SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2024, Vol. 67, Iss. 3, 132105:1–132105:15
https://doi.org/10.1007/s11432-022-3717-9

Unbalanced private set intersection with linear communication complexity

Quanyu ZHAO[1], Bingbing JIANG[2*], Yuan ZHANG[1], Heng WANG[1], Yunlong MAO[1] & Sheng ZHONG[1]

[1]*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023*, China;*
[2]*Purple Mountain Lab, Nanjing* 211111*, China*

Received 13 December 2022/Revised 17 February 2023/Accepted 7 March 2023/Published online 5 February 2024

Abstract The private set intersection (PSI) protocol allows two parties holding a set of integers to compute the intersection of their sets without revealing any additional information to each other. The unbalanced PSI schemes consider a specific setting where a client holds a small set of the size n and a server holds a much larger set of the size m ($n \ll m$). The communication overhead of state-of-the-art balanced PSI schemes is $O(m+n)$ and the unbalanced PSI schemes are $O(n\log m)$. In this paper, we propose a novel secure unbalanced PSI protocol based on a hash proof system. The communication complexity of our protocol grows only linearly with the size of the small set. In other words, our protocol achieves communication overhead of $O(n)$. We test the performance on a personal computer (PC) machine with a local area network (LAN) setting for the network. The experimental results demonstrate that the client only takes 2.01 s of online computation, 4.27 MB of round trip communication to intersect 1600 pieces of 32-bit integers with 2^{20} pieces of 32-bit integers with the security parameter $\lambda = 512$. Our protocol is efficient and can be applied to resource-constrained devices, such as cell phones.

Keywords unbalanced private set intersection, Hash proof system, linear communication complexity, small set, resource-constrained devices

1 Introduction

The private set intersection (PSI) protocol allows two parties, commonly referred to as server and client, to have a private set of integers and obtain the intersection of their sets without revealing any additional information. It has a high reputation as a special application scenario for secure multi-party computation. PSI is widely used as a security tool in a large number of emerging areas, including private contact discovery [1], DNA testing and pattern matching [2], passwords leakage validation [3], and privacy-preserving location sharing [4].

Meadows first proposes a PSI scheme based on public key infrastructure (PKI) in 1986 [5]. Despite the multiplicative homomorphic of Diffie-Hellman key exchange and the expensive modular exponentiation operations, especially the computational complexity grows linearly with the size of its sets, this scheme remains extremely important to study for the following reasons: (1) a significant advantage in terms of communication cost; (2) the application on resource-constrained devices.

A naive approach is described as: client and server encode integers into hash values and compare the hash values to obtain the intersection directly. However, this approach leaks privacy. Recently, a series of PSI schemes [6–21] based on different tools have been extensively studied. We always categorize them into four basic types, including PSI based on fully homomorphic encryption (FHE), the third parties, hash function, and oblivious transfer (OT) or oblivious transfer extension.

FHE-based PSI schemes [6,22] are widely studied for their low communication overhead. To the best of our knowledge, the communication complexity of FHE-based PSI schemes [6, 22] is $O(n\log m)$. However, the computational overhead grows rapidly with the size of the input and the depth of the arithmetic circuit. This reason hinders the implementation of these schemes in practical applications.

* Corresponding author (email: jiangbingbing@pmlabs.com.cn)

The third parties-based PSI schemes [8, 23–25] reduce the computation and communication overhead. However, introducing a third party may not be suitable for some practical applications, and the server may collude with parties, resulting in a privacy leaking.

Hash function-based PSI schemes convert the integers of two sets into long binary strings of special probabilistic data structures (Cuckoo hashing or Boolean circuits) [26,27]. These approaches reduce computational overhead. However, the multi-layer circuit is essential for the correct intersection, deploying a multi-layer circuit in advance is a challenge for resource-constrained devices. In addition, the lack of deletion function is also a drawback of these solutions.

OT or OT extension-based PSI schemes [7–9, 28–31] are proposed to reduce the computational complexity. Some OT extension schemes obtain many oblivious transfers with a small number of public-key operations only relating to the security parameters, independent of the input set size. Generally, OT-based PSI protocols require more communication than Diffie-Hellman(DH)-PSI schemes. However, Pinkas et al. [32] present an OT-based PSI scheme with less communication than DH-PSI schemes.

According to the participants' set sizes, PSI schemes are classified into two categories: balanced and unbalanced PSI schemes. The balanced PSI scheme considers a situation where two parties hold two sets with roughly equivalent sizes. The two parties may be two companies with similar capabilities, including computation, storage, and communication.

Previous schemes [14–16, 33, 34] rely on public key encryption, oblivious polynomial evaluation, fully homomorphic encryption, and number-theoretic assumptions to compute the intersection while guaranteeing the privacy. However, these methods usually require high communication and computation overheads. Some schemes [9, 10, 28, 30, 35] employ oblivious pseudo-random function (OPRF), boolean circuit, and oblivious transfer to reduce communication and computation. However, these schemes require both the server and the client to encode the integers together and compute the intersection. The communication complexity of these schemes is $O(m+n)$. Many studies [16,27,28,33,35–37] have been proposed to reduce communication and computational costs in PSI schemes. Hazay et al. [16] proposed a framework with communication complexity of $O(n+m)$ and computational complexity of $O(n+m(\text{loglog}(n+p(t))))$ modular exponentiations, where $p(t)$ denotes oblivious transfers for realizing the oblivious pseudorandom function evaluation. Falkl et al. [28] constructed a hash function-based PSI scheme that reduces the communication complexity from $\omega(m\lambda)$ to $O(m\lambda)$, where λ is a security parameter. Schneider et al. [35] and Pinkas et al. [27] utilized circuit and OPRF to construct PSI schemes with communication complexity of $O(m)$ and computational complexity of $O(m(\text{loglog}n)^2)$. The linear communication in these schemes implies that it increases linearly with the size of the two sets.

The unbalanced PSI scheme considers a situation where the size of the client's set is much smaller than that of the server's set. In addition, its capabilities including computational, storage, and communication are much weaker than those of the server. The client may not be able to afford the high communication and computation overheads. For example, a resource-constrained device (e.g., cellphone) with a smaller set desires to perform PSI with a service provider (e.g., Whatsapp) with a larger set. Therefore, balanced PSI schemes [6–10, 12, 16, 19, 20, 27, 28, 30, 33, 35, 37] may not be an effective way to accomplish this PSI task.

To optimize the communication overhead of unbalanced PSI schemes, Resende et al. [36] utilized OPRF to code the integers. Client obtains the ciphertext $Y' = \{\text{OPRF}_k(y), y \in Y\}$ without revealing Y, and server computes $X' = \{\text{OPRF}_k(x), x \in X\}$ locally by using the shared key k. Despite that it compresses X' before forwarding to the client, the communication still increases linearly with the size of the server set. The communication complexity may not be acceptable for resource-constrained devices. In addition, reducing communication through compression techniques introduces a certain amount of false positives. These false positives may not be acceptable in certain applications of PSI such as sample alignment in machine learning. Similar to the scheme [36], protocol [38] uses OPRF and relatively conservative compression techniques to construct the PSI scheme. The computation and communication overhead is divided into two phases, the setup and the online phase. However, the setup phase contains a preprocessing phase whose communication increases linearly with the size of the larger set. Moreover, this protocol also introduces a false positive.

In addition, Chen et al. [6] established an unbalanced PSI scheme based on FHE. This PSI scheme employs several techniques, including batch operations and hashing, to reduce circuit depth with predetermined multiplication to optimize performance. It achieves good performance for PSI schemes with only 32-bit integers and scales well to large sets on the server side. The protocol [22] supports integers of arbitrary length with communication complexity of $O(n\text{log}m)$. FHE is still far from being a generic

Table 1 Notations used in our protocol

Notations	Meaning
HPS	Hash proof system
HPK/HSK	Public key space/private key space
hsk	An s-dimensional private key vector chosen from the private key space
hpk	A t-dimensional public key vector chosen from the public key space
$\boldsymbol{A}$	A matrix of $s \times t$
$\boldsymbol{w}$	A t-dimensional witness vector
$\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t$	s-dimensional vectors
$\boldsymbol{D}/\boldsymbol{L}$	s-dimensional vector set/a subset of D
N	A large prime
$N_1, N_2, \ldots, N_n$	Secure RSA modules, N is bigger than N_1, $N_2, \ldots, N_n$
$\phi(\cdot)$	The Euler function
λ	The security parameter
c_i, c_j, $z_{i,j}$, $\mathbf{tc}_j$, C_j	Ciphertexts
$i \in [m]$, $j \in [n]$, $k \in [t]$	$i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, $k = 1, 2, \ldots, t$

solution for encrypting data, but that does not prevent it from being an important tool for specific applications, e.g., evaluating the AES circuit [39], or computing the edit distance [40].

