ldots, g_{n-1}\} \overset{\$}{\leftarrow} \mathbb{G}^n$ as the seed hash key, as shown in Figure 2. Then, computes a corresponding encoding key as $ek = \{g_0^t, \ldots, g_k^t g^v, \ldots, g_{n-1}^t\}$, where for every $i \in \{0, \ldots, n-1\}$, g_i^t is g_i to the power t. The only exception is $g_k^t g^v$ which is set as g_k^t times g^v. Receiver sends g^t, hk, and ek to sender. Sender samples $r \overset{\$}{\leftarrow} \mathbb{Z}_p$ and calculates the hash value h and the encoding value e, as follows:

$$h = g^r \prod_{i=0}^{n-1} hk[i]^{y[i]}, \tag{3}$$

$$e = (g^t)^r \prod_{i=0}^{n-1} ek[i]^{y[i]}. \tag{4}$$

Collision resistance of Function (3) can be routinely established from the discrete logarithm assumption in $\mathbb{G}$.

Then, the sender sends h and e to the receiver, who verifies whether $e = h^t$ or $e = h^t g^v$, where $e = h^t$ means $y[k] = 0$ and $e = h^t g^v$ means $y[k] = 1$. For each item in the set of the receiver, the two parties invoke PIR to compute whether it belongs to the set of the sender.

4. The Proposed PSI Protocol

The proposed PSI protocol contains offline phase and online phase. In the offline phase, the PSI preparation is performed which does not involve the set items of two parties. In the online phase, two parties complete PSI with their private sets. Receiver and sender hold the private sets $X = \{x_0, \ldots, x_{n_1-1}\}$ and $Y = \{y_0, \ldots, y_{n_2-1}\}$, respectively. Let $\mathbb{F} = \{0, \ldots, n-1\}$ be the input domain, containing all the possible items of X and Y. The parameters of the proposed protocol are shown in Table 1.

Table 1: The parameters of the proposed PSI protocol.

Parameter	Definition
X	The receiver's set
Y	The sender's set
x_i	The i^{th} item of X
y_j	The j^{th} item of Y
n_1	The sizes of X
n_2	The sizes of Y
$\mathbb{F}$	The input domain
n	The size of $\mathbb{F}$ and the order of Latin square
c_i	The row number of the Latin square related to x_i
(h_i, e_i)	The hash value and the encoding value of x_i
r_i	The result bit of (h_i, e_i)
S	Latin square, a $n \times n$ matrix
S_i	The i^{th} row of S, an n-dimensional vector
G	The seed key of the receiver, a $2 \times n$ matrix, consisting of the hash key and the encoding key
(t, v)	The trapdoor of the receiver
k	The initial column number
G^i	The key related to x_i, a $2 \times n$ matrix
λ	Statistical security parameter
κ	Computational security parameter

The framework of the proposed protocol is described in Figure 3. In the offline phase, receiver obtains row 0 of Latin square $S_{0,\cdot}$ by shuffling $\{0, \ldots, n-1\}$, samples the trapdoor $t, v \overset{\$}{\leftarrow} \mathbb{Z}_p$, the initial column number $k \overset{\$}{\leftarrow} \mathbb{Z}_n$ and the seed matrix G. Then receiver sends $S_{0,\cdot}$, G, and g^t to sender.

In the online phase, both parties employ PIR to determine whether each item x_i in the receiver's set X belongs to the sender's set Y. First, the receiver maps the set item x_i to the specific row number c_i of Latin square and sends it to the sender. Then, the sender generates the c_ith row $S_{c_i,\cdot}$ of Latin square, which satisfies that $S_{c_i,x_i} = S_{0,k}$. The sender takes $S_{0,\cdot}$ and $S_{c_i,\cdot}$ as the permutation rule to permute the seed key matrix G to the key matrix G^i of x_i. Then, the sender encodes G^i to obtain the hash value h_i and the encoding value e_i of x_i, and send them to the receiver. Finally, the receiver decodes (h_i, e_i) and obtains whether x_i belongs to $X \cap Y$.

4.1. The Offline Phase. Let algorithm $\mathcal{G}$ be a prime-order group generator that takes as an input a security parameter 1^λ and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a multiplicative cyclic group, order p is a prime number, and g is the generator. The algorithm Shuffle() takes a vector as an input and shuffles all the elements in the vector to form a new vector.

The offline phase consists of the following steps.

Step 1. Using the method described in Section 3.4, the receiver generates an n-dimensional vector $S_{0,\cdot} = \text{shuffle}(0, \ldots, n-1)$ and sends it to sender.

Step 2. The receiver samples the trapdoor $t, v \overset{\$}{\leftarrow} \mathbb{Z}_p$, and samples the initial column number $k \overset{\$}{\leftarrow} \mathbb{Z}_n$.

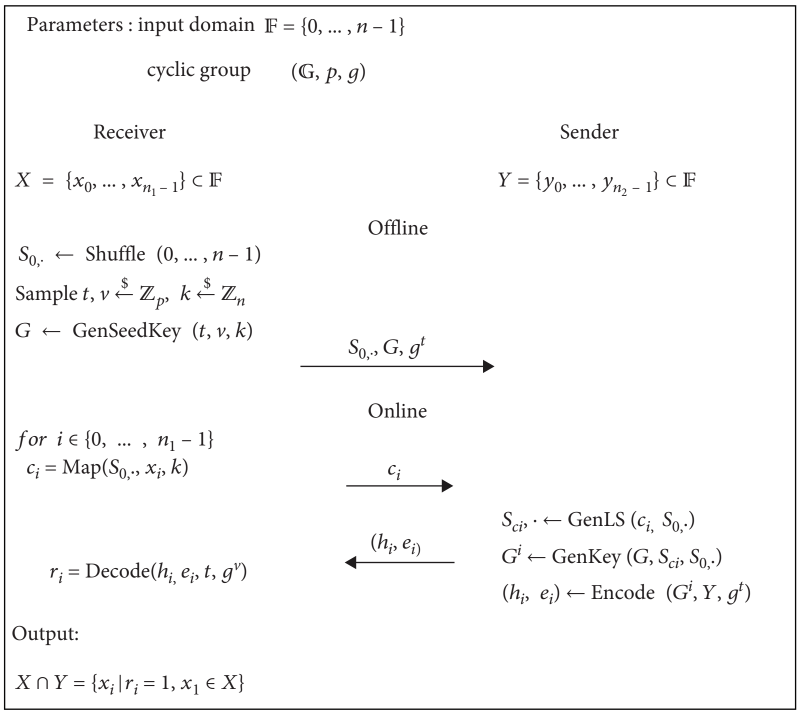
Parameters : input domain $\mathbb{F} = \{0, \ldots, n-1\}$
cyclic group $(\mathbb{G}, p, g)$
Receiver
Sender
$X = \{x_0, \ldots, x_{n_1-1}\} \subset \mathbb{F}$
$Y = \{y_0, \ldots, y_{n_2-1}\} \subset \mathbb{F}$
Offline
$S_{0,\cdot} \leftarrow$ Shuffle $(0, \ldots, n-1)$
Sample $t, v \xleftarrow{\$} \mathbb{Z}_p,\ k \xleftarrow{\$} \mathbb{Z}_n$
$G \leftarrow$ GenSeedKey (t, v, k)
$S_{0,\cdot}, G, g^t$
Online
$for\ i \in \{0, \ldots, n_1-1\}$
$c_i = \mathrm{Map}(S_{0,\cdot}, x_i, k)$
c_i
$S_{ci}, \cdot \leftarrow$ GenLS $(c_i, S_{0,\cdot})$
(h_i, e_i)
$G^i \leftarrow$ GenKey $(G, S_{ci}, S_{0,\cdot})$
$r_i = \mathrm{Decode}(h_i, e_i, t, g^v)$
$(h_i, e_i) \leftarrow$ Encode (G^i, Y, g^t)
Output:
$X \cap Y = \{x_i \mid r_i = 1, x_1 \in X\}$

