Multi-Party Privacy-Preserving Set Intersection with Quasi-Linear Complexity

Jung Hee Cheon[1], Stanislaw Jarecki[2], Jae Hong Seo[1]

[1]Department of Mathematical Sciences, Seoul National University,
{jhcheon,jhsbhs0}@snu.ac.kr
[2]Department of Computer Science, University of California, Irvine
stasio@ics.uci.edu

Abstract. Secure computation of the set intersection functionality allows n parties to find the intersection between their datasets without revealing anything else about them. An efficient protocol for such task could have multiple potential applications, in commerce, health-care, and security. However, all currently known secure set intersection protocols for $n > 2$ parties have computational costs that are quadratic in the (maximum) number of entries in the dataset contributed by each party, rendering secure computation of set intersection impractical on anything but small datasets.

In this paper we describe the first multi-party protocol for securely computing the set intersection functionality with both the communication and the computation costs that are quasi-linear in the size of the datasets. Specifically, our protocols require $O(n^2k\lambda)$ bits of communication and $\tilde{O}(n^2\lambda + (n\lambda + n^2)k)$ group multiplications per player in the malicious adversary setting, where k is the size of each dataset and λ is security parameter. Our protocol follows the basic idea of the protocol proposed by Kissner and Song [16], but we gain efficiency by using different representation of the polynomials associated with users' datasets, and careful employment of algorithms that interpolate or evaluate polynomials on multiple points more efficiently.

Keywords: multi-party set intersection, privacy-preserving set operation

1 Introduction

Privacy-Preserving Set Intersection (PPSI) is one of the most interesting and useful invention of multi-party computations. PPSI protocol enables each user to broadcast her encoded private dataset and to obtain only intersection of all users' private sets at the end of protocol. It can be used when several commercial companies want to share intersection of their customer lists while their own list except the set intersection is protected.

There are general multi-party computations that implement secure computation for any function to be able to express as a circuit [26, 12], so that they can be solutions for PPSI. In general, however, the overall overheads is often unacceptable since the complexity is depending on the size of circuit and the protocol requires expensive operations such as several oblivious transfers. Especially, the complexity of general multi-party computation is more higher in the presence of the malicious adversary (who can do anything arbitrarily) than the honest-but-curious adversary (who follows protocol's directions correctly).

If we restrict the number of protocol participants to two, there are several improvements for PPSI to attain better efficiency than general multi-party computation. From the work of Freedman, Nissim, and Pinkas [8], recently Hazay and Nissim [14] proposed protocols efficiently computing set intersection between two players. The protocol having the best performance in terms of complexity in the input size (that is, it's communication and computation overhead are linear in input size.) was proposed by Hazay and Lindell [13], next Jarecki and Liu [15] complemented its provable security against the malicious adversary.

In the case for multiple participants more than two (we simply call the multi-party case), the improvement with respect to the complexity has gradually be done, compared with the two-party case. Kissner and Song [16, 17] first addressed PPSI in the multi-party case using outstanding techniques basing on the work by Freedman et al. Recently Sang and Shen [21, 22] advanced Kissner and Song's result by reducing complexity in terms of the number of players. In contrast with the two-party case, however, all proposed protocols for PPSI in the multi-party case have quadratic complexity in the size of private dataset, and finding a efficient solution for PPSI with linear complexity in the input size in the multi-party case remains still open (even in the honest-but-curious adversary model).

In multi-party cryptography against the malicious adversary, the robustness has been considered as an important property [3, 7]. When a corrupted player's malicious behavior is detected, a non-robust protocol will stop, and then the protocol should restart excluding a corrupted player. On the other hand, a robust protocol can output the desired result without restarting even if some subset of corrupted players' malicious behaviors are detected.

In this paper our contributions are two folds. First, we present a PPSI protocol and show that each player in our PPSI protocol requires to transfer $O(n^2k)$ encryptions and to compute $\tilde{O}(n^2\lambda+(n\lambda+n^2)k)$ group multiplications. (k: the size of user's dataset, n: the number of users, λ: security parameter) Especially, if we focus on the online computation (that is, computations except the local computations used for input/output data representation conversions), each player is required to compute $O(n^2\lambda+(n\lambda+n^2)k)$ group multiplications. Second, we prove that the proposed PPSI protocol is robust in the presence of malicious adversary. That is, even if corrupted players leave before the end of protocol, our protocol outputs set intersection among players who proved that their input datasets are well-formed.

Technical Overview Let us explain key ideas of our construction. When one employs a polynomial to represent her private dataset, multiplications over encrypted polynomials and their zero-knowledge proofs are the most expensive in the system. In particular, all previous works dealing with PPSI in the multi-party setting by using polynomials to represent datasets consider a polynomial as a tuple of its coefficients, so that a multiplication over encrypted polynomials impose quadratic computational and communicational complexities in the degree of polynomials [16, 17, 20–22].

Our approach to reduce the complexity is to represent a polynomial $f(x)$ by several points on the curve $y = f(x)$. This gives us two benefits: one is that polynomial multiplications can be efficiently performed and the other is that it allows simple zero-knowledge proofs for multiplications over encrypted polynomials. For the first part, if we write polynomials $f(x)$ and $g(x)$ by $\{(s, f(s))\}_{s\in S}$ and $\{(s, g(s))\}_{s\in S}$, respectively for an index set S, a multiplication of two polynomials is simply done by $\{(s, f(s) \cdot g(s))\}_{s\in S}$. Hence the computational complexity of the protocol except the input/output data representation conversions becomes linear. Because of the simple computation for a polynomial multiplication, we can construct a simple zero-knowledge proof for a multiplication over encrypted polynomials. It impose only linear computation and communication overheads to the protocol.

In the proposed protocol, each user is required to convert her private dataset into a suggested form (a set of points on the curve $y = f(x)$) at the beginning of the protocol, and to convert the suggested form into a set at the end of the protocol. We call such conversions by Input Conversion and Output Conversion, respectively. These conversions use algorithms for polynomial interpolation and polynomial evaluation, and they require quadratic operations in input size k. In general, the cost for polynomial arithmetics in the underlying field will

be dominated by cryptographic operations such as encryptions, exponentiations, etc. In our protocol, however, each player should perform polynomial arithmetics such as evaluation and interpolation when coefficients or values are given in the exponent. For example, given a set of group elements $\{g^{a_0}, \cdots, g^{a_k}\}$ and a field element α one should compute $g^{f(\alpha)}$ where $f(x) = \sum_{i=0}^{k} a_i x^i$ and g is a group generator. In this case, for polynomial arithmetics group multiplications and group exponentiations should be used instead of field additions and scalar multiplications. Therefor the cost for polynomial arithmetics in our protocol is as significant as that of cryptographic operations. We will deal with a way how to speed up these conversions, and show that the way is appropriate to our protocol.

The goal of robust PPSI protocol is to compute set intersection among players who proved their well-formed input (i.e. a set of pairs of a point and an encrypted value $\{(s, c_s)\}_{s \in S}$ satisfies that for each $s \in S$ an equality $c_s = \mathsf{Enc}_{pk}(h(s))$ holds for some k-degree polynomial $h(x)$ where an index set S is given by the protocol.) even when some subset of corrupted players leave protocol after proving their well-formed input. One may think that this problem is easily solved by using Verifiable Secret Sharing (VSS) scheme, such as Pedersen VSS (Ped-VSS) [19]. It is, however, difficult to deal with encryptions that are shared by VSS. More precisely, it is uneasy to construct zero-knowledge proofs on encrypted messages that are shared by VSS before recovering shared encryption. For example, let c be an encryption of m, and suppose that a dealer wish to share c by using Ped-VSS. Then, the dealer should encode c as an element of some field $\mathbb{F}_q$ of size q and commit $G^c H^d$ where G, H are elements in some group of order q and d is chosen at random at random from $\mathbb{F}_q$. If we used an additive homomorphic encryption to encrypt m for c, then usually the message is located in the exponent of encryption. (For example, messages are located in exponents of encryptions if we use Paillier encryption [18] and the modified ElGamal encryption which is explained in the section 2.) Then, m in the shared encryption should be located in the exponent of exponent of committed value $G^c H^d$. To the best of our knowledge, there is no known zero-knowledge proofs for the relation associated with such m, e.g. zero-knowledge proof for knowledge of such m.

We construct a robust protocol in the presence of the malicious adversary by proposing simple VSS scheme which allows to share encryptions and supports zero-knowledge proofs on shared encryptions. To share an encryption $\mathsf{Enc}_{pk}(m)$ the dealer can broadcast $\mathsf{Com}'_{ck}(c)$ and $\mathsf{Enc}_{pk}(m + c)$ where Com'_{ck} is an additive homomorphic trapdoor commitment, Enc_{pk} is an additive homomorphic encryption, and c is a random integer. Thereafter, the dealer verifiably secret-shares c by using VSS. To recover $\mathsf{Enc}_{pk}(m)$, all players recover c and next compute $\mathsf{Enc}_{pk}(m+c) \oplus \mathsf{Enc}_{pk}(-c)$ where $\oplus$ denotes an add operation between corresponding plaintexts. Since both Com'_{ck} and Enc_{pk} have the additive homomorphic property, we can efficiently construct zero-knowledge proofs for the statement associated with m, e.g. zero-knowledge proof for knowledge of m. Furthermore, in the security proof, the simulator can equivocate on m as it can equivocate on c if it has a trapdoor of Com'_{ck}. Therefore, we can construct a robust PPSI protocol by using such a simple VSS.