Recently, the main goal of the current research [6, 22, 26, 36] on unbalanced PSI schemes is to optimize the communication and computation performance. We propose an unbalanced PSI scheme in this paper, whose communication complexity is only linearly related to the size of the small set, but not to the size of the large set. We implement our experiments and the experimental results demonstrate that our communication overhead is only linearly related to the size of the small set.

We propose a two-party private set intersection protocol based on a hash proof system. The client utilizes the hash proof system to encode integers as ciphertexts and transmits them to the server. The server matches them with the local ciphertexts, combines the matching results, and transmits the results back to the client. The client performs some lightweight computations to obtain the intersection of their two sets. Thus, the communication complexity only increases linearly with the size of the smaller set. Moreover, the main computational tasks will be performed by the server in a large data center, while the client performs only lightweight computations. Our protocol is well suited for resource-constrained clients. We summarize our contributions below.

• We propose a PSI protocol based on a hash proof system instead of using high-cost or high-complexity cryptographic tools such as homomorphic circuits and OT-hybrid models.

• We prove the security and correctness of our PSI scheme, and our protocol determines the correct intersection for two sets without the false positive.

• In our protocol, the communication overhead increases linearly only with the size of the smaller set. Moreover, it requires only one round of communication between the server and the client. Meanwhile, our scheme reduces the latency of network communication.

• Our experimental results demonstrate that our PSI protocol is an efficient approach in terms of communication. In addition, the communication and computation overheads are also acceptable for resource-constrained devices such as mobile phones.

For a clear understanding of our protocol, the notations used in our protocol are listed in Table 1.

2 Preliminaries

Fine-grained cryptographic primitives [41,42] are based on weak complexity-theoretic assumptions such as NP $\nsubseteq$ BPP, and these primitives are secure against any probabilistic polynomial time (PPT) adversaries. One-way permutations, hash proof systems [41–43], and trapdoor one-way functions are regarded as three key constructs of fine-grained cryptographic primitives. In other words, using a hash proof system to construct a cryptographic scheme possesses these properties of fine-grained cryptographic primitives. The hash proof systems [41–43] are treated as a non-interactive zero-knowledge proof system for a language. It is first proposed to construct a CCA secure PKE scheme [44, 45]. Subsequently, it is often used to solve subset membership problems (SMP) [43].

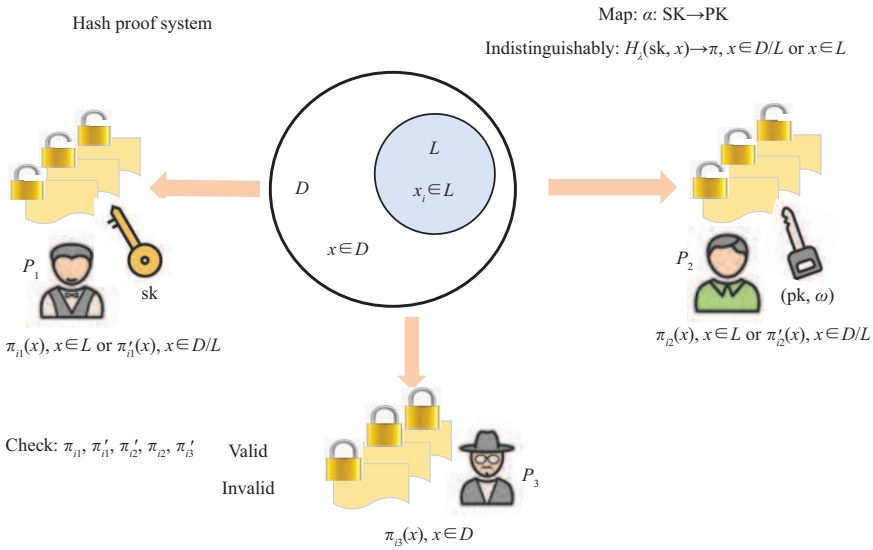

Figure 1 Different parties checking the validity of the witness.

Hash proof system. We will employ the notations of a hash proof system similar to [44]. A complete hash proof system is a PPT algorithm consisting of an array HPS $= \{\boldsymbol{D}, \boldsymbol{L}, \boldsymbol{W}, R, \mathbf{HSK}, \mathbf{HPK}, \pi, \alpha, H, F,$ aux$\}$. HPS chooses an efficiently computable map $\alpha : \mathbf{HPK} \longleftarrow \mathbf{HSK}$, **HSK** and **HPK** denote private key space and the corresponding public key space. $\boldsymbol{D}$ denotes a non-empty finite field and $\boldsymbol{L}$ is a subset of $\boldsymbol{D}$ ($\boldsymbol{L} \subset \boldsymbol{D}$). $\boldsymbol{W}$ is witness space, a statement $\boldsymbol{x} \in \boldsymbol{L}$ iff there exists a witness $\boldsymbol{w} \in \boldsymbol{W}$ for $(\boldsymbol{x}, \boldsymbol{w}) \in R$. R is an efficiently computable binary relation. HPS efficiently samples a private key **hsk** randomly from **HSK**, and chooses a family of efficiently computable function $H_\lambda : \{\pi | \pi = H_\lambda(\mathbf{hsk}, \boldsymbol{x}), \mathbf{hsk} \in \mathbf{HSK}, \boldsymbol{x} \in \boldsymbol{L}\}$. HPS chooses an efficiently computable map $F_\lambda : \{\pi | \pi = F_\lambda(\boldsymbol{x}, \boldsymbol{w}, \mathbf{hpk})\}$, π is the proof space, and aux denotes the auxiliary information. HPS allows one to generate a valid proof π for a statement $\boldsymbol{x}$. A valid proof π proves $\boldsymbol{x} \in \boldsymbol{L}$ by using $\boldsymbol{w}$ and **hpk** or only using **hsk**. Each statement $\boldsymbol{x}$ is sampled indistinguishably from inside and outside $\boldsymbol{L}$. This means that one has the ability to prove $\boldsymbol{x} \in \boldsymbol{L}$ based on $\boldsymbol{w}$ and **hpk**, and one holding **hsk** also has the same ability to prove $\boldsymbol{x} \in \boldsymbol{L}$.

We detail the hash proof system in Figure 1. It includes three types of parties P_1, P_2, P_3 with different abilities to prove $\boldsymbol{x} \in \boldsymbol{L}$ or $\boldsymbol{x} \in \boldsymbol{D}/L$. P_1 holds a golden key (private key **hsk**) who obtains valid evidence π by computing $\pi_{i1}(\boldsymbol{x}) = H_\lambda(\mathbf{hsk}, \boldsymbol{x}), \boldsymbol{x} \in \boldsymbol{L}$ or $\pi'_{i1}(\boldsymbol{x}) = H_\lambda(\mathbf{hsk}, \boldsymbol{x}), \boldsymbol{x} \in \boldsymbol{D}/\boldsymbol{L}$. P_2 holds a silver key (public key **hpk** and the corresponding witness $\boldsymbol{w}$) who obtains valid evidence π by computing $\pi_{i2}(\boldsymbol{x}) = F_\lambda(\boldsymbol{x}, \boldsymbol{w}, \mathbf{hpk}), \boldsymbol{x} \in \boldsymbol{L}$. In addition, P_2 gets invalid evidence π by computing $\pi'_{i2}(\boldsymbol{x}) = F_\lambda(\mathbf{hpk}, \boldsymbol{w}', \boldsymbol{x}), \boldsymbol{x} \in \boldsymbol{D}/\boldsymbol{L}$; $\boldsymbol{w}'$ is chosen randomly from the witness space $\boldsymbol{W}$. P_3 does not hold a key who obtains invalid evidence π by computing $\pi'_{i3}(\boldsymbol{x}) = F_\lambda(\boldsymbol{x}, \boldsymbol{w}', \mathbf{hpk}), \boldsymbol{x} \in \boldsymbol{D}/L$. Finally, checking the validity of the evidence will judge $\boldsymbol{x} \in \boldsymbol{L}$ or $\boldsymbol{x} \in \boldsymbol{D}/L$. A hash proof system consists of three algorithms.

• Key Generation algorithm $I(1^\lambda)$: The key generation algorithm obtains a pair of public and private keys $(\mathbf{hpk}, \mathbf{hsk})$ of length λ, and α: $\alpha(\mathbf{hsk}) = \mathbf{hpk}$, defining a function $H_\lambda(\mathbf{hsk}) : \boldsymbol{D} \to \pi$ for **hsk**.