FIGURE 3: The framework of the proposed PSI protocol.

Step 3. The receiver generates the seed key matrix $G \leftarrow \mathrm{GenSeedKey}(t, v, k)$. In detail, the receiver samples an n-dimensional vector of random group elements as the seed hash key $(g_0, \ldots, g_{n-1}) \xleftarrow{\$} \mathbb{G}^n$. Then calculates the seed encoding key as follows:

$$(u_0, \ldots, u_k, \ldots, u_{n-1}) = \left(g_0^t, \ldots, g_k^t g^v, \ldots, g_{n-1}^t\right), \tag{5}$$

where for every $i \in \{0, \ldots, n-1\}$, $u_i = g_i^t$. The only exception is u_i which is set as $g_k^t g^v$. Let the seed hash key be the 0th row of matrix G, while the seed encoding key is taken as the first row of matrix G. We have the equation as folllows:

$$G = \begin{pmatrix} g_0 & \cdots & g_{n-1} \\ u_0 & \ldots & u_{n-1} \end{pmatrix}. \tag{6}$$

Then, the receiver sends $(S_{0,\cdot}, G, g^t)$ to the sender.

4.2. The Online Phase. In the online phase, the receiver and the sender calculate the intersection of their sets. There are n_1 items in receiver's set $X = \{x_0, \ldots, x_{n_1-1}\} \subset \mathbb{F}$, and n_2 items in sender's set $Y = \{y_0, \ldots, y_{n_2-1}\} \subset \mathbb{F}$. For each item x_i, both parties invoke PIR to determine whether it belongs to the sender's set Y.

Recall that the receiver sends the seed key matrix G to the sender in the offline phase, and g^v is in the k^{th} column of G. To calculate x_i, we should permute the columns of G to obtain G^i such that g^v is in the x_i^{th} column of G^i. Each row in the Latin square contains all the items in $\mathbb{Z}_n$ in different order. Let row c_i of Latin square be the target row such that $S_{c_i,x_i} = S_{0,k}$. The process to obtain c_i is shown in Algorithm 1.

```
Input: $S_{0,k}$, $x_i$, $k$
Output: $c_i$
  $a = 0$
  While $S_{a,x_i} \neq S_{0,k}$
    $a = a + 1$
    $S_{a,x_i} = (S_{a-1,x_i} + 1)\ mod\ n$
  EndWhile
  $c_i = a$
```

ALGORITHM 1: Map().

Let $S_{c_i}, \cdot \leftarrow \mathrm{GenLS}(c_i, S_{0,\cdot})$ be an algorithm which calculates row c_i by row 0 of the Latin square. For each column b:

$$S_{c_i,b} = \left(S_{0,b} + c_i\right) \bmod n. \tag{7}$$

We can obtain a permutation rule from row $S_{0,\cdot}$ and row $S_{c_i,\cdot}$. For each column $m (m \in \mathbb{Z}_n)$ of G, let m' be the index of column such that $S_{c_i,m'} = S_{0,m}$. Then, set the m'^{th} row of G^i equal to the m^{th} column of G, namely $(G'_{0,m'}, G'_{1,m'})^T = (G_{0,m}, G_{1,m})^T$. The process is shown in Algorithm 2.

The 0th row of matrix G^i is the hash key, while the first row of matrix G^i takes as the encoding key. Let $(h_i, e_i) \leftarrow \mathrm{Encode}(G^i, Y, g^t)$ be the algorithm which takes the hash value and the encoding value as output. More specifically,

```
Input: $G, S_{0,\cdot}, S_{c_i,\cdot}$
Output: $G^i$
  For $m = 0$ to $n-1$
    $m' = 0$
    While $S_{c_i,m'} \neq S_{0,m}$
      $m' = m' + 1$
    EndWhile
    $G^i_{0,m'} = G_{0,m}$
    $G^i_{1,m'} = G_{1,m}$
EndFor
```

Algorithm 2: GenKey ().

the sender samples $r \xleftarrow{\$} \mathbb{Z}_p$, and calculates the hash value and the encoding value, as follows:

$$h_i = g^r \prod_{j=0}^{n_2-1} G^i_{0,y_j}, \tag{8}$$

$$e_i = (g^t)^r \prod_{j=0}^{n_2-1} G^i_{1,y_j}. \tag{9}$$

Then, sender sends (h_i, e_i) to receiver.

Observe that when $x \in Y$, then e_i is equal to $h_i^t g^v$, and that otherwise, it is equal to h_i^t. Let $r_i = \mathrm{Decode}(h_i, e_i, t, g^v)$ be the algorithm which decodes (h_i, e_i) by the trapdoor and outputs the result bit r_i. Let r_i denotes the three cases above:

$$r_i = \begin{cases} 0, & e_i = h_i^t \\ 1, & e_i = h_i^t g^v \\ \perp, & \text{other} \end{cases}. \tag{10}$$

In summary, the steps of the online phase are as follows. For each item $x_i \in X$:

Step 1. The receiver calculates c_i from x_i by algorithm Map(), and sends c_i to the sender.

Step 2. The sender generates row c_i by algorithm GenLS (), and generates the key matrix G^i by algorithm GenKey(). Therefore, the sender obtains the hash key and the encoding key of x_i.

Step 3. The sender calculates the hash value h_i and the encoding value e_i by Encode() and sends them to the receiver.

Step 4. The receiver decodes (h_i, e_i) and obtains whether x_i belongs to $X \cap Y$.

For every item in X, the receiver and the sender repeat the steps above and obtain $X \cap Y$ as follows:

$$X \cap Y = \{x_i | r_i = 1, x_i \in X\}. \tag{11}$$

5. Example Analysis

In this section, we offer an illustrative instance of the proposed PSI protocol aimed at showcasing its practical feasibility.