It is worth to note that our PPSI protocol uses a so-called 'common reference string' since we use Pedersen commitment scheme as a building block to design PPSI protocol, and we assume that there exists a broadcast channel.

Related Works of PPSI Protocols Except the general multi-party computation approach [12], Kissner and Song proposed the first efficient privacy-preserving set intersection protocol secure in the honest-but-curious (HBC) adversary model [16] and extended to be

secure in the malicious adversary model using general zero-knowledge proof techniques [17]. Sang and Shen [20–22] proposed protocols having better efficiency than [17]. However, they only reduced the complexity in terms of the number of players, and their protocols still keep quadratic complexity in the input size.

In the two-party case of PPSI, there are several notable researches. Freedman, Nissim, and Pinkas proposed the first asymmetric two-party set intersection protocol using oblivious polynomial evaluation [8]. However, their schemes are only secure against malicious client, or secure against malicious server (not both). Dachman-Soled et al. improved security so that proposed a protocol secure against both malicious parities [5]. Hazay and Nissim advanced complexity so that they attains almost linear complexity in the input size [14]. Camenisch and Zaverucha considered the case that no malicious adversaries can choose arbitrary their private inputs [2]. Using an oblivious pseudorandom function, Hazay and Lindell [13] provided two-party protocols which are secure in the standard model against covert parties [1]. Jarecki and Liu [15] improved its security so that secure in the standard model against both malicious parties. Cristofaro and Tsudik [4] presented two-party protocols with linear complexity by extending privacy-preserving information transfer, and they also proposed the authorized set intersection protocol which allows the set intersection only for authorized private set.

Outline We give definitions and tools of cryptography. In Section 3 we propose a new representation, call point representation, to describe private datasets, and possible operations over point representation. We present two protocols, each for different adversary models, so-called Honest-But-Curious adversary and the malicious adversary, in Section 4. In Section 5 we show how to speed up local computations of protocols, and analyze the complexities of the proposed protocol.

2 Definitions and Primitives

Notations Throughout this paper we will use several notations. Let n be the number of players participating in the protocol, k be the number of entries in each player,[1] $\mathbb{Z}_p$ be a finite field with prime p of size 2λ, $\mathbb{Z}_p[x]$ be a polynomial ring over $\mathbb{Z}_p$, $\mathbb{Z}_p^d[x]$ be a set of all polynomials of degree at most d in $\mathbb{Z}_p[x]$. For a set A, $a \xleftarrow{\$} A$ denotes a is uniformly chosen at random from A. $[a, b]$ denotes a set $\{\alpha \in \mathbb{Z} | a \leq \alpha \leq b\}$.

Adversary Model In this paper we are interested in the malicious adversary model rather than the Honest-But-Curious (HBC) adversary model, but we present a HBC protocol as an intermediate protocol to obtain a malicious one. In HBC adversary model, the adversary follows protocol's prescribed directions and he can utilize all information obtained during the protocol procedure for purpose to gain other information than the protocol's desired result, set intersection in PPSI protocol. In contrast to HBC adversary model, malicious adversary may corrupted some subset of players less than half and gain full controls over the corrupted players. She may not follow the given prescribed actions of protocol and may behave arbitrarily. We cannot prevent refusing of malicious adversary to participate in the protocol, running the protocol along with arbitrary values as his private input, or prematurely aborting the protocol.

[1] We assume that all players have same size private dataset. If each player needs, then he can add dummy inputs having invalid encoding, e.g. elements not in $\mathtt{P}$ which is the domain of private data.

Informally, we say an PPSI protocol is secure in the presence of malicious adversary if for any malicious adversary fully controlling a set of colluders in the real world, there exists a probabilistic polynomial running-time algorithm that translates all strategies of a malicious adversary in the real world to the strategies in the the ideal world, and follows in the ideal world. Furthermore, the real execution is computationally indistinguishable from the execution in the ideal world to the malicious adversary. We refer to [11] for the formal definitions for the HBC adversary model and the malicious adversary model.

Additive Homomorphic Encryption Our construction utilizes an additive homomorphic encryption scheme. In particular we describe our protocol by using the modified encryption scheme widely used in the set intersection protocol [8, 2, 5].

The modified ElGamal encryption consists of three algorithms.

• $\mathsf{Setup}(\lambda)$: Group generating algorithm takes the security parameter λ and generates group description $(\mathbb{G}, g, 1_{\mathbb{G}}, p)$ where g is a generator of a cyclic group $\mathbb{G}$ of prime order p, and $1_{\mathbb{G}}$ is the identity element in $\mathbb{G}$. Choose a random integer x from $\mathbb{Z}_p$ and set $pk \leftarrow \{g^x\}$, $sk \leftarrow \{x\}$.

• $\mathsf{Enc}_{pk}(m; r)$: $CT \leftarrow (g^r, pk^r g^m)$.

• $\mathsf{Dec}_{sk}(CT)$: Parse CT as (C_0, C_1) and output $C_1 \cdot C_0^{-sk}$.

This modified ElGamal encryption is additively homomorphic on message, and randomness as well:

$$\mathsf{Enc}_{pk}(m_1; r_1) \oplus \mathsf{Enc}_{pk}(m_2; r_2) = \mathsf{Enc}_{pk}(m_1 + M_2 \bmod p; r_1 + r_2 \bmod p)$$

where $\oplus$ operation denotes component-wise group multiplications. To denote additions of several ciphertexts CT_i for $i \in [1, \ell]$, we use a notation $\oplus_{i\in[1,\ell]} CT_i$. A scalar multiplication is also easily allowed by repeatedly computing $\oplus$ operation, that is, given a scalar $c \in \mathbb{Z}_p$,

$$\underbrace{\mathsf{Enc}_{pk}(m; r) \oplus \cdots \oplus \mathsf{Enc}_{pk}(m; r)}_{c \text{ times}} = \mathsf{Enc}_{pk}(cm \bmod p; cr \bmod p)$$

For a ciphertext CT and an integer c we simply denote a scalar multiplication of CT by c as CT^c. Since Dec algorithm outputs g^m instead of the plaintext m, it does not support a complete decryption. However, its the additive homomorphic property still allows to enjoy our PPSI protocol without a complete decryption algorithm. Further, the modified ElGamal encryption allows that ciphertexts can be re-randomize without the secret key.

Distributed Key Generation and Threshold Decryption We use the threshold version of the modified ElGamal encryption for which there exists efficient distributed key generation and the threshold decryption protocols which are secure in the malicious adversary setting. We take the distributed key generation protocol, denoted DKG, from [10]. For the threshold decryption protocol, denoted TDec, we use the protocol of [6], amended by the use of a zero-knowledge proof of correctness of partial decryption. We use the distributed version of this zero-knowledge proof which is suitable to the multi-party computation setting, as we explain in Section 4.2. For completeness we note that the proof of correctness of partial decryption is a proof of discrete-logarithm equality, which is actually a simplification of the proof HVZKPK-P we explain in that section.

Verifiable Secret Sharing We use the verifiable secret sharing (VSS) for two reasons, one for distributed zero-knowledge proofs in the section 4.2 and the other for the robustness in the presence of the malicious adversary. Specifically, we use Pedersen VSS (Ped-VSS) [19].

3 Representation for Efficient Private Data Manipulation

Let $\mathtt{X_i} \subset \mathbb{Z}_p$ be the set of private inputs of a player $\mathcal{P}_i$. We associate the set $\mathtt{X_i}$ with a polynomial $f_i(x) = \prod_{a_j \in \mathtt{X_i}}(x - a_j)$, called the private polynomial of $\mathcal{P}_i$.

All previous protocol [8, 16, 2, 5] that use polynomials to encode private datasets utilize the coefficient representation: a set of coefficients of the private polynomial. When we adopt the coefficient representation, a multiplication between two polynomials $f(x) = \sum_{i=0}^{k} a_i x^i$ and $g(x) = \sum_{i=0}^{k} b_i x^i$ is given by

$$f(x) \cdot g(x) = \sum_{\ell=0}^{2k} \sum_{i+j=\ell} a_i b_j x^\ell.$$

This makes the computational complexity be quadratic in k. Furthermore, the zero-knowledge proof proving this computation makes the communication complexity be quadratic in k, too.

To achieve linear complexity in k, we propose a new representation for private datasets, called Point Representation (PR).

Point Representation Given a set $\mathtt{X}$, and a public index set S of size ℓ, we define PR of $\mathtt{X}$ by the result of conversion $\mathsf{Conv_{StoP}}$ on input $\mathtt{X}$ and S where $\mathsf{Conv_{StoP}}$ is defined as follows: To compute $\mathsf{Conv_{StoP}}$, we construct a polynomial f having $\mathtt{X}$ as a set of roots, and next we evaluate f at all elements in S as $\{f(s)\}_{s\in S}$. Then, the resulting a set of pairs of a point in S and a corresponding value on f, $\{(s, f(s))\}_{s\in S}$ is PR of $\mathtt{X}$ with S. We denote this conversion by

$$\begin{aligned} \mathsf{Conv_{StoP}} : \mathbb{Z}_p^k \times \mathbb{Z}_p^\ell &\to \mathbb{Z}_p^{2\ell} \\ (\mathtt{X}, S) &\mapsto \{(s, f(s))\}_{s\in S} \end{aligned}$$

where $k < \ell$.

