Efficient Private Set Intersection Using Point-Value Polynomial Representation

Ou Ruan and Hao Mao

School of Computer Science, Hubei University of Technology, Wuhan 430068, China

Correspondence should be addressed to Ou Ruan; ruanou@163.com

Received 11 June 2020; Revised 15 July 2020; Accepted 20 July 2020; Published 1 September 2020

1. Introduction

Private set intersection (PSI) can be described that participants complete computation based on their private inputs and cannot learn additional information other than the set intersection. PSI has a wide range of applications such as privacy-preserving contact tracing for infection detection [1, 2], private contact discovery [3], similar document detection [4], suspects detection [5], relationship path discovery in social networks [6], and satellite collisions matching [7].

PSI has been well studied. Several cryptographic technologies have been proposed to implement PSI. According to cryptographic techniques involved, PSI protocols are mainly divided into the following three categories:

1. PSI based on the public-key technology: the main cryptographic technique was homomorphic encryption. The protocols were designed in such a way that the sender encrypted sets and the receiver performed some operations on the ciphertexts using the property of homomorphic encryption; then, the sender decrypted them by using his private key and got the intersection. With small communication complexity, these protocols were suitable for the scenario where the participants had strong computing power but the communication bandwidth was a bottleneck. However, the protocols had a higher time complexity because of using public-key cryptography.

2. PSI based on the generic circuit: the protocols transformed any function into garbled Boolean circuit and then completed the generic secure computation. The circuit generator encrypted each circuit gate using a double symmetric cryptosystem and generated a garbled circuit; the evaluator computed keys for the output wires by decrypting the appropriate ciphertexts without learning any intermediate values. The key technique used in the protocols was symmetric cryptosystem. The advantage of the general circuit protocol was that it made
the protocol easier to design and implement. But as a
general solution, the garbled circuit could not
achieve scalability, and the protocols were inefficient.

(3) PSI based on the oblivious transfer (OT) scheme: this
kind of protocols introduced some variants of OT.
The protocols were that elements were stored in
some data structures, and parties ran an OT for each
bit of inputs to get private outputs. Then, each party
performed XOR operations with random values and
its own elements. Lastly, the sender sent the results to
the receiver, who locally checked the existence of its
inputs. To improve efficiency, most of OT variants
were implemented by using the symmetric crypto-
system. Thus, these protocols had lower time com-
plexity and communication complexity. Nevertheless,
such protocols required additional keys-related
computations such as secret key negations.

From the above analysis, PSI protocols based on public-
key cryptosystem suffer from two constraints: low efficiency
and needing a complicated system for private/public-keys
management. On the other hand, PSI protocols based on
symmetric cryptosystem have higher efficiency, but nego-
tiating or secure transferring of secret keys leads to addi-
tional computations and communications. Furthermore,
the secure storage of keys will burden the system. In the paper,
we transform the problem of the intersection of sets into the
problem of finding the roots of polynomials by using point-
value polynomial representation and propose an efficient
PSI protocol without any cryptosystem.

1.1. Application Scenarios. Our work can be applied to the
following several practical scenarios.

1.1.1. Contact Tracing for Infection Detection. The COVID-
19 pandemic has posed an unprecedented challenge for
humans. Due to the highly contagious nature of the virus,
social distancing is one fundamental measure that has al-
ready been adopted by many countries. Based on the
matching of location information between infected patients
and regular people, contact tracing for infection detection
enables users to securely upload their data to the server, and
later, in case one user got infected, other users can check if
they had ever got in contact with the infected user in the past.
To protect users’ private location information, PSI can be
applied to securely compute shared location data.

1.1.2. Suspects Detection. Two national law enforcement
bodies have a list of suspected terrorists. Due to national
laws, they may not be allowed to disclose their whole lists,
even when collaborating. Using a PSI protocol, both agencies
can find commonly suspected terrorists and share their
information, while other relevant information will not be
disclosed.

1.1.3. Satellite Collisions. Different space agencies have their
own orbiting satellites. In order to determine the collision
problem of the same orbiting satellite pair and adjust the
orbit of the satellite appropriately, these agencies need to
share more detailed information. However, each agency
does not want to disclose anything other than whether there
was a collision in orbital information. Thus, it is necessary to
use PSI for computing the probability of a collision among
satellites without revealing their other private information.

1.2. Contributions. We transform the problem of the in-
tersection of sets into the problem of finding roots of
polynomials by using point-value polynomial representation
and propose a new approach to PSI protocol without any
cryptosystem. Then, we optimize our protocol based on the
permutation-based hashing technique that reduces the
length of the stored elements and the degree of the poly-
nomial. Eventually, our protocol and the related PSI pro-
tocols are implemented on the Linux platform. The main
contributions are as follows.

1.2.1. A New Approach to PSI Protocol. We propose a new
approach for designing PSI protocol based on point-value
polynomial representation and pseudorandom function.
Firstly, we represent sets as polynomials’ point-value pairs.
Each party denotes \( d \) elements \((s_1, \ldots, s_d)\) as a \( d \)-degree
polynomial \( \rho(x) = \prod_{i=1}^{d} (x - s_i) \) and represents \( \rho(x) \) as \( n \)
distinct point-value pairs \( \{(x_1, \rho(x_1)), \ldots, (x_n, \rho(x_n))\} \)
where \( n > d \). Secondly, we blind polynomials’ point-value pairs
for secure transportation and computation. Each party
blinds them as \( \{(x_1, \rho(x_1) + z_1), \ldots, (x_n, \rho(x_n) + z_n)\} \) by
using pseudorandom function and exchanges the blinded
point-value pairs. Thirdly, we compute the sum of two
blinded polynomials’ point-value pairs. Through computa-
tion and transportation, one party can get the sum of two
blinded polynomials’ point-value pairs. Lastly, we can learn
the polynomial by interpolation and get the intersection by
computing the roots of the polynomial. With this repre-
sentation, we could get the set intersection without any
cryptosystem.