• Private evaluation algorithm $H_\lambda(\cdot)$: For any $(\boldsymbol{x}, \boldsymbol{w}) \in R$, each private key holder can efficiently compute $\pi = H_\lambda(\mathbf{hsk}, \boldsymbol{x})$ using a PPT algorithm.

• Public evaluation algorithm $F_\lambda(\cdot)$: For any $(\boldsymbol{x}, \boldsymbol{w}) \in R$, the prover efficiently computes $\pi = F_\lambda(\mathbf{hpk}, \boldsymbol{x}, \boldsymbol{w})$ based on **hpk** and $\boldsymbol{w}$ with the absence of **hsk** using a PPT algorithm.

Then, $\forall \boldsymbol{x} \in \boldsymbol{L}$, $H_\lambda(\mathbf{hsk}, \boldsymbol{x}) = F_\lambda(\mathbf{hpk}, \boldsymbol{x}, \boldsymbol{w})$. Furthermore, HPS satisfies two properties, universality and smoothness. Universality means that the entropy of π should be large enough for fixed $\boldsymbol{x} \notin \boldsymbol{L}$ and **hpk**. Smoothness means that the distribution of π must be close to the uniform distribution of the proof space π for $\boldsymbol{x} \notin \boldsymbol{L}$ and **hsk**. Our PSI protocol requires the property of universality and smoothness.

Definition 1. HPS satisfies universality iff the probability of an adversary Adver successfully generating the proof π from the guessing $\boldsymbol{x}$ using a PPT algorithm without private key $\mathbf{hsk}^\star$ or witness $\boldsymbol{w}^\star$ is negligible.

$$\Pr[H_\lambda^{\mathrm{Adver}}(\mathbf{hsk}^\star, \boldsymbol{x}) = \pi \wedge F_\lambda^{\mathrm{Adver}}(\mathbf{hpk}, \boldsymbol{x}, \boldsymbol{w}^\star) = \pi] < \varepsilon,$$

where ε is a negligible parameter, $\mathbf{hsk}^\star$ and $\boldsymbol{w}^\star$ are chosen randomly in the private key and proof space.

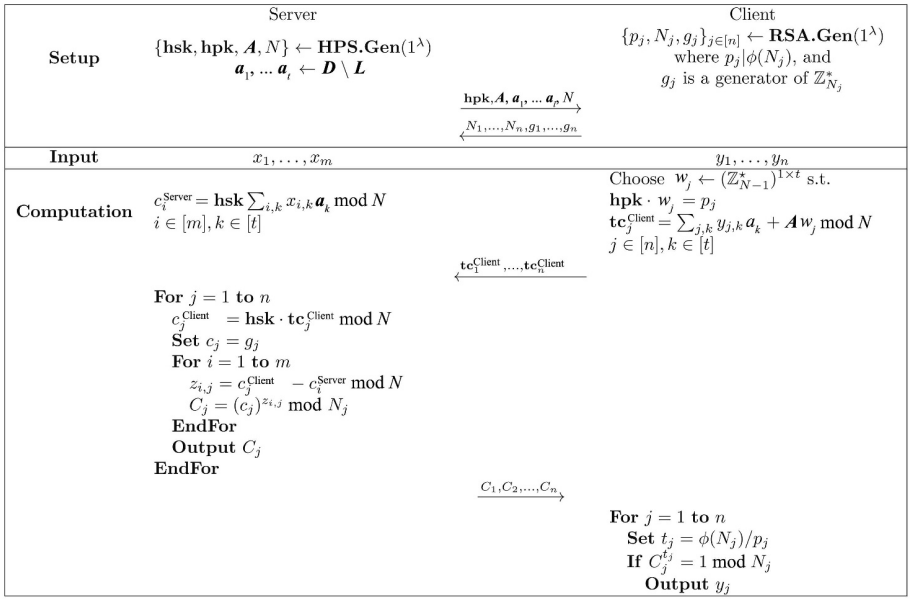

Figure 2 Details of our protocol.

Definition 2. HPS satisfies smoothness iff the probability of an adversary Adver successfully distinguishing the proof π and $\pi^\star$ from the statement $\boldsymbol{x}$ and $\boldsymbol{x}^\star$, $\boldsymbol{x} \neq \boldsymbol{x}^\star$ using PPT algorithm with private key $\mathbf{hsk}^\star$ or witness $\boldsymbol{w}^\star$ is negligible.

$$\Pr[H_\lambda^{\text{Adver}}(\mathbf{hsk}, \boldsymbol{x}) = \pi \wedge F_\lambda^{\text{Adver}}(\mathbf{hsk}^\star, \boldsymbol{x}^\star) = \pi^\star] < \varepsilon(\lambda),$$

$$\Pr[F_\lambda^{\text{Adver}}(\mathbf{hpk}, \boldsymbol{x}, \boldsymbol{w}) = \pi \wedge F_\lambda^{\text{Adver}}(\mathbf{hpk}, \boldsymbol{x}^\star, \boldsymbol{w}^\star) = \pi^\star] < \varepsilon(\lambda),$$

where $\alpha(\mathbf{hsk}) = \mathbf{hpk}$ and $\varepsilon(\lambda)$ is a negligible function.

3 Unbalanced private set intersection with linear communication complexity

In this section, we construct a two-party private set intersection protocol using a hash proof system. The two parties are called the server and the client, and both parties hold a set. The size of the client's set is much smaller than that of the server's set. Our protocol outputs the intersection of their two sets to the client without revealing anything to the server. The communication complexity increases only linearly with the size of the client's set. The computational complexity is also well suitable for clients with limited resources. For the completeness of our protocol, we present the security model and details of our protocol below.

3.1 Security model

The communications between the server and the client are on a public channel. The server and the client hold sets X and Y, respectively, of sizes m and n. m and n are public, and we assume $m \gg n$. Each element in both sets is an integer in $[0, 2^t - 1]$ consisting of t-bit binary strings. The capabilities of both parties include computational, communication, and storage capabilities. We consider only the server and the client as adversaries in our protocol. We elaborate on them as follows.

The server. The server is a semi-honest entity with a large size of set. Server plays a role of having sufficient computation, storage, and communication resources. Nevertheless, it is always eager to obtain some additional information from the client using some polynomial time arithmetic.

The client. The client is a semi-honest entity with a smaller size of set. Client has limited computing, storage, and communication resources. It usually subscribes to some services on the server and is willing to obtain some additional information or services.

3.2 Our PSI protocol

In this subsection, we divide our PSI protocol into the Setup phase and Computation phase, and the details of our protocol are summarized in Figure 2.

Setup phase. In the setup phase, the server and client obtain parameters. Their parameters consist of public and private key pairs of length λ and can be reused in different rounds of computation.

Step 1. Server chooses a private key **hsk** from private key space **HSK** and runs Key Generation algorithm $I(1^\lambda)$ to generate a public-private key pairs $(\mathbf{hpk}, \mathbf{hsk})$ satisfying $\mathbf{hpk} = \mathbf{hsk} \cdot \boldsymbol{A} \mod N$. **hsk** is an s-dimensional vector, and $\mathbf{hsk} \in (\mathbb{Z}_{N-1})^{1\times s}$. $\boldsymbol{A}$ is a randomly matrix of size $s \times t$, and $\boldsymbol{A} \in (\mathbb{Z}_N^\star)^{s\times t}$. N is the modulus of **HPS**. $\mathbf{hsk} \cdot \boldsymbol{A}$ denotes dot product of a vector **hsk** and a matrice $\boldsymbol{A}$. Thus, **hpk** is a t-dimensional vector, and $\mathbf{hpk} \in (\mathbb{Z}_N^\star)^{1\times t}$. Server chooses t s-dimensional vectors $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t\}$ randomly from $\boldsymbol{D}/\boldsymbol{L}$, where t is the bit length of an integer. $\boldsymbol{D}$ denotes the set of s-dimensional vector, and $\boldsymbol{D} = (\mathbb{Z}_N^\star)^{1\times s}$. $\boldsymbol{L}$ represents a subset of $\boldsymbol{D}$ satisfying $\boldsymbol{A} \cdot \boldsymbol{B} \in \boldsymbol{L}$, $\boldsymbol{B}$ is a t-dimensional vector selected randomly, and $\boldsymbol{B} \in (\mathbb{Z}_{N-1}^\star)^{1\times t}$. Thereafter, server sends $\{\mathbf{hpk}, \boldsymbol{A}, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t, N\}$ to client.