Before considering about the conversion that is a return from PR to a set, let us consider the last format of interactions in our protocol that will be described in the next section. We will use the modified ElGamal encryption which does not support a complete decryption. It causes each participants in the protocol to obtain PR of some polynomial (precisely, the intersection polynomial explained below) in the exponent instead of PR itself at the end of interactions among players in the protocol. In other words, $\{s, g^{f(s)}\}_{s\in S}$ is given instead PR $\{s, f(s)\}_{s\in S}$ for some polynomial $f(x)$ where g is a generator of some cyclic group. Therefore we need to convert PR into a set when PR is given in the exponent.

Now we explain how to convert into a set from given PR in the exponent $\{(s, g^{f_s})\}_{s\in S}$ where g is a generator of a cyclic group $\mathbb{G}$. We can find $\{g^{a_i}\}_{i\in[0,\ell-1]}$ by using polynomial interpolation where $f(x) = \sum_{i=0}^{\ell-1} a_i x^i$ and $f(s) = f_s$. Since polynomial interpolation consists of additions and scalar multiplications, we can apply polynomial interpolation even when PR is in the exponent. Then, we make an effort to extract all $f(x)$'s roots in $\mathbb{Z}_p$ such that all roots are in $\mathtt{P}$, where $\mathtt{P}$ is a domain of private data and exactly defined below. If we have a set A containing all roots of f, then we can decide whether each element $a \in A$ is a root of f by evaluating $g^{f(a)}$. If $g^{f(a)} = 1_\mathbb{G}$ where $1_\mathbb{G}$ is the identity element of $\mathbb{G}$, then a is a root of f. Otherwise, a is not a root of f. We know that such a set A exists since at least $\mathbb{Z}_p$ can be A. If we use, however, $\mathbb{Z}_p$ as A, computing $\mathsf{Conv_{PtoS}}$ will be very inefficient. (That is, it takes

exponential time in the security parameter.) In our PPSI protocol we will use a smaller set of size $k < \ell$ than $\mathbb{Z}_p$ as A. We denote this conversion by

$$\begin{aligned}\mathsf{Conv}_{\mathsf{PtoS}} : (\mathbb{Z}_p \times \mathbb{G})^{\ell} &\rightarrow \mathbb{Z}_p^{\ell-1} \\ \{(s, g^{f(s)})\}_{s\in S} &\mapsto \{x \in \mathtt{P} | f(x) = 0\}\end{aligned}$$

To reduce a chance that a random element belongs to a private dataset accidentally in the midst of the protocol, private data is restricted to choose from a subset of $\mathbb{Z}_p$, denoted by $\mathtt{P}$, which has a special encoding. For instance, we can define $\mathtt{P}$ a set of the form $a\|0^{\lambda}$ where λ is the security parameter. We usually set the size of p is equal to 2λ, so thus the chance that a random element hits $\mathtt{P}$ is negligible in λ.

This restriction of $\mathtt{P}$ gives some advantage: Let f_i be a private polynomial of $\mathcal{P}_i$ and r_i be a random polynomial for each i. Then, a set of all roots of the polynomial $\sum_i f_i \cdot r_i$, called the intersection polynomial, is equal to intersection of $\mathcal{P}_i$ with overwhelming probability in λ. Lemma 1 prove above and show that $\sum_i f_i \cdot r_i$ leaks no information except for set intersection.

Lemma 1 ([16]) *Let $f, h \in \mathbb{Z}_p^{\alpha}[x]$ and $\gcd(f, h) = 1$, and $r, s \xleftarrow{\$} \mathbb{Z}_p^{\beta}[x]$ where $\beta \geq \alpha$. Then, $f \cdot r + h \cdot s = \gcd(f, h) \cdot u$ where u uniformly distributes in $\mathbb{Z}_p^{\alpha+\beta}[x]$.*[2]

Arithmetic Operations on Point Representation If PRs of two k-degree polynomials f and h, $\{(s, f(s))\}_{s\in S}$ and $\{(s, h(s))\}_{s\in S}$ respectively, are given, an addition and a multiplication of two polynomials are simply done by $\{(s, f(s) + h(s))\}_{s\in S}$ and $\{(s, f(s) \cdot h(s))\}_{s\in S}$, respectively where $|S| \geq 2k+1$. They require $|S|$ operations which is linear in degree k. This is our key observation to reduce complexity from quadratic to linear during Online phase. (Online phase contains all processes in the protocol except the local computation for input/output data conversions and distributed key generations.) Moreover, to prove this linear operations, zero-knowledge proof also requires only linear size communications and computations.

Arithmetic Operations on Encrypted Polynomials Let $\mathsf{Enc}_{pk}(\cdot)$ be an additive homomorphic encryption with a public key pk, and S be a public index set for PR. When f_i is the private polynomial of $\mathcal{P}_i$, we denote the encryption of PR of f_i by $\mathsf{Enc}_{pk}^S(f_i)$ that is a set of pairs $\{(s, \mathsf{Enc}_{pk}(f_i(s)))\}_{s\in S}$. We define two operations over encrypted PR:
(1) $\boxed{\oplus_{i\in[1,n]}\mathsf{Enc}_{pk}^S(f_i)}$: Given encrypted PRs $\{\mathsf{Enc}_{pk}^S(f_i)\}_{i\in[1,n]}$, we can compute the sum of encrypted PRs as $\{(s, \oplus_{i\in[1,n]}\mathsf{Enc}_{pk}(f_i(s)))\}_{s\in S} = \{(s, \mathsf{Enc}_{pk}(\sum_{i\in[1,n]} f_i(s)))\}_{s\in S}$ and denote it by $\oplus_{i\in[1,n]}\mathsf{Enc}_{pk}^S(f_i)$
(2) $\boxed{h \otimes \mathsf{Enc}_{pk}^S(f)}$: Given an unencrypted PR $\{(s, h(s))\}_{s\in S}$ and an encrypted PR $\mathsf{Enc}_{pk}^S(f)$, we can compute the product as $\{(s, \mathsf{Enc}_{pk}(f(s))^{h(s)})\}_{s\in S} = \{(s, \mathsf{Enc}_{pk}(h(s) \cdot f(s)))\}_{s\in S}$ and denote it by $h \otimes \mathsf{Enc}_{pk}^S(f)$.

4 Application to PPSI Protocols

Point representation is useful to design PPSI protocols. In this section we present two protocols to perform intersection of players' private sets with preserving each player's privacy.

[2] The original lemma in [16] states for arbitrary ring satisfying some conditions that $\mathbb{Z}_p$ satisfies. In this paper we use a field $\mathbb{Z}_p$, so we just state the lemma for the case of $\mathbb{Z}_p$.

Public Parameters: The number of players n, maximum threshold of corrupted players t satisfying $2t+1 \leq n$, maximum size of each dataset k, set S of size at least $2k+1$, and descriptions of Enc_{pk} (the modified ElGamal), $\mathbb{G}$ (a base group), and $\mathtt{P} \subset \mathbb{Z}_p$ (the encoding for private inputs).
Private Input of Player $\mathcal{P}_i$: A set $\mathtt{X_i}$ of k values in $\mathtt{P}$.

(Initialization). All players perform the DKG protocol to generate a public key pk of Enc_{pk}. Private output of player $\mathcal{P}_i$ is denoted sk_i.
(Input Data Conversion). Each player $\mathcal{P}_i$ carries out local computations.

(1). Compute $\mathsf{Conv}_{\mathsf{StoP}}(\mathtt{X_i}, S) = \{(s, f_i(s))\}_{s\in S}$ and encrypt them to $\mathsf{Enc}^S_{pk}(f_i)$.
(2). Choose random k-degree polynomials $\{r_{i\ell}(x) \in \mathbb{Z}_p[x]\}_{\ell\in[1,n]}$, encode them to $\{\{r_{i\ell}(s)\}_{s\in S}\}_{\ell\in[1,n]}$.

(Online phase). Each player $\mathcal{P}_i$ carries out computing an encryption of the intersection polynomial $I(x) = \sum_{i,\ell\in[1,n]} r_{i\ell} f_\ell$, and then performs the threshold decryption protocol.

(1). Send $\mathsf{Enc}^S_{pk}(f_i)$ to all players.
(2). For each $\ell \in [1,n]$, compute $C_{i\ell} = r_{i\ell} \otimes \mathsf{Enc}^S_{pk}(f_\ell)$ and broadcast them to all players.
(3). Compute $C = \oplus_{i,\ell} C_{i\ell} = \{(s, \mathsf{Enc}_{pk}(\sum_{i,\ell\in[1,n]} r_{i\ell} f_\ell(s)))\}_{s\in S}$.
(4). Perform the threshold decryption protocol TDec on each ciphertext in C to obtain a set of values $\{(s, g^{I(s)})\}_{s\in S}$.

(Output Data Conversion). Each player $\mathcal{P}_i$ computes $\mathsf{Conv}_{\mathsf{PtoS}}(\{(s, g^{I(s)})\}_{s\in S})$.

