symmetry

MDPI

Article

A Secure and Lightweight Multi-Party Private Intersection-Sum Scheme over a Symmetric Cryptosystem

Junwei Zhang [1], Xin Kang [1], Yang Liu [1,*], Huawei Ma [2], Teng Li [1], Zhuo Ma [1] and Sergey Gataullin [3]

[1] School of Cyber Engineering, Xidian University, Xi'an 710071, China
[2] Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou 510006, China
[3] Faculty of Digital Economy and Mass Communications, Moscow Technical University of Communications and Informatics, 123423 Moscow, Russia
* Correspondence: bcds2018@foxmail.com

Abstract: A private intersection-sum (PIS) scheme considers the private computing problem of how parties jointly compute the sum of associated values in the set intersection. In scenarios such as electronic voting, corporate credit investigation, and ad conversions, private data are held by different parties. However, despite two-party PIS being well-developed in many previous works, its extended version, multi-party PIS, has rarely been discussed thus far. This is because, depending on the existing works, directly initiating multiple two-party PIS instances is considered to be a straightforward way to achieve multi-party PIS; however, by using this approach, the intersection-sum results of the two parties and the data only belonging to the two-party intersection will be leaked. Therefore, achieving secure multi-party PIS is still a challenge. In this paper, we propose a secure and lightweight multi-party private intersection-sum scheme called SLMP-PIS. We maintain data privacy based on zero sharing and oblivious pseudorandom functions to compute the multi-party intersection and consider the privacy of associated values using arithmetic sharing and symmetric encryption. The security analysis results show that our protocol is proven to be secure in the standard semi-honest security model. In addition, the experiment results demonstrate that our scheme is efficient and feasible in practice. Specifically, when the number of participants is five, the efficiency can be increased by 22.98%.

Keywords: private intersection-sum; secure computation; cloud computing

Citation: Zhang, J.; Kang, X.; Liu, Y.; Ma, H.; Li, T.; Ma, Z.; Gataullin, S. A Secure and Lightweight Multi-Party Private Intersection-Sum Scheme over a Symmetric Cryptosystem. *Symmetry* **2023**, *15*, 319. https://doi.org/10.3390/sym15020319

Academic Editors: Lianyong Qi, Wajid Rafiq, Wenwen Gong, Maqbool Khan and Lorentz Jäntschi

Received: 7 December 2022
Revised: 19 January 2023
Accepted: 19 January 2023
Published: 23 January 2023

1. Introduction

With the rapid development of artificial intelligence [1–6] and cloud computing [7–11], as a carrier of collaborative computing, the use of data is becoming increasingly important in various industries [12–17]. However, these data involve a large amount of private information, and the leakage of private data will cause serious security problems. Therefore, privacy computing [18–21] has been widely considered the optimal solution for maintaining data security [22–25] and protecting privacy [26–29]. Private intersection-sum (PIS) is one of the most commonly used privacy-preserving protocols in specific scenarios. It allows each party to hold a private set of data, and non-leadership parties additionally hold a private integer value associated with each datum, to jointly compute the sum of the associated integer values in the intersection.

Up to now, the two-party PIS has been applied in various fields, such as ad conversions [30] and electronic voting [31]. However, in other scenarios, private data are held by different parties. For example, in enterprise credit investigations, it is necessary to count the total amount of loans of multiple legal persons across different banks. Therefore, the multi-party PIS protocol must be considered. In addition, it is difficult for parties to participate in online computing in real time. Therefore, the outsourcing of data to the cloud by the participant, i.e., outsourced multi-party PIS, is a potential solution.

However, on the one hand, the existing PIS protocols primarily consider the two-party scenarios [30–33]. The function of multi-party PIS can be realized by executing the two-party PIS protocol multiple times, but using this approach, the intersection-sum results of the two parties and the data only belonging to the two-party intersection will be leaked. On the other hand, cloud systems store a large amount of user data and present greater temptation for attackers to conduct data mining and other attacks on private user data in the cloud. Therefore, it is challenging to design an outsourcing multi-party PIS protocol.

To summarize, we achieve the following contributions:

- We propose a secure and lightweight multi-party private intersection-sum scheme, called SLMP-PIS, which avoids the data privacy leakage problem of only repeatedly conducting existing two-party PIS schemes.
- SLMP-PIS is client-agnostic. The requester can ask the cloud server to obtain the computation result without the help of data owners, and data owners can maintain their offline status as long as their data have been outsourced securely to the cloud.
- SLMP-PIS is based on symmetric cryptosystem only. Therefore, the larger the number of participants, the more efficient SLMP-PIS is. Specifically, when the number of participants is five, the efficiency can be increased by 22.98%.

The remaining parts of this paper are organized as follows. Section 2 shows the related work associated with our work. Section 3 describes preliminaries used in our scheme, and we introduce the problem's formulations, including the system model and adversary model in Section 4. Section 5 presents the scheme's construction in detail. In Section 6, we conduct a security analysis, followed by Section 7, which shows the performance analysis. Finally, the conclusions of this whole paper are summarized in Section 8.

2. Related Work

We first introduce the related work on PIS. Ion et al. [32] described three PIS with cardinality protocols. One relies on a Diffie–Hellman-style double masking, and the other two use random oblivious transfer and encrypted Bloom filters. Miao et al. [33] proposed a private intersection-sum scheme with cardinality based on a shuffled, distributed, oblivious, pseudorandom function (DOPRF); Pedersen commitments; ElGamal encryptions; and Camenisch–Shoup encryptions. Niu et al. [34] described a privacy-preserving statistical computing protocol for the private set intersection to complete the relevant statistical computations of the intersection of two private sets, including cardinality, sum, average, variance, range, and so forth.

Currently, the PIS protocol has been applied in specific fields. Ion et al. [30] proposed a private intersection-sum protocol with the purpose of attributing aggregate ad conversions. Lu et al. [31] described a private intersection weighted sum protocol for privacy-preserving score-based voting with perfect ballot secrecy. Kulshrestha et al. [35] proposed a non-strict private intersection-sum protocol to compute $\sum_{x \in I} T[x]$, where $I = X_0 \bigcap (\bigcup_{i=1}^{n-1} X_i)$ and T is associated with values of X; the protocol was applied for estimating incidental collection in foreign intelligence surveillance.

In summary, the current state of the research shows that there have been PIS schemes based on different cryptographic primitives, and existing schemes are dedicated to minimizing the computational cost and communication overhead of the protocol while satisfying the basic security properties of PIS. However, none of them consider the strict multi-party privacy intersection-sum problem.

3. Preliminaries

In this section, we introduce some cryptographic tools, including oblivious transfer (OT), oblivious pseudorandom function (OPRF), and arithmetic sharing (AS).

3.1. Oblivious Transfer

Oblivious transfer [36,37] is a two-party protocol between a sender and a receiver. For 1-out-of-2 OT, the sender uses two private inputs, m_0 and m_1, and the receiver uses one private input bit $\sigma(\sigma \in \{0,1\})$. After the protocol, the receiver obtains m_σ. Moreover, 1-out-of-2 OT must satisfy the following properties:

(1) Receiver's indistinguishability security. For any $\sigma, \tau \in \{0,1\}$ and for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ executing the sender's part, the views that $\mathcal{A}$ sees in case the receiver tries to obtain m_σ and in case the receiver tries to obtain m_τ are computationally indistinguishable given m_0 and m_1.
(2) Sender's indistinguishability security.For any adversary $\mathcal{A}$ substituting the receiver and a simulator $\mathcal{A}^{'}$ playing the receiver's role in the ideal model, the outputs of $\mathcal{A}$ and $\mathcal{A}^{'}$ are statistically indistinguishable given m_0 and m_1.

3.2. Oblivious Pseudorandom Function

The oblivious pseudorandom function [38] involves two parties: one is the sender (denoted by S), and the other is the receiver (denoted by R). R takes data y as input. After the protocol is executed, R obtains the pseudorandom function value $F_y = F(k, y)$ of the data y, but cannot obtain any information about k. S obtains the key k of the pseudorandom function, so the pseudorandom function value $F_x = F(k, x)$ can be calculated for any data x.

3.3. Arithmetic Sharing

The arithmetic sharing protocol involves two parties: one is P_0, and the other is P_1. For a secret x, P_0 and P_1 each obtain a shared value. No one can obtain x alone, and they can recover x together.

Arithmetic sharing contains three algorithms, the shared values, sharing, and reconstruction, which are described as follows:

- Shared Values: on input secret $x(x \in Z_N)$, output sharing values $\langle x \rangle_0^A, \langle x \rangle_1^A$ satisfying $\langle x \rangle_0^A + \langle x \rangle_1^A = x \mod N$.
- Sharing: $P_i(i \in \{0,1\})$ chooses $r(r \in Z_N)$ and sets $\langle x \rangle_i^A = (x - r) \mod N$. Then, P_i sends r to P_{1-i}, that is, $\langle x \rangle_{1-i}^A = r \mod N$.
- Reconstruction: P_{1-i} sends $\langle x \rangle_{1-i}^A$ to P_i, which computes $x = (\langle x \rangle_0^A + \langle x \rangle_1^A) \mod N$.

4. Problem Formulations

In this section, we introduce the system model and adversary model of SLMP-PIS.