Step 2. Client obtains parameters $\{p_j, N_j, g_j\}$, $j \in [n]$, where $N_j = (2p_j + 1)(2q_j + 1)$, p_j and q_j are two primes of λ bit. p_j is a factor of $\phi(N_j)$. g_j is a generator of $\mathbb{Z}_{N_j}^*$. N_j is a large integer of $2\lambda + 2$ bit. n is the size of the set Y. Client transfers $\{N_1, N_2, \ldots, N_n, g_1, g_2, \ldots, g_n\}$ to server.

Computation phase. In this phase, server and client input $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, respectively. Finally, the client obtains the intersection $X \cap Y$, and the server does not obtain any additional information.

Step 1. Client chooses a t-dimensional vector $\boldsymbol{w}_j$ randomly, and $\boldsymbol{w}_j \in (\mathbb{Z}_{N-1}^\star)^{1\times t}$, satisfying $\mathbf{hpk} \cdot \boldsymbol{w}_j = p_j$, and calculates

$$\mathbf{tc}_j^{\text{Client}} = \sum_k y_{j,k}\boldsymbol{a}_k + \boldsymbol{A}\boldsymbol{w}_j \mod N, \tag{1}$$

where $j \in [n]$, $k \in [t]$, and $y_{j,k}$ is the kth bit of y_j. Then, client sends $\mathbf{tc}_j^{\text{Client}}$, $j \in [n]$ to server.

Step 2. Server computes c_i^{Server}, $i \in [m]$, $k \in [t]$; $x_{i,k}$ is the kth bit of x_i.

$$c_i^{\text{Server}} = \mathbf{hsk} \sum_k x_{i,k}\boldsymbol{a}_k \mod N. \tag{2}$$

Step 3. When receiving $\mathbf{tc}_j^{\text{Client}}$, $j \in [n]$, server computes c_j^{Client}, $j \in [n]$ as

$$c_j^{\text{Client}} = \mathbf{hsk} \cdot \mathbf{tc}_j^{\text{Client}} \mod N, \tag{3}$$

and computes $z_{i,j}$, $i \in [m], j \in [n]$ for each c_j^{Client} as

$$z_{i,j} = c_j^{\text{Client}} - c_i^{\text{Server}} \mod N. \tag{4}$$

Step 4. Server sets $c_j = g_j$ and computes C_j by performing the loop operations,

$$C_j = c_j^{\prod_{i=1}^m z_{i,j}} \mod N_j, \tag{5}$$

for $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$. Thereafter, server sends $\{C_1, C_2, \ldots, C_n\}$ to client.

Step 5. Client sets $t_j = \phi(N_j)/p_i$, and checks whether the equation

$$C_j^{t_j} = 1 \mod N_j \tag{6}$$

holds. If yes, client accepts $y_j \in X \cap Y$ and outputs y_j. Server does not obtain any additional information.

4 Analysis

This section describes the security and communication complexity of our scheme. The security includes correctness, and the privacy of server and client. The communication complexity contains the cryptography tools and communication overhead. We will detail the computational overhead of our scheme applied to resource-constrained devices in Subsection 5.3.

4.1 Security

The security of our PSI scheme consists of two aspects. First, our PSI protocol outputs the correct intersection $X \cap Y$ to the client. Second, the privacy of both parties is protected. Specifically, the client does not obtain any additional information except the intersection $X \cap Y$. The server does not obtain any additional information by executing our PSI protocol.

Correctness. We first prove the correctness of our scheme. In other words, the client will obtain the correct intersection of their two sets, and the server will receive the corresponding earnings iff both of them perform properly our protocol. For that, we introduce Theorem 1 below.

Theorem 1. If the server and client follow our PSI scheme honestly, the client will obtain the correct intersection of their two sets.

Proof. Theorem 1 implies that our PSI scheme satisfies the correctness. In other words, the client obtains all integers $y_j \in X \cap Y$ by checking $C_j^{t_j} = 1 \bmod N_j$. This is the beginning of this proof. After receiving $\mathbf{tc}_j^{\text{Client}}$ from client, server obtains c_j^{Client}, $j \in [n]$ by computing

$$\begin{aligned} c_j^{\text{Client}} &= \mathbf{hsk} \cdot \mathbf{tc}_j^{\text{Client}} \bmod N = \mathbf{hsk}\left(\sum_k y_{j,k}\boldsymbol{a}_k + \boldsymbol{A}\boldsymbol{w}_j\right) \bmod N \\ &= \mathbf{hsk}\sum_k y_{j,k}\boldsymbol{a}_k + \mathbf{hsk} \cdot \boldsymbol{A}\boldsymbol{w}_j \bmod N = \mathbf{hsk}\sum_k y_{j,k}\boldsymbol{a}_k + \mathbf{hpk} \cdot \boldsymbol{w}_j \bmod N \\ &= \mathbf{hsk}\sum_k y_{j,k}\boldsymbol{a}_k + p_j \bmod N. \end{aligned} \tag{7}$$

After that server gets $z_{i,j}$, $i \in [m], j \in [n]$ by computing

$$\begin{aligned} z_{i,j} &= c_j^{\text{Client}} - c_i^{\text{Server}} = \mathbf{hsk}\sum_k y_{j,k}\boldsymbol{a}_k + p_j - \mathbf{hpk}\sum_k x_{i,k}\boldsymbol{a}_k \bmod N \\ &= \mathbf{hsk}\left(\sum_k y_{j,k}\boldsymbol{a}_k - \sum_k x_{i,k}\boldsymbol{a}_k\right) + p_j \bmod N = \mathbf{hsk}\left(\sum_k y_{j,k} - \sum_k x_{i,k}\right)\boldsymbol{a}_k + p_j \bmod N \\ &= \mathbf{hsk}\sum_k (y_{j,k} - x_{i,k})\boldsymbol{a}_k + p_j \bmod N. \end{aligned} \tag{8}$$

If there exists an integer $y_j \in X \cap Y$, implying that $y_{j,k} = x_{i,k}$ and the equation $z_{i,j} = p_j \bmod N$ holds. Server generates C_j, $j = 1, 2, \ldots, n$ by computing

$$C_j = g_j^{\prod_{i=1}^n z_{i,j}} \bmod N_j = g_j^{z_{1,j}z_{2,j}\cdots z_{i-1,j}p_j z_{i+1,j}\cdots z_{m,j}} \bmod N_j. \tag{9}$$

Afterwards, server sends $\{C_1, C_2, \ldots, C_n\}$ to client, and client computes $t_j = \phi(N_j)/p_j$, $j \in [n]$ and checks

$$\begin{aligned} C_j^{t_j} &= C_j^{\phi(N_j)/p_j} \bmod N_j = g_j^{z_{1,j}z_{2,j}\cdots z_{i-1,j}p_j z_{i+1,j}\cdots z_{m,j}\phi(N_j)/p_j} \bmod N_j \\ &= (g_j^{p_j\phi(N_j)/p_j})^{z_{1,j}z_{2,j}\cdots z_{i-1,j}z_{i+1,j}\cdots z_{m,j}} \bmod N_j = (g_j^{\phi(N_j)})^{z_{1,j}z_{2,j}\cdots z_{i-1,j}z_{i+1,j}\cdots z_{m,j}} \bmod N_j \\ &= (g_j^{\phi(N_j) \bmod \phi(N_j)})^{z_{1,j}z_{2,j}\cdots z_{i-1,j}z_{i+1,j}\cdots z_{m,j}} = (g_j^0)^{z_{1,j}z_{2,j}\cdots z_{i-1,j}z_{i+1,j}\cdots z_{m,j}} = 1. \end{aligned} \tag{10}$$

If $x_i \neq y_j$ holds, then $z_{i,j} \neq p_j$ for $i = 1, 2, \ldots, m$, and

$$\begin{aligned} C_j^{t_j} &= C_j^{\phi(N_j)/p_j} \bmod N_j = g_j^{z_{1,j}z_{2,j}\cdots z_{i-1,j}z_{i,j}z_{i+1,j}\cdots z_{m,j}\phi(N_j)/p_j} \bmod N_j \\ &= (g_j^{z_{i,j}\phi(N_j)/p_j})^{z_{1,j}z_{2,j}\cdots z_{i-1,j}z_{i+1,j}\cdots z_{m,j}} \bmod N_j. \end{aligned} \tag{11}$$