Fig. 1. PPSI-HBC protocol against HBC adversary

4.1 Construction in HBC Adversary Model

We give a PPSI-HBC protocol secure against HBC adversary in the figure 1.

Look into Output Data Conversion: After finishing Online phase each player $\mathcal{P}_i$ has $\{(s, g^{I(s)})\}_{s\in S}$ where $I(x)$ is the intersection polynomial of degree $2k$. $\mathsf{Conv}_{\mathsf{PtoS}}$ should output all roots in $I(x)$ which are contained in $\mathtt{P}$. Therefore, to extract all roots of $I(x)$ which are contained $\mathtt{P}$, first $\mathcal{P}_i$ performs polynomial interpolation to recover $\{g^{a_i}\}_{i\in[0,2k]}$ where $I(x) = \sum_{i\in[0,2k]} a_i x^i$, second he carries out multiple evaluations $g^{I(\alpha)}$ at all elements $\alpha \in \mathtt{X_i}$, and then concludes for $\forall \alpha \in \mathtt{X_i}$,

$$\alpha \in \begin{cases} \cap_{j\in[1,n]} \mathtt{X}_j & \text{if } g^{I(\alpha)} = 1_{\mathbb{G}} \\ (\cap_{j\in[1,n]} \mathtt{X}_j)^c & \text{otherwise} \end{cases}$$

Correctness: Lemma 1 says $I(x) = \sum_{i,\ell\in[1,n]} r_{i\ell} \cdot f_\ell$ is equal to $\gcd(f_1, \cdots, f_n) \cdot u$ for some random polynomial $u(x)$. Each root of u will be included in $\mathtt{P}$ with only negligible probability in λ, and hence a set of all roots of $I(x)$ that are contained in $\mathtt{P}$ is equal to $\cap_{j\in[1,n]} \mathtt{X}_j$ with overwhelming probability in λ. Since $\cap_{j\in[1,n]} \mathtt{X}_j \subset \mathtt{X_i}$ for $\forall \mathtt{i}$, each player $\mathcal{P}_i$ can find set intersection in Output Data Conversion phase with overwhelming probability in λ.

Security: As we use a semantically secure encryption Enc_{pk} and all private inputs are encrypted by Enc_{pk}, no player can obtain non-trivial information about other player's private inputs during the protocol PPSI-HBC with non-negligible probability. The intersection polynomial $\sum_{i,\ell\in[1,n]} r_{i\ell} \cdot f_\ell$ leaks no information except for set intersection. This comes from Lemma 1 and the fact that for $\forall \ell$, $\sum_{i\in[1,n]} r_{i\ell}$ is indistinguishable from a random polynomial in $\mathbb{Z}^k_p[x]$ in the point of view of a coalition of HBC adversaries less than n. Therefore no HBC adversary can obtain any other information than the intersection with more than negligible probability.

4.2 Distributed Zero-Knowledge Proofs

To adapt PPSI-HBC protocol in HBC model to the malicious adversary model we need zero-knowledge proofs for the following statements involving polynomials encrypted by an additive homomorphic encryption Enc_{pk}:

- $\mathsf{P}[\mathsf{Enc}^S_{pk}(f_1), \mathsf{Enc}^S_{pk}(f_2), \mathsf{Enc}^S_{pk}(f_3)]$ holds if $\mathsf{Enc}^S_{pk}(f_3) = f_1 \otimes \mathsf{Enc}^S_{pk}(f_2)$.
- $\mathsf{D}[\mathsf{Enc}^S_{pk}(f)]$ holds if $f \in \mathbb{Z}^k_p[x]$.
- $\mathsf{DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f))]$ holds if $f \in \mathbb{Z}^k_p[x]$ and $f(1) = 1$ (i.e.$f \neq 0$)

where $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f))$ is a commitment to an encrypted polynomial $\mathsf{Enc}^S_{pk}(f)$ defined as follows:

$$\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f)) = \{(s, \mathsf{Com}'_{ck}(c_s), \mathsf{Enc}_{pk}(f(s) + c_s))\}_{s \in S} \text{ for } c_s \xleftarrow{\$} \mathbb{Z}_p$$

where Com'_{ck} is a trapdoor commitment, ck is public parameter of Com_{ck}, and c_s is a random integer. We would use Pedersen commitment scheme [19] as Com'_{ck} in this paper. To open committed encryption, for $\forall s \in S$ open c_s, then the receiver can recover $\mathsf{Enc}^S_{pk}(f)$ by computing $\mathsf{Enc}_{pk}(f(s) + c_s) \oplus \mathsf{Enc}_{pk}(-c_s; 0)$. If Com'_{ck} is perfectly hiding, then Com_{ck} also perfectly hides an encrypted polynomial, and if Com'_{ck} is computationally binding, then Com is, too.[3] Moreover, if Com' is an additive homomorphic commitment, then Com is, too as Enc_{pk} is an additive homomorphic encryption.

For each statement we construct a so-called sigma protocol, i.e. a 3-round public-coin interactive proof with special honest-verifier zero-knowledge and special strong soundness. We denote these sigma protocols as, respectively, HVZKPK-P, HVZKPK-D, HVZKPK-DO. We relegate the descriptions of these sigma protocols to Appendix A since these proof systems are simple extensions of well-known sigma protocols for proving knowledge of discrete logarithms, representations, and arithmetic relations between these representations. For example the sigma protocol for $\mathsf{HVZKPK\text{-}P}[\mathsf{Enc}_{pk}(f_1), \mathsf{Enc}_{pk}(f_2), \mathsf{Enc}_{pk}(f_3)]$ is a conjunction of $|S|$ instances, one for each $s \in S$, of a sigma protocol for proving knowledge of plaintexts $f_1(s), f_2(s), f_3(s)$ encrypted in $\mathsf{Enc}_{pk}(f_1(s)), \mathsf{Enc}_{pk}(f_2(s)), \mathsf{Enc}_{pk}(f_3(s))$, which satisfy relation $f_1(s) \cdot f_2(s)$ is equal to $f_3(s)$ as an element in $\mathbb{Z}_p$, the plaintext domain.

We use a generic conversion from a sigma protocol to a 'distributed zero-knowledge proof' protocol for the same statement: First, the prover broadcasts the initial message of the sigma protocol. Then the challenge is generated by a distributed coin-toss, i.e. each player in parallel verifiably secret-shares a random value in $\mathbb{Z}_p$, each sharing is then reconstructed in parallel, and the challenge is the sum of the shared values. (Moreover, using the VSS of Pedersen [19] homomorphic on $\mathbb{Z}_p$ one can first add all the VSS instances and reconstruct the result.) Finally, the prover broadcasts its response to this challenge, and each player verifies the proof. We denote the distributed proofs for the above statements as D-ZKPK-P, D-ZKPK-D, D-ZKPK-DO. It is easy to see that the distributed proof protocol has an efficient straight-line simulator (the simulator who controls more players than the half of protocol participants can steer the distributed coin-toss to hit a random challenge of its choice, and therefore it can use the special HVZK simulation of the underlying sigma protocol) and an efficient extraction of adversary's witnesses (the simulator can run a second execution on the side which produces different challenges then the main execution path, and sigma protocol guarantees witness extraction from successful responses to two different challenges).

[3] Com_{ck} does not hide the randomness used in Enc_{pk}, but, the randomness does contain no information about f. Therefore, Com_{ck} perfectly hides f and we need only this property.

4.3 Construction in Malicious Adversary Model

We give a protocol PPSI-MAL securely computing multi-party set intersection in the presence of malicious adversary in the figure 2. PPSI-MAL is based on PPSI-HBC presented in Section 4, with the following essential changes made: We add distributed protocols for zero-knowledge proofs of statements $\mathsf{DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i))]$, $\mathsf{D}[\mathsf{Enc}^S_{pk}(r_{i\ell})]$, and $\mathsf{P}[\mathsf{Enc}^S_{pk}(r_{i\ell}), \mathsf{Enc}^S_{pk}(f_\ell), \mathsf{Enc}^S_{pk}(r_{i\ell} \cdot f_\ell)]$, we rely on the fact that the distributed key generation and the threshold decryption protocols DKG and TDec of [10, 6] are secure in the malicious model, and furthermore, to prevent players from setting their private polynomials to zero polynomials, we add a constraint that every private polynomial must pass point $(1, 1)$. Namely, every player $\mathcal{P}_i$, given its private set $\mathtt{X_i}$, constructs its private polynomial as

$$f_i(x) = (\prod_{a_t \in \mathtt{X_i}} (x - a_t)(1 - a_t)^{-1})$$

It is easy to see that $f_i(1) = 1$. Since the zero-knowledge proofs of $\mathsf{DO}[\mathsf{Enc}_{pk}(f_i)]$ ensures that $f_i(1) = 1$ (and that the degree of f_i is at most k), it follows that f_i cannot be equal a zero polynomial.