4.1. System Model

The system model of SLMP-PIS is shown in Figure 1. It contains three entities: a requester (Re), a cloud server (CS), and data owners (DOs).

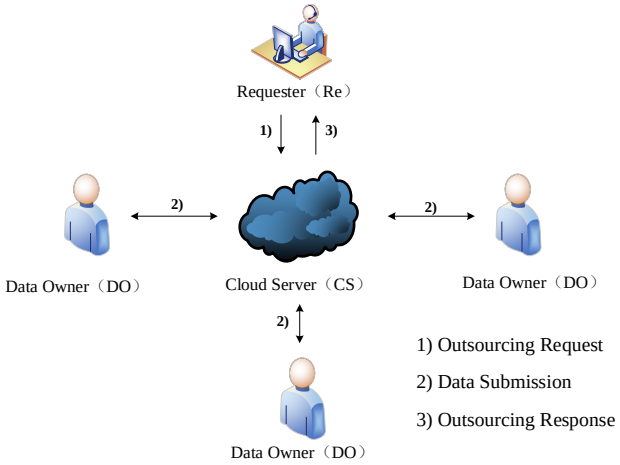

Figure 1. System model.

(1) Requester. The Re is responsible for submitting the processed data set to CS. In addition, the Re obtains the sharing value of the intersection-sum from the CS and computes the private intersection-sum result.
(2) Cloud Server. The CS acts as a connection between the Re and DOs. After receiving the processed data set from the Re, the CS assists the DOs in performing data processing and obtaining the processed data sets from the DOs. In addition, the CS computes the sharing value of the intersection-sum and sends it to the Re.
(3) Data Owners. Each DO holds a private set of data, and additionally holds a private integer value associated with each element. Each DO is responsible for submitting the processed data set to the CS.

4.2. Adversary Model

In the adversary model of SLMP-PIS, the Re, CS, and DOs are semi-honest parties. They strictly follow the protocol. However, the Re is interested in learning the data for the DOs and the private integer value associated with each element in the DOs' sets. Each DO is willing to learn the data of the Re and the other DOs. The CS remains curious about the raw data of the Re and DOs.

Therefore, we introduce three active adversaries in our model: $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. The goal of $\mathcal{A}_1$ is to learn the data of the Re. The goal of $\mathcal{A}_2$ is to obtain the data and associated values of the DOs. The goal of $\mathcal{A}_3$ is to obtain the private intersection-sum result. $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ possess the following capabilities:

(1) $\mathcal{A}_1$,$\mathcal{A}_2$, and $\mathcal{A}_3$ may eavesdrop on all communication links to obtain data owned by participants.
(2) $\mathcal{A}_1$ may compromise the CS to learn the data of the Re.
(3) $\mathcal{A}_2$ may compromise the CS to learn the DOs' data and associated values, or compromise one DO to learn the data and associated values of the other DOs.
(4) $\mathcal{A}_3$ may compromise the CS and DOs to obtain the intersection-sum result.

5. The Protocol Framework

In this section, we first provide some notations used in the protocol framework in Table 1, and we present the overview of our scheme in Figure 2. The details will be described in the following sections.

Table 1. Notations.

Notation	Description
$\|X\|$	The size of the data set X
x_i	The i-th data of X
$[m]$	The set $\{1, 2, \ldots, m\}$
$DO^{(u)}$	The u-th DO of the DOs
$Y^{(u)}$	The data set of the $DO^{(u)}$
$y_i^{(u)}$	The i-th data of $Y^{(u)}$
Enc	The asymmetric encryption algorithm
Dec	The asymmetric decryption algorithm
$MTgen$	Merkle tree generation algorithm
$F(\cdot)$	Pseudorandom function

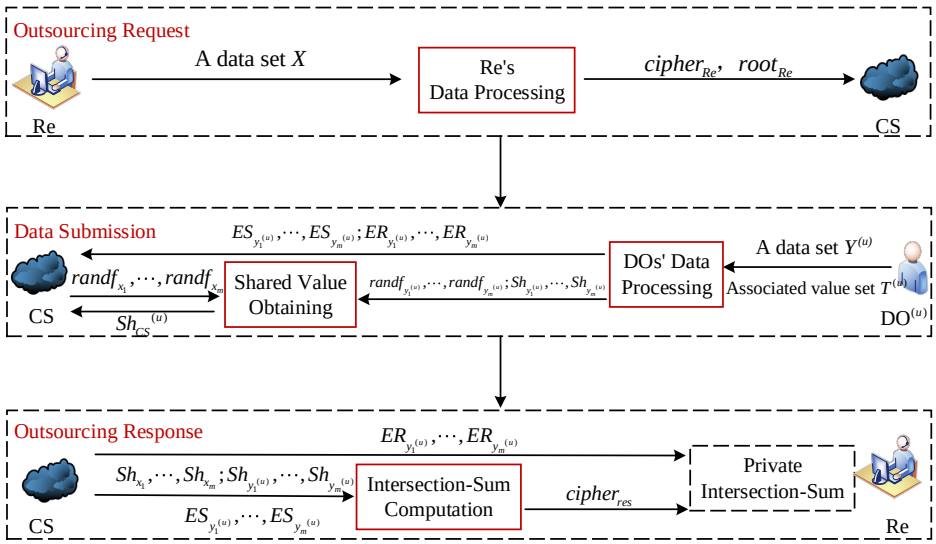

Figure 2. Scheme overview.

5.1. Initialization

In this phase, multiple parties (i.e., the Re and DOs) negotiate a symmetric key $k_{ij}|i \neq j$ with each other. The CS chooses a random private key a, $DO^{(u)}$ chooses $b^{(u)}$, and the Re chooses c. The CS negotiates the session key $k_{ab^{(u)}}$ with $DO^{(u)}$. The Re negotiates the session key k_{ac} with the CS. The Re and DOs share a group key K ($|K| = l$). In addition, several functions are selected: a pseudorandom function (PRF) $F(K, x) : \{0,1\}^l \times \{0,1\}^* \rightarrow \{0,1\}^{d_1}$, and hash functions $H : \{0,1\}^w \rightarrow \{0,1\}^{d_2}$.

5.2. Outsourcing Request

In this phase, the Re runs data processing on the input of data set $X(|X| = m)$ to generate $\{Enc_{Re}, root_{Re}\}$. The details are shown in Algorithm 1, and we divide this stage into several steps as follows:

(1) Generate the pseudorandom function value. The Re computes the pseudorandom function value $randf_{x_i}$ of the data x_i based on the group key K; namely, $randf_{x_i} = F(K, x_i)$.

(2) Generate the intersection sharing value. For each datum x_i, the Re first computes the pseudorandom function value F_j based on the symmetric key $k_{0j}(j \in [n])$; namely, $F_j = F(k_{0j}, x_i)$. Then, the Re XORs multiple F_j according to the formula (1) to generate the intersection sharing value Sh_{x_i}:

$$Sh_{x_i} = F_1 \oplus \cdots \oplus F_n \tag{1}$$

(3) Generate ciphertext and verification information for pseudorandom function values and intersection sharing values. The Re calculates the ciphertext $\{cipher_{Re_1}, \cdots, cipher_{Re_m}\}$ and the root $root_{Re}$ of the Merkle tree according to the formula (2) based on pseudorandom function values and intersection sharing values $\{randf_{x_1}||Sh_{x_1}, \cdots, randf_{x_m}||Sh_{x_m}\}$:

$$\begin{cases} cipher_{Re_i} = Enc(randf_{x_i}||Sh_{x_i}, k_{ac}), i \in [m] \\ root_{Re} = MTgen(randf_{x_1}||Sh_{x_1}, \cdots, randf_{x_m}||Sh_{x_m}) \end{cases} \tag{2}$$

Algorithm 1 Re's data processing

Input: A data set X with $|X| = m$
Output: $Enc_{Re}, root_{Re}$
for $i = 1$ to m **do**
  $randf_{x_i} \leftarrow F(K, x_i)$
  $Sh_{x_i} \leftarrow 0$
  for $j = 1$ to n **do**
    $F_j \leftarrow F(k_{0j}, x_i)$
    $Sh_{x_i} \oplus = F_j$
  end for
  $Enc_{Re_i} \leftarrow Enc(PRF_{x_i}||Sh_{x_i}, k_{ac})$
end for
$root_{Re} \leftarrow MTgen(PRF_{x_1}||Sh_{x_1}, \cdots, PRF_{x_m}||Sh_{x_m})$
$cipher_{Re} \leftarrow \{cipher_{Re_1}, \cdots, cipher_{Re_m}\}$
return $\{cipher_{Re}, root_{Re}\}$

5.3. Data Submission

In this phase, the CS receives the ciphertext and verification information of pseudorandom function values and intersection sharing values $\{cipher_{Re}, root_{Re}\}$ from Re.

Firstly, we introduce the validation of the pseudorandom function values and intersection sharing values. Assume that the CS decrypts Enc_{Re} to obtain $\left\{(randf_{x_1}||Sh_{x_1})', \ldots, (randf_{x_m}||Sh_{x_m})'\right\}$:

$$\left\{(randf_{x_i}||Sh_{x_i})'\right\} \leftarrow Dec(cipher_{Re_i}, k_{ac})(i \in [m]). \tag{3}$$

Then, the CS computes

$$root'_{Re} \leftarrow MTgen((randf_{x_1}||Sh_{x_1})', \ldots, (randf_{x_m}||Sh_{x_m})'). \tag{4}$$

If $root_{Re} = root'_{Re}$, the CS believes the verification passed and obtains $\{randf_{x_1}||Sh_{x_1}, \ldots, randf_{x_m}||Sh_{x_m}\}$.

