Efficiency improvement techniques for private intersection-sum protocol using Bloom filter

Hiroyuki Kano¹ · Keisuke Hakuta²

Received: 12 March 2021 / Accepted: 2 December 2021

Published online: 04 January 2022

© The Author(s) 2021 **OPEN**

Abstract

A private set intersection protocol is one of the secure multi-party computation protocols, and allows participants to compute the intersection of their sets without revealing them to each other. Ion et al. proposed the private intersection-sum protocol (PI-Sum). The PI-Sum is one of the two-party private set intersection protocol. In the PI-Sum, two parties (say Alice and Bob) have the private sets A and B . Moreover, Bob additionally has a rational integer associated with each element of B . The PI-Sum allows Bob to obtain the sum of the rational integers associated with the elements of $A \cap B$. This paper proposes the efficiency improvement techniques for the PI-Sum. The proposed techniques are based on Bloom filters which are probabilistic data structures. More precisely, this paper proposes three protocols which are modifications of the PI-Sum. The proposed protocols are more efficient than the PI-Sum.

Keywords Private set intersection (PSI) · Private intersection-sum · Homomorphic encryption · Bloom filter · Secure multi-party computation (SMC)

Mathematics Subject Classification 94A60 · 68P99

1 Introduction

Secure multi-party computation (SMC for short) protocols allow participants to compute a public function jointly while keeping their inputs private. A Private Set Intersection (PSI for short) protocol is one of the SMC protocols, and allows participants to compute the intersection of their sets without revealing them to each other. Since PSI protocols have many real-world applications (see for example [18, Sect. 2.6]), PSI protocols have extensively been studied in the last decade.

The first PSI protocol was proposed by Freedman et al. in [12] based on polynomial interpolation and an additive homomorphic encryption scheme. Later, Kissner et al. in

[17] generalized the first PSI protocol in [12] to the multi-party setting. Le et al. in [19] proposed a two-party PSI protocol with an untrusted third party. There exists several techniques to construct PSI protocols such as public-key techniques [7, 12, 13, 17, 22], Map-To-Prime technique [16], oblivious transfer technique [25], and so on.

1.1 Related work

Ion et al. in [13] proposed the Private Intersection-Sum Protocol (PI-Sum for short). The PI-Sum is one of the two-party PSI. In the PI-Sum, two parties (say Alice and Bob) have the private sets A and B . Moreover, Bob additionally has a rational integer associated with each element of B .

✉ Keisuke Hakuta, hakuta@cis.shimane-u.ac.jp; Hiroyuki Kano, hiroyuki.kanou.m@gmail.com | ¹Graduate School of Natural Science and Technology, Shimane University, 1060 Nishikawatsu-cho, Matsue, Shimane 690-8504, Japan. ²Institute of Science and Engineering, Academic Assembly, Shimane University, 1060 Nishikawatsu-cho, Matsue, Shimane 690-8504, Japan.

The PI-Sum allows Bob to obtain the sum of the rational integers associated with the elements of $A \cap B$. The PI-Sum ([13, Figure 2] or Algorithm 1) is constructed by Precomputation Algorithm (Algorithm 2), Intersection Set Algorithm (Algorithm 3), and Intersection Sum Algorithm (Algorithm 4). Precomputation Algorithm (Algorithm 2) encrypts all the elements in A and B , respectively. Intersection Set Algorithm (Algorithm 3) computes $A \cap B$ without revealing the private sets A and B to each other. Intersection Sum Algorithm (Algorithm 4) computes the sum of the rational integers associated with the elements of $A \cap B$ without revealing the private sets A and B to each other. The PI-Sum is based on Diffie-Hellman key exchange [9]. The PI-Sum requires to compute the cardinality $\#(A \cap B)$ of the intersection set $A \cap B$ of input sets A and B for obtaining the output. By DDH (Decisional Diffie-Hellman) assumption, the security of the PI-Sum in the honest-but-curious model was proven by Ion et al. in [13]. Participants in honest-but-curious model follow the steps in cryptographic protocol honestly, however the participants try to obtain the private information of other participant. For the input sets A and B , the PI-Sum requires to compute $2(\#A + \#B)$ exponentiations for obtaining the output. In computing an online ad revenue, since the input set A is the set of the user who saw online ad, one may assume that the cardinality of A and B to be several hundreds of thousands to million. Therefore the PI-Sum needs to improve the efficiency for computing an online ad revenue. Other PI-Sum protocols can be found in [14, 21].

1.2 Our contribution

This paper proposes the efficiency improvement techniques for the PI-Sum ([13, Figure 2]). We have two interesting contributions as follows:

- We propose a new PI-Sum protocol, say the proposed protocol 1 (Algorithm 7). The proposed protocol 1 is constructed by Precomputation Algorithm (Algorithm 2), the proposed algorithm for computing the private set intersection (Algorithm 8), and Intersection Sum Algorithm (Algorithm 4). Thus, the only difference between the PI-Sum ([13, Figure 2] or Algorithm 1) and the proposed protocol 1 (Algorithm 7) is the phase of the computation of private set intersection (Algorithm 3 and Algorithm 8). The proposed protocol 1 uses a Bloom filter for computing the cardinality of the intersection of input sets A and B . Since Bloom filters are probabilistic data structures, the computational cost of Algorithm 8 is smaller than that of Algorithm 3. Thus, the proposed protocol 1 is more efficient than the PI-Sum (See Eqs. (9 and 10)). We assume that $\#A = \#B$, and let us define $T := \#A (= \#B)$. Then the computational

cost of the proposed protocol 1 is about 1% smaller than that of the PI-Sum if $T = 2^{16}$, and the computational cost of the proposed protocol 1 is about 50% smaller than that of the PI-Sum if $T = 2^{22}$.

- This paper also proposes the proposed protocol 2 (Algorithm 10) and the proposed protocol 3 (Algorithm 11) for preventing the security problem described in Sect. 5. By adding an aborting functionality in the proposed protocol 2 and the proposed protocol 3, we can avoid the security problem described in Sect. 5. The computational costs of the proposed protocol 2 and the proposed protocol 3 are also discussed (See Eqs. (14 and 15)).

The rest of this paper is organized as follows: In Sect. 2, we describe the notations in this paper, and we review the PI-Sum [13] and a Bloom filter [4]. Section 3 proposes the proposed protocol 1 (Algorithm 7) that applying a Bloom filter in order to improve the efficiency of the computation about the intersection $A \cap B$, and show the correctness of the proposed protocol 1. In Sect. 4, we estimate the computational costs of the PI-Sum and the proposed protocol 1, and compare them. We show that the proposed protocol 1 is more efficient than the PI-Sum [13, Figure 2]. In Sect. 5, we propose the proposed protocol 2 and the proposed protocol 3 for preventing the above security problem. In Sect. 6, we discuss the efficiency of the proposed protocols. Section 7 concludes the paper.

2 Mathematical preliminaries

In this section we describe the notations that will be used in this paper. For more details, we refer the reader to [20].

For a finite set X , we denote by $\#X$ the cardinality of X . We use the symbol $\text{Inj}(X, Y)$ to represent the set of all the injective maps from a set X to a set Y , and use the symbol $\text{Bij}(X)$ to represent the set of all the bijective maps on X . Let $\mathbb{Z}$ be the rational integer ring. We denote by $\mathbb{N}_+$ the set $\{1, 2, \dots\}$. For a positive integer $m \geq 2$, we set $\mathbb{Z}_m := \{0, 1, \dots, m-1\}$ (the residue class ring modulo m). For a prime p , $\mathbb{F}_p := \{0, 1, \dots, p-1\}$ denotes the finite field with p elements. We define the map $\text{int} : \{0, 1\}^{256} \rightarrow \mathbb{Z}_{256}$, $(x_1, \dots, x_{256}) \mapsto \sum_{i=0}^{255} x_{256-i} 2^i$, where int is the bijective map from $\{0, 1\}^{256}$ to $\mathbb{Z}_{256}$. $\text{Set}[k] := \{1, \dots, k\}$ for each $k \in \mathbb{N}_+$. For $n \in \mathbb{N}_+$, $\{0, 1\}^n$ denotes the set of all the n -bit strings when $n \geq 1$, and $\{0, 1\}^n$ denotes the empty set $\emptyset$ when $n = 0$. In addition, we define $\{0, 1\}^* := \bigcup_{n \geq 0} \{0, 1\}^n$ as the set of all the finite bit strings. We let “ $\perp$ ” denote an abort symbol.

Let us denote by $(\mathcal{X}, P_{\mathcal{X}})$ the probability space, where $\mathcal{X}$ is a finite set and $P_{\mathcal{X}}$ is the map from $\mathcal{X}$ to the real closed interval $[0, 1]$. For each $x \in \mathcal{X}$, we have $0 \leq P_{\mathcal{X}}(x) \leq 1$ and $P_{\mathcal{X}}(\mathcal{X}) = 1$.

A subset $\mathcal{A} \subseteq \mathcal{X}$ is said to be an event, and we denote by $\Pr[\mathcal{A}] := \sum_{x \in \mathcal{A}} P_{\mathcal{X}}(x)$ the probability of the event $\mathcal{A}$.