1.2.2. Efficient Hashing PSI Protocol. We optimize the new
PSI protocol using the permutation-based hashing method,
which converts the hashed elements into shorter strings
without collisions and reduces the degree of polynomials.
The hashing is to create a two-dimensional table
\( HT[b][\max_y] \) and map each element to its hashed bins,
resulting in \( b \cdot \max_y \) stored elements, which split an \( n \)-degree
polynomial into \( b \cdot \max_y \)-degree polynomials. This approach
improves efficiency remarkably.

1.2.3. Implementation of Our Hashing PSI Protocol. We
implement our hashing protocol and other related protocols
in C/C++ on the Linux platform. We use Number Theory
Library (NTL) [8] along with GNU Multiprecision (GMP)
library [9] for polynomial arithmetic. Based on the detailed
experimental data, we conclude that our protocol is more
efficient than public-key-based and circuit-based PSI pro-
tocols and is more efficient than OT-based PSI protocols at
set elements less than \( 2^{12} \).
2. Related Work

According to the underlying cryptographic techniques, PSI protocols can be divided into the following three categories.

2.1. PSI Based on the Public-Key. In 1986, Meadows [10] introduced a PSI protocol that could solve the problem of authentication of mutually suspicious parties. But, they revealed the cardinality of sets during the authentication. To solve this problem, an improved PSI protocol [11] was proposed.

The PSI protocol based on oblivious polynomial evaluation [12] was proposed in 2004, which used the homomorphic encryption, balanced hashing, and properties of polynomials. They represented its elements as roots of polynomials and used interpolation to find out the coefficients of polynomials and sent the ciphertexts of coefficients of polynomials to the server by using ElGamal [13] or Pailler [14] encryption. In this protocol, it would lead to a high cost of exponential calculation in homomorphic encryption if the degree of the polynomial was large. An extended version [15] was presented, where the client and the server used the cuckoo hashing technique to reduce its computational complexity.

In 2009, Jarecki et al. [16] showed a PSI protocol based on the composite residual hypothesis. The protocol used additive homomorphic and zero-knowledge proof to realize the pseudorandom function and then performed the intersection operation on the random values of the set. The client and the server carried on the parallel oblivious pseudorandom function (OPRF) to get the intersection. However, the protocol relied on the common reference model.

In 2010, Cristofaro et al. proposed PSI and Authorized PSI (APSI) protocols [17, 18]. But, these PSI protocols revealed the client’s set cardinality. To hide the client’s set cardinality, Ateniese et al. [19] presented a PSI protocol that was to batch the hash value of the client. In 2012, Cristofaro et al. [20] used RSA and OPRF techniques to reduce the total cost of cryptographic operations based on Cristofaro et al.’s constructions [17, 18].

In 2017, Chen et al. [21] gave a PSI protocol with a low communication complexity based on the fully homomorphic encryption technology. In 2018, Chen et al. [22] implemented an unbalanced labeled PSI protocol against malicious adversaries by using OPRF into a preprocessing phase.

2.2. PSI Based on the Generic Circuit. The two main approaches were Yao’s garbled circuits [23, 24] and Goldreich protocol [25], which were to replace arbitrary functions with Boolean circuit computations. The communication overhead and the number of cryptographic operations depended on nonlinear gates’ number in the circuit. Thus, compared with the most special-purpose PSI protocols, the running time and communication complexity became more prominent problems for PSI protocols based on generic secure computation.

In 2012, Huang et al. [26] proposed several Boolean circuits for PSI protocols and evaluated based on Yao’s circuit, which used homomorphic encryption and adopted various circuit optimization techniques. The main method was that the client and the server sorted the elements in their sets locally and merged them in order through the garbled circuit and determined the equality of adjacent elements in the merged set. If they were equal, they would be the elements in the intersection. In 2015, Pinkas et al. [27] presented a circuit-phasing PSI protocol, which was up to 5 times faster than [26].

In 2018, Pinkas et al. [28] used a two-dimensional cuckoo hashing technique to realize a PSI based on the generic circuit, where it was asymptotically with better efficiency and could be extended to multiparty. For the general assumption of linear communication, Hemenway et al. [29], based on Pinkas et al.’s construction [27], represented a simple and generic circuit-based PSI protocol in 2019.

2.3. PSI Based on the Oblivious Transfer (OT) Scheme. In 2001, Naor et al. [30] proposed an OT protocol with asymmetric cryptographic operations, which spent expensive public-key operations when performing OT. Huberman et al. [11] used OT extensions (OTs) technology [31] to reduce expensive public-key operations by using more efficient symmetric cryptographic operations.

In 2013, Dong et al. [32] showed a PSI protocol that could process elements up to a size of 100 million. This protocol was based on bloom filter (BF), garbled bloom filter (GBF), secret sharing, and OTs. The linear complexity and high scalability of the protocol came from the effective symmetric cryptosystem and parallel processing, respectively. But, there was a problem with this protocol that the server might cause a selective failure to terminate the protocol in the malicious setting when the client performed a specific input. Thus, Rindal et al. [33] brought up an efficient fix using the cut-and-choose approach. Based on the method [30], Pinkas et al. [34] optimized it by replacing OTs with random OT, which did not need to save the GBF structure, but let the server and the client generate BF structure as the input of OT.

In 2015, Pinkas et al. [27] applied the phase and permutation hashing methods, which resulted in a reduction of computation and memory. Kolesnikov et al. [35] improved Pinkas et al.’s construction based on efficient OPRF. Subsequently, Kolesnikov et al. [36] proposed an extended version based on the literature [35], which gave a lightweight
protocol. Rindal et al. [33] gave the first implementation of PSI protocol against malicious adversaries. In 2018, Pinkas et al. [3] analyzed the current exiting protocols in detail and optimized PSI protocol using OPRF and the hashing techniques. A new PSI protocol was constructed by Pinkas et al. [37] in 2019, which used the 2-choice hashing [38], sparse OT extension, and the polynomial slice and stream techniques to reduce the communication cost and improve the efficiency of the protocol. In 2020, Pinkas et al. [39] proposed a PSI protocol based on a probe-and-XOR of strings (PaXoS) data structure, which not only had linear communication and computational complexity, but also can safely resist the malicious adversary in a nonprogrammable random oracle.