Then, on the input of the data set $(Y^{(u)}, T^{(u)})(|Y^{(u)}| = |T^{(u)}| = m)$, $DO^{(u)}$ runs data processing to generate $\left\{ES^{(u)}, Sh^{(u)}, ER^{(u)}\right\}$. The details are shown in Algorithm 2. We take a DO as an example and divide this stage into several steps as follows:

Algorithm 2 DO's Data Processing

Input: A data set $(Y^{(u)}, T^{(u)})$ with $|Y^{(u)}| = |T^{(u)}| = m$
Output: $ES^{(u)}, Sh^{(u)}, ER^{(u)}$

for $i = 1$ to m **do**
  $r_i^{(u)} \xleftarrow{\$}$ a random number
  $randf_{y_i^{(u)}} \leftarrow F(K, y_i^{(u)})$
  $ES_{y_i^{(u)}} \leftarrow Enc(randf_{y_i^{(u)}}, t_i^{(u)} - r_i^{(u)})$
  $ER_{y_i^{(u)}} \leftarrow Enc(k_{0u}, r_i^{(u)})$
  $Sh_{y_i^{(u)}} \leftarrow 0$
  for $j = 0$ to n, $j \neq u$ **do**
    $F_j \leftarrow F(k_{uj}, y_i^{(u)})$
    $Sh_{y_i^{(u)}} \oplus = F_j$
  end for
end for
$ES^{(u)} \leftarrow \left\{ES_{y_1}^{(u)}, \cdots, ES_{y_m}^{(u)}\right\}$
$Sh^{(u)} \leftarrow \left\{Sh_{y_1}^{(u)}, \cdots, Sh_{y_m}^{(u)}\right\}$
$ER^{(u)} \leftarrow \left\{ES_{y_1}^{(u)}, \cdots, ER_{y_m}^{(u)}\right\}$
return $\left\{ES^{(u)}, Sh^{(u)}, ER^{(u)}\right\}$

(1) Generate calculation value. For each datum y_i, the DO first computes the pseudorandom function value $randf_{y_i}$ based on the group key K; namely, $randf_{y_i} = F(K, y_i)$. Then, the DO chooses a random number r_i for associated datum t_i and encrypts the difference between t_i and r_i based on $randf_{y_i}$ to generate the calculation value ES_{y_i}; namely, $ES_{y_i} = Enc(RPF_{y_i}, t_i - r_i)$.

(2) Generate intersection sharing value. For each datum y_i, the DO first computes the pseudorandom function value F_j based on the symmetric key negotiated with the Re and other DOs $k_{uj} (j \in [n], j \neq u)$; namely, $F_j = F(k_{uj}, y_i)$. Then, the DO XORs multiple F_j according to the formula (5) to generate the intersection sharing value Sh_{y_i}:

$$Sh_{y_i} = F_0 \oplus \cdots \oplus F_{u-1} \oplus F_{u+1} \cdots \oplus F_n \tag{5}$$

(3) Generate calculation sharing value. For each datum y_i, the DO chooses a random number r_i and generates calculation sharing value ES_{y_i} based on the symmetric key k_{0u} negotiated with the Re; namely, $ER_{y_i} = Enc(k_{0u}, r_i)$.

Next, we introduce the CS to obtain the intersection sharing value. CS obtains the pseudorandom function value of x_i by verifying the ciphertext and verification information sent by the Re. Each DO obtains the calculation value, the calculation sharing value, and the intersection sharing value through data processing. The CS and each DO obtain shared values, as shown in Algorithm 3. As a result, the CS obtains the shared value Sh_{CS}. We take a DO as an example and divide this stage into several steps as follows:

Algorithm 3 Obtaining Shared Values

Input: CS:$randf_{x_i}$ DO:$randf_{y_i}, ES_{y_i}||Sh_{y_i}||ER_{y_i}$ $(i \in [m])$
Output: Sh_{CS}

CS and DO execute:
  for $i = 1$ to m **do**
    CS sends $randf_{x_i}$ to F_{OPRF}
    CS receives $OF_{x_i} \leftarrow F(k_{OPRF}, randf_{x_i})$
  end for
  DO receives k_{OPRF}
  for $i = 1$ to m **do**
    DO computes $OF_{y_i} \leftarrow F(k_{OPRF}, randf_{y_i})$
  end for
DO executes:
  for $i = 1$ to m **do**
    $H_{DO_i} \leftarrow H(OF_{y_i})$
    $Th_{DO_i} \leftarrow OF_{y_i} \oplus Sh_{y_i}$
  end for
  $h_d \leftarrow \{0,1\}^d$
  if $h_d \notin \{H_{DO_1}, \cdots, H_{DO_m}\}$ **then**
    $Th_d \leftarrow$ a random number r
  end if
  $H \leftarrow \{H_{DO_1}, \cdots, H_{DO_m}, h_d\}$
  $cipher_H \leftarrow Enc(H, k_{ab})$
  $Th \leftarrow \{Th_{DO_1}, \cdots, Th_{DO_m}, Th_d\}$
  sends hash table $T = \{cipher_H, Th\}$ to CS
  $ES \leftarrow \{ES_{y_1}, \cdots, ES_{y_m}\}$
  $ER \leftarrow \{ES_{y_1}, \cdots, ER_{y_m}\}$
  sends $\{ES, ER\}$ to CS
CS executes:
  $H \leftarrow Dec(cipher_H, k_{ab})$
  for $i = 1$ to m **do**
    $H_{CS_i} \leftarrow H(OF_{x_i})$
    $Sh_{CS_i} \leftarrow Th_{CS_i} \oplus OF_{x_i}$
  end for
  $Sh_{CS} \leftarrow \{Sh_{CS_1}, \cdots, Sh_{CS_m}\}$
  return $\{Sh_{CS}\}$

(1) The CS and DO perform an oblivious pseudorandom function algorithm F_{OPRF}. First, the CS takes the pseudorandom function value $randf_{x_i}$.Then, as the output of F_{OPRF}, the DO receives the key k_{OPRF}, and the CS receives the value OF_{x_i} corresponding to $randf_{x_i}$; namely, $OF_{x_i} = F(k_{OPRF}, SE_{x_i})$. In particular, the CS can only obtain OF_{x_i} corresponding to $randf_{x_i}$, and cannot obtain the key k_{OPRF}.
(2) The DO generates an oblivious pseudorandom function value. Based on the k_{OPRF} output by F_{OPRF} and the pseudorandom function value $randf_{y_i}$, the DO generates the corresponding oblivious pseudorandom function value OF_{y_i}; namely, $OF_{y_i} = F(k_{OPRF}, SE_{y_i})$.
(3) The DO generates the hash table T. Firstly, the DO generates hash value H_{DO_i} based on the oblivious pseudorandom function value OF_{y_i}; namely, $H_{DO_i} = H(OF_{y_i})$. Then, Th_{DO_i} is generated based on OF_{y_i} and intersection sharing value Sh_{y_i}; namely, $Th_{DO_i} = OF_{y_i} \oplus Sh_{y_i}$. In addition, for the binary bit string h_d, but not in $\{H_{DO_1}, \cdots, H_{DO_m}\}$, the DO selects the random number r as h_d corresponding to Th_d. Finally, the DO encrypts H and generates $cipher_H$ based on the session key k_{ab}, where $H = \{H_{DO_1}, \cdots, H_{DO_m}, h_d\}$. The DO takes $cipher_H, Th$ as two columns to generate a hash table T and sends it to the CS.
(4) The CS obtains the intersection sharing value. Firstly, the CS obtains the hash table T based on $cipher_H$ and k_{ab}. Then, based on the oblivious pseudorandom function

value OF_{x_i} and T, the CS obtains the intersection sharing value Sh_{CS_i}; namely, $Sh_{CS_i} = Th_{CS_i} \oplus OF_{x_i}$. According to the steps of the DO generation hash table T, it can be proven that Sh_{CS_i} satisfies the formula (6):

$$Sh_{CS_i} = \begin{cases} Th_{CS} \oplus OF_x = OF_y \oplus Sh_y \oplus OF_x = Sh_y, & x = y \\ Th_{CS} \oplus OF_x = r \oplus OF_x, & x \neq y. \end{cases} \tag{6}$$

At this point, the data submission phase ends. The CS obtains the data sharing value $\left\{Sh_{CS}^{(u)}, ES^{(u)}, ER^{(u)}\right\}$ for each $DO^{(u)}$.