For a public key encryption scheme, let us denote the Alice's public key and the Alice's secret key (resp. Bob's public key and the Bob's secret key) by pk_A and sk_A (resp. pk_B and sk_B), respectively. Let λ be a security parameter. We use the symbol $\text{HomEnc}_{pk_B}(m)$ to represent the ciphertext of a plaintext m using Bob's public key pk_B . We assume that an Abelian group $\mathbb{G}$ is parameterized by the security parameter λ .

2.1 Private intersection-sum Protocol

Before the description of the PI-Sum, we need some additional notations. Alice and Bob share the security parameter λ . The set of user identifiers held by Alice is $A = \{u_i\}_{i \in [I]}$, and the set of all the pairs of user identifier and an integer associated with each user identifier is $B = \{(v_j, t_j)\}_{j \in [J]}$. In this situation, $u_i \in [I] \in U$, $v_j \in [J] \in U$, and $t_j \in [J] \in \mathbb{N}_+$ hold, where U is the set of user identifiers. The private exponent held by Alice (resp. Bob) is k_1 (resp. k_2). In the PI-Sum, the Paillier encryption scheme [24] is used for an additive homomorphic encryption scheme. Bob generates the pair of the public key pk_B and the secret key sk_B using the key generation algorithm of the Paillier

encryption scheme. Bob shares the public key pk_B with Alice, and privately holds the secret key sk_B .

The PI-Sum uses the random oracle $\text{RO} : U \rightarrow \mathbb{G}$ from the set of user identifiers U to the Abelian group $\mathbb{G}$. We will describe the details of the Abelian group $\mathbb{G}$ in Sect. 4. The map from the set of user identifiers U to the Abelian group $\mathbb{G}$ is composed with the cryptographic hash function SHA-256 and the Montgomery curve Curve25519 [1]. By [1, Theorem 2.1], we can regard each element of $\mathbb{F}_p$ as the x -coordinate of the corresponding point on Curve25519. For each $x \in \{0, 1\}^{256}$, we can treat $\text{int}(x) \in \mathbb{F}_p$ as the x -coordinate of a point on Curve25519. Therefore, we can represent each 256 bit string as the point on Curve25519. We compute the hash value of each user identifier using SHA-256, and represent each 256 bit string as the point on Curve25519 via the map int . From the above discussion, we can compose the map from the user identifier space U to the Abelian group $\mathbb{G}$. In [13], it is assumed that the map from the user identifier space U to the Abelian group $\mathbb{G}$ is the random oracle for the security of the PI-Sum in honest-but-curious model [13]. For this reason, we also assume that the map from the user identifier space U to the Abelian group $\mathbb{G}$ is the random oracle $\text{RO} : U \rightarrow \mathbb{G}$.

We describe the PI-Sum [13] in Algorithm 1.

Algorithm 1 Private Intersection-Sum Protocol [13, Figure 2]

Alice's input: $A = \{u_i\}_{i \in [I]}, k_1, pk_B$.

Bob's input: $B = \{(v_j, t_j)\}_{j \in [J]}, k_2, pk_B, sk_B$.

Alice's output:

$$\mathcal{J} = \left\{ \phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \left\{ \text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2} \right\}_{i \in [I]} \right\}.$$

Bob's output: the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.

- 1: Alice and Bob perform Precomputation Algorithm (Algorithm 2):
 Alice inputs $A = \{u_i\}_{i \in [I]}, k_1$ and Bob inputs $B = \{(v_j, t_j)\}_{j \in [J]}, k_2$.
 Alice obtains $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$
 and $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$.
 - 2: Alice and Bob perform Intersection Set Algorithm (Algorithm 3):
 Alice inputs $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}, \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$.
 Alice obtains $\mathcal{J}$.
 - 3: Alice and Bob perform Intersection Set Algorithm (Algorithm 4):
 Alice inputs $\{\text{HomEnc}_{pk_B}(t_{\phi_3(j)})\}_{j \in [J]}, \mathcal{J}$ and Bob inputs sk_B .
 Bob obtains the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.
 - 4: Bob outputs the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.
-

One can see that the PI-Sum ([13, Figure 2] or Algorithm 1) is constructed by Precomputation Algorithm (Algorithm 2), Intersection Set Algorithm (Algorithm 3), and Intersection Sum Algorithm (Algorithm 4). Next, we explain Precomputation Algorithm (Algorithm 2), Intersection Set Algorithm (Algorithm 3), and Intersection Sum Algorithm (Algorithm 4), respectively.

We describe Precomputation Algorithm in Algorithm 2. Precomputation Algorithm encrypts all the elements in A and B , respectively. Precomputation Algorithm outputs $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ and $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$ from Alice's input set $A = \{u_i\}_{i \in [I]}$ and Bob's input set $B = \{(v_j, t_j)\}_{j \in [J]}$.

Algorithm 2 Precomputation Algorithm**Alice's input:** $A = \{u_i\}_{i \in [I]}$, the private exponent k_1 .**Bob's input:** $B = \{(v_j, t_j)\}_{j \in [J]}$, the private exponent k_2 .**Alice's output:** $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}, \{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}.$ **Bob's output:** N/A.

```

1: for  $i$  from 1 to  $I$  by +1 do
2:   Alice computes  $\text{RO}(u_i)^{k_1}$  and stores  $\text{RO}(u_i)^{k_1}$ .
3: end for
4: Alice generates  $\phi_1 \in \text{Bij}([I])$ .
5: Alice sends  $\{\text{RO}(u_{\phi_1(i)})^{k_1}\}_{i \in [I]}$  to Bob.
6: for  $i$  from 1 to  $I$  by +1 do
7:   Bob computes  $\text{RO}(u_{\phi_1(i)})^{k_1 k_2}$  and stores  $\text{RO}(u_{\phi_1(i)})^{k_1 k_2}$ .
8: end for
9: Bob generates  $\phi_2 \in \text{Bij}([I])$ .
10: Bob generates  $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$  from  $\{\text{RO}(u_{\phi_1(i)})^{k_1 k_2}\}_{i \in [I]}$ .
11: Bob sends  $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$  to Alice.
12: for  $j$  from 1 to  $J$  by +1 do
13:   Bob computes  $\text{RO}(v_j)^{k_2}$  and stores  $\text{RO}(v_j)^{k_2}$ .
14:   Bob encrypts  $t_j$  and obtains  $\text{HomEnc}_{pk_B}(t_j)$ .
15: end for
16: Bob generates  $\phi_3 \in \text{Bij}([J])$ .
17: Bob sends  $\{(\text{RO}(v_{\phi_3(j)})^{k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$  to Alice.
18: for  $j$  from 1 to  $J$  by +1 do
19:   Alice computes  $\text{RO}(v_{\phi_3(j)})^{k_1 k_2}$  and stores  $\text{RO}(v_{\phi_3(j)})^{k_1 k_2}$ .
20: end for
21: Alice outputs  $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ 
    and  $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}.$ 

```

We describe Intersection Set Algorithm in Algorithm 3. Intersection Set Algorithm computes $A \cap B$ without revealing the private sets A and B to each other. Intersection Set Algorithm outputs $\mathcal{J}$ from Alice's input $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}, \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$. The set $\mathcal{J}$ defined by the following equation:

$$\mathcal{J} = \left\{ \phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \left\{ \text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2} \right\}_{i \in [I]} \right\}.$$

By doing so, Alice obtains $\mathcal{J}$.

Algorithm 3 Intersection Set Algorithm**Alice's input:** $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}, \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}.$ **Bob's input:** N/A.**Alice's output:** $\mathcal{J} = \{\phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}\}.$ **Bob's output:** N/A.

```

1: Alice initializes  $\mathcal{J}$  with  $\emptyset$ .
2: for  $j$  from 1 to  $J$  by +1 do
3:   if  $\text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$  then
4:     Alice updates  $\mathcal{J}$  with  $\mathcal{J} \cup \{\phi_3(j)\}$ .
5:   end if
6: end for
7: Alice outputs  $\mathcal{J}$ .

```

We describe Intersection Sum Algorithm in Algorithm 4. Intersection Sum Algorithm computes the sum of the rational integers associated with the elements of $A \cap B$ without revealing the private sets A and B to each other.

Intersection Sum Algorithm outputs the intersection sum $\sum_{t_{\phi_3(j)} \in \mathcal{J}} t_{\phi_3(j)}$ from Alice's input $\{\text{HomEnc}_{pk_B}(t_{\phi_3(j)})\}_{j \in [J]}$, intersection set $\mathcal{J}$ and Bob's input sk_B .

Algorithm 4 Intersection Sum Algorithm**Alice's input:** $\{\text{HomEnc}_{pk_B}(t_{\phi_3(j)})\}_{j \in [J]}$, intersection set $\mathcal{J}$.**Bob's input:** sk_B .**Alice's output:** N/A.**Bob's output:** the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.

```

1: Alice initializes Sum with 0.
2: Alice encrypts Sum and obtains  $\text{HomEnc}_{pk_B}(\text{Sum})$ .
3: for  $j$  from 1 to  $J$  by +1 do
4:   if  $\phi_3(j) \in \mathcal{J}$  then
5:     Alice computes
        $\text{HomEnc}_{pk_B}(\text{Sum} + t_{\phi_3(j)}) = \text{HomEnc}_{pk_B}(\text{Sum}) \cdot \text{HomEnc}_{pk_B}(t_{\phi_3(j)})$ 
       and stores  $\text{HomEnc}_{pk_B}(\text{Sum} + t_{\phi_3(j)})$ .
6:   Alice updates  $\text{HomEnc}_{pk_B}(\text{Sum})$  with  $\text{HomEnc}_{pk_B}(\text{Sum} + t_{\phi_3(j)})$ .
7:   end if
8: end for
9: Alice sends  $\text{HomEnc}_{pk_B}(\text{Sum})$  to Bob.
10: Bob decrypts  $\text{HomEnc}_{pk_B}(\text{Sum})$  and obtains  $\text{Sum} = \sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$ .
11: Bob outputs  $\text{Sum} = \sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$ .