3. Preliminaries

3.1. Representing Set with Polynomial Point-Value Pairs.
We give the transformation from operations of sets to operations of polynomials. This representation allows us to represent a set using a random point evaluation polynomial.

Definition 1. Polynomial representation of a set. Given a set $S = \{s_1, \ldots, s_d\}$, whose set cardinality is $d$; then, we define its characteristic polynomial as

$$\rho(x) = \prod_{i=1}^{d} (x - s_i),$$

and thus every element $s_i \in S$ for $1 \leq i \leq d$ is a root of $\rho(x)$.

Definition 2. Polynomial in point-value pairs: distinct point-value pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ can represent a degree $d$ polynomial $\rho(x) = \prod_{i=1}^{d} (x - s_i)$, where $y_j = \rho(x_j)$ for $1 \leq j \leq n, n > d$. If $\mathbf{x}$ is fixed, the vector $\mathbf{y} = (y_1, \ldots, y_n)$ represents the polynomial $\rho(x)$.

Definition 3. Set intersection: let $S^{(A)}$ and $S^{(B)}$ be two sets with the same degree $d$ and represented as polynomials $\rho^{(A)}(x)$ and $\rho^{(B)}(x)$, respectively. Then, the intersection $S^{(A)} \cap S^{(B)}$ could be learnt by finding the roots of the following polynomial:

$$\rho^{(A)}(x) \cdot \rho^{(B)}(x) + \rho^{(A)}(x) \cdot \rho^{(B)}(x),$$

where $\rho^{(A)}(x)$ and $\rho^{(B)}(x)$ are $d$-degree polynomials, $\rho^{(A)}(x) = \prod_{i=1}^{d} (x - r^{(A)}_i)$ and $\rho^{(B)}(x) = \prod_{i=1}^{d} (x - r^{(B)}_i)$, and $r^{(A)}_i$ and $r^{(B)}_i$ are random values picked uniformly.

3.2. Hashing Techniques

3.2.1. The Simple Hashing. The simple hashing maps each element $x$ to its hashed positions. In particular, when hashing an element $x$, it stores $x$ in the bin $HT[H(x)]$, where $H$ is a random function: $[0, 1]^* \mapsto [1, b]$. To contain the multiple elements, $HT$ will be denoted as a double array $HT[i][\max_b]$ for $1 \leq i \leq b$. The insert function of simple hashing can be described as

for each element $x$ do,

if $HT[H(x)] \cdot \text{length} \leq \max_b$, \hspace{1cm} (3)

$x \mapsto HT[H(x)]$,

where $\max_b$ is introduced to represent the maximum number of each bin of hash table.

3.2.2. Permutation-Based Hashing. Permutation-based hashing technique is to allow the hashed elements to be converted shorter strings that can be stored in the hash table for reducing storage space and computation complexity, which was proposed by Arbitman et al. [40]. Originally, an element $x = x_1|x_2$ is represented as bits, where $|x_1| = \log_b b$ is the bins’ size in the hash table. Then, the element $x$ gets the index, $x_1 \oplus H(x_2)$, where $H$ is a random function: $[0, 1]^* \mapsto [1, b]$. Finally, the value stored in the bin is $x_2$, $|x_2| = |x| - |x_1|$. Thus, the stored data’s length is significantly reduced and efficiency will be improved.

3.3. Security Definitions. This section focuses on the security definition of PSI protocol.

3.3.1. Adversary. We consider a semihonest adversary who follows the protocol specifications while trying to obtain extra information from the exchanging messages.

3.3.2. Functionality. The functionality being implemented in this paper is $f_\pi(S^{(A)}, S^{(B)}) = (S^{(A)} \cap S^{(B)}), \bot$, that is, two parties; $A$ has a set $S^{(A)} = \{s_1^{(A)}, \ldots, s_d^{(A)}\}$ and gets the intersection, and $B$ has a set $S^{(B)} = \{s_1^{(B)}, \ldots, s_d^{(B)}\}$ but does not learn any output.

Definition 4. Simulations: $\text{Sim}_A$ and $\text{Sim}_B$, such that

$$\text{veiw}_a^{\pi}(S^{(A)}, S^{(B)}) \equiv \text{Sim}_A(S^{(A)}, f_\pi(S^{(A)}, S^{(B)})),$$

$$\text{veiw}_b^{\pi}(S^{(A)}, S^{(B)}) \equiv \text{Sim}_B(S^{(B)}, \bot),$$

where "$\equiv$" represents two views that are computationally indistinguishable.

4. The New PSI Protocol Based on Point-Value Polynomial Representation

The functionality that is implemented in the new PSI protocol is $f^{\pi}_\tau(S^{(A)}, S^{(B)}) = f(S^{(A)} \cap S^{(B)}, \bot)$. Let $I \in [A, B]$, $S^{(I)} = \{s_1^{(I)}, \ldots, s_d^{(I)}\}$, where $|S^{(I)}| = d$. The new protocol is shown in Figure 1 and has the following four steps.
(a) Setup: party $A$ constructs a public finite field $F_p$ where $p$ is a large prime, a pseudorandom function $\text{PRF}: \{0,1\}^* \times \mathbb{Z}_p^* \rightarrow F_p$ that generates pseudorandom values in $F_p$, and a vector $x \mapsto \{x_n\}$ with $n=2d+1$ distinct nonzero values picked randomly from $F_p$. Then, it publishes $F_p$, PRF, and $x \mapsto \{x_n\}$.