5.4. Outsourcing Response

In this phase, the CS first computes the private set intersection based on the intersection sharing values of the Re and DOs. Then, the private sum calculation result is computed based on the calculation value and calculation sharing value generated by the DO. The details of the CS computing the privacy sum are shown in the Algorithm 4. We take a DO as an example and divide this stage into several steps as follows:

(1) Compute private set intersection. The CS, according to formula (7), executes XOR based on the intersection sharing value Shx_i generated by the Re and $Sh_{CS_i}^{(u)}$, which is obtained by the interaction with each $DO^{(u)}$, and then generates res_{x_i}. If $res_{x_i} = 0$, then the datum x_i belong to the set intersection; namely, $x_i \in X \bigcap (\bigcap_{u=1}^{n} Y^{(u)})$:

$$res_{x_i} = Shx_i \oplus Sh_{CS_i}^{(1)} \oplus \cdots \oplus Sh_{CS_i}^{(n)} \tag{7}$$

(2) Compute private intersection-sum. The CS decrypts the calculation value ES_{x_i} based on pseudorandom function value $randf_{x_i}$; namely, $DS_{x_i} = Dec(randf_{x_i}, ES_{x_i})$. Then, the CS sums multiple calculation values to obtain sum.

(3) Generate the ciphertext of the private sum result. Based on the private intersection-sum result sum and the session key k_{ac} negotiated with the Re, the CS generates the ciphertext of the private sum result; namely, $cipher_{res} = Enc(sum, k_{ac})$. Then, the CS generates ER_{Re} based on calculated sharing values ER_{x_i} corresponding to the intersection data x_i and sends $\{cipher_{res}, ER_{Re}\}$ to the Re.

Algorithm 4 Private Sum Computation

Input: $Sh_{CS}^{(u)}$,Shx_i
Output: $cipher_{res}$

```
for i = 1 to m do
    sum ← 0
    ER_Re ← null
    res_{x_i} ← Shx_i
    for u = 1 to n do
        res_{x_i} ⊕= Sh_{CS_i}^{(u)}
    end for
    if res_{x_i} == 0 then
        DS_{x_i} ← Dec(randf_{x_i}, ES_{x_i}^{(u)})
        sum+ = DS_{x_i}
    end if
end for
cipher_res ← Enc(sum, k_ac)
return {cipher_res, ER_Re}
```

After receiving $\{cipher_{res}, ER_{Re}\}$ from the CS, the Re first decrypts each ER_{x_i} based on the group key K to obtain r_i and add all r_i named sum_1. Next, the Re decrypts $cipher_{res}$

based on the session key k_{ac} to obtain sum named sum_0. Finally, Re adds sum_0 and sum_1 to obtain the final result of privacy intersection-sum.

6. Security Analysis

In this section, we analyze the security of our proposed protocol from three points: the security of the Re's data, the security of the DOs' data, and the security of intersection-sum result.

6.1. Security of Re's Data

SLMP-PIS guarantees the security of the Re's data. In other words, no one except for the Re learns X.

In the outsourcing request phase, for each datum x_i, the Re generates $randf_{x_i}$ and intersection sharing value Sh_{x_i}. For $randf_{x_i}$, Re executes pseudorandom function for x_i based on the group key K, and obtains $randf_{x_i}$ corresponding to x_i. For Sh_{x_i}, Re first calculates F_u based on the symmetric key $k_{0u}(i \in [n])$ negotiated with $DO^{(u)}$. Then, the Re performs XOR on the n pseudorandom function values to obtain the data sharing value, i.e., $Sh_{x_i} = F_1 \oplus \cdots \oplus F_n$. If the adversary $\mathcal{A}_1$ wants to learn x_i, they need to obtain key K and k_{ou}. However, due to the security of PRF, the probability of an adversary's challenge being successful is negligible.

In the data submission phase, Re computes $cipher_{Re_i} = Enc(randf_{x_i}||Sh_{x_i}||r_i, k_{ac})$ based on the session key k_{ac} negotiated between the Re and CS and sends it to the CS. Therefore, the adversary $\mathcal{A}_1$ can only obtain the ciphertext of the pseudorandom function value and intersection sharing value of the Re's data. In the absence of k_{ac}, the confidentiality of symmetric encryption ensures that the probability that outsiders obtain x_i is negligible.

To sum up, SLMP-PIS guarantees the security of the Re's data based on the security of PRF and symmetric encryption.

6.2. Security of DOs' Data

SLMP-PIS guarantees the security of the DOs' data. On the one hand, it is difficult for the CS and Re to learn any of the DOs' data, including datum Y and the related datum T. On the other hand, it is hard for each DO to learn the data of other DOs.

In the data submission phase, the CS first performs OPRF with the DO. The CS obtains OF_{x_i} without k_{OPRF} with negligible probability. The DO calculates OF_{y_i} based on k_{OPRF}. Then, the DO generates a hash table based on OF_{y_i} and Sh_{y_i}. During this process, the DO calculates the hash value of OF_{y_i} as the first column of the hash table, and the XOR value of OF_{y_i} and Sh_{y_i} as the second column of the hash table. Finally, the CS obtains Sh_{y_i} or random numbers through the XOR operation based on the hash table. If the adversary $\mathcal{A}_2$ compromises CS's ability to learn y_i, it needs to break the one-way hash function and the security of OPRF, and this probability is negligible.

In the initialization phase, each $DO^{(u)}$ negotiates the session key $k_{ab^{(u)}}$ with the CS based on the Diffie–Hellman key exchange protocol. $DO^{(u)}$ encrypts the first column of the hash table based on $k_{ab^{(u)}}$. If the adversary $\mathcal{A}_2$ would like to learn the data through the first column of hash table, the key $k_{ab^{(u)}}$ of the DO needs to be obtained. However, the probability is negligible. In addition, each $DO^{(u)}$ negotiates a symmetric key $k_{iu}(i \in [0, n], i \neq u)$ with Re and other DOs, and calculates the $Sh_{y_i^{(u)}}$ based on $n-1$ symmetric keys. The possibility for any two DOs to obtain equal symmetric keys is negligible. Therefore, it is difficult for other DOs to learn the DO's data through the second column of the hash table.

In summary, SLMP-PIS guarantees the security of the DOs' data based on OPRF security, the one-way hash function, key security, and the discrete logarithm problem.

6.3. Security of Intersection-Sum Result

SLMP-PIS guarantees the security of the private intersection-sum result. In other words, no one except for the Re obtains the intersection-sum result.

In the outsourcing request phase, the Re generates a pseudorandom function value for each datum x_i. In the data submission phase, each DO generates a calculation value ES_{x_i} and a calculation sharing value ER_{x_i}. After running the obtaining shared values algorithm, the CS computes the intersection of $X \bigcap (\bigcap_{u=1}^{n} Y^{(u)})$. Then, the CS computes the sharing value of intersection-sum based on ES_{x_i}, and sends the ciphertext to the Re. In the outsourcing response phase, the Re generates another sharing value. Therefore, the adversary $\mathcal{A}_3$ compromises the CS to learn the intersection-sum result with negligible probability due to the arithmetic secret sharing.

In conclusion, SLMP-PIS guarantees the security of the intersection-sum result based on arithmetic secret sharing and symmetric encryption.

7. Performance Analysis

In this section, we introduce the experimental settings and results of the SLMP-PIS.

7.1. Experimental Settings

We set the Re's data set $\{x_i = i | i \in [m]\}$. We set the DO's data set Y as formula (8) and associated values T as $\{t_i = i | i \in [m]\}$. We set the AES arithmetic to AES-256 and the hash algorithm to MD5 by default.

$$y_i^{(u)} = \begin{cases} i, & 1 \leq i \leq m/2 \\ random - number, & m/2 < i \leq m. \end{cases} \tag{8}$$

In addition, we introduce the experimental arrangement. First, we study the computational cost in the outsourcing request phase, the data submission phase involving the DOs processing data and obtaining shared values, and the outsourcing response phase. Second, we also research the parameters and algorithms in our scheme. We study the effects of different AES algorithms and hash functions on the computational cost. Next, we compare the computational cost of our scheme in online and offline phases with different data sizes m. We set $m = 2^9$ and $m = 2^{11}$ as examples.

Finally, we compare our scheme with some related work, including Ion et al. [32], Lu et al. [31], and Kulshrestha et al. [35]. The comparison results are shown in Table 2. "$\surd$" means satisfied, and "$\times$" means dissatisfied. Ion et al. [32] proposed a strict two-party private intersection-sum protocol, Lu et al. [31] described a private intersection weighted sum protocol, and Kulshrestha et al. [35] proposed a non-strict private intersection-sum protocol to compute $\sum_{x \in I} T[x]$, where $I = X_0 \bigcap (\bigcup_{i=1}^{n-1} X_i)$ and T is associated with the values of X. Therefore, we compare our scheme with the computational cost of executing the model multiple times, as in Ion et al. [32], since they implement the same function.

Table 2. Comparison with related work.

Schemes	Ion [32]	Lu [31]	Kulshrestha [35]	*SLMP-PIS*
Data privacy	$\surd$	$\surd$	$\surd$	$\surd$
Strictly PIS	$\surd$	$\times$	$\times$	$\surd$
Party offline	$\times$	$\times$	$\times$	$\surd$
Data update	$\times$	$\times$	$\times$	$\surd$
Symmetric cryptosystem	$\times$	$\times$	$\times$	$\surd$

The experiments are implemented on a PC (CPU, Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz 1.80 GHz; RAM, 8.00 G; OS, Windows 10) with centos7 virtual machines using python 3.7.4. The code is shown in github: https://github.com/mokx1874/SLMP-PIS.git, accessed on 19 January 2023.

7.2. Experimental Results

7.2.1. Computational Costs in Different Phases

At this stage, we evaluate the effect of data set size m and the number of DOs on the computational cost in the outsourcing request phase, the data submission phase involving the DO processing data and obtaining shared values, and the outsourcing response phase. The computational cost of the SLMP-PIS is shown in Figure 3. We select m from 2^8 to 2^{12} and $|DOs|$ from 2 to 5.