```

The next assumption (Assumption 1) and the additive homomorphic property of the Paillier encryption scheme is needed in steps 3–8 of Algorithm 4. More precisely, Assumption 1 guarantees that the sum (not modular addition) of the rational integers associated with the elements of $A \cap B$ is not exceed the modulus N .

Assumption 1 Set $\mathcal{M} := \{m \in \mathbb{Z}_N \mid m \ll N\} \subset \mathbb{Z}_N$. We assume that $t_{j \in [J]} \in \mathcal{M}$ for each $j \in [J]$, and we also assume the following inequality condition:

$$\sum_{m \in \mathcal{M}} m < N.$$

Remark that by Assumption 1, we always have $\text{Sum} < N$ in step 6 of Algorithm 4. By doing so, Bob obtains the intersection sum $\sum_{t_{\phi_3(j) \in \mathcal{J}}} t_{\phi_3(j)}$. Bob outputs the intersection sum $\sum_{t_{\phi_3(j) \in \mathcal{J}}} t_{\phi_3(j)}$ in step 4 of the PI-Sum.

Thus, by the PI-Sum (Algorithm 1), Alice obtains $\mathcal{J}$ (the set of all indices of the elements $v_{\phi_3(j)} \in A \cap B$), and

Bob obtains the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$, where $\phi_3 \in \text{Bij}([J])$ is generated in step 16 of Algorithm 2.

2.2 Bloom filter

A Bloom filter is the probabilistic data structure using the k hash functions $H_{\mu \in [k]} : \{0, 1\}^* \rightarrow \mathbb{Z}_l$. If there exists $x \in \{0, 1\}^*$ such that $H_{\mu}(x) \neq H_{\mu'}(x)$ for all $\mu, \mu' \in [k]$ with $\mu \neq \mu'$, then we call the k hash functions *independent*. We use the k independent hash functions $H_{\mu \in [k]} : \{0, 1\}^* \rightarrow \mathbb{Z}_l$ for a Bloom filter. We use the k appropriate cryptographic hash functions for a Bloom filter. For an element $x \in X$, we can check whether $x \in Y$ or not using a Bloom filter. A Bloom filter is constructed using Algorithm 5 and Algorithm 6. Algorithm 5 stores a set Y in a l boolean array $\text{BF}[l]$, and Algorithm 6 checks whether $x \in Y$ or not. We use [22, Algorithm 5] (resp. [22, Algorithm 6]) for Algorithm 5 (resp. Algorithm 6). The l boolean array $\text{BF}(Y)$ denotes the boolean array $\text{BF}[l]$ storing a set Y using Algorithm 5.

Algorithm 5 Const Algorithm [22, Algorithm 1]**Input:** $Y \subset \{0, 1\}^*$, $H_{\mu \in [k]} : \{0, 1\}^* \rightarrow \mathbb{Z}_l = \{0, 1, \dots, l-1\}$.**Output:** Bloom Filter $\text{BF}(Y)$.

```

1: for  $\alpha$  from 0 to  $l-1$  by +1 do
2:    $\text{BF}[\alpha] \leftarrow 0$ .
3: end for
4: for all  $y \in Y$  do
5:   for  $\mu$  from 1 to  $k$  by +1 do
6:      $\beta \leftarrow H_{\mu}(y)$ .
7:     if  $\text{BF}[\beta] = 0$  then
8:        $\text{BF}[\beta] \leftarrow 1$ .
9:     end if
10:   end for
11: end for
12: return  $\text{BF}(Y)$ .

```

Algorithm 6 ElementCheck Algorithm [22, Algorithm 2]**Input:** Bloom Filter $\text{BF}(Y)$, $x \in X$, $H_{\mu \in [k]} : \{0, 1\}^* \rightarrow \mathbb{Z}_l = \{0, 1, \dots, l-1\}$.**Output:** 1 (if $x \in Y$) or 0 (otherwise).

```

1: for  $\mu$  from 1 to  $k$  by +1 do
2:    $\beta \leftarrow H_{\mu}(x)$ .
3:   if  $\text{BF}(Y)[\beta] = 0$  then
4:     return 0.
5:   end if
6: end for
7: return 1.
```

A Bloom filter may cause a false positive. If $\#Y = w$ then the probability of a false positive ϵ is given by the following equation:

$$\epsilon = \left\{ 1 - \left(1 - \frac{1}{l} \right)^{wk} \right\}^k. \quad (1)$$

The probability of a false positive ϵ has a negligible probability for $l \geq 128w$ [4, Section 2.2]. It is well-known that for given l and w , the probability ϵ is minimum when $k = \frac{l}{w} \log_e 2$ [4, Section 2.1]. We refer the reader to [2, 4] for details.

3 Proposed protocol 1

In this section, we propose the proposed protocol 1 that applying a Bloom filter in order to improve the efficiency of the PI-Sum (Algorithm 1). Inspired by the technique using a Bloom filter proposed by Miyaji, Nakasho, and Nishida in [22], we shall improve the efficiency of Intersection Set Algorithm (Algorithm 3) by using a Bloom filter.

3.1 Description of proposed protocol 1

The proposed protocol 1 is described as Algorithm 7.

Algorithm 7 The Proposed Protocol 1**Alice's input:** $A = \{u_i\}_{i \in [I]}$, k_1 , pk_B .**Bob's input:** $B = \{(v_j, t_j)\}_{j \in [J]}$, k_2 , pk_B , sk_B .**Alice's output:** $\mathcal{J} = \{\phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}\}$.**Bob's output:** the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.

```

1: Alice and Bob perform Precomputation Algorithm (Algorithm 2):
   Alice inputs  $A = \{u_i\}_{i \in [I]}$ ,  $k_1$  and Bob inputs  $B = \{(v_j, t_j)\}_{j \in [J]}$ ,  $k_2$ .
   Alice obtains  $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ 
   and  $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$ .
2: Alice and Bob perform the proposed algorithm (Algorithm 8):
   Alice inputs  $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ ,  $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$ .
   Alice obtains  $\mathcal{J}$ .
3: Alice and Bob perform Intersection Set Algorithm (Algorithm 4):
   Alice inputs  $\{\text{HomEnc}_{pk_B}(t_{\phi_3(j)})\}_{j \in [J]}$ ,  $\mathcal{J}$  and Bob inputs  $sk_B$ .
   Bob obtains the intersection sum  $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$ .
4: Bob outputs the intersection sum  $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$ .
```

The PI-Sum is constructed by Algorithm 2, Algorithm 3, and Algorithm 4, whereas the proposed protocol 1 is constructed by Algorithm 2, Algorithm 8, and Algorithm 4. Thus, the only difference between the PI-Sum (Algorithm 1) and the proposed protocol 1 (Algorithm 7) is the phase of the computation of private set intersection (Algorithm 3 and Algorithm 8). Algorithm 8 (the proposed algorithm) is used for the phase of the computation of private set intersection in the proposed protocol 1, and is based

on a Bloom filter. In the proposed protocol 1, Alice obtains $\mathcal{J}$ (the set of all indices of the elements $v_{\phi_3(j)} \in A \cap B$), and Bob obtains the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$. Let $\text{RO} : U \rightarrow \mathbb{G}$ be defined as in Sect. 2. Alice and Bob share a security parameter λ , and perform the setup phase in exactly the same manner as in the PI-Sum. Since the proposed protocol 1 uses a Bloom filter, the proposed protocol 1 requires Alice to generate $\psi \in \text{Inj}(\mathbb{G}, \{0, 1\}^*)$ and share ψ with Bob.

3.2 Main idea of proposed protocol 1

In this section, we explain the details and the main idea of the proposed protocol 1. In the proposed protocol 1, Alice and Bob perform Precomputation Algorithm (Algorithm 2) in step 1 of the proposed protocol 1 in exactly the same manner as in the PI-Sum, and Alice obtains $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ and $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_g}(t_{\phi_3(j)}))\}_{j \in [J]}$ by this procedure. Alice and Bob also perform Intersection Sum Algorithm (Algorithm 4) in step 3 of the proposed

protocol 1 in exactly the same manner as in the PI-Sum, and Bob obtains the intersection sum $\sum_{t_{\phi_3(j)} \in \mathcal{J}} t_{\phi_3(j)}$ by this procedure. On the other hand, Alice obtains $\mathcal{J}$ in step 2 of the proposed protocol 1 by performing Algorithm 8 (the proposed algorithm), whereas Intersection Set Algorithm (Algorithm 3) are used in the PI-Sum. The proposed algorithm is described as Algorithm 8. We refer the reader to Sect. 3.1 for the definition of $\psi \in \text{Inj}(\mathbb{G}, \{0, 1\}^*)$ in Algorithm 8.

Algorithm 8 Proposed Algorithm

Alice's input: $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$, $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$.