Furthermore, PPSI-MAL has the robust property. That is, even if corrupted players behave maliciously, our protocol outputs set intersection among players who proved that their input data is well-formed ($\mathsf{DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i))]$). Each player makes an encrypted private polynomial and next verifiably secret-shares his encrypted private polynomial, and then proves shared encryption is well-formed, i.e., runs $\mathsf{DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i))]$. More precisely, to share an encrypted polynomial $\mathsf{Enc}^S_{pk}(f_i)$ for private polynomial f_i of player $\mathcal{P}_i$, $\mathcal{P}_i$ commits to $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f)) = \{(s, \mathsf{Com}'_{ck}(c_s), \mathsf{Enc}_{pk}(f(s) + c_s; r_s))\}_{s \in S}$, and verifiably secret-share $\{c_s\}_{s \in S}$ by using Ped-VSS. Then, zero-knowledge proofs for the statement that f is well-formed can be easily done by $\mathsf{D\text{-}ZKPK\text{-}DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f))]$. If a corrupted player $\mathcal{P}_i$ leaves after passing $\mathsf{D\text{-}ZKPK\text{-}DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i))]$, then remaining players can recover $\mathsf{Enc}^S_{pk}(f_i)$ by recovering $\{c_s\}_{s \in S}$ and then computing $\{\mathsf{Enc}_{pk}(f(s) + c_s; r_s) \oplus \mathsf{Enc}_{pk}(-c_s; 0)\}_{s \in S}$. Therefore our protocol is robust against corrupted players' leaving.

Correctness: Since $\mathsf{Enc}^S_{pk}(\sum_{i \in R, \ell \in M} r_{i\ell} \cdot f_\ell) = \{(s, \mathsf{Enc}_{pk}(\sum_{i \in R, \ell \in M}(r_{i\ell} \cdot f_\ell)(s)))\}_{s \in S}$ and $|S|$ is larger than degree of polynomial $2k$, it is an encryption of valid PR of a polynomial $\sum_{i \in R, \ell \in M} r_{i\ell} \cdot f_\ell$. M is a set of indices of all players being proved their well-formed input and R is a set of indices of all players being passed all zero-knowledge proofs protocols. For $\forall \ell \in M$, $\sum_{i \in R} r_{i\ell}$ is a random polynomial in the view of corrupted players since we assume that honest players are majority, and hence by Lemma 1 $\sum_{i \in R, \ell \in M} r_{i\ell} \cdot f_\ell$ is the intersection polynomial of all players $\mathcal{P}_i$ such that $i \in M$.

Security: The following theorem proves PPSI-MAL protocol is secure in the presence of the malicious adversary when we assume the majority of honest players.

Theorem 1 *(Security of PPSI-MAL) If Enc_{pk} is semantically secure additive homomorphic encryption, protocol PPSI-MAL is a secure computation protocol for computing the PPSI functionality in the presence of any coalition C of t corrupt players such that $2t + 1 \leq n$. Specifically, for any arbitrarily malicious adversarial algorithm $\mathcal{A}$ controlling players in C, there exists an efficient simulator $\mathcal{S}$ such that for any set of inputs $\{\mathtt{X_i}\}_{i \notin C}$ to the honest players, the outputs of adversary $\mathcal{A}$ and of the honest players interacting in the PPSI-MAL protocol are computationally indistinguishable from the outputs of $\mathcal{S}$ and of the honest players in the ideal world interacting with the ideal PPSI functionality f_{PPSI}.*

We relegate the proof of Theorem 2 to Appendix B.

Public Parameters: The number of players n, maximum threshold of corrupted players t satisfying $2t+1 \leq n$, maximum size of each dataset k, set S of size at least $2k+1$, and descriptions of Enc_{pk} (the modified ElGamal), $\mathbb{G}$ (a base group), Com_{ck} (commitment scheme using Pedersen commitment scheme), and $\mathtt{P} \subset \mathbb{Z}_p$ (the encoding for private inputs).
Private Input of Player $\mathcal{P}_i$: A set $\mathtt{X_i}$ of k values in $\mathtt{P}$.

(Initialization). All players perform the DKG protocol to generate a public key pk of Enc_{pk}. Private output of player $\mathcal{P}_i$ is denoted sk_i.
(Input Data Conversion). Each $\mathcal{P}_i$ performs the following local computations:

(1). Compute $\mathsf{Conv}_{\mathsf{StoP}}(\mathtt{X_i}, S) = \{(s, f_i(s))\}_{s\in S}$ for private polynomial $f_i(x) = (\prod_{a_t \in \mathtt{X_i}}(x-a_t)(1-a_t)^{-1})$, and encrypt them to $\mathsf{Enc}^S_{pk}(f_i)$.
(2). Choose n random polynomials in $\mathbb{Z}_p^k[x]$, $\{r_{i\ell}(x)\}_{\ell\in[1,n]}$ and encrypt them to $\{\mathsf{Enc}^S_{pk}(r_{i\ell})\}_{\ell\in[1,n]}$.

(Online phase). Each player $\mathcal{P}_i$ computes an encryption of the intersection polynomial $I(x) = \sum_{i\in R, \ell\in M} r_{i\ell} \cdot f_\ell$, where M is a set of indices of players proving their well-formed input (that is, verifiably secret-share their input and pass the proof D-ZKPK-DO), and R is a set of index of players passing all zero-knowledge proofs protocols. Next, $\mathcal{P}_i$ decrypts it via threshold decryption protocol together with other players:

(1). Set $M = [1,n]$ and $R = [1,n]$, commit to $\mathsf{Enc}^S_{pk}(f_i)$ to all players, (that is, send $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i)) = \{(s, \mathsf{Com}'_{ck}(c_{is}), \mathsf{Enc}_{pk}(f_i(s)+c_{is}))\}_{s\in S})$, and verifiably secret-share $\{c_{is}\}_{s\in S}$. If player $\mathcal{P}_j$ fails verifiable secret-sharing of his input, then set $M := M \setminus \{j\}$ and $R := R \setminus \{j\}$.
(2). Play the prover part in proof protocol D-ZKPK-DO$[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i))]$ and the verifier part in D-ZKPK-DO$[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_j))]$ for $j \in M \setminus \{i\}$. All n instances of this distributed proof protocol proceed in parallel. If player $\mathcal{P}_j$ fails to pass D-ZKPK-DO$[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_j))]$, then set $M := M \setminus \{j\}$ and $R := R \setminus \{j\}$.
(3). Each $\mathsf{Enc}^S_{pk}(f_j)$ for $j \in M$ is computed by recovering shared values $\{c_{js}\}_{s\in S}$.
(4). Compute $\{C_{i\ell} = r_{i\ell} \otimes \mathsf{Enc}_{pk}(f_\ell)\}_{\ell\in M}$ and broadcast them together with $\{\mathsf{Enc}_{pk}(r_{i\ell})\}_{\ell\in M}$ to all players.
(5). Run D-ZKPK-D$[\mathsf{Enc}^S_{pk}(r_{i\ell})]$ and D-ZKPK-P$[\mathsf{Enc}^S_{pk}(r_{i\ell}), \mathsf{Enc}^S_{pk}(f_\ell), C_{i\ell}]$ protocols as the prover for $\ell \in M$, and also play the verifier part in D-ZKPK-D$[\mathsf{Enc}^S_{pk}(r_{j\ell})]$ and D-ZKPK-P$[\mathsf{Enc}^S_{pk}(r_{j\ell}), \mathsf{Enc}^S_{pk}(f_\ell), C_{j\ell}]$ protocols for $j \in M \setminus \{i\}, \ell \in M$. All n instances of this distributed proof protocol proceed in parallel. If player $\mathcal{P}_j$ fails to pass D-ZKPK-D$[\mathsf{Enc}^S_{pk}(r_{j\ell})]$ or D-ZKPK-P$[\mathsf{Enc}^S_{pk}(r_{j\ell}), \mathsf{Enc}^S_{pk}(f_\ell), C_{j\ell}]$, then set $R := R \setminus \{j\}$.
(6). Compute $C = \oplus_{i\in R, \ell\in M} C_{i\ell} = \mathsf{Enc}^S_{pk}(\sum_{i\in R,\ell\in M} r_{i\ell} \cdot f_\ell)$.
(7). Participate in $|S|$ parallel instances of the threshold decryption protocol TDec, using its private input sk_i, on each ciphertext in C, to obtain the set of values $\{(s, g^{I(s)})\}_{s\in S}$.

(Output Data Conversion). Each player $\mathcal{P}_i$ compute $\mathsf{Conv}_{\mathsf{PtoS}}(\{(s, g^{I(s)})\}_{s\in S})$.

Fig. 2. PPSI-MAL protocol against the malicious adversary

5 Refinement for Efficiency's Sake

We use notations exp and mul to respectively denote the number of exponentiations and multiplications in $\mathbb{G}$. In PPSI-MAL protocol except Input/Output Data Conversion phases, each player is required to perform DKG, TDec, verifiable secret-sharings, the distributed zero-knowledge proofs, and multiplications/additions of encrypted polynomials. Almost operations are dominated by the distributed zero-knowledge proofs that require $O(n^2 + nk)$ exp. ($O(n)$ exp for DKG, $O(nk)$ exp for TDec, $O(nk)$ exp for verifiable secret-sharings and recovering, $O(nk)$ exp for multiplications of n number of k-degree encrypted polynomials.) Each player also computes $O(n^2k)$ mul in the step (6) for additions of n^2 number of $2k$-degree encrypted polynomials. If we assume that $O(1)$ exp is equal to $O(\lambda)$ mul, then the overall computational costs except Input/Output Data Conversion phases is $O(n^2\lambda + (n\lambda + n^2)k)$ mul.