Even though $z_{i,j} \neq p_j$, $z_{i,j}$ may be a multiple of p_j. When $z_{i,j}/p_j$, the equation $g_j^{z_{i,j}\phi(N_j)/p_j} = 1$ and $C_j^{t_j} = 1 \bmod N_j$ holds. Then, the client will receive $y_j \in X \cap Y$. Next, we prove that the probability of $z_{i,j}/p_j$ is negligible, and the client will receive the right intersection. Assumption $z_{i,j}/p_j$ for an integer, the number of $z_{i,j}$ is approximately equal to N_j/p_j. The probability of $z_{i,j}/p_j$ for an integer is

$$\Pr = \frac{\frac{N_j}{p_j}}{N_j} = \frac{1}{p_j}. \tag{12}$$

The probability of $z_{i,j} \nmid p_j$ for an integer is $1 - \Pr$, and the probability of $z_{i,j} \nmid p_j$ for all integers is $(1 - \Pr)^n$. Thus, the probability of misjudging at least an integer in the intersection is

$$\begin{aligned} \Pr' = 1 - (1 - \Pr)^n = 1 - \left(1 - \frac{1}{p_j}\right)^n = 1 - \left(C_n^0 + C_n^1 \frac{1}{(-p_j)} + C_n^2 \frac{1}{(-p_j)^2} + \cdots + C_n^n \frac{1}{(-p_j)^n}\right) \\ < 1 - \left(C_n^0 - C_n^1 \frac{1}{p_j}\right) = 1 - \left(1 - n\frac{1}{p_j}\right) = \frac{n}{p_j}. \end{aligned} \tag{13}$$

C_n^* in this equation denotes a combination formula and follows the law of combination calculation. Since p_j is a λ-bit prime where λ denotes the security parameter, Pr' is negligible. The client will obtain an integer string $C_j^{t_j}$ and it is not equal to 1. Thereafter, client receives the correct intersection including all integers $y_j \in X \cap Y$ by checking $C_j^{t_j} = 1 \bmod N_j$. This operation completes the proof of Theorem 1.

Privacy. In this part, we prove that the privacy of both parties is protected in the semi-honest model.

Theorem 2. The privacy for both parties is protected in the semi-honest model. In other words, the client cannot obtain any information except the intersection of the two sets. The server cannot get any information about the intersection and client's set except the exchanged parameters.

Proof. Theorem 2 has two implications. One is that if the client is an adversary, it cannot obtain any additional information except the intersection of their two sets. The other is that if the server is an adversary, it cannot get any information about the client's set and the intersection. We divide the proof of Theorem 2 into two parts, the privacy of the client and the privacy of the server.

Primarily, we prove that the client's privacy is protected while the server only obtains the exchanged information. We divide the privacy of the client into two parts, including the privacy of the client's set and the privacy of the intersection.

We first prove that the server cannot get any information about the set of the client. Client randomly samples the witness $\boldsymbol{w}_j$ from $(\mathbb{Z}_{N-1}^\star)^{1\times t}$. These witnesses $\boldsymbol{w}_j$ are independent of the public **hpk** and private key **hsk**, and are also independent of the integers in client's set. Each witness $\boldsymbol{w}_j$ satisfies only $\mathbf{hpk} \cdot \boldsymbol{w}_j = p_j$. Despite that the server has **hpk**, **hsk**, and $\{N_1, N_2, \ldots, N_n\}$, it cannot get the factors $\{p_1, p_2, \ldots, p_n\}$. Obtaining p_j, $j \in [n]$ from $\{N_1, N_2, \ldots, N_n, g_1, g_2, \ldots, g_n\}$ is equivalent to solving the factoring problem on the product of large primes. Then, the server cannot get $\boldsymbol{w}_j$ without p_j.

$\boldsymbol{A}$ is a random matrix of size $s \times t$ and the vector $\boldsymbol{w}_j$ is randomly chosen from $(\mathbb{Z}_{N-1}^\star)^{1\times t}$. Applying the Leftover Hash Lemma and Entropy Smoothing Lemma [46], we obtain the result that $\boldsymbol{A}\boldsymbol{w}_j \bmod N$ is an s-dimensional vector and is uniformly distributed over the group $(\mathbb{Z}_N^\star)^{1\times s}$. Thus, the ciphertexts $\{\mathbf{tc}_1^{\text{Client}}, \mathbf{tc}_2^{\text{Client}}, \ldots, \mathbf{tc}_n^{\text{Client}}\}$ are uniformly distributed over the group $(\mathbb{Z}_N^\star)^{1\times s}$. When the server obtains $\{\mathbf{tc}_1^{\text{Client}}, \mathbf{tc}_2^{\text{Client}}, \ldots, \mathbf{tc}_n^{\text{Client}}\}$ in the computation phase, it is impossible for server to obtain any information about client's set.

Now we prove the server cannot get any information about the intersection $X \cap Y$. As described in the computation phase, if $x_i = y_j$ holds, there exists a ciphertext $z_{i,j}$ satisfying $z_{i,j} = p_j$ for $i = 1, 2, \ldots, m$. The client accepts $y_j \in X \cap Y$ under the condition that there exists a prime p_j in $z_{i,j}$, $i \in [m]$ satisfying $p_j | N_j$. Despite obtaining p_j, $j \in [n]$ from $\{N_1, N_2, \ldots, N_n\}$ is equivalent to solving the factoring problem on the product of large primes, the server cannot obtain p_j, $j \in [n]$. However, there may be a doubt as described below. The server may determine $y_j \notin X \cap Y$ by determining that there does not exist a prime in $z_{i,j}$, $i \in [m]$. In other words, the existence of a prime in $z_{i,j}$, $i \in [m]$ may imply $y_j \in X \cap Y$ and the absence of a prime p_j in $z_{i,j}$, $i \in [m]$ means $y_j \notin X \cap Y$. We will show that even if $x_i \neq y_j$, there is still a prime in $z_{i,j}$, $i \in [m]$, and this doubt does not exist.

As we all know that the number of primes in $\mathbb{Z}_N$ is approximately equal to $N/\ln N$. The probability that an element in $z_{i,j}$ is a prime is

$$\Pr_1 = \frac{\frac{N}{\ln N}}{N} = \frac{1}{\ln N}. \tag{14}$$

The probability that an element in $z_{i,j}$ is not a prime is $1 - \Pr_1$, and the probability that all elements in $z_{i,j}$, $i \in [m]$ are not prime is $(1 - \Pr_1)^m$. Thus,

$$(1 - \Pr_1)^m = \left(1 - \frac{1}{\ln N}\right)^m, \tag{15}$$

where λ is the security parameter and $|X| = m$, then $N \approx 2^{\lambda}$,

$$(1 - \Pr_1)^m = \left(1 - \frac{1}{\ln N}\right)^m = \left(1 - \frac{1}{\ln 2^{\lambda}}\right)^m = \left(1 - \frac{1}{\lambda \ln 2}\right)^m = \left(1 - \frac{1}{0.7\lambda}\right)^m. \tag{16}$$

Assuming $\lambda = 128$ and $m = 2^{16}$, then the probability $(1 - \Pr_1)^m = (1 - \frac{1}{0.7\lambda})^m$ is negligible, and the probability that all elements in $z_{i,j}$, $i \in [m]$ are not prime is negligible. Thus, the server cannot determine $y_j \notin X \cap Y$ by determining that there does not exist a prime in $z_{i,j}$, $i \in [m]$, and cannot get any information about the intersection $X \cap Y$.

In addition, $\phi(N_j)$ and p_j are unknown to the server, and the server cannot obtain t_j, $j \in [n]$ and determine $C_j^{t_j} \neq 1 \bmod N_j$. Then, the server does not have an efficient strategy to infer any information about the intersection and the client's set. Our protocol protects the privacy of the client's set and the privacy of the intersection.

Secondly, we prove that the server's privacy is protected while the client only obtains the intersection. We divide the server's privacy into two parts. The client cannot obtain the private key **hsk** from public parameters in the setup phase and it also cannot obtain any additional information about the server's set in the computation phase.

The client receives the parameters $\{\mathbf{hpk}, \boldsymbol{A}, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t, N\}$ in setup phase. $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t$ are chosen randomly from $\boldsymbol{D}/\boldsymbol{L}$, and are independent of the private key **hsk** and server's set. Client cannot obtain any information about the private key **hsk** and the server's set from $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t$. The security of HPS system guarantees that the client cannot obtain any information about **hsk** from $\boldsymbol{A}$, **hpk**, and N. In addition, the parameters $\{\mathbf{hpk}, \boldsymbol{A}, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t, N\}$ are not related to the server's set. The client cannot utilize it to infer any additional information about the server's set.