Bob's input: N/A.

Alice's output: $\mathcal{J} = \{\phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}\}$.

Bob's output: N/A.

- 1: Alice performs Const Algorithm (Algorithm 5):
 Alice inputs $\psi(\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]})$ and
 outputs $\text{BF}(\psi(\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}))$.
 - 2: Alice initializes $\mathcal{J}$ with $\emptyset$.
 - 3: **for** j from 1 to J by +1 **do**
 - 4: Alice performs ElementCheck Algorithm (Algorithm 6):
 Alice inputs $\psi(\text{RO}(v_{\phi_3(j)})^{k_1 k_2})$, $\text{BF}(\psi(\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}))$
 and outputs 0 or 1.
 - 5: **if** Alice obtains 1 **then**
 - 6: Alice updates $\mathcal{J}$ with $\mathcal{J} \cup \{\phi_3(j)\}$.
 - 7: **end if**
 - 8: **end for**
 - 9: Alice outputs $\mathcal{J}$.
-

As we see above, the only difference between the PI-Sum and the proposed protocol 1 is the phase of the computation of private set intersection (Algorithm 3 and Algorithm 8). The proposed algorithm (Algorithm 8) outputs $\mathcal{J}$ from Alice's input $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$, $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$ using a Bloom filter. Since Alice computes $\mathcal{J}$ by checking whether $\delta_1 = \delta_2$ or not for each $(\delta_1, \delta_2) \in \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]} \times \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$, Alice exactly obtains $\mathcal{J}$ in Intersection Set Algorithm (Algorithm 3). However, the computational cost of Algorithm 3 is relatively large because of $I \times J$ times computation of checking whether $\delta_1 = \delta_2$ for each $(\delta_1, \delta_2) \in \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]} \times \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$.

Inspired by the technique using a Bloom filter proposed by Miyaji, Nakasho, and Nishida in [22], we use a Bloom filter in the proposed algorithm (Algorithm 8) in order to improve the efficiency of Intersection Set Algorithm (Algorithm 3). More precisely, Algorithm 8 requires a computation of Const Algorithm (Algorithm 5) and J times computation of ElementCheck Algorithm (Algorithm 6). Since Algorithm 5 and Algorithm 6 essentially need multiple computation of hashing which is relatively small, the computational cost of the proposed algorithm (Algorithm 8) is

smaller than that of Algorithm 3. This is the main observation which allows us to improve the efficiency of Intersection Set Algorithm (Algorithm 3).

On the other hand, since a Bloom filter may cause a false positive, $\phi_3(j) \in [J]$ satisfying $\text{RO}(v_{\phi_3(j)})^{k_1 k_2} \notin \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ might be included in $\mathcal{J}$ in Algorithm 8. We will discuss this point in Sect. 3.2 (Lemma 1).

Proposition 1 Step 1 of Algorithm 7 (the proposed protocol 1) corresponds exactly to step 1 of Algorithm 1 (the PI-Sum). Steps 3–4 of Algorithm 7 (the proposed protocol 1) also corresponds exactly to steps 3–4 of Algorithm 1 (the PI-Sum). On the other hand, step 2 of Algorithm 7 (the proposed protocol 1) only differ from step 2 of Algorithm 1 (the PI-Sum).

Remark 1 Let $\text{Enc}_b : \{0, 1\}^b \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^b$ be a pseudorandom permutation. If $\lambda = 128$ and $b = 128$, then we can easily construct k independent hash functions $H_{\mu \in [k]} : \{0, 1\}^* \rightarrow \mathbb{Z}_{256}$ by using SHA-256 $H : \{0, 1\}^* \rightarrow \{0, 1\}^{256}$ and $\text{Enc}_{128, \mathcal{K}} : \{0, 1\}^{128} \times \mathcal{K} = \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$, where $\mathcal{K} \in \{0, 1\}^{128}$. For $x \in \{0, 1\}^*$, we first compute $H(x) = H_{\text{left}}(x) \parallel H_{\text{right}}(x) \in \{0, 1\}^{256}$. Here, $H_{\text{left}}(x) \in \{0, 1\}^{128}$

(resp. $H_{\text{right}}(x) \in \{0, 1\}^{128}$) is the left (resp. the right) most significant 128-bits of $H(x)$. Since $H_{\text{left}}(x) \in \{0, 1\}^{128}$ and $H_{\text{right}}(x) \in \{0, 1\}^{128}$, we have $\text{Enc}_{128, \mathcal{K}}(H_{\text{left}}(x)), \text{Enc}_{128, \mathcal{K}}(H_{\text{right}}(x)) \in \{0, 1\}^{128}$. Thus, we can construct a hash function from $\{0, 1\}^*$ to $\mathbb{Z}_{256}$ by taking $\text{int}(\text{Enc}_{128, \mathcal{K}}(H_{\text{left}}(x)) \parallel \text{Enc}_{128, \mathcal{K}}(H_{\text{right}}(x))) \in \mathbb{Z}_{256}$

for each $x \in \{0, 1\}^*$. We can construct $\text{Enc}_{128, \mathcal{K}}$ using 128-bit AES [8], and we can construct k independent hash functions by changing the key $\mathcal{K} \in \{0, 1\}^{128}$.

3.3 Correctness of proposed protocol 1

In this section, we shall prove the correctness of the proposed protocol 1 in Theorem 1. The correctness of the PI-Sum was proven by Ion et al. in [13, Section 3]. Since the only difference between the Algorithm 1 and Algorithm 7 is the phase of the computation of private set intersection, it suffices for the proof of the correctness of Algorithm 7 to show the following lemma (Lemma 1).

Lemma 1 Let $\mathcal{A}$ denote the event that in an execution of Algorithm 7, it is determined in step 4 of Algorithm 8 that the number $\phi_3(j) \in [J]$ satisfying $\text{RO}(v_{\phi_3(j)})^{k_1 k_2} \notin \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ belongs to $\mathcal{J}$. Let $\mathcal{B}$ denote the event that Alice inputs $\psi(\text{RO}(v_{\phi_3(j)})^{k_1 k_2})$ satisfying $\text{RO}(v_{\phi_3(j)})^{k_1 k_2} \notin \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ and $\text{BF}(\psi(\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}))$ in Algorithm 5, and obtains 1 in step 4 of Algorithm 8. Namely, the event $\mathcal{B}$ is that the Bloom filter determines the number $\text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$ belongs to $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ although $\text{RO}(v_{\phi_3(j)})^{k_1 k_2} \notin \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$. Then, we have the following inequality:

$$\Pr[\mathcal{A}] \leq \Pr[\mathcal{B}] = \left\{ 1 - \left(1 - \frac{1}{l} \right)^{lk} \right\}^k.$$

In other words, the probability $\Pr[\mathcal{A}]$ is negligible for sufficiently large l .

Proof If the event $\mathcal{A}$ occurs, then the event $\mathcal{B}$ also occurs in step 4 of Algorithm 8. From this, we have $\Pr[\mathcal{A}] \leq \Pr[\mathcal{B}]$. A Bloom filter has a false positive. If the event $\mathcal{B}$ occurs, then a false positives occurs in Algorithm 6. If a false positives occurs in Algorithm 6, then the event $\mathcal{B}$ occurs. Thus the following equation holds:

$$\Pr[\mathcal{B}] = \left\{ 1 - \left(1 - \frac{1}{l} \right)^{lk} \right\}^k. \quad (2)$$

By Inequality $\Pr[\mathcal{A}] \leq \Pr[\mathcal{B}]$ and by Eq. (2), the following inequality holds:

$$\Pr[\mathcal{A}] \leq \Pr[\mathcal{B}] = \left\{ 1 - \left(1 - \frac{1}{l} \right)^{lk} \right\}^k. \quad (3)$$

By setting $l = 128l$ and $k = 128 \log_e 2$, we have $\Pr[\mathcal{B}] = \{1 - (1 - 1/l)^{lk}\}^k \approx 1/2^{128}$ (see [4, section 2.2]). Therefore, the probability $\Pr[\mathcal{B}] = \{1 - (1 - 1/l)^{lk}\}^k$ is negligible. Thus, by Inequality (3), the probability $\Pr[\mathcal{A}]$ is also negligible. This concludes the proof of Lemma 1. $\square$

By Lemma 1, the following theorem (Theorem 1) holds.

Theorem 1 The proposed protocol 1 (Algorithm 7) satisfies the correctness.

3.4 Security analysis of proposed protocol 1

In this section, we briefly discuss the security of the proposed protocol 1. Our analysis is similar to that of the

Table 2 The number of computations in Algorithm 2 and Algorithm 4

	Algorithm 2	Algorithm 4
Num. of scalar multiplications on Curve25519	$2(I + J)$	N/A
Num. of encryption	J	1
Num. of multiplications between ciphertexts	N/A	J
Num. of decryption	N/A	1

Table 1 Notation for the estimation

$M^{(p)}$	The computational cost of one multiplication on $\mathbb{F}_p$
S	The computational cost of one squaring on $\mathbb{F}_p$
ECDBL	The computational cost of one point doubling on Curve25519
ECADD	The computational cost of one point addition on Curve25519
Enc	The computational cost of one encryption
Dec	The computational cost of one decryption
Exp	The computational cost of one exponentiation on $\mathbb{Z}/N^2\mathbb{Z}$
$M^{(N^2)}$	The computational cost of one multiplication between ciphertexts

PI-Sum [13]. The security of the PI-Sum was proven by Ion *et al.* in [13, Section 3.1]. More precisely, the authors prove the security of the PI-Sum in the honest-but-curious model in [13, Theorem 1] and [13, Theorem 2]. Since the only difference between the PI-Sum ([13, Figure 2] or Algorithm 1) and the proposed protocol 1 (Algorithm 7) is the phase of the computation of private set intersection (Algorithm 3 and Algorithm 8), it suffices to consider the phase of the computation of private set intersection (Algorithm 8) for the security of the proposed protocol 1. The following theorem holds for the security of the proposed protocol 1.