(b) Initialization: each party $I$ performs the following steps:

1. Select a dummy number $tk^{(i)}$ and compute pseudorandom values $r_i^{(A)} = \text{PRF}(tk^{(i)}, i)$ for $1 \leq i \leq d$ and then generate random polynomial $\gamma^{(i)}(x) = \prod_{j=1}^d (x - r_j^{(i)})$

2. Construct polynomial $\rho^{(i)}(x) = \prod_{j=1}^d (x - s_j^{(i)})$

3. Compute vectors $\{d_1, \ldots, d_n\}$ and $\{s_1, \ldots, s_n\}$ with values $\rho^{(i)}(x_j)$ and $\gamma^{(i)}(x_j)$, respectively, for $1 \leq j \leq n$, which are used to represent polynomials $\gamma^{(i)}(x)$ and $\rho^{(i)}(x)$.

4. Pick another random number $mk^{(i)}$ and generate pseudorandom values $z_j^{(i)} = \text{PRF}(mk^{(i)}, j)$ for $1 \leq j \leq n$ that are used to blind polynomial values.

Figure 1: A PSI Protocol based on point-value polynomial representation.
(c) Intersection interaction: party A tries to get the sum of two blinded polynomials point-value pairs whose roots are the intersection with party B. To do so, the following computations will be performed.

1. Party A computes $p_j = (y^{(A)}(x_j) + \rho^{(A)}(x_j)) \cdot z_j^{(A)}$ for $1 \leq j \leq n$ and sends the vector $\vec{p} = (p_1, \ldots, p_n)$ to party B.

2. Receiving party A’s message, party B blinds the vector $\vec{\phi}^{(B)}$ as follows:

$$d_j = \rho^{(B)}(x_j) \cdot z_j^{(B)} + y^{(B)}(x_j),$$
$$e_j = \rho^{(B)}(x_j) + y^{(B)}(x_j) \cdot z_j^{(B)}.$$  

(5)

where $1 \leq j \leq n$. Then, it gets the blinded vectors $\vec{\phi} = (e_1, \ldots, e_n)$ and $\vec{d} = (d_1, \ldots, d_n)$ and sends them to party A.

3. Party A computes the blinded vector $\vec{q}$, whose elements $q_j$ for $1 \leq j \leq n$ are computed as follows:

$$q_j = (e_j \cdot y^{(A)}(x_j) - \rho^{(A)}(x_j) \cdot d_j) \cdot z_j^{(A)}.$$  

(6)

4. Party B removes the blinding factors $z_j^{(B)}$ for $1 \leq j \leq n$ as follows:

$$c_j = \left( p_j \cdot o_j + q_j \right) \cdot (1 + z_j^{(B)})^{-1} = \left( \rho^{(B)}(x_j) \cdot y^{(A)}(x_j) - \rho^{(A)}(x_j) \cdot y^{(B)}(x_j) \right) \cdot z_j^{(A)}.$$  

(7)

Then, it sends the vector $\vec{\phi}$ to party A.

(d) Intersection result: party A gets set intersection.

1. Party A unblinds the blinding factors $z_j^{(A)}$ for $1 \leq j \leq n$ as follows:

$$y_j = (e_j) \cdot (z_j^{(A)})^{-1} = \rho^{(B)}(x_j) \cdot y^{(A)}(x_j) - \rho^{(A)}(x_j) \cdot y^{(B)}(x_j).$$  

(8)

2. Party A restores polynomials $\phi(x)$ by using point-value pair interpolation $(x_j, y_j)$ for $1 \leq j \leq n$.

3. Party A checks each element $s_i^{(A)}$ for $1 \leq i \leq d$ whether it is a root of $\phi(x)$. If it holds, it is an element of intersection; otherwise it is not.

4.1. Correctness of the New Protocol. Because for $1 \leq j \leq n$,

$$q_j = (e_j \cdot y^{(A)}(x_j) - \rho^{(A)}(x_j) \cdot d_j) \cdot z_j^{(A)}$$

$$= (\rho^{(B)}(x_j) + y^{(B)}(x_j) \cdot z_j^{(B)}) \cdot y^{(A)}(x_j) \cdot z_j^{(A)} - (y^{(B)}(x_j) + \rho^{(B)}(x_j) \cdot z_j^{(B)}) \cdot \rho^{(A)}(x_j) \cdot z_j^{(A)}.$$  

(9)

Next,

$$y_j = c_j \cdot z_j^{(A)} \cdot \rho^{(A)}(x_j).$$  

From Definition 3, we can get the intersection by computing the roots of the polynomial $\phi(x)$.

Thus, the new protocol is correct.

5. Efficient PSI Protocol Using Hashing

We optimize the above protocol using the permutation-based hashing. At first, each party constructs a two-dimensional hash table $HT$, where the first dimension is the
index of the hashed element and the second dimension stores the elements. Then, each party pads the second dimension with random values to the maximum load. The permutation-based hashing makes each party break down its original set into several small subsets. Thus, it will greatly reduce the degree of polynomials and then significantly improve the efficiency of the protocol. Let \( I \in \{ A, B \} \), \( S^{(l)} = \{ s_1^{(l)}, \ldots, s_d^{(l)} \} \), where \( |S^{(l)}| = d \). The hashing protocol is shown in Figure 2, and the details include the following steps.

(a) Setup: party A selects a permutation-based hashing function \( H \) with the parameters \( b \) and \( \max_b \) for the hash table, where \( b \) is bins’ size in the hash table and \( \max_b \) denotes the maximum length in a bin. Next, it constructs a public finite field \( F_p \) where \( p \) is a large prime, a pseudorandom function \( \text{PRF}: \{0, 1\}^* \times \mathbb{Z}_p \rightarrow F_p \) that generates pseudorandom values in \( F_p \), and a vector \( \overline{x} \) with \( n = 2\max_b + 1 \) distinct nonzero values picked randomly from \( F_p \). Then, it publishes \( H, b, \max_b, F_p, \overline{x} \), and \( \text{PRF} \).

(b) Hashing: each party \( I \) performs the following.