Now we prove the client only obtains the intersection in the computation phase. The server computes the loop operations

$$C_j = c_j^{\prod_{i=1}^m z_{i,j}} \bmod N_j \tag{17}$$

and transfers $\{C_1, C_2, \ldots, C_n\}$ to the client. Each $\{C_1, C_2, \ldots, C_n\}$ is the aggregation of $z_{i,j}$, $i \in [m]$; then the client can obtain the aggregation $\prod_{i=1}^m z_{i,j}$ by computing

$$C_j = c_j^{\prod_{i=1}^m z_{i,j}} \bmod N_j = c_j^{\prod_{i=1}^m z_{i,j} \bmod \phi(N_j)}. \tag{18}$$

However, server computes $z_{i,j}$, $i \in [m]$,

$$\begin{aligned} z_{i,j} &= c_j^{\text{Client}} - c_i^{\text{Server}} \bmod N = \mathbf{hsk} \sum_k y_{j,k} \boldsymbol{a}_k - \mathbf{hsk} \sum_k x_{i,k} \boldsymbol{a}_k + p_j \bmod N \\ &= \mathbf{hsk} \left(\sum_k y_{j,k} \boldsymbol{a}_k - \sum_k x_{i,k} \boldsymbol{a}_k \right) + p_j \bmod N = \mathbf{hsk} \left(\sum_k y_{j,k} - \sum_k x_{i,k} \right) \boldsymbol{a}_k + p_j \bmod N. \end{aligned} \tag{19}$$

Even though the client gets the aggregation $\prod_{i=1}^m z_{i,j}$, p_j, $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_t$, and N, it cannot get any information about c_j^{Client} and c_i^{Server} without **hsk**. The client does not have an efficient strategy to divide the aggregation $\prod_{i=1}^m z_{i,j}$ into $z_{i,j}$, $i \in [m]$. Therefore, the client cannot infer any additional information about the server's set x_i, $i \in [m]$ from C_j. Hence, the privacy of the server's set is guaranteed in our PSI scheme. This completes the proof of Theorem 2.

4.2 Communication complexity

We show the theoretical communication complexity of our PSI protocol in this subsection, and compare it with other schemes, including VKRK-16 [30], the Diffie-Hellman based PSI [37], Spot-19 [32], HKP-17 [6], HZK-18 [22], PaXos-20 [47], and SS-PSI [34] in the semi-honest model. These protocols communicate over an idealized network without considering the metadata, realistic encodings, byte alignment, etc. The comparison of the cryptographic tools and the communication complexity is detailed in Table 2. m and n denote the size of the input set for the server and client, respectively. OPRF represents the oblivious pseudorandom function, OT extension denotes the oblivious transfer extension, and FHE denotes fully homomorphic encryption.

Table 2 demonstrates the communication complexity is $O(n + m\log(mn))$ for VKRK-16, Spot-19-Fast, PaXos-20, and SS-PSI, $O(m + n\log(mn))$ for DH-PSI, $O(n + m\log(n))$ for Spot-19-Low, $O(m\log(n))$ for

Table 2 Comparison on communication complexity

Protocol	Cryptography tools	Communication complexity
VKRK-16 [30]	OPRF	$O(n + m\log(mn))$
DH-PSI [37]	Standard DDH assumption	$O(m + n\log(mn))$
Spot-19-Low commu [32]	OT extension, OPRF	$O(n + m\log(n))$
Spot-19-Fast [32]	OT extension, OPRF, Cuckoo hashing	$O(n + m\log(mn))$
HKP-17 [6]	FHE, OPRF	$O(n\log(m))$
HZK-18 [22]	FHE, OPRF	$O(n\log(m))$
PaXos-20 [47]	OT extension, Cuckoo hashing	$O(n + m\log(mn))$
SS-PSI [34]	Diffie-Hellman instantiation	$O(n + m\log(mn))$
Our protocol	Hash proof system, RSA assumption	$O(n)$

HKP-17 and HZK-18. The communication complexity of our PSI protocol is $O(n)$. This situation implies that the communication cost of our PSI protocol has an advantage over other schemes. Considering that the size of the server's set is equal to the size of the client's set, the communication complexity of our PSI protocol has a clear advantage over the VKRK-16 scheme, DH-PSI scheme, Spot-19-Low scheme, Spot-19-Fast scheme, PaXos-20 scheme, and SS-PSI scheme. In addition, our protocol also has a slight advantage over the HKP-17 scheme and HZK-18 scheme. As is well-known to all, expanding bandwidth and increasing computing power in the practical applications can be a huge challenge for devices with limited resources. Then, our PSI protocol has an advantage over other schemes in the practical applications in terms of communication overhead and network delay.

5 Experiments

5.1 Implementation

We implement our PSI protocol described in Figure 2. We choose the CryptoPP library [48–50] in C++ to implement the hash proof system and RSA assumption. All fixed-length integers in both sets are randomly chosen and encoded by utilizing `Integerclass` and `Integer` method: `Encode` and `MinEncodeSize`. Parameters including N, p, q are chosen by adopting `AutoRandomPool`.

We test performance on a PC machine with an Intel(R) Xeon(R) Gold 5115 CPU @ 2.40 GHz and 96 GB RAM. We use the Linux `tc` command to simulate the bandwidth. Specifically, we consider the network environment as an LAN setting, and two parties are connected via a local host with a bandwidth of 10 Gbps. We repeat our experiment six times and compute the average runtime.

We implement experiments for the intersection of two unbalanced private sets, as mentioned in Figure 2. The sizes of two sets in our experiments are implemented as $|X| = 2^{16}, 2^{20}, 2^{24}$ and $|Y| = 50, 100, 200, 400, 800, 1600$. Thus, there are at least three orders of magnitude differences in the sizes of their two sets. We statistically set the security levels $\lambda = 128, 256, 512$. We randomly sample the elements $w_{[i]}$, $i = 1, 2, \ldots, \lambda - 1$, where $w_{[i]}$ is the ith element of $\boldsymbol{w}_j$ and obtain the element $w_{[i]}$, $i = \lambda$ by computing equation $\mathbf{hpk} \cdot \boldsymbol{w}_j = p_j$.

Although the length of each integer in both sets, security parameters, and other computational parameters are designed as 32-bit strings in our experiments, our scheme is also applicable to other lengths of integers, and parameters. We utilize C++ programming language to implement our experiment. The computation and communication between the server and the client are implemented in hardware as a single thread. It is noticeable that our scheme can also be performed in parallel with multiple threads.

5.2 Evaluations

The execution time is recorded from the query operation is started by the client until obtaining the intersection from the server. To optimize the computation, we perform the setup phase in offline. We rigorously and carefully record the results of our experiments, and enumerate these in Table 3.

We list the communication unit (kb) and the runtime unit (ms) for the setup phase and computation phase under different λ. The communication in the setup phase and computation phase includes S→R and R→S. S →R means that the server transmits data to the client. R→S means that the client transmits data to the server. The runtime includes the computational times on the client's side.

Table 3 Cost of the communication and computation

Parameters		$\lambda = 128$					$\lambda = 256$					$\lambda = 512$				
		Communication (kb)				Time (ms)	Communication (kb)				Time (ms)	Communication (kb)				Time (ms)
		Setup phase		Comput. phase		Client	Setup phase		Comput. phase		Client	Setup phase		Comput. phase		Client
$\|X\|$	$\|Y\|$	S→R	R→S	S→R	R→S		S→R	R→S	S→R	R→S		S→R	R→S	S→R	R→S	
2^{16}	50	29.87	4.69	2.34	27.58	67.15	56.10	9.39	4.69	53.35	70.81	110.42	18.76	9.38	104.93	101.68
	100	30.34	9.38	4.69	55.17	96.53	56.39	18.76	9.38	106.74	129.94	114.39	37.51	18.75	209.86	199.16
	200	29.71	18.76	9.37	110.33	180.72	56.39	37.51	18.75	213.48	233.16	114.42	75.01	37.50	419.73	371.20
	400	29.87	37.52	18.75	220.70	367.04	56.39	75.01	37.50	426.95	508.53	116.42	150.01	75.00	839.45	784.78
	800	28.87	75.01	37.50	441.39	723.14	56.39	150.01	75.00	853.91	1041.74	112.42	300.01	150.00	1678.91	1128.40
	1600	29.87	150.01	75.00	882.81	1408.09	56.19	300.01	150.00	1707.53	1744.43	114.42	600.01	300.00	3357.81	2004.97
2^{18}	50	29.84	4.70	2.34	27.58	73.31	57.39	9.39	4.69	53.37	66.879	116.42	18.76	9.38	104.93	102.91
	100	29.84	9.39	4.69	55.17	96.80	57.39	18.76	9.38	106.74	139.02	112.42	37.51	18.75	209.86	201.61
	200	29.87	18.76	9.37	110.35	175.80	57.32	37.51	18.75	213.48	255.56	120.42	75.01	37.50	419.73	304.01
	400	29.87	37.51	18.75	220.70	355.12	58.36	75.01	37.50	426.95	365.61	104.39	150.01	75.00	839.44	574.65
	800	30.78	75.01	37.50	441.41	595.45	56.39	150.01	75.00	853.91	668.81	112.42	300.01	150.00	1678.91	1168.90
	1600	31.37	150.01	75.00	882.81	962.47	56.36	300.01	150.00	1707.81	1140.11	110.86	600.01	300.00	3355.20	2015.70
2^{20}	50	28.87	4.70	2.34	27.58	63.92	59.36	9.39	4.69	53.39	70.32	106.32	18.76	9.38	104.93	72.29
	100	29.37	9.38	4.69	55.18	98.17	55.39	18.76	9.38	106.74	124.59	110.29	37.51	18.76	209.85	136.41
	200	27.37	18.76	9.38	110.35	180.76	55.39	37.51	18.75	213.48	170.65	110.39	75.01	37.50	419.73	295.01
	400	29.87	37.51	18.75	220.70	302.69	55.39	75.01	37.50	426.95	291.65	112.42	150.01	75.00	839.45	502.18
	800	31.01	75.01	37.50	441.41	603.40	60.36	150.01	75.00	853.91	678.28	112.42	300.01	150.00	1678.91	979.84
	1600	28.87	150.01	75.00	882.81	782.91	59.36	300.01	150.00	1707.81	1186.20	110.73	600.01	300.00	3357.81	2010.04