Theorem 2 *Alice cannot obtain any information about Bob's secret in Algorithm 8. Moreover, the proposed protocol 1 is secure in the honest-but-curious model.*

Proof We first prove that Alice cannot obtain any information about Bob's secret $B = \{(v_j, t_j)\}_{j \in [J]}$. Alice performs Const Algorithm (Algorithm 5) in step 1 of Algorithm 8 and Alice performs ElementCheck Algorithm (Algorithm 6) in step 4 of Algorithm 8. Since Alice inputs the set of ciphertexts $\psi\left(\left\{\text{RO}\left(u_{\phi_2(\phi_1(i))}\right)^{k_1 k_2}\right\}_{i \in [I]}\right), \psi\left(\left\{\text{RO}\left(v_{\phi_3(j)}\right)^{k_1 k_2}\right\}_{j \in [J]}\right)$ in these algorithms, it is infeasible for Alice to obtain the Bob's secret $v_{\phi_3(j)} \in B, k_2$. In addition, the index $i \in [I]$ of $u_i \in A$ is not same as the index $\phi_2(\phi_1(i)) \in [I]$ of the ciphertext $\text{RO}\left(u_{\phi_2(\phi_1(i))}\right)^{k_1 k_2}$ of $u_i \in A$. Similarly, the index $j \in [J]$ of $v_j, t_j \in B$ is not same as the index $\phi_3(j) \in [J]$ of the ciphertext $\left(\text{RO}\left(v_{\phi_3(j)}\right)^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)})\right)$ of $v_j, t_j \in B$. Thus, it is infeasible for Alice to obtain any element in $A \cap B$ by comparing the index $\phi_3(j) \in [J]$ and the index $i \in [I]$. The latter part follows from the former part. $\square$

Remark 2 Ion *et al.* discuss the security of PI-Sum [13] in the honest-but-curious model. More precisely, they discuss that Alice cannot obtain any information about Bob's secret in [13, Theorem 1] and Bob cannot obtain any information about Alice's secret in [13, Theorem 2]. On the other hand, Bob does not input its own secret and does not obtain the output of Algorithm 8. In addition, Bob does not carry out any computation in Algorithm 8. Therefore, we only discuss the security of the phase of the computation of private set intersection (Algorithm 8) for the security of the proposed protocol 1 in Theorem 2.

4 Efficiency estimation of proposed protocol 1

In this section, we estimate the computational costs of the PI-Sum (Algorithm 1) and the proposed protocol 1 (Algorithm 7). In addition, we compare the computational costs

of the proposed protocol 1 with that of the PI-Sum. We use Curve25519 as Abelian group $\mathbb{G}$. Since $p = 2^{255} - 19$ and $\log_2 p = 255$, we assume that $\lambda = 128$ and $\lceil \log_2 N \rceil = 3072$. As mentioned in Sect. 2.1, we can regard each $x \in \mathbb{F}_p$ as the x -coordinate of a point on Curve25519 [1, Theorem 2.1]. Since a scalar multiplication of Curve25519 is not necessary to compute the y -coordinate of a point on Curve25519, the computational cost of the random oracle RO is negligibly small compared to other steps in the PI-Sum and the proposed protocol 1. Therefore, we can ignore the computational cost of the random oracle RO. By a similar argument, we can also ignore the computational costs of the mappings ϕ_1, ϕ_2, ϕ_3 , and ψ . We use the notation in Table 1 to estimate the computational costs of the PI-Sum and the proposed protocol 1.

Here, we estimate the computational costs of the PI-Sum and the proposed protocol 1 by counting the number of $\mathbf{M}^{(p)}$, $\mathbf{S}$, **ECDBL**, **ECADD**, **Enc**, **Dec**, **Exp**, and $\mathbf{M}^{(N^2)}$, respectively. We denote the computational cost of Precomputation Algorithm (Algorithm 2) (resp. Intersection Sum Algorithm (Algorithm 4)) by $\text{Cost}_{\text{Precomp}}$ (resp. $\text{Cost}_{\text{Inter-Sum}}$). We define

$$\text{Cost}_{\text{Common}} := \text{Cost}_{\text{Precomp}} + \text{Cost}_{\text{Inter-Sum}}.$$

We first estimate the computational cost $\text{Cost}_{\text{Common}}$. Table 2 shows the number of computations in Precomputation Algorithm and Intersection Sum Algorithm.

From [5, Section 3], we have $0.8\mathbf{M}^{(p)} \leq \mathbf{S} \leq \mathbf{M}^{(p)}$. Here, we assume that $\mathbf{S} = \mathbf{M}^{(p)}$. From [1, Section 5], we have **ECDBL** = $4\mathbf{S} + 1\mathbf{M}^{(p)} = 5\mathbf{M}^{(p)}$ and **ECADD** = $5\mathbf{M}^{(p)}$. These equations yield **ECDBL** = **ECADD** = $5\mathbf{M}^{(p)}$. Since we can use precomputing techniques in the Paillier encryption scheme by applying the technique of [24, Section 7], we assume that $1\mathbf{Enc} = 1\mathbf{Dec} = 1\mathbf{Exp}$. By [20, Algorithm 14.76], we have $1\mathbf{Exp} = \frac{3}{2} \lceil \log_2 N \rceil \mathbf{M}^{(N^2)}$, namely,

$$1\mathbf{Enc} = 1\mathbf{Dec} = 1\mathbf{Exp} = \frac{3}{2} \lceil \log_2 N \rceil \mathbf{M}^{(N^2)}. \quad (4)$$

For the computational cost of one scalar multiplication **ScalarMult**, the following equation holds by [6]:

$$\begin{aligned} 1\mathbf{ScalarMult} &= \log_2 p \times \mathbf{ECDBL} + \frac{\log_2 p}{2} \times \mathbf{ECADD} \\ &= \frac{15}{2} (\log_2 p) \mathbf{M}^{(p)}. \end{aligned} \quad (5)$$

By Eqs. (4 and 5), the computational cost $\text{Cost}_{\text{Precomp}}$ satisfies the following equation:

$$\begin{aligned} \text{Cost}_{\text{Precomp}} &= 2(I + J) \times \mathbf{ScalarMult} + J \times \mathbf{Enc} \\ &= 15(\log_2 p)(I + J) \mathbf{M}^{(p)} + \frac{3}{2} \lceil \log_2 N \rceil J \mathbf{M}^{(N^2)}. \end{aligned} \quad (6)$$

By a similar argument as above, the computational cost $\text{Cost}_{\text{Inter-Sum}}$ satisfies the following equation:

$$\begin{aligned}\text{Cost}_{\text{Inter-Sum}} &= J \times \mathbf{M}^{(N^2)} + 1\text{Enc} + 1\text{Dec} \\ &= (J + 3\lceil \log_2 N \rceil) \mathbf{M}^{(N^2)}.\end{aligned}\quad (7)$$

By Eqs. (6 and 7), the computational cost $\text{Cost}_{\text{Common}}$ satisfies the following equation:

$$\begin{aligned}\text{Cost}_{\text{Common}} &= \text{Cost}_{\text{Precomp}} + \text{Cost}_{\text{Inter-Sum}} \\ &= 15(\log_2 p)(I + J) \mathbf{M}^{(p)} \\ &\quad + \left(\left(\frac{3}{2} \lceil \log_2 N \rceil + 1 \right) J + 3 \lceil \log_2 N \rceil \right) \mathbf{M}^{(N^2)} \quad (8) \\ &= 3825(I + J) \mathbf{M}^{(p)} + (4609J + 9216) \mathbf{M}^{(N^2)}.\end{aligned}$$

Finally, we estimate the computational costs of the PI-Sum and the proposed protocol 1. Let $\text{Cost}_{\text{PI-Sum}}$ (resp. $\text{Cost}_{\text{Proposed1}}$) be the computational cost of Algorithm 1 (resp. Algorithm 7). In Intersection Set Algorithm, we compute $\mathcal{J}$ (the set of all indices of the elements $v_{\phi_3(j)} \in A \cap B$) by checking whether $\delta_1 = \delta_2$ or not for each $(\delta_1, \delta_2) \in \{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]} \times \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$. Hence, the computational cost $\text{Cost}_{\text{PI-Sum}}$ satisfies

$$\begin{aligned}\text{Cost}_{\text{PI-Sum}} &= \text{Cost}_{\text{Common}} + IJ\lambda \\ &= 3825(I + J) \mathbf{M}^{(p)} + (4609J + 9216) \mathbf{M}^{(N^2)} + IJ\lambda.\end{aligned}\quad (9)$$