Let $\mathbb{F} = \{0, \ldots, 7\}$ be the input domain, and $n = 8$ be the number of elements of $\mathbb{F}$. The receiver and the sender, respectively, hold sets $X = \{6, 1\}$ and $Y = \{4, 0, 2, 6, 7\}$ of sizes $n_1 = 2$ and $n_2 = 5$. Let $\mathbb{G} = \{1, 5^1, 5^2, \ldots, 5^{16}\}$ be a multiplicative cyclic group of order $p = 17$ and with generator $g = 5$.

In the offline phase, receiver shuffles $(0, \ldots, 7)$ and obtains the 0th row $S_{0,\cdot} = (7, 0, 2, 5, 1, 6, 4, 3)$ of Latin square S.

After sampling the trapdoor $t = 3, v = 7$ and the initial column number $k = 4$, the receiver samples $(5^6, 5^2, 5^9, 5^1, 5^7, 5^3, 5^4, 5^{13})$ from $\mathbb{G}$ as the 0th row of the seed key matrix G and calculates $(u_0, \ldots, u_k, \ldots, u_{n-1}) = (5^{6\times3}, 5^{2\times3}, 5^{9\times3}, 5^{1\times3}, 5^{7\times3+7}, 5^{3\times3}, 5^{4\times3}, 5^{13\times3})$, where for every $i \in \{0, \ldots, 7\}$, $u_i = g_i^t$. The only exception is u_i which is set as $g_k^t g^v$. Let $(u_0, \ldots, u_k, \ldots, u_{n-1})$ be the first row of the seed matrix G. We have the equation as follows:

$$G = \begin{pmatrix} 5^6 & 5^2 & 5^9 & 5^1 & 5^7 & 5^3 & 5^4 & 5^{13} \\ 5^1 & 5^6 & 5^{10} & 5^3 & 5^{11} & 5^9 & 5^{12} & 5^5 \end{pmatrix}. \tag{12}$$

Finally, the receiver sends $(S_{0,\cdot}, G, g^t)$ to sender.

In the online phase, the receiver determines the items of set X that also belong to $X \cap Y$. For the item $x_0 = 6$, the receiver finds out $c_0 = 5$ such that $S_{c_0,6} = S_{0,k}$, and sends c_0 to the sender.

The sender calculates the fifth row $(4, 5, 7, 2, 6, 3, 1, 0)$ of S from the 0th row $(7, 0, 2, 5, 1, 6, 4, 3)$ using Equation (7) and permutes the column vectors of G according to the 0th row and the fifth. It is evident that $S_{0,0} = S_{5,2}$, thus the second column of G' should be the same as the 0th column of G. Similarly, as $S_{0,1} = S_{5,7}$, the seventh column of G' should match the first column of G. In the same vein, we have the following equation:

$$G' = \begin{pmatrix} 5^4 & 5^1 & 5^6 & 5^9 & 5^3 & 5^{13} & 5^7 & 5^2 \\ 5^{12} & 5^3 & 5^1 & 5^{10} & 5^9 & 5^5 & 5^{11} & 5^6 \end{pmatrix}. \tag{13}$$

Then, the sender calculates the hash value $h_0 = \prod_{j=0}^{n_2-1} G'_{0,Y_j} = G'_{0,0}G'_{0,2}G'_{0,4}G'_{0,6}G'_{0,7} = 5^5$ and the encoding value $e_0 = \prod_{j=0}^{n_2-1} G'_{1,Y_j} = G'_{1,0}G'_{1,2}G'_{1,4}G'_{1,6}G'_{1,7} = 5^5$, and sends (h_0, e_0) to receiver.

The receiver calculates $h_0^t = 5^{5\times3} = 5^{15}$ and finds that $e_0 = h_0^t g^v$, thus $r_0 = 1$ and $x_0 \in Y$. For the item $x_1 = 1$, the receiver finds out $c_0 = 1$ such that $S_{c_0,1} = S_{0,k}$ and sends c_0 to sender.

Sender permutes the column vectors of G according to the 0th row $(7, 0, 2, 5, 1, 6, 4, 3)$ and the first row $(0, 1, 3, 6, 2, 7, 5, 4)$ of S. We can see that $S_{0,0} = S_{1,5}$, thus let the fifth column of G' be the 0th column of G. Similarly, since $S_{0,1} = S_{1,0}$, let the 0th column of G' be the first column of G. In the same vein, we have the following equation:

$$G' = \begin{pmatrix} 5^2 & 5^7 & 5^{13} & 5^3 & 5^9 & 5^6 & 5^1 & 5^4 \\ 5^6 & 5^{11} & 5^5 & 5^9 & 5^{10} & 5^1 & 5^3 & 5^{12} \\ \\ \end{pmatrix}. \quad (14)$$

Then, the sender calculates $h_1 = \prod_{j=0}^{n_2-1} G'_{0,y_j} = G'_{0,0}G'_{0,2}G'_{0,4}G'_{0,6}G'_{0,7} = 5^{12}$, $e_1 = \prod_{j=0}^{n_2-1} G'_{1,y_j} = G'_{1,0}G'_{1,2}G'_{1,4}G'_{1,6}G'_{1,7} = 5^2$, and sends (h_1, e_1) to receiver.

The receiver calculates $h_0^t = 5^{12\times 3} = 5^2$ and finds that $e_0 = h_0^t$, thus $r_0 = 0$ and $x_0 \notin Y$. Finally, the receiver obtains $X \cap Y = \{6\}$.

6. Proof of Security

Our protocol relies on the DDH assumption, which is resistant to semihonest attackers. Relying on the previous theory of security proof [37–39], this section proves the security of the proposed protocol against the corrupt sender and the corrupt receiver, respectively.

6.1. Security against the Corrupt Sender

Theorem 1. *The proposed protocol is resistant against the corrupt sender under the DDH assumption. Formally, we construct a simulator $\mathcal{S}_1$ that takes the inputs $(\mathbb{F}, \mathbb{G}, p, g)$ and the outputs (G, c_i) is indistinguishable from the real receiver.*

Proof. According to the proposed protocol, the messages that receiver sends to sender are the 0th row of Latin square $S_{0,\cdot}$, seed key matrix G, and the row number $c_i (i \in \{0, \dots, n-1\})$. $S_{0,\cdot}$ is generated by Shuffle() and thus indistinguishable from random. Consequently, we focus on the security of seed key matrix G and the row number c_i in this section. We prove $\mathcal{S}_1$ is indistinguishable from the real receiver via the following hybrid argument. □

Hybrid 0: Hybrid 0 is the real interaction. In the offline phase, receiver generates and sends seed key matrix G honestly. In the online phase, for each item x_i of X, receiver performs Map() and sends the row number c_i according to Section 4.

Hybrid 1: Same as Hybrid 0 except that G is replaced with a random matrix G'.