For local computations in PPSI-MAL, each player performs $O(nk^2)$ multiplications in $\mathbb{Z}_p$ for conversions and $O(nk)$ exp for encryptions in Input Data Conversion phase, and $O(k^2)$ exp for conversions in Output Data Conversion phase. For Input/Output Data Conversion phases we need to calculate polynomial interpolation, and we need to evaluate polynomial at multiple points, such that both computation require quadratic operations in degree of a polynomial. The one of the best known algorithm to perform polynomial interpolation or evaluation is using the Fast Fourier Transform (FFT) such that polynomial interpolation/evaluation using FFT require $O(k(\log k)^2)$ field operations for k-degree of polynomial. We refer to [24, 25] for details. We note that FFT can be applied to polynomials even when values or coefficients of polynomials are in the exponent since FFT uses only scalar multiplications and additions. Similarly, polynomial evaluation at multiple points or polynomial interpolation using FFT can be also applied to polynomials even when polynomial coefficient or values are in the exponent. Therefore we can utilize FFT in Output Data Conversion phase.

In fact, to utilize FFT we need a restriction for $\mathbb{Z}_p$ such that $\mathbb{Z}_p$ has a primitive 2^m-th root of unity where 2^m is larger than $2k$. The existence of a primitive 2^m-th root of unity in $\mathbb{Z}_p$ is equivalent to $2^m | p-1$. If we take $p = 2^m \cdot r + 1$ for a random r and perform a primality test until obtaining a prime, we will get such a prime in $O(\log p)$ trials. (That is, the running time to generate an prime p appropriate to FFT is linear in the security parameter, $O(\lambda)$.) Hence r must be at least $O(\log p)$, which means that $m < \log p - \log\log p$. It does not restrict our protocol's parameters since p is much larger than $2k$. The security of ElGamal encryption does not depend on the specific prime p. Lower bound of generic attacks to the decisional Diffie-Hellman problem whose hardness is equivalent to the semantic security of ElGamal encryption is independent of a format of p [23]. If we use an elliptic curve group, then there is not known non-generic attack to the decisional Diffie-Hellman problem. Therefore we can assume that the modified ElGamal encryption is still semantically secure when we are working on $\mathbb{Z}_p$ supporting FFT.

Remark. The most expensive operations in Input/Output Data Conversion are polynomial arithmetics when values are given in the exponent. If we use Paillier encryption [18] instead of the modified ElGamal encryption as an additive homomorphic encryption, then we do know need these expensive polynomial arithmetics for the case of values being in the exponent since Paillier encryption support a complete decryption. Then, if we use Paillier encryption, and do not use FFT and hence polynomial arithmetics are quadratic in k, polynomial arithmetics are field operations dominated by cryptographic operations such as encryption and exponentiations, so that we may obtain PPSI protocol with linear complexity. However, in the security proof of PPSI-MAL protocol the simulator should factor polynomials to extract their roots, but factoring a polynomial over $\mathbb{Z}_N$ where N is RSA modulus is an intractable problem. Therefore we cannot use Paillier encryption in our protocol without modifying the proof or the protocol. Kissner and Song's protocol [17] has a similar problem to ours if they use Paillier encryption. They suggested two options to prove security without factoring polynomials. One is to prove security in the random oracle model. If every root of polynomials is forced to be hash output, then the simulator can extract each root of polynomials by simulating hash oracle. Another is to prove that each polynomial is a product of degree-1 polynomials in zero-knowledge proof manner with strong soundness. Then, the simulator can extract all roots from the property of strong soundness of zero-knowledge proof. However, these zero-knowledge proofs impose $O(k^3)$ complexity.

Estimated Complexity By applying FFT to Input/Output Data Conversion phases, we can reduce computational overhead to $O(nk(\log k)^2)$ multiplications in $\mathbb{Z}_p$ and $O(nk + k(\log k)^2)$ exp, so that the total computational complexity of PPSI-MAL is $O(n^2\lambda + (n\lambda + n^2)k + \lambda k(\log k)^2)$ mul and $O(nk(\log k)^2)$ multiplications in $\mathbb{Z}_p$. It can be simply written by $\tilde{O}(n^2\lambda + (n\lambda + n^2)k)$ mul.

To estimate the communication overhead is easier than the computational overhead. Each player in PPSI-MAL is required to transfer $O(n^2k)$ encryptions in (4) and (5) of Online phase. Since other steps' communication overhead in Online phase is smaller than these steps, the overall communication overhead is $O(n^2k)$. We compare our PPSI protocol secure against malicious adversary with previous works in Table 1.[4]

Protocol	Communication (the number of bits)	Computation (the number of mul)
[17]	$O(n^2k^2\lambda)$	$O(n^2k + n\lambda k^2)$
[20]	$O(c^2k^2\lambda)$	$O(c^2\lambda k^2)$
[21, 22]	$O(nk^2\lambda)$	$O(n\lambda k^2)$
Ours	$O(n^2k\lambda)$	$\tilde{O}(n^2\lambda + (n\lambda + n^2)k)$

c: the number of corrupted players

Table 1. Comparisons PPSI in the presence of malicious adversary

6 Conclusion and Further Work

In this paper we proposed a privacy-preserving set intersection protocol satisfying (1) linear communication and quasi-linear computation overheads in the size of data input, (2) the robustness property in the presence of malicious adversary. The proposed protocol attains linear/quasi-linear complexities, however, there are some open issues.

One interesting open problem is to construct a PPSI protocol with linear complexity in both the size of data input and the number of players. Sang and Shen [22] attain linear complexity in the number of players, and our protocol achieves quasi-linear complexity in the size of data input. There is, however, no PPSI protocol with linear or quasi-linear complexity in both parameters.

Finding practical solutions for other privacy-preserving set operations, such as set union [9] and threshold set union [16], are also interesting open problems.

[4] In [17], Kissner and Song also proposed (without proof) that the cut-and-choose technique may be used to reduce communication complexity to $O(n^2k)$. Sang and Shen, however, pointed out that the cut-and-choose technique will give the adversary chances to behave maliciously in KS protocol [21, 22].

References

1. Y. Aumann and Y. Lindell. Security against covert adversaries: efficient protocols for realistic adversaries. In *TCC 2007*, LNCS, pages 137–156. Springer-Verlag, 2007.
2. J. Camenisch and G. M. Zaverucha. Private intersection of certified sets. In *Financial Cryptography 2009*, LNCS. Springer-Verlag, 2009.
3. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *FOCS*, 1985.
4. E. D. Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, LNCS, pages 143–159. Springer, 2010.
5. D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Efficient robust private set intersection. In M. A. et al., editor, *ACNS 2009*, volume 5536 of *LNCS*, pages 126–142. Springer-Verlag, 2009.
6. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, volume 435 of *LNCS*, pages 307–315. Springer, 1989.
7. P. Feldman. A practival scheme for non-interactive verifiable secure sharing. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.
8. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set-intersection. In C. C. et al., editor, *Advances in Cryptology-EuroCrypt'04*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, 2004.
9. K. B. Frikken. Privacy-preserving set union. In *ACNS*, LNCS, pages 237–252. Springer, 2007.
10. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Journal of Cryptology*, volume 20(1), pages 51–83. Springer New York, 2007.
11. O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.
12. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM STOC*, pages 218–229, 1987.
13. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against mailicious and covert adversaries. In *TCC 2008*, LNCS, pages 155–175. Springer-Verlag, 2008.
14. C. Hazay and K. Nissim. Efficient set operations in the presence of malicious adversaries. In *Public Key Cryptography*, LNCS, pages 312–331. Springer, 2010.
15. S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *TCC 2009*, LNCS. Springer-Verlag, 2009.
16. L. Kissner and D. Song. Privacy-preserving set operations. In V. Shoup, editor, *Advances in Cryptology-Crypto'05*, volume 3621 of *LNCS*, pages 241–257. Springer-Verlag, 2005.
17. L. Kissner and D. Song. Privacy-preserving set operations. Technical Report CMU-CS-05-133, Carnegie Mellon University, 2006.
18. P. Paillier. Public-key cryptosystems based on composite degree residuosity public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology-EuroCrypt'99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.
19. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology-Crypto'91*, volume 576 of *LNCS*, pages 129–140. Springer-Verlag, 1991.
20. Y. Sang and H. Shen. Privacy preserving set intersection protocol secure against malicious behaviors. In *Proceedings of the Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 461–468, Washington, DC, USA, 2007. IEEE Computer Society.
21. Y. Sang and H. Shen. Privacy preserving set intersection based on bilinear groups. In *Proceedings of the thirty-first Australasian conference on Computer science*, volume 74, pages 47–54, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.
22. Y. Sang and H. Shen. Efficient and secure protocols for privacy-preserving set operations. In *ACM Transactions on Information and Systems Security*, 2009.
23. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266. Springer, 1997.
24. V. Shoup. *A Conputational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.
25. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.
26. A. Yao. Protocols for secure computations. In *FOCS*, 1982.