In the proposed algorithm, we input $\psi(\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}) \subset \{0, 1\}^*$ in Const Algorithm. Const Algorithm requires I iterations in steps 4–11 of Algorithm 5. Thus, the computational cost of the whole processes in Const Algorithm is $I\lambda$. We input $\psi(\text{RO}(v_{\phi_3(j)})^{k_1 k_2})$ and $\text{BF}(\psi(\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}))$ in ElementCheck Algorithm. It is easy to see that the computational cost of J times performing of Algorithm 6 is $J\lambda$. Hence, the computational cost $\text{Cost}_{\text{Proposed1}}$ satisfies

$$\begin{aligned}\text{Cost}_{\text{Proposed1}} &= \text{Cost}_{\text{Common}} + (I + J)\lambda \\ &= 3825(I + J) \mathbf{M}^{(p)} + (4609J + 9216) \mathbf{M}^{(N^2)} + (I + J)\lambda.\end{aligned}\quad (10)$$

5 Countermeasures against repeatedly executing the PI-Sum

As explained in [13, Section 3], it may be possible that “repeatedly executing the PI-Sum in sequence will leak information due to correlated inputs in different sessions, and one strategy for resolving this issue would be to compose differential privacy techniques [11] with the cryptographic protocol, by adding appropriately sampled noise to the inputs”. In this section, we propose two protocols (Algorithm 10 and Algorithm 11) having the resistance to the above security problem. These proposed protocols are modifications of the “Reverse” Private Intersection-Sum protocol (RePI-Sum for short) in [13, Figure 4].

5.1 Description of proposed protocol 2 and proposed protocol 3

In this section, we propose the proposed protocol 2 (Algorithm 10) and the proposed protocol 3 (Algorithm 11). These protocols allow Bob to obtain the intersection sum. The proposed protocol 2 and the proposed protocol 3 are based on the proposed estimation algorithm for the intersection size (Algorithm 9) and the proposed protocol 1. Algorithm 9 is based on Bloom filter Bootstrap [15, Algorithm 2] for preventing the security problem in the RePI-Sum ([13, Figure 4]). The proposed estimation algorithm for the intersection size is described as Algorithm 9. We refer the reader to Sect. 3.1 for the definition of $\psi \in \text{Inj}(\mathbb{G}, \{0, 1\}^*)$ in Algorithm 9.

Algorithm 9 Proposed Estimation Algorithm for the Intersection Size

Alice's input: X, S (the number of observations).

Bob's input: Y, S (the number of observations).

Alice's output: intersection size $\hat{x} \approx \#(X \cap Y)$.

Bob's output: intersection size $\hat{x} \approx \#(X \cap Y)$.

- 1: **for** d from 1 to S by +1 **do**
- 2: Alice performs Const Algorithm (Algorithm 5):
Alice inputs $\psi(X)$ and obtains $\text{BF}(\psi(X))$.
- 3: Bob performs Const Algorithm (Algorithm 5):
Bob inputs $\psi(Y)$ and obtains $\text{BF}(\psi(Y))$.
- 4: Alice and Bob perform SSP ([10, Protocol 2]):
Alice inputs $\text{BF}(\psi(X))$ and Bob inputs $\text{BF}(\psi(Y))$.
Alice obtains $y_{a,d}$ and Bob obtains $y_{b,d}$.
- 5: Alice and Bob jointly compute $y_d = y_{a,d} + y_{b,d}$, and store y_d .
- 6: **end for**
- 7: Alice and Bob compute $\hat{\theta} = \left(1 + \sum_{d=1}^S y_d\right) / (2 + S)$.
- 8: Alice and Bob compute the intersection size
 $\hat{x} = \#X + \#Y - \frac{1}{k} \log_{1-\frac{1}{l}} \left(\hat{\theta} - 1 + \left(1 - \frac{1}{l}\right)^{k\#X} + \left(1 - \frac{1}{l}\right)^{k\#Y} \right)$.

In Algorithm 9, Alice has the private set X and Bob has the private set Y . Algorithm 9 allows Alice and Bob to obtain the estimation size of the intersection $\hat{x} \approx \#(X \cap Y)$. Kikuchi and Sakuma in [15, Section 5.1] show that the following inequality holds

$$|[\hat{x}] - \#(X \cap Y)| \leq 1$$

in Bloom filter Bootstrap [15, Algorithm 2]. So, we assume that the above inequality holds in Algorithm 9. In Algorithm 9, we use [10, Protocol 2] for Secure Scalar Product Protocol (SSP for short). In Algorithm 9, the input S is the number of executions of SSP. In steps 1–6 of Algorithm 9, Alice obtains $\text{BF}(\psi(X))$ and Bob obtains $\text{BF}(\psi(Y))$ in the d -th

execution of SSP, where $1 \leq d \leq S$. In steps 4–5 of Algorithm 9, y_d satisfies the following equation:

$$y_d = y_{a,d} + y_{b,d} = \#\{\alpha \in \mathbb{Z}_l \mid \text{BF}(\psi(X))[\alpha] = \text{BF}(\psi(Y))[\alpha] = 1\}.$$

For $\alpha \in \mathbb{Z}_l$, let us denote by θ the probability of the event $\text{BF}(\psi(X))[\alpha] = \text{BF}(\psi(Y))[\alpha] = 1$. In step 7 of Algorithm 9, $\hat{\theta}$ is the Bayesian estimation of the probability θ . We call the phase of computing y_d using SSP (steps 1–6 of Algorithm 9) *the secure scalar product phase*. We also call the phase of computing the output using $\hat{\theta}$ (steps 7–8 of Algorithm 9) *the intersection size phase*. We describe the proposed protocol 2 (resp. the proposed protocol 3). The detailed description of the algorithms are given in Algorithm 10 and Algorithm 11.

Algorithm 10 Proposed Protocol 2

Alice's input: $A = \{u_i\}_{i \in [I]}$, k_1 , pk_B , T .

Bob's input: $B = \{(v_j, t_j)\}_{j \in [J]}$, k_2 , pk_B , sk_B , T .

Alice's output:

$$\mathcal{J} = \left\{ \phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \left\{ \text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2} \right\}_{i \in [I]} \right\}.$$

Bob's output: the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.

- 1: Alice and Bob perform the proposed estimation algorithm for the intersection size (Algorithm 9):
 - Alice inputs $\{u_i\}_{i \in [I]}$ and Bob inputs $\{v_j\}_{j \in [J]}$.
 - Alice and Bob obtain $\hat{x} \approx \#(\{u_i\}_{i \in [I]} \cap \{v_j\}_{j \in [J]})$.
 - 2: Alice (or Bob) checks if $\hat{x} \leq T$ or not.
 - 3: **if** $\hat{x} \leq T$ **then**
 - 4: Alice (or Bob) outputs “ $\perp$ ”.
 - 5: **end if**
 - 6: Alice and Bob perform Precomputation Algorithm (Algorithm 2):
 - Alice inputs $A = \{u_i\}_{i \in [I]}$ and k_1 , and Bob inputs $B = \{(v_j, t_j)\}_{j \in [J]}$ and k_2 .
 - Alice obtains $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ and $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$.
 - 7: Alice and Bob perform the proposed algorithm (Algorithm 8):
 - Alice inputs $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ and $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$.
 - Alice obtains $\mathcal{J}$.
 - 8: Alice and Bob perform Intersection Set Algorithm (Algorithm 4):
 - Alice inputs $\{\text{HomEnc}_{pk_B}(t_{\phi_3(j)})\}_{j \in [J]}$ and $\mathcal{J}$, and Bob inputs sk_B .
 - Bob obtains the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.
 - 9: Bob outputs the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.
-

Fig. 1 Comparison result of the computational costs of $\text{Cost}_{\text{PI-Sum}}$ and $\text{Cost}_{\text{Proposed1}}$

Algorithm 11 Proposed Protocol 3

Alice's input: $A = \{u_i\}_{i \in [I]}$, k_1 , pk_B , T .

Bob's input: $B = \{(v_j, t_j)\}_{j \in [J]}$, k_2 , pk_B , sk_B , T .

Alice's output:

$$\mathcal{J} = \left\{ \phi_3(j) \in [J] \mid \text{RO}(v_{\phi_3(j)})^{k_1 k_2} \in \left\{ \text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2} \right\}_{i \in [I]} \right\}.$$

Bob's output: the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.

- 1: Alice and Bob perform Precomputation Algorithm (Algorithm 2):
 Alice inputs $A = \{u_i\}_{i \in [I]}$ and k_1 , and Bob inputs $B = \{(v_j, t_j)\}_{j \in [J]}$ and k_2 .
 Alice obtains $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$
 and $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$.
- 2: Alice and Bob perform the proposed estimation algorithm for the intersection size (Algorithm 9):
 Alice inputs $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$ and Bob inputs $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$.
 Alice and Bob obtain $\hat{x} \approx \#(\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]} \cap \{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]})$.
- 3: Alice (or Bob) checks if $\hat{x} \leq T$ or not.
- 4: **if** $\hat{x} \leq T$ **then**
- 5: Alice (or Bob) outputs " $\perp$ ".
- 6: **end if**
- 7: Alice and Bob perform the proposed algorithm (Algorithm 8):
 Alice inputs $\{\text{RO}(u_{\phi_2(\phi_1(i))})^{k_1 k_2}\}_{i \in [I]}$ and $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$.
 Alice obtains $\mathcal{J}$.
- 8: Alice and Bob perform Intersection Set Algorithm (Algorithm 4):
 Alice inputs $\{\text{HomEnc}_{pk_B}(t_{\phi_3(j)})\}_{j \in [J]}$ and $\mathcal{J}$, and Bob inputs sk_B .
 Bob obtains the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.
- 9: Bob outputs the intersection sum $\sum_{\phi_3(j) \in \mathcal{J}} t_{\phi_3(j)}$.