Recall that the 0th row of G is randomly sampled and indistinguishable from random. The elements in the first row of G are calculated by the elements in the 0th row. In this hybrid, the elements in the first row of G are replaced by n random elements $(u'_0, \dots, u'_{n-1}) \xleftarrow{\$} \mathbb{G}^n$ and have the following equation:

$$G' = \begin{pmatrix} g_0 & \cdots & g_{n-1} \\ u'_0 & \dots & u'_{n-1} \end{pmatrix}. \quad (15)$$

Let $w \in \{0, \dots, n-1\}$, and the matrix:

$$H_w = \begin{pmatrix} g_0 & \cdots & g_{w-1} & g_w & g_{w+1} & \cdots & g_{n-1} \\ u'_0 & \dots & u'_{w-1} & \boldsymbol{u'_w} & u_{w+1} & \dots & u_{n-1} \end{pmatrix}. \quad (16)$$

The 0th row of H_w equals to that of G. For the first row, the 0th element u'_0 to the wth element u'_w are equal to the 0th element to the wth element in row 1 of G', and the $(w+1)$th element u_{w+1} to the $(n-1)$th element u_{n-1} are equal to the $(w+1)$th element to the $(n-1)$th element of G. Obviously, $H_{n-1} = G'$. When $w \geq 1$, then:

$$H_{w-1} = \begin{pmatrix} g_0 & \cdots & g_{w-1} & g_w & g_{w+1} & \cdots & g_{n-1} \\ u'_0 & \dots & \boldsymbol{u'_{w-1}} & u_w & u_{w+1} & \dots & u_{n-1} \end{pmatrix}. \quad (17)$$

The distinction between H_{w-1} and H_w is the element in the first row and the wth column. When $w \neq k$, let $a_1, a_3 \xleftarrow{\$} \mathbb{Z}_p$ and $a_2 = t$. Recall that k and t are only held by the receiver. Let:

$$\widetilde{H}_w = \begin{pmatrix} g_0 & \cdots & g_{w-1} & \boldsymbol{g^{a_1}} & g_{w+1} & \cdots & g_{n-1} \\ u'_0 & \dots & u'_{w-1} & \boldsymbol{g^{a_3}} & u_{w+1} & \dots & u_{n-1} \end{pmatrix}, \quad (18)$$

$$\widetilde{H}_{w-1} = \begin{pmatrix} g_0 & \cdots & g_{w-1} & \boldsymbol{g^{a_1}} & g_{w+1} & \cdots & g_{n-1} \\ u'_0 & \dots & u'_{w-1} & \boldsymbol{g^{a_1 a_2}} & u_{w+1} & \dots & u_{n-1} \end{pmatrix}. \quad (19)$$

We have $g^{a_1} \overset{c}{\equiv} g_w$ and $g^{a_3} \overset{c}{\equiv} u'_w$ as the four values are generated randomly. It can be shown that $g^{a_1 a_2} = u_w$ as $a_2 = t$. Consequently, we have $H_w \overset{c}{\equiv} H'_w$ and $H_{w-1} \overset{c}{\equiv} H'_{w-1}$. Since $(g^{a_1}, g^{a_2}, g^{a_1 a_2})$ and $(g^{a_1}, g^{a_2}, g^{a_3})$ are indistinguishable under DDH assumption, $H'_w \overset{c}{\equiv} H'_{w-1}$. Then, we have $H_w \overset{c}{\equiv} H_{w-1}$. When $w = 1$, just let $a_3 = (a_3 - v) \bmod p$, and the conclusion is well-supported in the same vein. Then, we can observe the following equation:

$$G \overset{c}{\equiv} H_0 \overset{c}{\equiv} H_1 \overset{c}{\equiv} \dots \overset{c}{\equiv} H_{n-1} = G'. \quad (20)$$

Consequently, $G \overset{c}{\equiv} G'$, namely Hybrid 0 and Hybrid 1 are indistinguishable.

Hybrid 2: Same as Hybrid 0, except we replace all the row number $(c_0, \dots, c_{n_1-1})$ with random $(c'_0, \dots, c'_{n_1-1}) \xleftarrow{\$} \mathbb{Z}_n^{n_1}$.

We prove Hybrid 1 and Hybrid 2 are indistinguishable in two aspects. When $n_1 = 1$, namely there is only a single element in set X. In this case, the relationship among different row numbers is not considered, and we focus on the security of a single row number. When $n_1 > 1$, we focus on the relationship among different row numbers.

As described in Section 4.2, c_i is the row number of Latin square S, which denotes the permutation rule of the column vectors of seed key matrix G. Now the seed key matrix is replaced with G' in Hybrid 1. Let the permutation result of G' be P_i and P'_i according to c_i and c'_i, respectively. Since all the elements of G' are random, $P_i \overset{c}{\equiv} G'$ and $P'_i \overset{c}{\equiv} G'$. Consequently, when $n_1 = 1$, $P_i \overset{c}{\equiv} P'_i$ as well as c_i and c'_i are indistinguishable.

When $n_1 > 1$, for each $z \in \{0, \ldots, n_1 - 1\}$, let vector $C_z = (c'_0, \ldots, c'_z, c_{z+1}, \ldots, c_{n_1-1})$, where the 0th to the zth elements are random and the $(z+1)$th to the $(n_1 - 1)$th elements are true row numbers. Thus, $C_{z-1} = (c'_0, \ldots, c'_{z-1}, c_z, \ldots, c_{n_1-1})$, and the only difference between C_{z-1} and C_z is the zth element. Let c_a be an arbitrary element of C_{z-1} except c_z. Let $l = c_z - c_a$ and $l' = c'_z - c_a$. For an arbitrary column of S, we have $S_{c_z,d} = S_{c_a,d} + l$ and $S_{c'_z,d} = S_{c_a,d} + l'$. Every row of S contains all the elements of $\{0, \ldots, n-1\}$, hence there exist elements equal to $S_{c_z,d}$ and $S_{c'_z,d}$, respectively, in the row c_a. Let the column index of $S_{c_z,d}$ in row c_a be e, and let the column index of $S_{c'_z,d}$ in row c_a be e', namely $S_{c_z,d} = S_{c_a,e}$ and $S_{c'_z,d} = S_{c_a,e'}$. Hence:

$$S_{c_a,e} = S_{c_a,d} + l, \tag{21}$$

$$S_{c_a,e'} = S_{c_a,d} + l'. \tag{22}$$

$S_{c_a,e}$ and $S_{c_a,d}$ can be denoted by the elements of row 0 as $S_{c_a,e} = S_{0,e} + c_a$ and $S_{c_a,d} = S_{0,d} + c_a$. Then plug them into Equations (21) and (22), and further we have the following equations:

$$S_{0,e} = S_{0,d} + l, \tag{23}$$

$$S_{0,e'} = S_{0,d} + l', \tag{24}$$

where e and e' are column indexes. Since the 0th row is sorted randomly, the distinction between e and d is random under any l, resulting in the following equation:

$$(c_a, c_z) \overset{c}{\equiv} (c_a, c'_z). \tag{25}$$

It can be shown that:

$$\begin{aligned} & c_z \overset{c}{\equiv} c'_z \Rightarrow C_z \overset{c}{\equiv} C_{(z-1)} \\ \Rightarrow \quad & (c_0, \ldots, c_{n_1-1}) \overset{c}{\equiv} C_0 \overset{c}{\equiv} \ldots \overset{c}{\equiv} C_{n-1} = (c'_0, \ldots, c'_{n_1-1}). \end{aligned} \tag{26}$$

Consequently, Hybrid 1 and Hybrid 2 are indistinguishable.