In the outsourcing request phase, as shown in Figure 4, it is obvious that m and $|DOs|$ influence the computational cost. When setting $|DOs| = 5$, the computational cost is 0.63 s ($m = 2^8$) and 10.11 s ($m = 2^{12}$). When setting $m = 2^{10}$, the computational cost is 1.43 s ($|DOs| = 2$) and 2.47 s ($|DOs| = 5$). Therefore, the computational cost in the outsourcing request phase is positively correlated with m and $|DOs|$.

In the DOs' data processing in the data submission phase, as shown in Figure 5, it is obvious that m and $|DOs|$ influence the computation cost. When setting $|DOs| = 5$, the computational cost is 0.71 s ($m = 2^8$) and 11.35 s ($m = 2^{12}$). When setting $m = 2^{10}$, the computational cost is 1.76 s ($|DOs| = 2$) and 2.87 s ($|DOs| = 5$). Therefore, the computational cost of the DOs' data processing is a positive correlation with m and $|DOs|$. As shown in Figure 6, we evaluate the computational overhead of this stage under different m through multiple experiments. The computational costs are 0.79 s ($m = 2^8$), 1.37 s ($m = 2^9$), 2.35 s ($m = 2^{10}$), 3.51 s ($m = 2^{11}$), and 7.42 s ($m = 2^{12}$). Therefore, the computational cost of obtaining shared values is a positive correlation with m.

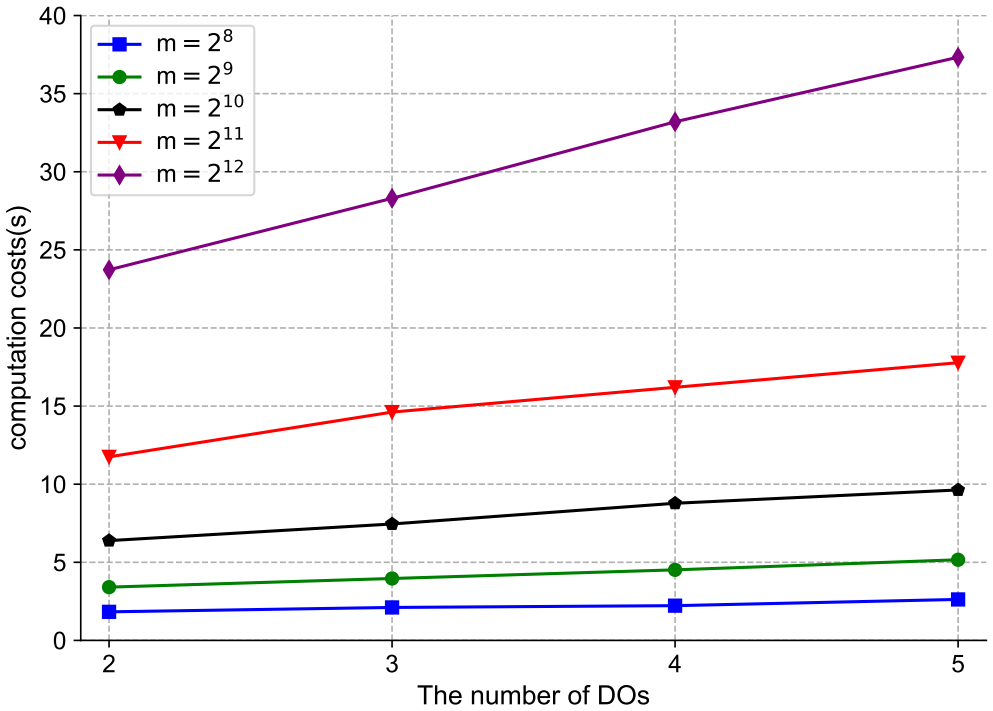

Figure 3. Computational cost of SLMP-PIS.

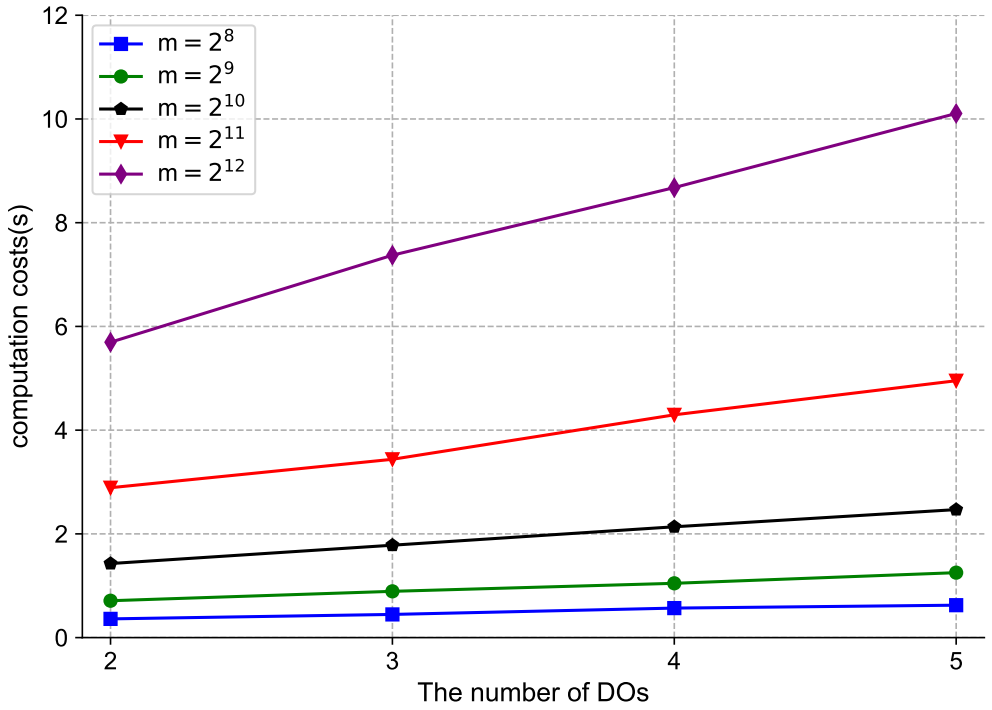

Figure 4. Computational cost of outsourcing requests.

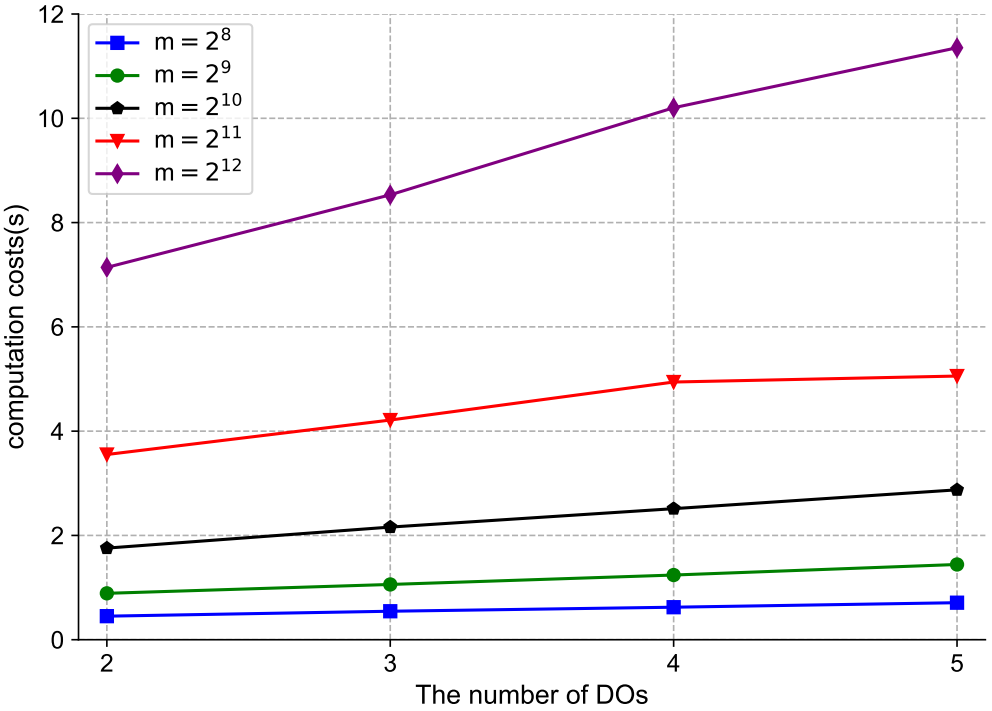

Figure 5. Computational cost of DOs' data processing.