1. Create a hash table \( HT^{(l)} \) by doing the following:
   \[
   H(s_i^{(l)}) = j, \quad s_i^{(l)} \rightarrow HT^{(l)}[j], \quad \text{for } 1 \leq i \leq d. \tag{12}
   \]

2. For every bin \( HT^{(l)}[j] (1 \leq j \leq b) \), get an array \( \text{flag}^{(l)}[j] = HT^{(l)}[j].\text{size} \) that holds the size of the actual mapped elements. If its size is less than \( \max_b \), pick dummy elements, \( a_j^{(l)}(\text{flag}^{(l)}[j] + 1 \leq t \leq \max_b) \), and pad them to the bin \( HT^{(l)}[j] \).

(c) Initialization: each party \( I \) chooses a dummy quantity \( t_k^{(l)} \), and for each bin, \( j (1 \leq j \leq b) \) does the following.

1. Generate a pseudorandom value \( k_j^{(l)} \) by using \( t_k^{(l)} \):
   \[
   k_j^{(l)} = \text{PRF}(t_k^{(l)}, j). \tag{13}
   \]

2. Generate \( \max_b \) pseudorandom values and construct a random polynomial as follows:
   \[
   r_j^{(l)} = \text{PRF}(k_j^{(l)}, i), \quad \text{for } 1 \leq i \leq \max_b, \quad y_j^{(l)}(x) = \prod_{i=1}^{\max_b} (x - r_j^{(l)}). \tag{14}
   \]

3. Construct a polynomial \( p_j^{(l)}(x) \) to represent the elements in the bin \( HT^{(l)}[j] \):
   \[
   p_j^{(l)}(x) = \prod_{i=1}^{\max_b} (x - HT^{(l)}[j][i]). \tag{15}
   \]

4. Choose a random number \( mk_j^{(l)} \) and get \( n \) pseudorandom values \( z_j^{(l)} \) that are used to blind the polynomial values:
   \[
   z_j^{(l)} = \text{PRF}(mk_j^{(l)}, t), \quad \text{for } 1 \leq t \leq n. \tag{16}
   \]

5. Compute vectors \( \overrightarrow{y}_j^{(l)} \) and \( \overrightarrow{\rho}_j^{(l)} \) with values \( y_j^{(l)}(x_i) \) and \( \rho_j^{(l)}(x_i) \), respectively, for \( 1 \leq t \leq n \), which are used to represent polynomials \( y_j^{(l)}(x) \) and \( \rho_j^{(l)}(x) \).

(d) Intersection interaction: party A tries to get the sum of two blinded polynomials point-value pairs whose roots are the intersection with party B. To do so, the following computation will be performed.

1. Party A computes \( p_{j,t} = (y_j^{(A)}(x_i) + \rho_j^{(A)}(x_i)) \cdot z_j^{(A)} \) for \( 1 \leq j \leq b \) and \( 1 \leq t \leq n \), and sends the vector \( \overrightarrow{p} = (\overrightarrow{p_1}, \ldots, \overrightarrow{p_b}) \) to party B.

2. Receiving party B’s message, party B blinds every value as follows:
   \[
   d_{j,t} = \rho_j^{(B)}(x_i) \cdot z_j^{(B)} + y_j^{(B)}(x_i), \tag{17}
   \]
   \[
   e_{j,t} = \rho_j^{(B)}(x_i) + y_j^{(B)}(x_i) \cdot z_j^{(B)}.
   \]
   where \( 1 \leq j \leq b \) and \( 1 \leq t \leq n \). Then, it gets the blinded vectors \( \overrightarrow{e} = (\overrightarrow{e_1}, \ldots, \overrightarrow{e_b}) \) and \( \overrightarrow{d} = (\overrightarrow{d_1}, \ldots, \overrightarrow{d_b}) \) and sends them to party A.

3. Party A computes the blinded vector \( \overrightarrow{a} \) as follows:
   \[
   q_{j,t} = (e_{j,t} - y_j^{(A)}(x_i) - \rho_j^{(A)}(x_i) \cdot d_{j,t}) \cdot z_j^{(A)}, \tag{18}
   \]
   where \( 1 \leq j \leq b \) and \( 1 \leq t \leq n \). Then, it gets the blinded vectors \( \overrightarrow{q} = (\overrightarrow{q_1}, \ldots, \overrightarrow{q_b}) \) and sends them to party B.

4. Party B removes the blinding factors \( z_j^{(B)} \) as follows:
   \[
   c_{j,t} = (\rho_j^{(B)}(x_i) \cdot y_j^{(A)}(x_i) - \rho_j^{(A)}(x_i) \cdot y_j^{(B)}(x_i) \cdot z_j^{(B)}), \tag{19}
   \]
   where \( 1 \leq j \leq b \) and \( 1 \leq t \leq n \). Then, it gets the vectors \( \overrightarrow{c} = (\overrightarrow{c_1}, \ldots, \overrightarrow{c_b}) \) and sends them to party A.

(e) Intersection result: for each bin \( j (1 \leq j \leq b) \), party A restores the subpolynomial by the interpolation and gets the intersection by computing the roots of the subpolynomial.

1. Party A removes the blinding factors \( z_j^{(A)} \) for \( 1 \leq t \leq n \) as follows:
   \[
   y_{j,t} = (c_{j,t})^{-1} = \rho_j^{(B)}(x_i) \cdot y_j^{(A)}(x_i) - \rho_j^{(A)}(x_i) \cdot y_j^{(B)}(x_i). \tag{20}
   \]

2. Party A restores the subpolynomial \( \phi_j(x) \) by using the point-value pairs interpolation \( (\overrightarrow{x}, \overrightarrow{y}_j) \).
Party $A$ finds the elements of the intersection $S^A \cap S^B$ by computing the roots of polynomial $\varphi_j(x)$ as follows:

\[
\text{if } \varphi_j(HT^A[j][i]) = 0, \quad HT^A[j][i] \in S^A \cap S^B, \quad (21)
\]

where $1 \leq i \leq \text{flag}^A[j]$, $\text{flag}^A[j]$ denotes the size of the actual elements in bin $j$.