Taken together, simulator $\mathcal{S}_1$ can be constructed to simulate receive, such that the simulation is indistinguishable from the real interaction. Consequently, the proposed protocol is resistant against the corrupt sender under the DDH assumption.

6.2. Security against the Corrupt Receiver

Theorem 2. *The proposed protocol is resistant against the corrupt receiver. Formally, we construct a simulator $\mathcal{S}_2$ that takes the inputs $(\mathbb{F}, \mathbb{G}, p, g, S_{0,\cdot}, G, g^t)$and the outputs (h_i, e_i) is indistinguishable from the real sender.*

Proof. To calculate each item x_i in X, the only message that receiver sends to sender is the hash value and the encoding value (h_i, e_i). In this section, we construct a simulator who holds (t, v, f_i), where f_i denotes whether x_i belongs to intersection. $f_i = 0$ denotes x_i does not belong to intersection, and $f_i = 1$ denotes x_i belongs to intersection. We prove simulation is indistinguishable from the real (h_i, e_i) via the following hybrid argument. □

Hybrid 3: The real interaction. To respond each c_i received from receiver, sender samples $r' \overset{\$}{\leftarrow} \mathbb{Z}_p$, generates and sends the verification information (h_i, e_i) honestly as shown in Equations (8) and (9).

Hybrid 4: Simulator $\mathcal{S}_2$ receives c_i, and samples $r' \overset{\$}{\leftarrow} \mathbb{Z}_p$. Then calculates:

$$h'_i = g^{r'}, \tag{27}$$

$$e'_i = g^{tr'} g^{vf_i}. \tag{28}$$

Simulator $\mathcal{S}_2$ sends (h'_i, e'_i) to the receiver. The corrupt receiver calculates:

$$(h\prime_i)^t = g^{r\prime t}. \tag{29}$$

When $f_i = 0$, $e'_i = (h\prime_i)^t$, the receiver obtains $r_i = 0$. When $f_i = 1$, $e'_i = (h\prime_i)^t g^v$, the receiver obtains $r_i = 1$. We have $r_i = f_i$, thus receiver cannot distinguish (h_i, e_i) and (h'_i, e'_i) from the relationship between h and e. Due to the collision resistance of TDH function, distinguishing (h_i, e_i) and (h'_i, e'_i) is the discrete logarithm problem. Consequently, we have $(h_i, e_i) \overset{c}{\equiv} (h'_i, e'_i)$.

Taken together, Hybrid 3 and Hybrid 4 are indistinguishable. The proposed protocol is resistant against the corrupt receiver.

7. Performance Evaluation

7.1. Comparison of Communication. To demonstrate communication performance of the proposed protocol, we report on it in comparison with the state-of-the-art PSI protocols. The communication costs of different protocols are shown in Tables 2 and 3. Since [2, 3, 21–23] proposed both malicious and semihonest protocols, we compare with the semihonest versions only.

We set the computational security $\kappa = 128$ and statistical security $\lambda = 40$. ϕ is the size of elliptic curve group elements (256 is used here). The costs of base OTs are independent of input size and equal to 5κ. n_1, n_2, n denote the sizes of

Table 2: Correspondence between parameter α and n_2.

n_2	2^{16}	2^{20}	2^{24}
α	2^5	2^7	2^8

Table 3: Correspondence between parameter s and n_2.

n_2	2^8	2^9	2^{10}	2^{12}	2^{16}	2^{20}	2^{24}
s	12	10	8	6	4	3	2

Table 4: Theoretical communication costs of PSI protocols with invariant n_2 (in bits).

Protocol	Communication	(n_1, n_2)					
		$(2^{24}, 2^{24})$	$(2^{22}, 2^{24})$	$(2^{21}, 2^{24})$	$(2^{20}, 2^{24})$	$(2^{18}, 2^{24})$	$(2^{16}, 2^{24})$
KKRT [12]	$(3+s)(\lambda+\log(n_1 n_2))n_1+1.2\ln_2+\lvert\text{baseOT}\rvert$	$920n_1$	$2{,}350n_1$	$4{,}365n_1$	$8{,}100n_1$	$31{,}130n_1$	$123{,}280n_1$
CLR17 [25]	$2(\log(3n_2/n'\alpha)+0.15\alpha)n_1\log q$	$16{,}248n_1$	$16{,}248n_1$	$16{,}248n_1$	$16{,}248n_1$	$16{,}248n_1$	$16{,}248n_1$
SpOT [20]	$1.02\ (\lambda+\log n_2+2)n_1+\ln_2+\lvert\text{baseOT}\rvert$	$467n_1$	$1667n_1$	$3{,}267n_1$	$6{,}467n_1$	$25{,}667n_1$	$102{,}467n_1$
PaXoS [21]	$(\lambda+\log(n_1 n_2))n_1+l(2.4n_2+\lambda+\chi)+\lvert\text{baseOT}\rvert$	$1{,}048n_1$	$3{,}926n_1$	$7{,}765n_1$	$15{,}444n_1$	$61{,}522n_1$	$245{,}840n_1$
CM20 [22]	$(\lambda+\log(n_1 n_2))n_1+4.8\kappa n_2+\lvert\text{baseOT}\rvert$	$702n_1$	$2{,}543n_1$	$5{,}000n_1$	$9{,}914n_1$	$39{,}403n_1$	$157{,}366n_1$
RS21 [3]	$(\lambda+\log(n_1 n_2))n_1+2^{17}\kappa n_2^{0.05}+\kappa n_2+\lvert\text{baseOT}\rvert$	*$218n_1$*	*$607n_1$*	$1{,}127n_1$	$2{,}168n_1$	$8{,}421n_1$	$33{,}436n_1$
RT21 [2]	$(\lambda+\log(n_1 n_2))n_1+\phi n_2+\phi$	$344n_1$	$1{,}110n_1$	$2{,}133n_1$	$4{,}180n_1$	$16{,}466n_1$	$65{,}616n_1$
GPR21 [23]	$(\lambda+\log(n_1 n_2))n_1+l(1.3n_2+0.5\log n_2+\lambda)+\lvert\text{baseOT}\rvert$	$608n_1$	$2{,}166n_1$	$4{,}245n_1$	$8{,}404n_1$	$33{,}362n_1$	$133{,}200n_1$
Authors	$(\log n+2\phi)n_1$	$612n_1$	$612n_1$	*$612n_1$*	*$612n_1$*	*$612n_1$*	*$612n_1$*

Italic values indicate the best results of each column.