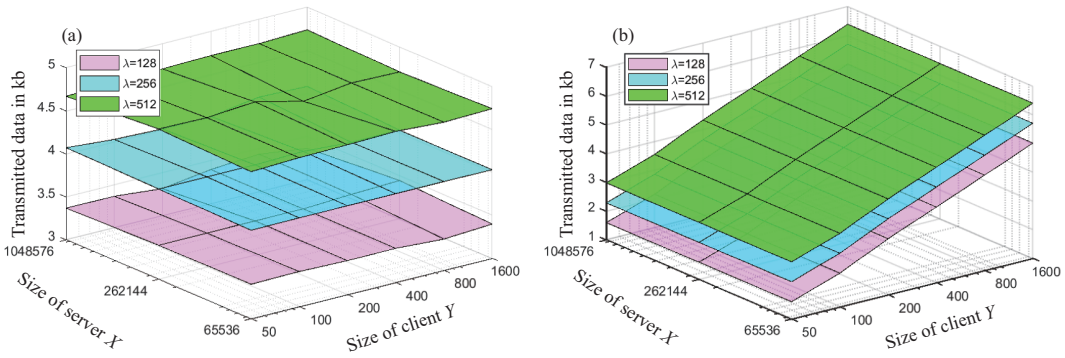

Figure 3 (Color online) Transmitted data in the setup phase at different λ (a) from server to client and (b) from client to server.

5.3 Experiments analysis

We analyze the experimental data in terms of both communication and computation as follows.

Communication analysis. We divide the communication overhead into two parts, including the communication in the setup and computation phases. Specifically, we detail the relationship between the communication complexity and the size of the two sets, and the security parameters. We describe the communication overhead of two parts: S→R and R→S.

Table 3 demonstrates that the communication overhead of S→R in the setup phase does not vary with the size of the two sets. The reason is that the server only transmits public parameters in this phase. The amount of parameters is independent of the size of the two sets, but is related to the length of the strings in the two sets.

Table 3 demonstrates that the communication overhead of R→S in the setup phase, S→R and R→S in the computation phase increase linearly with the size of the client's set, and do not vary with the size of the server's set. The reason is that the client transmits public parameters in the setup phase. The amount of parameters increases only linearly with the client's set size, and is independent of the server's set size. In the computation phase, the communication overhead of exchanging information always varies linearly with the size of the client's set, independent of the size of the server's set. In addition, the parameters, public and private keys can be reused in different rounds of the computation phase; it always reduces the communication overhead in multiple rounds of the computation phase. The experimental results show that the communication complexity of our PSI scheme is $O(n)$. In other words, the communication complexity varies linearly only with the client's set size, and is independent of the server's set size.

In addition, Table 3 also indicates that the communication overhead will increase linearly with the length of the parameter λ from 128 to 512. The reason is that longer security parameters will result in longer ciphertexts and higher communication overhead.

For the reason the size of the two sets grows exponentially, we perform logarithmic operations on the communication results in Table 3 to obtain clear experimental plots (Figures 3 and 4). The plots directly reflect the linear relationship between the communication complexity and the size of the two sets. Figures 3 and 4 can visually depict the results. In the setup phase, the communication complexity of the client only increases linearly with the size of the client's set. Moreover, the communication complexity of the server is constant and within a reasonable range. During the computation phase, the communication complexity of the server and the client only increases linearly with the size of the client's set. Security parameters with longer bits will result in increased communication overhead. This result will be widely accepted by the industry and the academia. Finally, as we can see from Table 3 or Figures 3 and 4, the communication overhead of the client in our scheme can be widely accepted by mobile devices, even low performance devices with limited communication capability. The low communication overhead is a good advantage of our scheme.

Computation analysis. We analyze the computational complexity of our scheme in this part. The experimental results indicate that the computational complexity is acceptable for both the client and the server, even for mobile or other resource-limited devices. We leverage `gettimeofday()` function to record the time from the start of the client's query operation to the time when the intersection of their two sets is obtained from the server. The runtime of the client is listed in Table 3.

The runtime listed in Table 3 demonstrates the computational complexity of the client varies linearly

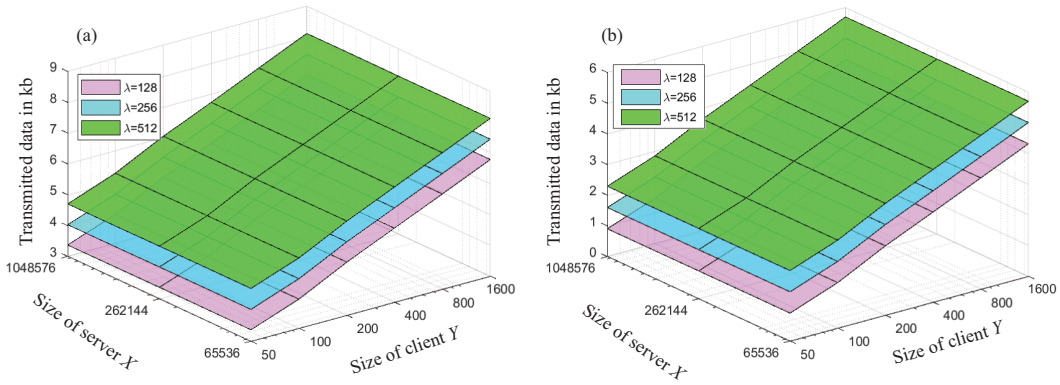

Figure 4 (Color online) Transmitted data in the computation phase at different λ (a) from client to server and (b) from server to client.

with the size of the client's set and does not vary with the size of the server's set. In other words, the computational complexity of the client is $O(n)$. Moreover, the results also indicate that the longer security parameters will result in increased computational complexity for the client. Furthermore, Table 3 also shows the computational complexity of our scheme is acceptable for devices with limited resources, such as mobile phones. As we view that the runtime is roughly 2 s in the case of security parameter $\lambda = 512$, the size of the client's set is 1600, and the size of the server's set varies from 2^{16} to 2^{20}.

On the other hand, the runtime is approximately less than twice the original runtime when the size of the client's set increases to twice the original size. We analyze our experiment and propose two possible reasons to explain this phenomenon. First, we use the algorithm from the CryptoPP library, which contains the optimization function. The probability that the data can be computed optimally varies with the size of the two sets, resulting in runtime that does not increase to twice the original size. The second reason is that our experiment is repeated six times. The average of the six experimental results is utilized to reflect the experimental results. However, the amount of experiments is not enough to reflect the whole results. This situation is determined by the amount of repeated experiments, and it does not indicate that our experiment is unreasonable. This phenomenon further illustrates that the computational overhead of our scheme can be well applied to devices with limited resources, such as mobile devices.

In our experiments, the runtime on the server side demonstrates that it can also be accepted by a computationally abundant server. We will research some computational optimization algorithms at the server's side in future work.