### 5.1. Correctness of the Hashing Protocol

Because for $1 \leq j \leq b$, $1 \leq t \leq n$,

| $A$ | $B$ |
|-----|-----|
| **Input**: $S(A) = \{ s_1^A, \ldots, s_d^A \}$ | **Input**: $S(B) = \{ s_1^B, \ldots, s_d^B \}$ |
| **Output**: $S^A \cap S^B$ | **Output**: $\perp$ |

**Hashing**

\[
H : (0, 1)^* \rightarrow [1, \ldots, b] ; \text{ hash function}
\]

$b$: bin's size, $\max_b$: the maximum load of each bin

\[
S^A = \{ x_1, \ldots, x_{\max} \}, \quad n = 2 + \max_b + 1
\]

**Initialization**

\[
k_1^A = \text{PRF}(k_A^A, j)
\]

\[
k_j^A = \text{PRF}(k_A^A, j)
\]

\[
t_j = \text{Flag}^A[j]
\]

where $1 \leq i \leq \text{flag}^A[j]$, $\text{flag}^A[j]$ denotes the size of the actual elements in bin $j$.

**Set Intersection**

\[
p_{j,t} = \rho_j^A(x_t) + \gamma_j^A(x_t) \cdot s_j^A
\]

\[
\{ p_1, \ldots, p_k \}
\]

\[
p_{j,t} \in \{ x_1, \ldots, x_{\max} \}
\]

\[
p_{j,t} = \rho_j^B(x_t) + \gamma_j^B(x_t)
\]

\[
\{ p_1, \ldots, p_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

\[
\{ q_1, \ldots, q_k \}
\]

**Figure 2**: Efficient PSI protocol using the permutation-based hashing.
Theorem 1. Security and Communication Networks 9

The simulator $\text{Sim}_A$ is constructed and performs the following steps:

1. Create an empty view and then append $S'(A)$ and $f_\gamma(\mathcal{S}(A), \mathcal{S}(B))$ to it.

2. Pick a set $\mathcal{S}'(B)$ with $d$ elements at random such that $S'(A) \cap \mathcal{S}'(B) = f_\gamma(\mathcal{S}(A), \mathcal{S}(B))$.

3. Construct polynomials $\rho'(A)(x)$ and $\rho'(B)(x)$ ($1 \leq j \leq b$) representing sets $\mathcal{S}'(A)$ and $\mathcal{S}'(B)$, respectively. And generate random polynomials $y'(A)(x)$ and $y'(B)(x)$ for $1 \leq j \leq b$.

4. Generate the Random Values $z'_j(A)$ and $z'_j(B)$ for $1 \leq j \leq b, 1 \leq t \leq n$.

5. Blind the polynomials’ values and get random vectors $\overline{p}'$, $\overline{q}'$, $\overline{c}'$, and $\overline{d}'$, where $p'_j, e'_j, d'_j$, and $q'_j$ ($1 \leq j \leq b, 1 \leq t \leq n$) are computed as follows:

$$
p'_j = (y'(A)(x) + \rho'(A)(x)) \cdot z'_j(A),
\quad d'_j = p'_j + y'(B)(x),
\quad e'_j = \rho'(B)(x),
\quad q'_j = (e'_j + y'(A)(x) - \rho'(A)(x)) \cdot z'_j(A).
$$

6. Compute the simulator’s view is

$$
\text{Sim}_A(S'(A), f_\gamma(\mathcal{S}(A), \mathcal{S}(B))) = \left(S'(A), \overline{p}', \overline{d}', \overline{c}', \overline{q}', \overline{d}', \overline{c}' \right).
$$

Note that, in both views, the input $\mathcal{S}(A)$ and the output $f_\gamma(\mathcal{S}(A), \mathcal{S}(B))$ are identical. Pseudorandom function is used to blind the elements in the real protocol. We can get blinded vectors $\overline{p}$, $\overline{q}$, $\overline{d}$, $\overline{c}$, $\overline{c}'$, $\overline{d}'$, $\overline{c}'$, $\overline{q}'$, and $\overline{d}'$ from the real protocol. On the other hand, through the calculation of the simulator, $\overline{p}$, $\overline{d}$, $\overline{c}$, $\overline{c}'$, $\overline{d}'$, $\overline{c}'$, $\overline{q}'$, and $\overline{d}'$ are random vectors. So, $\overline{p}$ and $\overline{p}'$, $\overline{q}$ and $\overline{q}'$, $\overline{d}$ and $\overline{d}'$, $\overline{c}$ and $\overline{c}'$, $\overline{c}'$ and $\overline{c}'$ are computationally indistinguishable.

Case 2. Corrupted party $B$: in the real protocol, party $B$’s view is

$$
\text{View}_B^\mathcal{S}(S'(A), \mathcal{S}(B)) = \left(S'(A), \overline{p}, \overline{d}, \overline{c}, \overline{q}, \overline{c}' \right).
$$

We construct a simulator $\text{Sim}_B$ who has the input $\mathcal{S}(B)$ and performs the following steps:

1. Create an empty view and then append $\mathcal{S}(B)$ to it.

2. Pick a set $\mathcal{S}'(A)$ with $d$ elements at random.
(3) Construct polynomials $\rho_{j}^{(A)}(x)$ and $\rho_{j}^{(B)}(x_{j})(1 \leq j \leq b)$, representing sets $S^{(A)}$ and $S^{(B)}$, respectively. And generate random polynomials $\gamma_{j}^{(A)}(x)$ and $\gamma_{j}^{(B)}(x)$ for $1 \leq j \leq b$.