Algorithm 10 (resp. Algorithm 11) is the protocol that Alice and Bob perform Algorithm 9 in just before step 1 (resp. just after step 1) of Algorithm 7. In Algorithm 10 and Algorithm 11, Alice and Bob share the security parameter λ , and perform the setup phase in exactly the same manner as in Algorithm 7. Algorithm 10 and Algorithm 11 require Alice and Bob to share the threshold T with respect to the cardinality of the intersection set of input sets A and B .

We describe the overview of Algorithm 10. Alice and Bob perform Algorithm 9 in step 1 of Algorithm 10. Alice (resp. Bob) inputs the private set $\{u_i\}_{i \in [I]}$ (resp. $\{v_j\}_{j \in [J]}$) in Algorithm 9. Thus Alice and Bob obtain the estimation size of the intersection $\hat{x} \approx \#(\{u_i\}_{i \in [I]} \cap \{v_j\}_{j \in [J]})$. Alice (or Bob) checks whether $\hat{x} \leq T$ or not in step 2 of Algorithm 10. If $\hat{x} \leq T$, then Alice (or Bob) outputs " $\perp$ " and aborts Algorithm 10. If Bob aborts Algorithm 10, then Alice and Bob add a new element into the input sets A and B , and repeat Algorithm 10. As similar to steps 1–4 of Algorithm 7, Bob

computes and outputs the intersection sum in steps 6–9 of Algorithm 10.

Next, we describe the overview of Algorithm 11. As similar to step 1 of Algorithm 7, Algorithm 11 encrypts $A = \{u_i\}_{i \in [I]}$ and $B = \{(v_j, t_j)\}_{j \in [J]}$, and outputs $\{\text{RO}(u_{\phi_2(\phi_1(i))}^{k_1 k_2})\}_{i \in [I]}$, $\{(\text{RO}(v_{\phi_3(j)})^{k_1 k_2}, \text{HomEnc}_{pk_B}(t_{\phi_3(j)}))\}_{j \in [J]}$ in step 1 of Algorithm 11. Alice holds $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$ in just after step 1 of Algorithm 11. Bob also holds $\{\text{RO}(u_{\phi_2(\phi_1(i))}^{k_1 k_2})\}_{i \in [I]}$ in step 10 of Algorithm 2. Alice and Bob perform Algorithm 9 in step 2 of Algorithm 11. Alice inputs $\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]}$ and Bob inputs $\{\text{RO}(u_{\phi_2(\phi_1(i))}^{k_1 k_2})\}_{i \in [I]}$ into Algorithm 9. Thus Alice and Bob obtain the estimation size of the intersection

$$\hat{x} \approx \#(\{\text{RO}(v_{\phi_3(j)})^{k_1 k_2}\}_{j \in [J]} \cap \{\text{RO}(u_{\phi_2(\phi_1(i))}^{k_1 k_2})\}_{i \in [I]}).$$

As similar to steps 2–5 of Algorithm 10, Algorithm 11 also checks whether $\hat{x} \leq T$ or not in steps 3–6 of Algorithm 11. If $\hat{x} \leq T$, then Alice (or Bob) outputs “ $\perp$ ” and aborts Algorithm 11. As similar to steps 1–4 of Algorithm 7, Algorithm 11 outputs the intersection sum in steps 7–9 of Algorithm 11. From the above discussion, Algorithm 10 (resp. Algorithm 11) performs Algorithm 9 in just before (resp. after) encrypting the input sets A and B .

In Algorithm 10 and Algorithm 11, Bob can obtain the estimation size of the intersection $\hat{x} \approx \#(A \cap B)$. Bob can compute the intersection sum if and only if $T \leq \#(A \cap B)$ holds. If Bob may know the element of the Alice's private set $u_{i \in [I]} \in A$ from the intersection sum, Bob can prevent leakage of the element of the Alice's private set $u_{i \in [I]} \in A$ by aborting step 4 of Algorithm 10 or step 5 of Algorithm 11 without computing the output. Bob can obtain the intersection sum in Algorithm 10 and Algorithm 11.

5.2 Efficiency estimation of proposed protocol 2 and proposed protocol 3

In this section, we estimate the computational costs of the proposed protocol 2 (Algorithm 10) and the proposed protocol 3 (Algorithm 11). Let us denote by $\mathbf{M}^{(p')}$ the computational cost of one multiplication on $\mathbb{F}_{p'}$. Here, p' is a prime satisfying $p' > l$ and $p' \approx l$, and is used for step 4 of Algorithm 9 ([10, Protocol 2]). By [15, Section 4.4], we assume that $S = 10$ and $k = 1$. We define

$$\text{tmp} := \hat{\theta} - 1 + (1 - 1/l)^{k\#X} + (1 - 1/l)^{k\#Y}.$$

Remark that tmp is appeared in the equation of step 8 of Algorithm 9. We denote the computational costs of the proposed estimation algorithm for the intersection size (Algorithm 9) by $\text{Cost}_{\text{Estimation}}$. Similarly, we denote the secure scalar product phase (steps 1–6 of Algorithm 9),

and the intersection size phase (steps 7–8 of Algorithm 9) by Cost_{SSP} and $\text{Cost}_{\text{Size}}$, respectively. As in Section 4, we use the symbol $\text{Cost}_{\text{Proposed2}}$ (resp. $\text{Cost}_{\text{Proposed3}}$) to represent the computational cost of the proposed protocol 2 (resp. the proposed protocol 3). Since the only difference between the proposed protocol 2 and the proposed protocol 3 is their timing of performing Algorithm 9, we have

$$\text{Cost}_{\text{Proposed2}} = \text{Cost}_{\text{Proposed3}}. \quad (11)$$

Therefore, we only estimate $\text{Cost}_{\text{Proposed2}}$ by counting the number of $\mathbf{M}^{(p')}$. We first estimate the computational cost $\text{Cost}_{\text{Estimation}}$. The computational cost of step 2 of Algorithm 9 (resp. step 3 of Algorithm 9) is $\lambda\#X$ (resp. $\lambda\#Y$). By [10, Section 4.3], the computational cost of step 4 of Algorithm 9 is $l^2/2\mathbf{M}^{(p')}$. Thus the computational cost of Cost_{SSP} satisfies the following equation:

$$\begin{aligned} \text{Cost}_{\text{SSP}} &= S \left(\lambda\#X + \lambda\#Y + \frac{l^2}{2}\mathbf{M}^{(p')} \right) \\ &= 1280(\#X + \#Y) + 5l^2\mathbf{M}^{(p')}. \end{aligned} \quad (12)$$

In step 8 of Algorithm 9, it need to compute $\hat{x} = \#X + \#Y - (\log_{1-1/l} \text{tmp})/k$. Since

$$\log_{1-1/l} \text{tmp} = \log_e \text{tmp} / \log_e(1 - 1/l),$$

we can precompute $1/\log_e(1 - 1/l)$. Therefore, we can ignore the computational cost of computing $1/\log_e(1 - 1/l)$. By [20, Table 2.1] and [3, Section 9], the computational cost of $\text{Cost}_{\text{Size}}$ satisfies the following equation:

$$\begin{aligned} \text{Cost}_{\text{Size}} &\approx (\log_2 S)^2 + (\log_2 \text{tmp})^3 + (\log_2(\log_e \text{tmp}))^2 \\ &\approx \mathbf{M}^{(p')} + (\log_2 \text{tmp})^3 + (\log_2(\log_e \text{tmp}))^2. \end{aligned} \quad (13)$$

By Eqs. (12 and 13), the computational cost of $\text{Cost}_{\text{Estimation}}$ satisfies the following equation:

$$\begin{aligned} \text{Cost}_{\text{Estimation}} &= \text{Cost}_{\text{SSP}} + \text{Cost}_{\text{Size}} \\ &\approx (5l^2 + 1)\mathbf{M}^{(p')} + 1280(\#X + \#Y) \\ &\quad + (\log_2 \text{tmp})^3 + (\log_2(\log_e \text{tmp}))^2. \end{aligned}$$

Hence, the computational cost of $\text{Cost}_{\text{Proposed2}}$ satisfies the following equation:

$$\text{Cost}_{\text{Proposed2}} = \text{Cost}_{\text{Proposed1}} + \text{Cost}_{\text{Estimation}}. \quad (14)$$

By Eqs. (11 and 14), we have

$$\text{Cost}_{\text{Proposed3}} = \text{Cost}_{\text{Proposed1}} + \text{Cost}_{\text{Estimation}}. \quad (15)$$

6 Results and discussion

In Sect. 3, we propose Algorithm 7 which is an efficiency improvement technique for the PI-Sum (Algorithm 1). By Table 2 and Eq. (9), the computational complexity of Algorithm 1 is $O((I + J)\lambda^3)$. Similarly, by Table 2 and Eq. (10), the computational complexity of Algorithm 7 is also $O((I + J)\lambda^3)$. Thus, the computational complexity of Algorithm 1 is same as the computational complexity of Algorithm 7. However, we claim that if I and J are relatively large for a fixed security parameter λ then the proposed protocol 1 (Algorithm 7) is more efficient than the PI-Sum ([13, Figure 2] or Algorithm 1). Next we explain why the claim holds.