A Honest-Verifier Zero-Knowledge Proofs

In this section we describe the sigma protocol, i.e. (special) honest verifier zero-knowledge proof systems with (special) strong soundness, for the statements $\mathsf{P}, \mathsf{D}, \mathsf{DO}$ needed in the PPSI-MAL protocol, as described in section 4.2. These proof systems are simple extensions of well-known sigma protocols for proving knowledge of discrete logarithms, representations, and arithmetic relations between these representations. For example the sigma protocol for HVZKPK-P$[\mathsf{Enc}_{pk}(f_1), \mathsf{Enc}_{pk}(f_2), \mathsf{Enc}_{pk}(f_3)]$ is a conjunction of $|S|$ instances, one for each $s \in S$, of a sigma protocol for proving knowledge of plaintexts $f_1(s), f_2(s), f_3(s)$ encrypted in $\mathsf{Enc}_{pk}(f_1(s)), \mathsf{Enc}_{pk}(f_2(s)), \mathsf{Enc}_{pk}(f_3(s))$, which satisfy relation $f_1(s) \cdot f_2(s)$ is equal to $f_3(s)$ as an element in $\mathbb{Z}_p$, the plaintext domain.

We use notation $\mathsf{Enc}_{pk}(\cdot;\cdot)$ and $\mathsf{Com}'_{ck}(\cdot;\cdot)$ instead of $\mathsf{Enc}_{pk}(\cdot)$ and $\mathsf{Com}'_{ck}(\cdot)$, respectively, to indicate randomizers in addition to messages in a encryption and a commitment scheme, respectively. We assume that both Enc_{pk} and Com'_{ck} have the additively homomorphic property on both message and randomness, for example the modified ElGamal encryption and Pedersen commitment scheme, respectively.

Sigma Protocol for Product Relation HVZKPK-P: We show a sigma protocol for relation HVZKPK-P between the prover P and the verifier V, namely to prove equality $\mathsf{Enc}^S_{pk}(f_3) = f_1 \otimes \mathsf{Enc}^S_{pk}(f_2)$ for given encrypted polynomials $\mathsf{Enc}^S_{pk}(f_1)$, $\mathsf{Enc}^S_{pk}(f_2)$, $\mathsf{Enc}^S_{pk}(f_3)$, is just a conjunction of $|S|$ proof systems, one for each triple of ciphertexts $\mathsf{Enc}_{pk}(f_1(s)), \mathsf{Enc}_{pk}(f_2(s)), \mathsf{Enc}_{pk}(f_3(s))$ for each $s \in S$. Therefore we give SHVZK for the statement that for given $A = \mathsf{Enc}_{pk}(a; r_a), B = \mathsf{Enc}_{pk}(b; r_b), C = \mathsf{Enc}_{pk}(c; r_c)$, $(A, C) \in L_B$ where $L_B = \{\ (A, C) \mid A = \mathsf{Enc}_{pk}(a; r_a),\ B = \mathsf{Enc}_{pk}(b; r_b),\ C = \mathsf{Enc}_{pk}(a \cdot b; a \cdot r_b + r)$ for $a,\ r_a,\ r \in \mathbb{Z}_p\ \}$.

HVZKPK-P

Common Input: Description of homomorphic encryption Enc including the plaintext domain $\mathbb{Z}_p$, a public key pk, and $A = \mathsf{Enc}_{pk}(a; r_a)$, $B = \mathsf{Enc}_{pk}(b; r_b)$, $C = \mathsf{Enc}_{pk}(a \cdot b; a \cdot r_b + r)$.
Prover Input: a, r_a and r.
Goal: Prove that $(A, C) \in L_B$.

- **P** $\rightarrow$ **V**: P chooses $a', r'_a, r' \xleftarrow{\$} \mathbb{Z}_p$. P computes $A' := \mathsf{Enc}_{pk}(a'; r'_a)$ and $C' := B^{a'} = \mathsf{Enc}_{pk}(a' \cdot b; a' \cdot r_b + r')$, and then, sends V A', C'.
- **V** $\rightarrow$ **P**: V chooses $e \xleftarrow{\$} \mathbb{Z}_p$, and then sends e to P.
- **P** $\rightarrow$ **V**: P computes $\tilde{a} := a' + e \cdot a$, $\tilde{r}_a := r'_a + e \cdot r_a$, and $\tilde{r} := r' + e \cdot r$, and then sends V $\tilde{a}, \tilde{r}_a, \tilde{r}$.
- **V**: V accepts if (1) $\mathsf{Enc}_{pk}(\tilde{a}; \tilde{r}_a) \stackrel{?}{=} A' \oplus A^e$ and (2) $B^{\tilde{a}} \oplus \mathsf{Enc}_{pk}(0; \tilde{r}) \stackrel{?}{=} C' \oplus C^e$.

Completeness: straightforward. □
(Special) Honest-Verifier Simulation: Given A, B, C and e, the simulator picks $\tilde{a}$, $\tilde{r}_a$ and $\tilde{r}$ at random, and computes corresponding A' and C' as follows;

$$A' = \mathsf{Enc}_{pk}(\tilde{a}; \tilde{r}_a) \oplus A^{-e}.$$
$$C' = B^{\tilde{a}} \oplus \mathsf{Enc}_{pk}(0; \tilde{r}) \oplus C^{-e}.$$

Then, $A' = \mathsf{Enc}_{pk}(\tilde{a} - e \cdot a; \tilde{r}_a - e \cdot r_a)$ and $C' = \mathsf{Enc}_{pk}((\tilde{a} - e \cdot a) \cdot b; (\tilde{a} - e \cdot a) \cdot r_b + \tilde{r} - e \cdot r)$. Hence $a' = \tilde{a} - ea$, $r'_a = \tilde{r}_a - er_a$ and $r' = \tilde{r} - er$. In the simulated distribution $(a', r'_a, r', A', C', \tilde{a}, \tilde{r}_a, \tilde{r})$, $\tilde{a}, \tilde{r}_a$, and $\tilde{r}$ are uniformly distributed, and others are uniquely determined, in particular, there are only one corresponding (a', r'_a, r', A', C') for each $(\tilde{a}, \tilde{r}_a, \tilde{r})$. Let us consider real distribution $(a', r'_a, r', A', C', \tilde{a}, \tilde{r}_a, \tilde{r})$ generated by the prover. In real distribution a', r'_a, and r' are uniformly distributed, and others are calculated using a', r'_a and r'. If we consider the restricted distribution $(\tilde{a}, \tilde{r}_a, \tilde{r})$, then $(\tilde{a}, \tilde{r}_a, \tilde{r})$ are uniformly distributed. Furthermore, there exists one-to-one correspondence between (a', r'_a, r') and $(\tilde{a}, \tilde{r}_a, \tilde{r})$. Therefore the simulated distribution is perfectly same to the real distribution. □

(Special) Strong Soundness: We show that if there exist $(e, \tilde{a}, \tilde{r}_a, \tilde{r})$ and $(e^*, \tilde{a}^*, \tilde{r}^*_a, \tilde{r}^*)$ such that
(1) $e \neq e^*$
(2) $A' \oplus A^e = \mathsf{Enc}_{pk}(\tilde{a}; \tilde{r}_a)$ and $C' \oplus C^e = B^{\tilde{a}} \oplus \mathsf{Enc}_{pk}(0; \tilde{r})$
(3) $A' \oplus A^{e^*} = \mathsf{Enc}_{pk}(\tilde{a}^*; \tilde{r}^*_a)$ and $C' \oplus C^{e^*} = B^{\tilde{a}^*} \oplus \mathsf{Enc}_{pk}(0; \tilde{r}^*)$, then $(A, C) \in L_B$. Moreover, the witness a can be efficiently extracted from the above two distinct related proof transcripts.

Let us show how to extract a. First, compute $((A' \oplus A^e) \oplus (A' \oplus A^{e^*})^{-1})^{(e-e^*)^{-1}}$ where $(e - e^*)^{-1}$ is the multiplicative inverse element of $(e - e^*)$ in the ring $\mathbb{Z}_p$. Then,

$$\begin{aligned} & A \\ &= ((A' \oplus A^e) \oplus (A' \oplus A^{e^*})^{-1})^{(e-e^*)^{-1}} \\ &= \mathsf{Enc}_{pk}((e - e^*)^{-1}(\tilde{a} - \tilde{a}^*); (e - e^*)^{-1}(\tilde{r}_a - \tilde{r}^*_a)) \end{aligned}$$

Since $\mathsf{Enc}_{pk}(\cdot;\cdot)$ is a one-to-one mapping, $a = (e - e^*)^{-1}(\tilde{a} - \tilde{a}^*)$ and $r_a = (e - e^*)^{-1}(\tilde{r}_a - \tilde{r}^*_a)$. Therefore we can extract the witness a.

Now we remain to show that $(A, C) \in L_B$. Compute C as follows:

$$\begin{aligned} & C \\ &= ((C' \oplus C^e) \oplus (C' \oplus C^{e^*})^{-1})^{(e-e^*)^{-1}} \\ &= \mathsf{Enc}_{pk}((\tilde{a} - \tilde{a}^*)b; (\tilde{a} - \tilde{a}^*)r_b + (\tilde{r} - \tilde{r}^*))^{(e-e^*)^{-1}} \end{aligned}$$

Since $a = (e - e^*)^{-1}(\tilde{a} - \tilde{a}^*)$ and $\mathsf{Enc}_{pk}(\cdot;\cdot)$ is a one-to-one mapping, C is an encryption of ab with a randomness $ar_b + (e - e^*)^{-1}(\tilde{r} - \tilde{r}^*)$. □

Sigma Protocol for Product Relation HVZKPK-D: We let $L_{(k,S)} = \{ \mathsf{Enc}^S_{pk}(f) \mid f \in \mathbb{Z}^k_p[x] \}$ be a set of encrypted polynomials of degree less than or equal to k.