(4) Generate the Random Values $z_{j}^{(A)}$ and $z_{j}^{(B)}$ for $1 \leq j \leq b, 1 \leq t \leq n$

(5) Blind the polynomials’ values and get random vectors $\bar{p'}$, $\bar{q'}$, $\bar{d'}$, and $\bar{e'}$, where $p_{j}^{t}$, $e_{j}^{t}$, $d_{j}^{t}$, and $q_{j}^{t}$ for $1 \leq j \leq b, 1 \leq t \leq n$ are computed as follows:

$$p_{j}^{t} = (\gamma_{j}^{(A)}(x_{j}) + \rho_{j}^{(A)}(x_{j})): z_{j}^{(A)},$$
$$d_{j}^{t} = \rho_{j}^{(B)}(x_{j}) \cdot z_{j}^{(B)},$$
$$e_{j}^{t} = (\gamma_{j}^{(B)}(x_{j}) + \gamma_{j}^{(B)}(x_{j})): z_{j}^{(B)},$$
$$q_{j}^{t} = (e_{j}^{t}: \gamma_{j}^{(A)}(x_{j}) - \rho_{j}^{(A)}(x_{j}) \cdot d_{j}^{t}: z_{j}^{(A)}).$$

(6) Compute $c_{j}^{t} = (\rho_{j}^{(B)}(x_{j}) \cdot \gamma_{j}^{(A)}(x_{j}) - \rho_{j}^{(B)}(x_{j}))$ for $1 \leq j \leq b, 1 \leq t \leq n$, and get random vector $\bar{c'}$, $\bar{q'}$, $\bar{d'}$, and $\bar{e'}$ to the view.

(7) Insert vector $\bar{p'}$, $\bar{q'}$, $\bar{d'}$, $\bar{e'}$, and $\bar{c'}$ to the view.

So, the view of the simulator Sim$_{B}$’s construction is

$$\text{Sim}_{B}(S^{(B)}, \bot) = (S^{(B)}, \bar{p'}, \bar{d'}, \bar{e'}, \bar{q'}, \bar{c'}, \bot).$$

Note that, in both views, the input $S^{(B)}$ is identical and the output $\bot$ is empty. Pseudorandom function is used to blind the elements in the real protocol. We can get blinded vectors $\bar{p}$, $\bar{q}$, $\bar{d}$, $\bar{e}$, and $\bar{c}$ are random vectors. Thus, $\bar{p}$ and $\bar{p'}$, $\bar{q}$ and $\bar{q'}$, $\bar{d}$ and $\bar{d'}$, $\bar{e}$ and $\bar{e'}$, and $\bar{c}$ and $\bar{c'}$ are computationally indistinguishable.

Combining the above, we can get that

$$\text{view}_{A}(S^{(A)}, S^{(B)}) \equiv \text{Sim}_{A}(S^{(A)}, f_{\text{view}}^{A}(S^{(A)}, S^{(B)})), \quad \text{view}_{B}(S^{(A)}, S^{(B)}) \equiv \text{Sim}_{B}(S^{(B)}, \bot).$$

Therefore, the hashing protocol is secure in the semi-honest model.

### 6. Evaluation

#### 6.1. Implementation

We ran our experiments in Ubuntu 18.04 with Linux 4.4.0-59 64-bit desktop PC. All protocols were implemented and executed using the same hardware equipped with Intel Core i7-7700K CPU with 3.6 GHz and 8 GB of RAM. We implemented our protocol and related protocols [3, 18, 26, 32, 34] in the same environment setting. Our protocol and related protocols had the same number of input elements, whose size was 32 bits. Our protocol was implemented using the Number Theory Library (NTL) along with the GNU Multiprecision (GMP) library for polynomial arithmetic.

We give the running times of related protocols in Table 1 and Figure 3. From them, it can be seen that our protocol is more efficient than public-key-based and circuit-based PSI protocols, and it is more efficient than OT-based PSI protocols with the set size less than $2^{12}$.

#### 6.2. Experimental Results

A detailed analysis with related PSI protocols is given in Table 2. We evaluate the performance in terms of four properties: needing cryptographic or not, simulated-based security, computation complexity, and communication complexity. From Table 2, our protocol enjoys the following advantages. (1) There is no need for a complicated cryptographic system in our protocol, which only uses hashing and pseudorandom function and provides a lightweight system. But, in other protocols, asymmetric encryption system or symmetric encryption system is needed. (2) Our protocol gives a detailed formal security proof by using the ideal simulation mechanism in the standard model while [3, 26, 34] only show an informal security analysis. (3) Computation complexity and communication complexity of our protocol are $O(d)$, while both of [26] are $O(d \cdot \log^d)$.

| Type   | Protocol Set size | 2^8 | 2^12 | 2^16 | 2^20 |
|--------|------------------|-----|------|------|------|
| Public-key | Cristofaro2010 [18] 779 12546 203036 3193920 |
| Circuit   | Huang2012 [26] 79 1377 32292 — |
|          | Dong2013 [32] 105 448 4179 65218 |
| OT        | Pinkas2014 [34] 95 346 2991 49171 |
| Novel     | Pinkas2018 [3] 311 362 702 5847 |
|          | Dong2013 [32] 105 448 4179 65218 |

**Figure 3:** Running time in ms with related PSI protocols.
7. Conclusion and Future Work

In this paper, we proposed a new approach to PSI protocol without any cryptographic based on point-value polynomial representation and pseudorandom function and optimized it based on hashing techniques. Our protocol had high performance with set elements less than $2^{12}$. In our protocols, there was a constraint that both parties should have the same set degree. In the future, we will extend our approach and study PSI protocols with a lightweight client where the server had a very large degree but the client’s degree is relatively small.

Data Availability

All the pseudocodes used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant nos. 61672010, 61702168, and 61701173 and the fund of Hubei Key Laboratory of Transportation Internet of Things (WHUT/IOT-2017B001).

| Type      | Protocol          | Needing cryptosystem | Simulated-based Security | Computation Complexity | Communication Complexity |
|-----------|-------------------|----------------------|--------------------------|------------------------|--------------------------|
| Public-key Circuit | Cristofaro2010 [18] | asym                 | Yes                      | $O(d)$                 | $O(d)$                   |
|           | Huang2012 [26]    | sym                  | No                       | $O(d \cdot \log^d)$     | $O(d \cdot \log^d)$      |
|           | Dong2013 [32]     | sym                  | Yes                      | $O(d)$                 | $O(d)$                   |
| OT        | Pinkas2014 [34]   | sym                  | No                       | $O(d)$                 | $O(d)$                   |
| Novel     | Our               | No                   | Yes                      | $O(d)$                 | $O(d)$                   |

asym: public-key cryptography. sym: symmetric cryptography. $d$: set cardinality.