Table 5: Theoretical communication costs of PSI protocols with invariant n_1 (in bits).

Protocol	Communication	(n_1, n_2)					
		$(2^8, 2^8)$	$(2^8, 2^9)$	$(2^8, 2^{10})$	$(2^8, 2^{12})$	$(2^8, 2^{16})$	$(2^8, 2^{20})$
KKRT [12]	$(3+s)(\lambda+\log(n_1 n_2))n_1+1.2\ln_2+\lvert\text{baseOT}\rvert$	$1{,}322n_1$	$1703n_1$	$2{,}560n_1$	$8{,}222n_1$	$123{,}330n_1$	$1{,}966{,}558n_1$
CLR17 [25]	$2(\log(3n_2/n'\alpha)+0.15\alpha)n_1\log q$	—	—	—	—	$1814n_1$	$7{,}856n_1$
SpOT [20]	$1.02\ (\lambda+\log n_2+2)n_1+ln_2+\lvert\text{baseOT}\rvert$	$453n_1$	$854n_1$	$1655n_1$	$6{,}457n_1$	$102{,}461n_1$	$1{,}638{,}465n_1$
PaXoS [21]	$(\lambda+\log(n_1 n_2))n_1+l(2.4n_2+\lambda+\chi)+\lvert\text{baseOT}\rvert$	$1{,}093n_1$	$2{,}054n_1$	$3{,}975n_1$	$15{,}497n_1$	$245{,}901n_1$	$3{,}932{,}305n_1$
CM20 [22]	$(\lambda+\log(n_1 n_2))n_1+4.8\kappa n_2+\lvert\text{baseOT}\rvert$	$672n_1$	$1{,}288n_1$	$2{,}518n_1$	$9{,}892n_1$	$157{,}352n_1$	$2{,}516{,}653n_1$
RS21 [3]	$(\lambda+\log(n_1 n_2))n_1+2^{17}\kappa n_2^{0.05}+\kappa n_2+\lvert\text{baseOT}\rvert$	$86{,}661n_1$	$89{,}840n_1$	$93{,}254n_1$	$101{,}444n_1$	$146{,}939n_1$	$655{,}430n_1$
RT21 [2]	$(\lambda+\log(n_1 n_2))n_1+\phi n_2+\phi$	*$313n_1$*	*$570n_1$*	$1{,}083n_1$	$4{,}157n_1$	$65{,}601n_1$	$1{,}048{,}645n_1$
GPR21 [23]	$(\lambda+\log(n_1 n_2))n_1+l(1.3n_2+0.5\log n_2+\lambda)+\lvert\text{baseOT}\rvert$	$647n_1$	$1{,}169n_1$	$2{,}210n_1$	$8{,}454n_1$	$133{,}261n_1$	$2{,}130{,}068n_1$
Authors	$(\log n+2\phi)n_1$	$612n_1$	$612n_1$	*$612n_1$*	*$612n_1$*	*$612n_1$*	*$612n_1$*

Italic values indicate the best results of each column.

receiver's set, sender's set, and input field $\mathbb{F}$, and we set $n = 2^{100}$. n', q, α are the parameters of FHE, where $n' = 2^{13}$ and $q = 2^{189} - 2^{21} + 9 \times 2^{15} + 1$. α increases as n_2 get higher, and Table 2 shows the distinct values of α under distinct n_2 according to CLR17 [25]. $l = \log n$ denotes the width of OT extension matrix. $\chi \log n_1$ is the upper bound on the number of cycles in a cuckoo graph of PaXoS. s is the maximum stash size for cuckoo hashing. When three hash functions are utilized to map n_2 elements to $1.2n_2$ bins, the relationship between n_2 and s is shown in Table 3 according to KKRT [12].

Table 4 shows the communication costs of different protocols when $n_2 = 2^{24}$ and n_1 ranges from 2^{24} to 2^{16}. It is apparent from this table that the communication cost of the proposed protocol is proportional to n_1 and is not related to n_2, thus the communication cost decreases as n_1 get smaller. When (n_1, n_2) equal to $(2^{22}, 2^{24})$ or $(2^{24}, 2^{24})$, the communication cost of the proposed protocol is higher than some of the other protocols. However, when $n_1 \le 2^{21}$, the communication cost of the proposed protocol is the lowest. The reason is the communication costs of the other protocols are related to both n_1 and n_2. When the sizes of the two sets are $(2^{21}, 2^{24})$, the ratio of them is 2^3, and the communication cost required by our protocol is 55.14% of the state-of-the-art protocol. When the size of the two sets is $(2^{16}, 2^{24})$, the communication cost required by our protocol is only 0.6% of the state-of-the-art protocol.

Table 5 shows the communication costs of different protocols when $n_1 = 2^8$ and n_2 ranges from 2^8 to 2^{24}. It is shown that the communication cost of the proposed protocol is invariant while the communication costs of other protocols rapidly increase with n_2. When $n_2 \ge 2^{10}$, the communication

TABLE 6: Experimental communication cost comparison.

Protocol	Communication cost (KB)			
Size	$(2^8, 2^8)$	$(2^8, 2^{10})$	$(2^8, 2^{12})$	$(2^8, 2^{14})$
KKRT [12]	43.8	101.8	224.6	542.7
SpOT [20]	30.72	71.68	235.52	870.4
PaXoS [21]	69.83	158.8	274.8	665.6
CM20 [22]	43	99.3	325.6	1234.9
RT21 [2]	*16.04*	40.04	136.04	520.04
Authors	17	*17*	*17*	*17*

Italic values indicate the best results of each column.

TABLE 7: Online computation cost comparison.

Protocol	Online time (s)											
Size	$(2^8, 2^8)$			$(2^8, 2^{10})$			$(2^8, 2^{12})$			$(2^8, 2^{14})$		
Bandwidth	LAN	1 Mbps	100 Kbps	LAN	1 Mbps	100 Kbps	LAN	1 Mbps	100 Kbps	LAN	1 Mbps	100 Kbps
KKRT [12]	0.071	0.795	7.177	*0.083*	2.247	18.682	0.103	7.295	63.194	*0.167*	28.012	236.499
SpOT [20]	0.068	0.187	1.504	0.217	0.915	6.379	0.989	3.441	28.674	4.34	14.203	161.215
PaXoS [21]	0.094	0.674	12.058	0.133	1.514	30.3	0.184	2.887	105.429	0.373	6.138	387.782
CM20 [22]	0.048	*0.109*	2.899	0.579	1.912	10.727	0.2003	4.407	38.193	0.277	17.614	162.895
RT21 [2]	*0.047*	0.305	*0.458*	0.563	*0.547*	2.932	*0.099*	*1.16*	11.466	0.298	*4.355*	44.233
Authors	0.587	0.686	1.609	1.791	1.879	*2.842*	8.389	8.402	*9.12*	29.32	30.272	*30.338*

Italic values indicate the best results of each column.