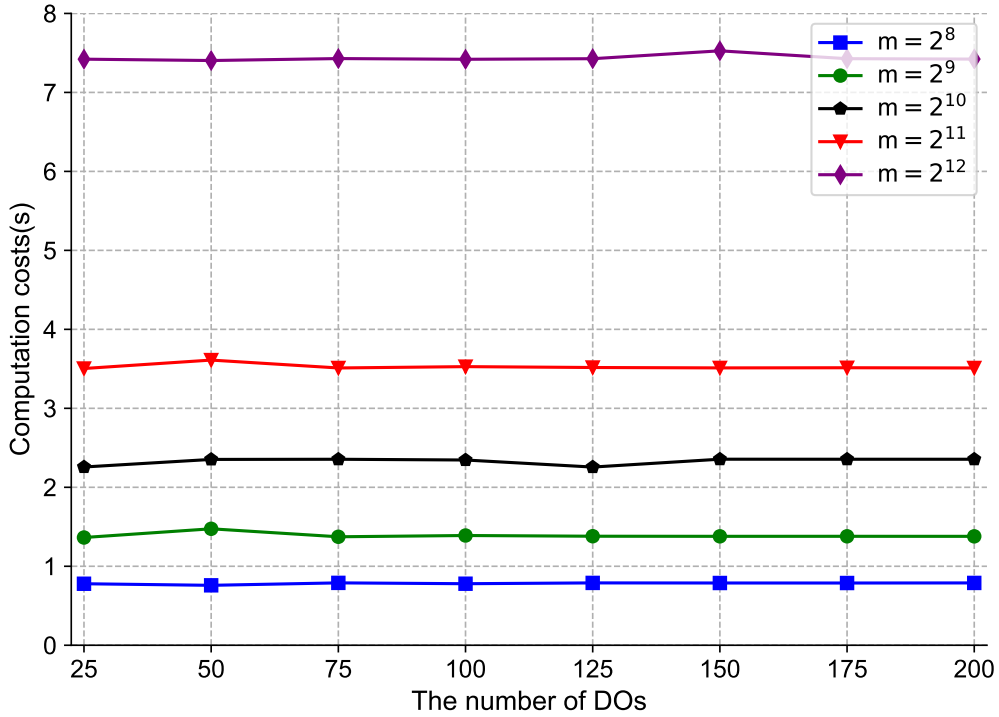

Figure 6. Computational cost of obtaining shared values.

In the outsourcing response phase, as shown in Figure 7, it is obvious that m and $|DOs|$ influence the computational cost. When setting $|DOs| = 5$, the computational cost is 0.56 s ($m = 2^8$) and 8.45 s ($m = 2^{12}$). When setting $m = 2^{10}$, the computational cost is 0.95 s ($|DOs| = 2$) and 2.03 s ($|DOs| = 5$). Therefore, the computational cost in the outsourcing response phase is positively correlated with m and $|DOs|$.

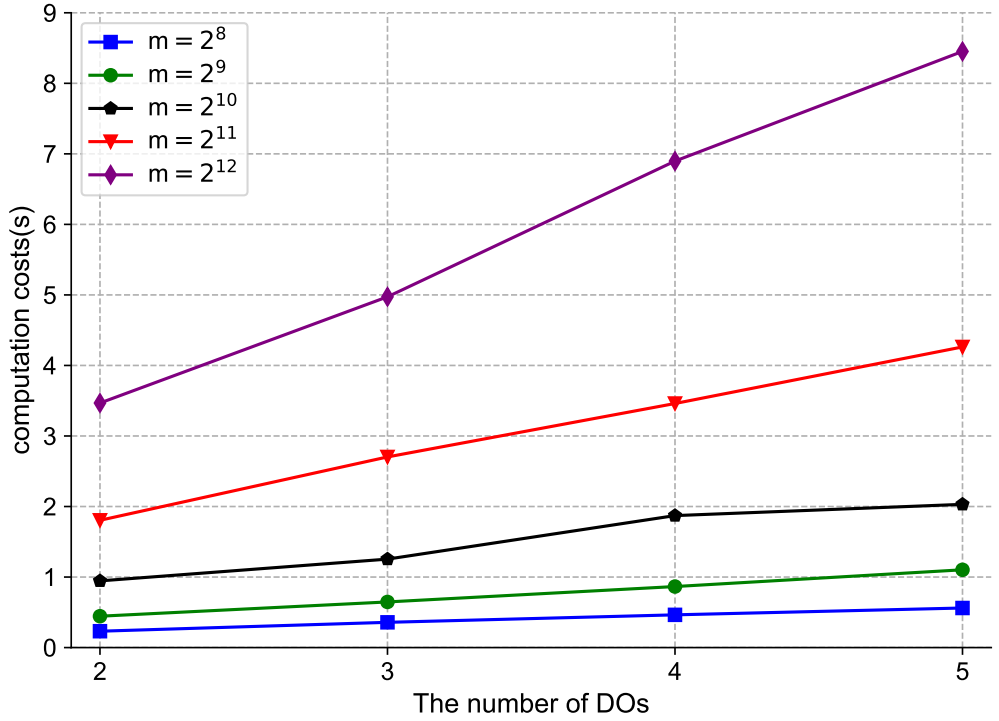

Figure 7. Computational cost of outsourcing responses.

As a result, the computational cost in different phases has a positive correlation with m and the number of DOs, and it is acceptable.

7.2.2. The Effect of *AES* and *Hash*

We set $m = 2^{10}$ in Figures 8 and 9. The results show that the impact of hash functions on computational cost is not obvious. When using MD5, the computational cost is 11.75 s, and it is 11.79 s using SHA-1, 11.83 s using SHA-256, and 11.87 s using SHA-512. The difference is negligible. For AES, we notice that AES algorithms have little impact on computation cost. When using AES-128, the computational cost is 11.72 s, 11.74 s using AES-192, and 11.81 s in using AES-256. The difference is negligible.

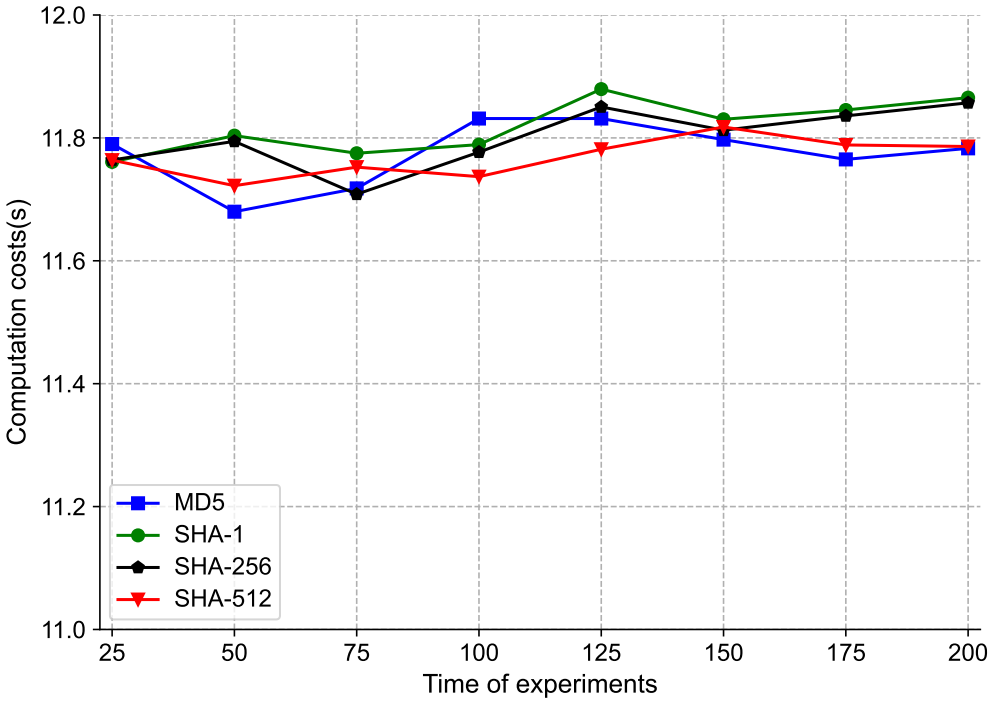

Figure 8. Computational cost with different hash functions.

As a result, the impact of different AES algorithms and hash functions on the computational cost is negligible.

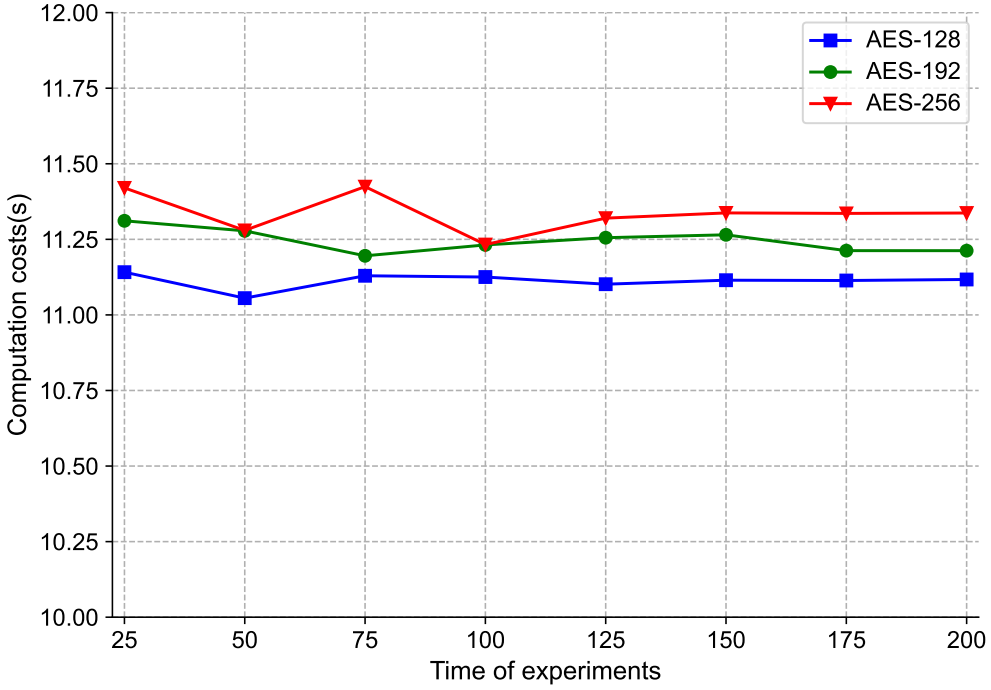

Figure 9. Computational cost with different AES algorithms.