6 Conclusion

In this paper, we construct a practical private set intersection protocol based on a hash proof system for clients with limited communication and computational resources. The communication complexity of our protocol increases only with the size of the small set, which is a significant progress compared to previous state-of-the-art protocols. The communication has good advantages not only in unbalanced PSI schemes but also in balanced PSI schemes. Its computational complexity can be accepted by clients with limited computational resources. We prove the security of our scheme, and the client obtains the correct intersection without a false element. We implement our PSI protocol, and the experimental results also demonstrate that our protocol will be accepted by devices with limited resources. We believe our work has a wide range of applications, especially in private contact discovery, and originality testing of papers. We will optimize the computational efficiency on the server's side in future work.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2020YFB1005900), Natural Science Foundation on Frontier Leading Technology Basic Research Project of Jiangsu (Grant No. BK20222001), Leading-edge Technology Program of Jiangsu National Science Foundation (Grant No. BK20202001), and National Natural Science Foundation of China (Grant Nos. 61872176, 62272215, 61872179, 62272222).

References

1 Demmler D, Rindal P, Rosulek M, et al. PIR-PSI: scaling private contact discovery. Proc Privacy Enhancing Technol, 2018, 2018: 159–178
2 Troncoso-Pastoriza J R, Katzenbeisser S, Celik M. Privacy preserving error resilient DNA searching through oblivious automata. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007. 519–528

3 Kontaxis G, Athanasopoulos E, Portokalidis G, et al. Sauth: protecting user accounts from password database leaks. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2013. 178–198
4 Narayanan A, Thiagarajan N, Lakhani M, et al. Location privacy via private proximity testing. In: Proceedings of the Network and Distributed System Security Symposium, San Diego, 2011. 1–17
5 Meadows C. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: Proceedings of IEEE Symposium on Security and Privacy, 1986
6 Chen H, Laine K, Rindal P. Fast private set intersection from homomorphic encryption. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2017. 1243–1255
7 Rindal P, Rosulek M. Malicious-secure private set intersection via dual execution. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2017. 1229–1242
8 Dong C Y, Chen L Q, Wen Z K. When private set intersection meets big data: an efficient and scalable protocol. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2013. 789–800
9 Pinkas B, Schneider T, Zohner M. Faster private set intersection based on OT extension. In: Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), 2014. 797–812
10 Pinkas B, Schneider T, Segev G, et al. Phasing: private set intersection using permutation-based hashing. In: Proceedings of the 24th USENIX Security Symposium (USENIX Security 15), 2015. 515–530
11 Lu S Q, Zheng J H, Cao Z F, et al. A survey on cryptographic techniques for protecting big data security: present and forthcoming. Sci China Inf Sci, 2022, 65: 201301
12 Giuseppe A, Cristofaro E D, Tsudik G. If size matters: size-hiding private set intersection. In: Proceedings of International Workshop on Public Key Cryptography. Berlin: Springer, 2011. 6571: 156–173
13 Jia Y, Sun S F, Zhou H S, et al. Shuffle-based private set union: faster and more secure. In: Proceedings of the 31st USENIX Security Symposium, 2022. 2947–2964
14 Aranha D F, Lin C, Orlandi C, et al. Laconic private set-intersection from pairings. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2022. 111–124
15 Ma J P K, Chow S S M. Secure computation friendly private set intersection from oblivious compact graph evaluation. In: Proceedings of the ACM on Asia Conference on Computer and Communications Security, 2022. 1086–1097
16 Hazay C, Nissim K. Efficient set operations in the presence of malicious adversaries. J Cryptol, 2012, 25: 383–433
17 Guo X J, Li J, Liu Z L, et al. Labrador: towards fair and auditable data sharing in cloud computing with long-term privacy. Sci China Inf Sci, 2022, 65: 152106
18 Zhang G-W, Chen W, Fan-Yuan G-J, et al. Polarization-insensitive quantum key distribution using planar lightwave circuit chips. Sci China Inf Sci, 2022, 65: 200506
19 Huang Y, Evans D, Katz J, et al. Faster secure two-party computation using garbled circuits. In: Proceedings of the 20th USENIX Security Symposium, 2011. 1–16
20 Huang Y, Evans D, Katz J. Private set intersection: are garbled circuits better than custom protocols? In: Proceedings of Network and Distributed Systems Security (NDSS) Symposium, 2012. 1–15
21 Ciampi M, Orlandi C. Combining private set-intersection with secure two-party computation. In: Proceedings of International Conference on Security and Cryptography for Networks. Cham: Springer, 2018. 464–482
22 Chen H, Huang Z, Laine K, et al. Labeled PSI from fully homomorphic encryption with malicious security. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2018. 1223–1237
23 Debnath S K, Dutta R. Towards fair mutual private set intersection with linear complexity. Security Comm Networks, 2016, 9: 1589–1612
24 Kamara S, Mohassel P, Raykova M, et al. Scaling private set intersection to billion-element sets. In: Proceedings of International Conference on Financial Cryptography and Data Security, 2014. 8437: 195–215
25 Le P H, Ranellucci S, Gordon S D. Two-party private set intersection with an untrusted third party. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2019. 2403–2420
26 Pinkas B, Schneider T, Weinert C, et al. Efficient circuit-based PSI via cuckoo hashing. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2018. 125–157
27 Pinkas B, Schneider T, Tkachenko O, et al. Efficient circuit-based PSI with linear communication. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer, 2019. 122–153
28 Falk B H, Noble1 D, Ostrovsky R. Private set intersection with linear communication from general assumptions. In: Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, 2019. 14–25
29 Asharov G, Lindell Y, Schneider T, et al. More efficient oblivious transfer and extensions for faster secure computation. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2013. 535–548
30 Kolesnikov V, Kumaresan R, Rosulek M, et al. Efficient batched oblivious PRF with applications to private set intersection. In: Proceedings of the ACM Conference on Computer and Communications Security, 2016. 818–829
31 Pinkas B, Schneider T, Zohner M. Scalable private set intersection based on OT extension. ACM Trans Priv Secur, 2018, 21: 1–35
32 Pinkas B, Rosulek M, Trieu N, et al. SpOT-Light: lightweight private set intersection from sparse OT extension. In: Proceedings of Annual International Cryptology Conference. Cham: Springer, 2019. 401–431
33 Cristofaro E D, Tsudik G. Practical private set intersection protocols with linear complexity. In: Proceedings of International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2010. 143–159
34 Rosulek M, Trieu N. Compact and malicious private set intersection for small sets. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2021. 1166–1181
35 Ferhat K, Alptekin K. Linear complexity private set intersection for secure two-party protocols. In: Proceedings of International Conference on Cryptology and Network Security. Cham: Springer, 2020. 409–429
36 Resende A C D, Aranha D F. Faster unbalanced private set intersection. In: Proceedings of International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2018. 203–221
37 Cristofaro E D, Kim J, Tsudik G. Linear-complexity private set intersection protocols secure in malicious model. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2010. 213–231
38 Kiss Á, Liu J, Schneider T, et al. Private set intersection for unequal set sizes with mobile applications. Proc Privacy Enhancing Technologies, 2017, 2017: 177–197
39 Gentry C, Halevi S, Smart N P. Homomorphic evaluation of the AES circuit. In: Proceedings of Annual Cryptology Conference. Berlin: Springer, 2012. 850–867

40 Cheon J H, Kim M, Lauter K. Homomorphic computation of edit distance. In: Proceedings of International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2015. 194–212
41 Egashira S, Wang Y, Tanaka K. Fine-grained cryptography revisited. J Cryptol, 2021, 34: 1–43
42 Degwekar A, Vaikuntanathan V, Vasudevan P N. Fine-grained cryptography. In: Proceedings of Annual International Cryptology Conference. Berlin: Springer, 2016. 533–562
43 Hesse J, Hofheinz D, Kohl L. On tightly secure non-interactive key exchange. In: Proceedings of Annual International Cryptology Conference. Cham: Springer, 2018. 65–94
44 Cramer R, Shoup V. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2002. 2332: 45–64
45 Hong H B, Shao J, Wang L C, et al. A CCA secure public key encryption scheme based on finite groups of Lie type. Sci China Inf Sci, 2022, 65: 119102
46 Ajtai M. Generating hard instances of lattice problems. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 1996. 99–108
47 Pinkas B, Rosulek M, Trieu N, et al. PSI from PaXoS: fast, malicious private set intersection. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer, 2020. 739–767
48 Dai W. Crypto++ Library 8.2. Free C++ class library of cryptographic schemes. https://www.cryptopp.com/. 2014
49 Merkle R C. Secure communications over insecure channels. Commun ACM, 1978, 21: 294–299
50 Diffie W, Hellman M E. New directions in cryptography. IEEE Trans Inform Theory, 1976, 22: 644–654