Figure 1 shows the computational costs of $\text{Cost}_{\text{PI-Sum}}$ and $\text{Cost}_{\text{Proposed1}}$. For the sake of simplicity we assume that $I = J = 2^x$ and the security parameter $\lambda = 128$. In Fig. 1, the graph of $y = f(x)$ (resp. $y = g(x)$) is described, where $\text{Cost}_{\text{PI-Sum}} = 2^y$ (resp. $\text{Cost}_{\text{Proposed1}} = 2^y$). By [20, Table 2.5] and by [1, Section 1], the multiplication cost $\mathbf{M}^{(p)}$ is $O((\lceil \log_2 p \rceil)^2)$. Since $p = 2^{255} - 19$, by omitting the constant cost of the multiplication cost $\mathbf{M}^{(p)}$, we may assume that

$$\mathbf{M}^{(p)} \approx (\lceil \log_2 p \rceil)^2 \approx (\lceil \log_2 (2^{255} - 19) \rceil)^2 \approx (255)^2. \quad (16)$$

From [20, Table 2.5] and by [24, Section 4], the multiplication cost $\mathbf{M}^{(N^2)}$ is $O((\lceil 2 \log_2 N \rceil)^2)$. It is known that if $\lambda = 128$ then $\lceil \log_2 N \rceil \approx 3072$ (See [23]). By omitting the constant cost of the multiplication cost $\mathbf{M}^{(N^2)}$, we may assume that

$$\mathbf{M}^{(N^2)} \approx (\lceil 2 \log_2 N \rceil)^2 \approx (2 \times 3072)^2 \approx (6144)^2. \quad (17)$$

In this situation, we have

$$\mathbf{M}^{(N^2)} \approx 144\mathbf{M}^{(p)}.$$

Substituting Eqs. (16 and 17) into Eq. (9) yields

$$\text{Cost}_{\text{PI-Sum}} = 128 \times 2^{2x} + 174481365474 \times 2^x + 347892350976. \quad (18)$$

Similarly, substituting Eqs. (16 and 17) into Eq. (10) yields

$$\text{Cost}_{\text{Proposed1}} = 174481365730 \times 2^x + 347892350976. \quad (19)$$

By Eqs. (18 and 19), if $x = 16$ then the computational cost of the proposed protocol 1 is about 1% smaller than that of the PI-Sum, namely,

$$\text{Cost}_{\text{Proposed1}} \approx \frac{99}{100} \times \text{Cost}_{\text{PI-Sum}}.$$

If $x = 22$ then the computational cost of the proposed protocol 1 is about 50% smaller than that of the PI-Sum, namely,

$$\text{Cost}_{\text{Proposed1}} \approx \frac{1}{2} \times \text{Cost}_{\text{PI-Sum}}.$$

Thus, the proposed protocol 1 (Algorithm 7) is faster than the PI-Sum (Algorithm 1).

In Sect. 5, we propose other efficiency improvement techniques for the PI-Sum. As explained in Sect. 5, repeatedly executing the PI-Sum in sequence will leak information. The proposed protocol 2 (Algorithm 10) and the proposed protocol 3 (Algorithm 11) are modifications of the RePI-Sum [13, Figure 4], and can prevent the above security problem. Moreover, by adding an aborting functionality in step 4 of Algorithm 10 and step 5 of Algorithm 11, Algorithm 10 and Algorithm 11 are faster than the RePI-Sum [13, Figure 4].

7 Conclusion

In this paper, we proposed the efficiency improvement techniques for the Ion et al. private intersection-sum protocol (PI-Sum). The proposed protocol 1 is more efficient than the PI-Sum. The computational cost of the proposed protocol 1 is about 1% smaller than that of the PI-Sum if the base 2 logarithm of the number of the private set is 16, and the computational cost of the proposed protocol 1 is about 50% smaller than that of the PI-Sum if the base 2 logarithm of the number of the private set is 22. To our best knowledge, there exists only two private intersection-sum protocols. Thus, the proposed protocol 1 is the fastest private intersection-sum protocol. We also proposed the proposed protocol 2 and the proposed protocol 3 for preventing the security problem described in Section 5. By adding an aborting functionality in the proposed protocol 2 and the proposed protocol 3, the proposed protocol 2 and the proposed protocol 3 can be more efficient than the "Reverse" Private Intersection-Sum protocol proposed by Ion et al. Implementation of our proposed protocols is left for future work.

Acknowledgements The authors would like to thank the anonymous reviewers for their helpful comments to improve the quality of the paper.

Funding This work was partially supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (Grant Number 19K20270).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing,

adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bernstein DJ (2006) Curve25519: new Diffie-Hellman speed records. In: Yung M, Dodis Y, Kiayias A, Malkin T (eds) Public key cryptography—PKC 2006. Springer, Berlin, pp 207–228
- Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. *Comm ACM* 13(7):422–426
- Brent RP (2010) Multiple-precision zero-finding methods and the complexity of elementary function evaluation. CoRR [arXiv:1004.3412](https://arxiv.org/abs/1004.3412)
- Broder A, Mitzenmacher M (2003) Network applications of bloom filters: a survey. *Internet Math* 1(4):485–509
- Cohen H, Miyaji A, Ono T (1998) Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta K, Pei D (eds) *Advances in cryptology: ASIACRYPT '98*. Springer, Berlin, pp 51–65
- Cohen H, Frey G, Avanzi R, Doche C, Lange T, Nguyen K, Vercauteren F (2006) *Handbook of elliptic and hyperelliptic curve cryptography*. CRC Press, Boca Raton
- De Cristofaro E, Tsudik G (2010) Practical private set intersection protocols with linear complexity. In: Sion R (ed) *Financial cryptography and data security: FC 2010*. Springer, Berlin, pp 143–159
- Daemen J, Rijmen V (2002) *The design of Rijndael: AES—the advanced encryption standard*. Springer, Berlin, Heidelberg
- Diffie W, Hellman ME (1976) New directions in cryptography. *IEEE Trans Inf Theor* 22(6):644–654
- Du W, Han YS, Chen S (2004) Privacy-preserving multivariate statistical analysis: linear regression and classification. In: *Proceedings of the 2004 SIAM international conference on data mining*; pp. 222–233
- Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. *Found Trends Theor Comput Sci* 9(3–4):211–407
- Freedman MJ, Nissim K, Pinkas B (2004) Efficient private matching and set intersection. In: Cachin C, Camenisch JL (eds) *Advances in cryptology: EUROCRYPT 2004*. Springer, Berlin, pp 1–19
- Ion M, Kreuter B, Nergiz E, Patel S, Saxena S, Seth K, Shanahan D, Yung M (2017) Private intersection-sum protocol with applications to attributing aggregate ad conversions. *IACR Cryptol ePrint Arch* 2017:738
- Ion M, Kreuter B, Nergiz E, Patel S, Raykova M, Saxena S, Seth K, Shanahan D, Yung M (2019) On deploying secure computing: private intersection-sum-with-cardinality. *IACR Cryptol ePrint Arch* 2019:723
- Kikuchi H, Sakuma J (2013) Bloom filter bootstrap: privacy-preserving estimation of the size of an intersection. In: Wang L, Shafiq B (eds) *Data and applications security and privacy XXVII—DBSec 2013*. Springer, Berlin, pp 145–163
- Kim M, Lee HT, Cheon JH (2012) Mutual private set intersection with linear complexity. In: Jung S, Yung M (eds) *International workshop on information security applications: WISA 2011*. Springer, Berlin, pp 219–231
- Kissner L, Song D (2005) Privacy-preserving set operations. In: Shoup V (ed) *Advances in cryptology: CRYPTO 2005*. Springer, Berlin, pp 241–257
- Lambæk M (2016) Breaking and fixing private set intersection protocols. *IACR Cryptol ePrint Arch* 2016:665
- Le PH, Ranellucci S, Gordon SD (2019) Two-party private set intersection with an untrusted third party. In: *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (CCS '19)*; pp. 2403–2420
- Menezes AJ, Van Oorschot PC, Vanstone SA (1996) *Handbook of applied cryptography*. CRC Press, Boca Raton
- Miao P, Patel S, Raykova M, Seth K, Yung M (2020) Two-sided malicious security for private intersection-sum with cardinality. In: Micciancio D, Ristenpart T (eds) *Advances in cryptology: CRYPTO 2020*. Springer, Berlin, pp 3–33
- Miyaji A, Nakasho K, Nishida S (2017) Privacy-preserving integration of medical data: a practical multiparty private set intersection. *J Med Syst* 41(3):37
- National Institute of Standards and Technology, Recommendation for Key Management Part 1: General (Revision 4), National institute of standards and technology special publication 800-57, January 2016
- Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed) *Advances in cryptology: EUROCRYPT '99*. Springer, Berlin, pp 223–238
- Pinkas B, Schneider T, Zohner M (2014) Faster private set intersection based on OT extension. In: Fu K, Jung J (eds) *Proceedings of the 23rd USENIX security symposium (USENIX Security 2014)*; pp. 797–812

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.