7.2.3. Online and Offline

The current PIS basically needs to be executed online, which places more stringent requirements on the participants. Our solution enables parties to be offline after submitting their data. As shown in Figure 10, we compare the calculation cost of online and offline processes with different m. When setting $m = 2^9$ and $|DOs| = 5$, the computational cost is 3.21 s in online and 4.35 s in offline processes. When setting $m = 2^{11}$ and $|DOs| = 5$, the computational cost is 10.81 s for online and 23.17 s for offline.

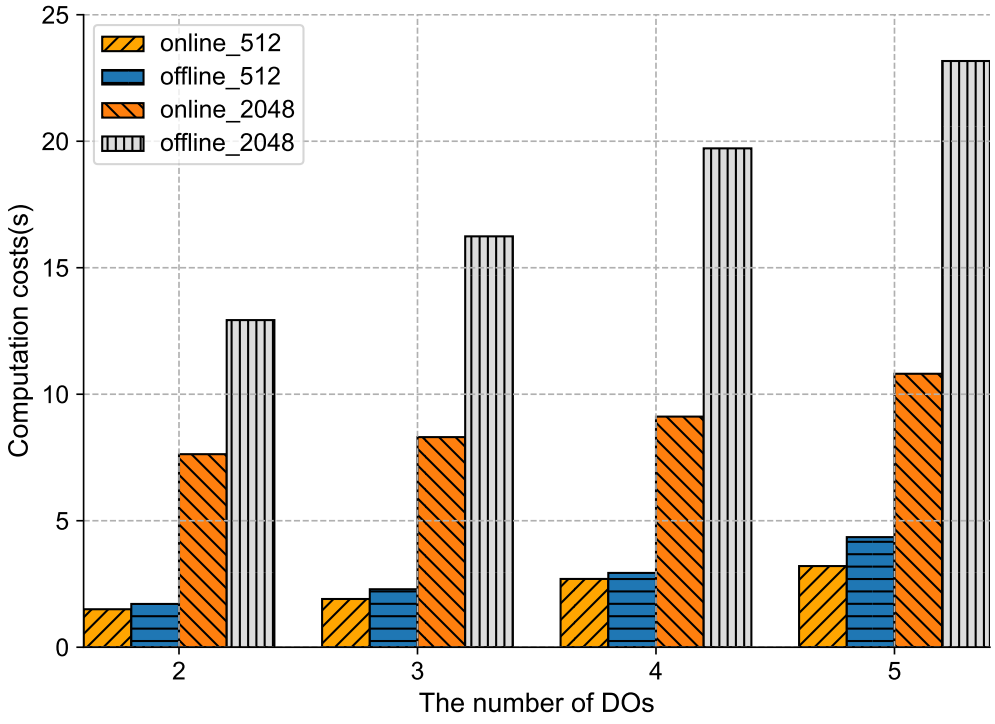

Figure 10. Computational cost of online and offline processes.

As a result, we find that the difference between the computational cost of offline processes and the computational cost of online processes gradually increases with the increase in m, and the computational cost of online processes is smaller than that of offline processes.

7.2.4. Comparison with Related Work

As shown in Figure 11, we compare the computational cost of our proposed scheme and Ion's [32] with different m. When setting $m = 2^9$ and $|DOs| = 2$, the computational cost is 3.80 s in our scheme and 3.99 s in Ion's. When setting $m = 2^9$ and $|DOs| = 5$, the computational cost is 7.56 s for our scheme and 11.29 s for Ion's. When setting $m = 2^{11}$ and $|DOs| = 2$, the computational cost is 18.98 s for our scheme and 19.06 s for Ion's. When setting $m = 2^{11}$ and $|DOs| = 5$, the computational cost is 33.97 s for our scheme and 44.11 s for Ion's.

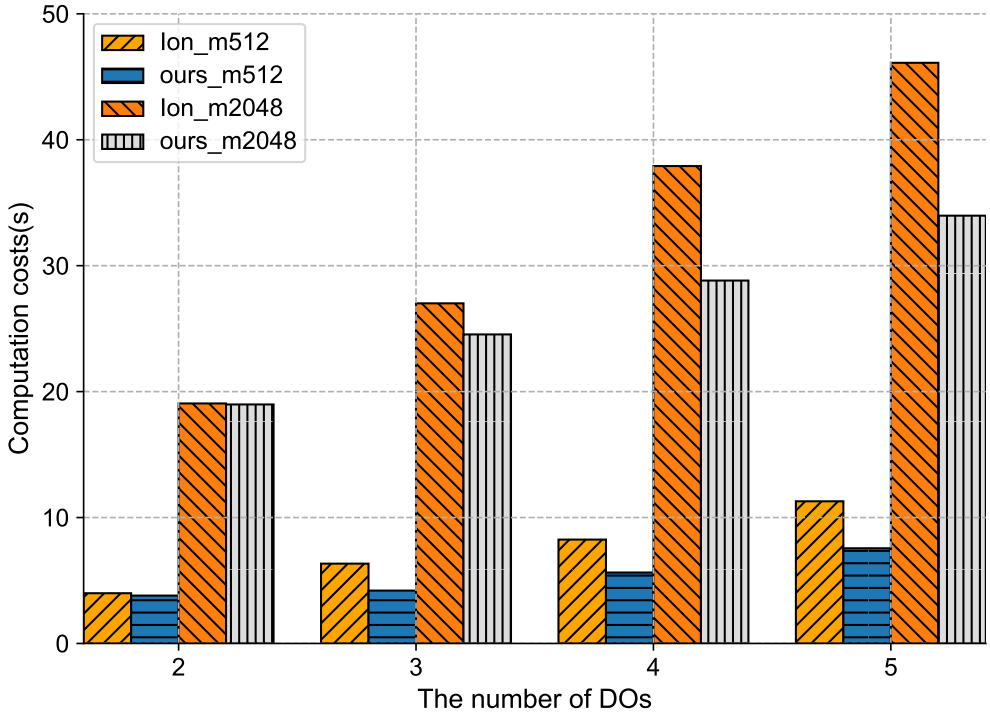

Figure 11. Computational cost of our scheme and *Ion*'s [32].

As a result, we find that the larger the number of DOs, the more efficient our scheme is.

8. Conclusions

In this paper, we demonstrate a secure and lightweight multi-party private intersection-sum scheme that computes the sum of associated values based on the private set intersection. On the one hand, the scheme achieves an outsourced multi-party private intersection-sum. On the other hand, our scheme allows participants to be offline and still update data. Specifically, we protect the privacy of outsourced data based on zero sharing and an oblivious pseudorandom function, and consider the privacy of associated values based on arithmetic sharing and symmetric encryption. We introduce the cloud computing technique into our scheme, which allows data owners to stay offline after outsourcing their processed data. Finally, the security analysis shows that the protocol is secure. The performance results illustrate the efficiency and feasibility of our scheme. Specifically, when the number of participants is five, the efficiency can be increased by 22.98%. As part of our future work, we will continue to study how to design a PIS protocol with malicious security.