References

[1] T. Altuwaiyan, M. Hadian, L. Liang et al., “EPIC: efficient privacy-preserving contact tracing for infection detection,” in Proceedings of the ICC 2018, IEEE, Kansas, MO, USA, pp. 1–6, May 2018.
[2] T. Qiang, “Privacy-preserving contact tracing: current solutions and open questions,” 2020, https://arxiv.org/abs/2004.06818.
[3] 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.
[4] S. P. Kim, M. S. Gil, H. Kim et al., “Efficient two-step protocol and its discriminative feature selections in secure similar document detection,” Security and Communication Networks, vol. 2017, Article ID 6841216, 12 pages, 2017.
[5] M. Fischlin, B. Pinkas, A. R. Sadeghi, T. Schneider, and I. Visconti, “Secure set intersection with untrusted hardware tokens,” in Proceedings of the CT-RSA 2011, vol. 6558, LNCS, San Francisco, CA, USA, pp. 1–16, February 2011.
[6] G. Mezzour, A. Perrig, V. D. Gligor, and P. Papadimitratos, “Privacy-preserving relationship path discovery in social networks,” in Proceedings of the CANS 2009, vol. 5888, LNCS, Kanazawa, Japan, pp. 189–208, December 2009.
[7] L. Kamm and J. Willemsen, “Secure floating point arithmetic and private satellite collision analysis,” International Journal of Information Security, vol. 14, no. 6, pp. 531–548, 2015.
[8] V. Shoup, “NTL: number theory library,” 2016, http://www.shoup.net/ntl/.
[9] T. Granlund, “GMP: the GNU multiple precision arithmetic library,” 2020, http://gmplib.org/.
[10] C. Meadows, “A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party,” in Proceedings of the SP 1986, IEEE, Oakland, CA, USA, pp. 134–137, April 1986.
[11] B. A. Huberman, M. Franklin, and T. Hogg, “Enhancing privacy and trust in electronic communities,” in Proceedings of the EC 1999, ACM, Denver, CO, USA, pp. 78–86, November 1999.
[12] M. J. Freedman, K. Nissim, and B. Pinkas, “Efficient private matching and set intersection,” in Proceedings of the Advances in Cryptology-EUROCRYPT 2004, vol. 3027, pp. 1–19, LNCS, Interlaken, Switzerland, May 2004.
[13] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469–472, 1985.
[14] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Proceedings of the EUROCRYPT 1999, vol. 1592, LNCS, Prague, Czech Republic, pp. 223–238, May 1999.
[15] M. J. Freedman, C. Hazay, K. Nissim, and B. Pinkas, “Efficient set intersection with simulation-based security,” Journal of Cryptology, vol. 29, no. 1, pp. 115–155, 2016.
[16] S. Jarecki and X. Liu, “Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection,” Theory of Cryptography, Springer, vol. 5444, pp. 577–594, Berlin, Germany, 2009.
[17] E. D. Cristofaro, J. Kim, and G. Tsudik, “Linear-complexity private set intersection protocols secure in malicious model,” in Proceedings of the ASIACRYPT 2010, vol. 6477, LNCS, Springer, Singapore, pp. 213–231, December 2010.
[18] E. D. Cristofaro and G. Tsudik, “Practical private set intersection protocols with linear complexity,” in Proceedings of the FC 2010, vol. 6052, LNCS, Canary Islands, Spain, pp. 143–159, January 2010.
[19] G. Ateniese, E. De Cristofaro, and G. Tsudik, “If size matters: size-hiding private set intersection,” Public Key Cryptograph-
[20] E. D. Cristofaro and G. Tsudik, “Experimenting with fast private set intersection,” in *Proceedings of the TRUST*, vol. 7344, LNCS, Vienna, Austria, pp. 55–73, June 2012.

[21] H. Chen, K. Laine, and P. Rindal, “Fast private set intersection from homomorphic encryption,” in *Proceedings of the CCS 2017*, ACM, Dallas, TX, USA, pp. 1243–1255, October 2017.

[22] H. Chen, Z. Huang, K. Laine, and P. Rindal, “Labeled PSI from fully homomorphic encryption with malicious security,” in *Proceedings of the CCS 2018*, ACM, Toronto, Canada, pp. 1223–1237, January 2018.

[23] A. C. Yao, “How to generate and exchange secrets,” in *Proceedings of the FOCS 1986*, IEEE, Toronto, Canada, pp. 162–167, October 1986.

[24] L. Xin, L. Shundong, C. Xiubo et al., “Efficient solutions to two-party and multiparty millionaires’ problem,” *Security and Communication Networks*, vol. 2017, Article ID 5207386, 11 pages, 2017.

[25] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game or a completeness theorem for protocols with honest majority,” in *Providing Sound Foundations for Cryptography: on the Work of Shafi Goldwasser and Silvio Micali*, pp. 411–496, ACM, New York, NY, USA, 2019.

[26] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, “Efficient circuit-based PSI via cuckoo hashing,” in *Proceedings of the EUROCRYPT 2018*, vol. 10822, LNCS, Tel Aviv, Israel, pp. 125–157, April 2018.

[27] B. Pinkas, M. Rosulek, and A. Yanai, “Faster private set intersection based on OT extension,” in *Proceedings of the 23rd USENIX Security Symposium*, USENIX, San Diego, CA, USA, pp. 797–812, August 2014.

[28] V. Kolesnikov and R. Kumaresan, “Improved OT extension for transferring short secrets,” in *Advances in Cryptology-CRYPTO 2013*, Springer, vol. 8043, pp. 54–70, Berlin, Germany, 2013.