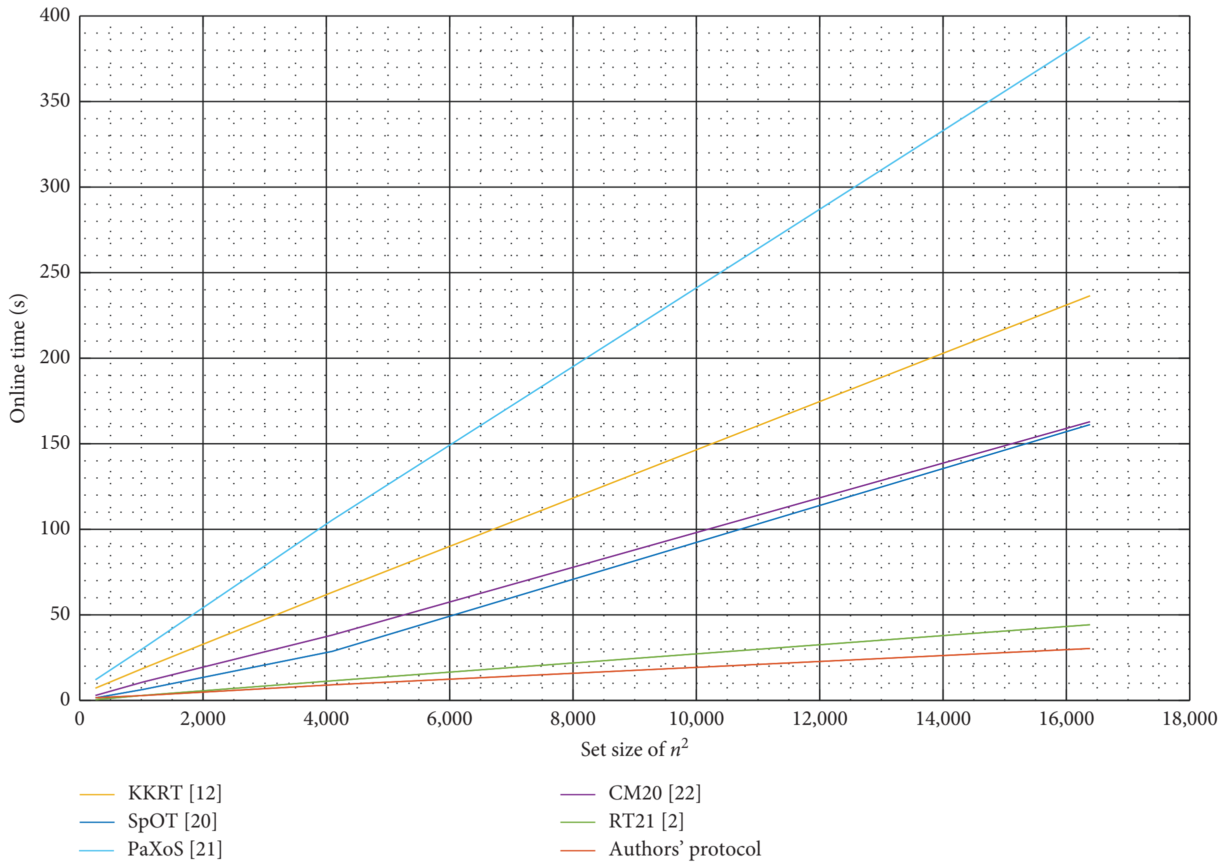

FIGURE 4: Online computation cost comparison when $n_1 = 2^8$.

costs of other protocols are higher than the proposed protocol. In addition, the advantage of the proposed protocol is increasingly apparent as n_1 get larger.

7.2. Experimental Results. In order to evaluate the performance of our PSI protocol, we built and evaluated an implementation. Our source code is available on GitHub: https://github.com/TAN-OpenLab/Unbanlanced-PSI.

We implement our protocol in C++, and run our protocol on Ubuntu 16.04 with 8 GB RAM. We set $n = 2^{16}$, and other parameters are the same as in Section 7.1. We set the values of n_1 and n_2, and record communication cost and online time. As Table 6 shows, the communication cost of the proposed protocol is 17 KB when $n_1 = 2^8$ and is not related to n_2. The advantage of the proposed protocol over communication cost is increasingly apparent as n_2 increases. Particularly, when $n_1 = 2^8$ and $n_2 = 2^{10}$, the communication cost of our protocol is 42.5% of the best existing protocol RT21 [2]. When $n_2 = 2^{14}$, our protocol requires only 3.3% communication cost of RT21 [2]. Although it has been shown from online time that the computation cost still remains to be reduced.

Due to the low communication cost, the proposed protocol is more suitable for the scenarios with low network bandwidth. As shown in Table 7, for the specific set sizes, the online time changes little with network LAN, 1 Mbps and 100 Kbps bandwidths. Although our protocol is not the fastest with network LAN and 1 Mbps bandwidth, we gain an apparent advantage with 100 Kbps bandwidth. With the set sizes $(2^8, 2^{14})$ and 100 Kbps bandwidth, our protocol achieves a 7.8x speedup compared to KKRT [12], a 5.31x speedup compared to SpOT [20], a 12.78x speedup compared to PaXoS [21], a 5.37x speedup compared to CM20 [22] and 1.46x speedup compared to RT21 [2]. Thus, our protocol is applicable to low bandwidth networks. With unbalanced sets $(2^8, 2^{10})$, $(2^8, 2^{12})$, and $(2^8, 2^{14})$, our protocol is faster than other protocols under 100 Kbps bandwidth, and we achieve 1.03x, 1.26x, and 1.46x speedup compared to RT21 [2], which proves our protocol is applicable to two sets with larger difference.

We present the computation cost intuitively in Figure 4. When we fix the value of n_1 to 2^8 and set the bandwidth to 100 Kbps, it is evident that for all protocols, the computation cost rises as the set size increases. The relationship between the set size of n_2 and the online time is linear, and the online time of our protocol is the lowest compared to the other protocols.

8. Conclusion

We propose a semihonest efficient PSI protocol for unbalanced sets based on trapdoor hashing and Latin square, which relies on the DDH assumption. By employing trapdoor hashing, the communication cost is only dependent on the smaller set, effectively eliminating the impact of the larger set size on communication cost. The use of Latin square reduces the number of times encoding keys need to be sent, enhancing communication efficiency. The results of the performance analysis clearly indicate that the proposed protocol exhibits optimization in terms of communication cost specifically for unbalanced sets on low bandwidth. Furthermore, the advantage of our protocol becomes more prominent as the disparity between the sizes of n_1 and n_2 increases. In future work, our focus will be directed toward reducing the computation time and storage cost associated with our proposed protocol.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was funded by the National Key Research and Development Program of China under Grant No. 2023YFC3306201, the Fundamental Research Funds for the Central Universities No. N2317004, and the National Natural Science Foundation of China No. 61772125.