Author Contributions: Conceptualization, X.K., Y.L., J.Z., H.M., T.L., Z.M. and S.G.; methodology, X.K., Z.M., J.Z., Y.L. and T.L.; software, X.K., J.Z., Y.L. and H.M.; validation, J.Z., Y.L. and H.M.; formal analysis, X.K. and Y.L.; investigation, J.Z., Y.L. and H.M.; resources, X.K., Y.L. and H.M.; writing—original draft preparation, X.K., Z.M., Y.L. and J.Z.; writing—review and editing, X.K., Z.M., J.Z., Y.L. and S.G.; visualization, X.K., Z.M. and Y.L.; supervision, X.K., Z.M., J.Z., Y.L. and S.G.; project administration, Z.M. J.Z. and H.M.; funding acquisition, Z.M., J.Z. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Key Research and Development Program of China (Grant No. 2021YFB3101100, 2022YFB3103500), the National Natural Science Foundation of China (U21A20464, 61872283), the Natural Science Basic Research Program of Shaanxi (2023-JC-JQ-49, 2022JZ-33, 2021JC-22), CNKLSTISS, the China 111 Project (No. B16037).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sun, L.; Gupta, R.K.; Sharma, A. Review and potential for artificial intelligence in healthcare. *Int. J. Syst. Assur. Eng. Manag.* **2022**, *13*, 54–62.
2. Boute, R.N.; Gijsbrechts, J.; Van Mieghem, J.A. Digital lean operations: Smart automation and artificial intelligence in financial services. In *Innovative Technology at the Interface of Finance and Operations*; Springer: Cham, Switzerland, 2022; pp. 175–188.
3. Ye, H.; Liu, J.; Wang, W.; Li, P.; Li, T.; Li, J. Secure and efficient outsourcing differential privacy data release scheme in cyber–physical system. *Future Gener. Comput. Syst.* **2020**, *108*, 1314–1323. [CrossRef]
4. Li, T.; Li, J.; Liu, Z.; Li, P.; Jia, C. Differentially private Naive Bayes learning over multiple data sources. *Inf. Sci.* **2018**, *444*, 89–104. [CrossRef]
5. Heidari, A.; Jabraeil Jamali, M.A. Internet of Things intrusion detection systems: A comprehensive review and future directions. *Clust. Comput.* **2022**. [CrossRef]
6. Heidari, A.; Navimipour, N.J.; Unal, M.; Zhang, G. Machine Learning Applications in Internet-of-Drones: Systematic Review, Recent Deployments, and Open Issues. *ACM Comput. Surv.* **2022**. [CrossRef]
7. Li, Y.; Jiang, Z.L.; Wang, X.; Fang, J.; Zhang, E.; Wang, X. Securely outsourcing ID3 decision tree in cloud computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 2385150. [CrossRef]
8. Xie, R.; He, C.; Xie, D.; Gao, C.; Zhang, X. A secure ciphertext retrieval scheme against insider kgas for mobile devices in cloud storage. *Secur. Commun. Netw.* **2018**, *2018*, 7254305.
9. Cai, Z.; Yan, H.; Li, P.; Huang, Z.A.; Gao, C. Towards secure and flexible EHR sharing in mobile health cloud under static assumptions. *Clust. Comput.* **2017**, *20*, 2415–2422. [CrossRef]
10. Zhu, Y.; Zhang, Y.; Li, X.; Yan, H.; Li, J. Improved collusion-resisting secure nearest neighbor query over encrypted data in cloud. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e4681. [CrossRef]
11. Althobaiti, O.S.; Mahmoodi, T.; Dohler, M. Intelligent Bio-Latticed Cryptography: A Quantum-Proof Efficient Proposal. *Symmetry* **2022**, *14*, 2351. [CrossRef]
12. Khan, N.U.; Shah, M.A.; Maple, C.; Ahmed, E.; Asghar, N. Traffic Flow Prediction: An Intelligent Scheme for Forecasting Traffic Flow Using Air Pollution Data in Smart Cities with Bagging Ensemble. *Sustainability* **2022**, *14*, 4164.
13. Makin, S.; Brack, C.; Kynn, M.; Murchie, P. 1013 DIAGNOSTIC TEST ACCURACY OF FRAILTY SCREENING TOOLS USING DATA IN ELECTRONIC PRIMARY CARE RECORDS. *Age Ageing* **2022**, *51*, 005.
14. Huang, H. Cryptosystems Based on Tropical Congruent Transformation of Symmetric Matrices. *Symmetry* **2022**, *14*, 2378. [CrossRef]
15. Almaiah, M.A.; Al-Zahrani, A.; Almomani, O.; Alhwaitat, A.K. Classification of cyber security threats on mobile devices and applications. In *Artificial Intelligence and Blockchain for Future Cybersecurity Applications*; Springer: Cham, Switzerland, 2021; pp. 107–123.
16. Gabr, M.; Younis, H.; Ibrahim, M.; Alajmy, S.; Khalid, I.; Azab, E.; Elias, R.; Alexan, W. Application of DNA Coding, the Lorenz Differential Equations and a Variation of the Logistic Map in a Multi-Stage Cryptosystem. *Symmetry* **2022**, *14*, 2559. [CrossRef]
17. Bahig, H.M.; Hazber, M.A.G.; Al-Utaibi, K.; Nassr, D.I.; Bahig, H.M. Efficient Sequential and Parallel Prime Sieve Algorithms. *Symmetry* **2022**, *14*, 2527. [CrossRef]
18. Chen, W.; Li, J.; Huang, Z.; Gao, C.; Yiu, S.; Jiang, Z.L. Lattice-based unidirectional infinite-use proxy re-signatures with private re-signature key. *J. Comput. Syst. Sci.* **2021**, *120*, 137–148. [CrossRef]
19. Wang, X.; Li, J.; Yan, H. An improved anti-quantum MST3 public key encryption scheme for remote sensing images. *Enterp. Inf. Syst.* **2021**, *15*, 530–544.

20. Yan, H.; Chen, M.; Hu, L.; Jia, C. Secure video retrieval using image query on an untrusted cloud. *Appl. Soft Comput.* **2020**, *97*, 106782.
21. Almaiah, M.A.; Dawahdeh, Z.; Almomani, O.; Alsaaidah, A.; Al-Khasawneh, A.; Khawatreh, S. A new hybrid text encryption approach over mobile ad hoc network. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 6461–6471. [CrossRef]
22. Yuan, H.; Chen, X.; Wang, J.; Yuan, J.; Yan, H.; Susilo, W. Blockchain-based public auditing and secure deduplication with fair arbitration. *Inf. Sci.* **2020**, *541*, 409–425.
23. Yu, J.; Xue, H.; Liu, B.; Wang, Y.; Zhu, S.; Ding, M. Gan-based differential private image privacy protection framework for the internet of multimedia things. *Sensors* **2020**, *21*, 58. [PubMed]
24. Xu, L.; Sun, Z.; Li, W.; Yan, H. Delegatable searchable encryption with specified keywords for EHR systems. In *Wireless Networks*; Spinger: New York, NY, USA, 2020; pp. 1–13.
25. Ali, A.; Almaiah, M.A.; Hajjej, F.; Pasha, M.F.; Fang, O.H.; Khan, R.; Teo, J.; Zakarya, M. An Industrial IoT-Based Blockchain-Enabled Secure Searchable Encryption Approach for Healthcare Systems Using Neural Network. *Sensors* **2022**, *22*, 572. [CrossRef] [PubMed]
26. Jing, Z.; Gu, C.; Yu, Z.; Shi, P.; Gao, C. Cryptanalysis of lattice-based key exchange on small integer solution problem and its improvement. *Clust. Comput.* **2019**, *22*, 1717–1727.
27. Li, J.; Tang, X.; Wei, Z.; Wang, Y.; Chen, W.; Tan, Y.A. Identity-based multi-recipient public key encryption scheme and its application in IoT. *Mob. Netw. Appl.* **2021**, *26*, 1543–1550.
28. Bahig, H.M.; Nassr, D.I.; Mahdi, M.A.; Bahig, H.M. Small Private Exponent Attacks on RSA Using Continued Fractions and Multicore Systems. *Symmetry* **2022**, *14*, 1897. [CrossRef]
29. Mahad, Z.; Ariffin, M.R.K.; Ghafar, A.H.A.; Salim, N.R. Cryptanalysis of RSA-Variant Cryptosystem Generated by Potential Rogue CA Methodology. *Symmetry* **2022**, *14*, 1498. [CrossRef]
30. Ion, M.; Kreuter, B.; Nergiz, E.; Patel, S.; Saxena, S.; Seth, K.; Shanahan, D.; Yung, M. Private intersection-sum protocol with applications to attributing aggregate ad conversions. *Cryptology ePrint Archive*. 2017. Available online: https://eprint.iacr.org/2017/738 (accessed on 1 August 2017).
31. Lu, S.; Li, Z.; Miao, X.; Han, Q.; Zheng, J. PIWS: Private Intersection Weighted Sum Protocol for Privacy-Preserving Score-Based Voting With Perfect Ballot Secrecy. *IEEE Trans. Comput. Soc. Syst.* **2022** , *31*, 1–18. [CrossRef]
32. Ion, M.; Kreuter, B.; Nergiz, A.E.; Patel, S.; Saxena, S.; Seth, K.; Raykova, M.; Shanahan, D.; Yung, M. On deploying secure computing: Private intersection-sum-with-cardinality. In Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P), Genoa, Italy, 7–11 September 2020; pp. 370–389.
33. Miao, P.; Patel, S.; Raykova, M.; Seth, K.; Yung, M. Two-Sided Malicious Security for Private Intersection-Sum with Cardinality. In *Proceedings of the Advances in Cryptology—CRYPTO 2020*; Micciancio, D., Ristenpart, T., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–33.
34. Niu, Z.; Wang, H.; Li, Z.; Song, X. Privacy-preserving statistical computing protocols for private set intersection. *Int. J. Intell. Syst.* **2021**. [CrossRef]
35. Kulshrestha, A.; Mayer, J. Estimating Incidental Collection in Foreign Intelligence Surveillance: Large-Scale Multiparty Private Set Intersection with Union and Sum. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*; USENIX Association: Boston, MA, USA, 2022; pp. 1705–1722.
36. Döttling, N.; Garg, S.; Hajiabadi, M.; Masny, D.; Wichs, D. Two-round oblivious transfer from CDH or LPN. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Cham, Switzerland, 2020; pp. 768–797.
37. Wang, X.; Kuang, X.; Li, J.; Li, J.; Chen, X.; Liu, Z. Oblivious transfer for privacy-preserving in VANET's feature matching. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 4359–4366. [CrossRef]
38. Freedman, M.J.; Ishai, Y.; Pinkas, B.; Reingold, O. Keyword Search and Oblivious Pseudorandom Functions. In *Proceedings of the Theory of Cryptography*; Kilian, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 303–324.