References

[1] E. David, K. Vladimir, and R. Mike, *A Pragmatic Introduction to Secure Multi-Party Computation*, Foundations & Trends in Privacy & Security, 2018.

[2] M. Rosulek and N. Trieu, "Compact and Malicious Private Set Intersection for Small Sets," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1166–1181, Association for Computing Machinery, 2021.

[3] P. Rindal and P. Schoppmann, "VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE," in *Advances in Cryptology—EUROCRYPT 2021*, pp. 901–930, Springer International Publishing, 2021.

[4] J. Gao, C. Surana, and N. Trieu, "Secure contact tracing platform from simplest private set intersection cardinality," *IET Information Security*, vol. 16, no. 5, pp. 346–361, 2022.

[5] M. Ion, B. Kreuter, A. E. Nergiz et al., "On deploying secure computing: private intersection-sum-with-cardinality," in *2020 IEEE European Symposium on Security and Privacy*, pp. 370–389, IEEE, Genoa, Italy, 2020.

[6] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert, "Mobile private contact discovery at scale," in *28th USENIX Security Symposium*, pp. 1–20, USENIX, 2019.

[7] C. Meadows, "A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party," in *IEEE Symposium on Security & Privacy*, pp. 134–134, IEEE, 1986.

[8] B. A. Huberman, M. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 78–86, Association for Computing Machinery, 1999.

[9] S. Jarecki and X. Liu, "Fast secure computation of set intersection," in *Security and Cryptography for Networks*, J. A. Garay and R. De Prisco, Eds., vol. 6280 of *Lecture Notes in Computer Science*, pp. 418–435, Springer, 2010.

[10] E. De Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in Malicious model," in *Advances in Cryptology—ASIACRYPT 2010*, M. Abe, Ed.,

vol. 6477 of *Lecture Notes in Computer Science*, pp. 213–231, Springer, 2010.

[11] B. Pinkas, T. Schneider, and M. Zohner, *Faster private Set Intersection Based on OT Extension*, vol. 21, pp. 797–812, ACM Transactions on Privacy & Security, 2014.

[12] 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*, pp. 818–829, Association for Computing Machinery, 2016.

[13] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, *Phasing: Private Set Intersection Using Permutation-Based Hashing*, pp. 515–530, USENIX Security Symposium, 2015.

[14] P. Rindal and M. Rosulek, "Improved private set intersection against malicious adversaries," in *Advances in Cryptology—EUROCRYPT 2017*, J. S. Coron and J. Nielsen, Eds., vol. 10210 of *Lecture Notes in Computer Science*, pp. 235–259, Springer, Cham, 2017.

[15] P. Rindal and M. Rosulek, "Malicious-secure private set intersection via dual execution," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1229–1242, Association for Computing Machinery, 2017.

[16] M. Orrù, E. Orsini, and P. Scholl, "Actively secure 1-out-of-N OT extension with application to private set intersection," in *Topics in Cryptology—CT-RSA 2017*, H. Handschuh, Ed., vol. 10159 of *Lecture Notes in Computer Science*, pp. 381–396, Springer, 2017.

[17] C. Hazay and M. Venkitasubramaniam, "Scalable multi-party private set-intersection, 2017, public-key cryptography—PKC," in *Public-Key Cryptography – PKC 2017*, S. Fehr, Ed., vol. 10174 of *Lecture Notes in Computer Science*, pp. 175–203, Springer, 2017.

[18] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on OT extension," *ACM Transactions on Privacy and Security*, vol. 21, no. 2, pp. 1–35, 2018.

[19] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based PSI via cuckoo hashing," in *EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds., vol. 10822 of *LNCS*, pp. 125–157, Springer, 2018.

[20] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "SpOT-light: lightweight private set intersection from sparse OT extension," in *Advances in Cryptology—CRYPTO 2019*, vol. 11694 of *Lecture Notes in Computer Science*, pp. 401–431, Springer, Cham, 2019.

[21] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "PSI from PaXoS: fast, malicious private set intersection," in *Advances in Cryptology—EUROCRYPT 2020*, A. Canteaut and Y. Ishai, Eds., vol. 12106 of *Lecture Notes in Computer Science*, pp. 739–767, Springer, 2020.

[22] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious PRF," in *Advances in Cryptology—CRYPTO 2020*, D. Micciancio and T. Ristenpart, Eds., vol. 12172 of *Lecture Notes in Computer Science*, pp. 34–63, Springer, Cham, 2020.

[23] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology – CRYPTO 2021*, T. Malkin and C. Peikert, Eds., vol. 12826 of *Lecture Notes in Computer Science*, pp. 395–425, Springer, Cham, 2021.

[24] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology—EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, pp. 1–19, Springer, Berlin, Heidelberg, 2004.

[25] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1243–1255, Association for Computing Machinery, 2017.

[26] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K.-K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9390–9401, 2020.

[27] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1885–1899, 2022.

[28] Q. Wang, D. Wang, C. Cheng, and D. He, "Quantum2FA: efficient quantum-resistant two-factor authentication scheme for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 193–208, 2023.

[29] N. Döttling, S. Garg, Y. Ishai, G. Malavolta, T. Mour, and R. Ostrovsky, "Trapdoor hash functions and their applications," in *Advances in Cryptology—CRYPTO 2019*, A. Boldyreva and B. Micciancio, Eds., vol. 11694 of *Lecture Notes in Computer Science*, pp. 3–32, Springer, Cham, 2019, 2019.

[30] M. Chase, S. Garg, M. Hajiabadi, J. Li, and P. Miao, "Amortizing rate-1 OT and applications to PIR and PSI," in *Theory of Cryptography*, K. Nissim and B. Waters, Eds., vol. 13044 of *Lecture Notes in Computer Science*, pp. 126–156, Springer, Cham, 2021.

[31] V. Kolesnikov and R. Kumaresan, "Improved OT extension for transferring short secrets," in *Advances in Cryptology—CRYPTO 2013*, R. Canetti and J. A. Garay, Eds., vol. 8043 of *Lecture Notes in Computer Science*, pp. 54–70, Springer, Berlin, Heidelberg, 2013.

[32] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Advances in Cryptology—CRYPTO 2003*, D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, pp. 145–161, Springer, Berlin, Heidelberg, 2003, 2003.

[33] O. Goldreich, *Foundations of Cryptography: Volume 2: Basic Applications*, Cambridge University Press, 2004.

[34] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[35] R. A. Brualdi, *Introductory Combinatorics*, University of Wisconsin, 5th edition, 2009.

[36] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: single database, computationally-private information retrieval," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pp. 364–373, IEEE, Miami Beach, FL, USA, 1997.

[37] Q. Wang and D. Wang, "Understanding failures in security proofs of multi-factor authentication for mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 597–612, 2023.

[38] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2015.

[39] N. Koblitz and A. J. Menezes, "Another look at "Provable Security"," *Journal of Cryptology*, vol. 20, no. 1, pp. 3–37, 2007.