HVZKPK-D

Common Input: Description of homomorphic encryption Enc including the plaintext domain $\mathbb{Z}_p$, a public key pk, and $\mathsf{Enc}^S_{pk}(f) := \{(s, C_s)\}_{s\in S}$ where S is an arbitrary index set such that $|S| \geq k+1$, f is a k-degree polynomial f, and $C_s = \mathsf{Enc}_{pk}(f(s); r_s)$ for a random $r_s \in \mathbb{Z}_p$.
Prover Input: f and $\{r_s\}_{s\in S}$.
Goal: Prove that $\mathsf{Enc}^S_{pk}(f) \in L_{(k,S)}$.

- $\mathbf{P} \to \mathbf{V}$: P chooses $f' \xleftarrow{\$} \mathbb{Z}_p^k[x]$ and $r'_s \xleftarrow{\$} \mathbb{Z}_p$ for $s \in S$. P computes $\mathsf{Enc}^S_{pk}(f') = \{(s, C'_s)\}_{s\in S}$, where $C'_s := \mathsf{Enc}_{pk}(f'(s); r'_s)$, and sends $\mathsf{Enc}^S_{pk}(f')$ to V.
- $\mathbf{V} \to \mathbf{P}$: V chooses $e \xleftarrow{\$} \mathbb{Z}_p$, and then sends e to P.
- $\mathbf{P} \to \mathbf{V}$: P computes $\tilde{f} := f' + e \cdot f$ and $\tilde{r}_s := r'_s + e \cdot r_s$ for $\forall s \in S$, and then, sends V $\tilde{f}, \tilde{r}_s$.
- $\mathbf{V}$: V accepts if (1) $\tilde{f} \in \mathbb{Z}_p^k[x]$ and (2) $\mathsf{Enc}_{pk}(\tilde{f}; \tilde{r}_s) = C'_s \oplus C_s^e$ for $\forall s \in S$.

Completeness: It is straightforward. □
(Special) HVZK Simulatability: Given $\{C_s\}_{s\in S}, e$, the simulator picks a polynomial $\tilde{f}(x)$ of degree less than or equal to k, and integers $\{\tilde{r}_s\}_{s\in S}$ at random. Then, computes corresponding $\{C'_s\}_{s\in S}$ as follows;

$$C'_s = \mathsf{Enc}_{pk}(\tilde{f}(s); \tilde{r}_s) \oplus C_s^{-e}.$$

C'_s is equal to $\mathsf{Enc}_{pk}(\tilde{f}(s) - e\cdot f(s); \tilde{r}_s - e\cdot r_s)$, and thus $f'(s) = \tilde{f}(s) - e\cdot f(s)$ and $r'(s) = \tilde{r}_s - e\cdot r_s$. Since f and $\tilde{f}$ are degree less than or equal to k, f' is, too. In the simulated distribution of protocol transcripts $(\{C'_s\}_{s\in S}, \tilde{f}, \{\tilde{r}_s\}_{s\in S})$, $\tilde{f}$ and $\{\tilde{r}_s\}_{s\in S}$ are uniformly distributed, but $\{C'_s\}_{s\in S}$ is uniquely determined by $\tilde{f}$ and $\{\tilde{r}_s\}_{s\in S}$. In real protocol f' and $\{r'_s\}_{s\in S}$ are chosen at random, and then $\tilde{f}$ and $\{\tilde{r}_s\}_{s\in S}$ are computed. If we consider only the distribution of $\tilde{f}$ and $\{\tilde{r}_s\}_{s\in S}$, it is uniformly distributed, and there exists one-to-one correspondence between ($\tilde{f}$, $\{\tilde{r}_s\}_{s\in S}$) and (f', $\{r'_s\}_{s\in S}$). Therefore, there exists unique values of (f', $\{r'_s\}_{s\in S}$, $\{C'_s\}_{s\in S}$) for each ($\tilde{f}$, $\{\tilde{r}_s\}_{s\in S}$), so that the simulated distribution is identical to the real distribution. □
(Special) Strong Soundness: We show that if there exist $(e, \tilde{f}, \{\tilde{r}_s\}_{s\in S})$ and $(e^*, \tilde{f}^*, \{\tilde{r}^*_s\}_{s\in S})$ such that
(1). $e \neq e^*$,
(2). $\tilde{f} \in \mathbb{Z}_p^k[x]$ and $\mathsf{Enc}_{pk}(\tilde{f}; \tilde{r}_s) = C'_s \oplus C_s^e$ for $\forall s \in S$,
(3). $\tilde{f}^* \in \mathbb{Z}_p^k[x]$ and $\mathsf{Enc}_{pk}(\tilde{f}^*; \tilde{r}^*_s) = C'_s \oplus C_s^{e^*}$ for $\forall s \in S$,
then $\mathsf{Enc}^S_{pk}(f) \in L_{(k,S)}$. Moreover, polynomial f can be extracted given the above two proof transcripts.

From (2) and (3) we know for $s \in S$,

$$C'_s \oplus C_s^e = \mathsf{Enc}_{pk}(\tilde{f}(s); \tilde{r}_s),$$
$$C'_s \oplus C_s^{e^*} = \mathsf{Enc}_{pk}(\tilde{f}^*(s); \tilde{r}^*_s).$$

Compute C_s as follows:

$$\begin{aligned} & C_s \\ &= ((C'_s \oplus C_s^e) \oplus (C'_s \oplus C_s^{e^*})^{-1})^{(e-e^*)^{-1}} \\ &= \mathsf{Enc}_{pk}((e-e^*)^{-1}(\tilde{f}(s) - \tilde{f}^*(s)); (e-e^*)^{-1}(\tilde{r}_s - \tilde{r}^*_s)) \end{aligned}$$

Since $\tilde{f}, \tilde{f}^* \in \mathbb{Z}_p^k[x]$, $(e-e^*)^{-1}(\tilde{f}-\tilde{f}^*) \in \mathbb{Z}_p^k[x]$, too. Therefore $\{C_s\}_{s\in S}$ is also an encryption of a polynomial in $\mathbb{Z}_p^k[x]$, and we can extract the witness polynomial by computing $(e-e^*)^{-1}(\tilde{f}-\tilde{f}^*)$. □

Sigma Protocol for Product Relation HVZKPK-DO: Protocol HVZKPK-DO is a trivial modification of the above protocol HVZKPK-D: The prover chooses a polynomial f' s.t. $f'(1)=0$, and uses Com instead of Enc_{pk}. Then, the verifier accepts only if $\tilde{f}(1)=e$.

We let $LO_{(k,S)}$=$\{\ \mathsf{Enc}^S_{pk}(f) \mid f \in \mathbb{Z}_p^k[x]$ and $f(1)=1\}$ be a set of valid encrypted polynomials passing a point $(1,1)$ of degree less than or equal to k. We use the Pedersen commitment scheme Com'_{ck}, and use the notation $\mathsf{Com}'_{ck}(c_s;d_s)$ to indicate randomizer d_s as well as message c_s.

HVZKPK-DO

Common Input: Description of Com_{ck}, Enc_{pk} including $\mathbb{Z}_p$, ck, and pk, and $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f)) := \{(s, D_s, C_s))\}_{s\in S}$, where $D_s = \mathsf{Com}'_{ck}(c_s;d_s)$, $C_s = \mathsf{Enc}_{pk}(f(s)+c_s;r_s)$, S is an index set such that $|S| \geq k+1$, f is a k-degree polynomial f, and r_s is chosen at random $\mathbb{Z}_p$.
Prover Input: f, $\{(c_s, d_s, r_s)\}_{s\in S}$.
Goal: $\mathsf{Enc}^S_{pk}(f) \in LO_{(k,S)}$.

- $\mathbf{P} \to \mathbf{V}$: P chooses $f' \xleftarrow{\$} \mathbb{Z}_p^k[x]$ passing a point $(1,0)$ and $r'_s, c'_s \xleftarrow{\$} \mathbb{Z}_p$ for $s \in S$, and then computes $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f')) = \{(s, D'_s, C'_s)\}_{s\in S}$, where $C'_s := \mathsf{Enc}_{pk}(f'(s)+c'_s;r'_s)$ and $D'_s = \mathsf{Com}'_{ck}(c'_s;d'_s)$, and then sends $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f'))$ to V.
- $\mathbf{V} \to \mathbf{P}$: V chooses $e \xleftarrow{\$} \mathbb{Z}_p$, and then sends e to P.
- $\mathbf{P} \to \mathbf{V}$: P computes $\tilde{f} := f' + e \cdot f$, $\tilde{r}_s := r'_s + e \cdot r_s$, $\tilde{c}_s := c'_s + e \cdot c_s$, and $\tilde{d}_s := d'_s + e \cdot d_s$ for $\forall s \in S$, and then, sends V $\tilde{f}, \tilde{r}_s, \tilde{c}_s, \tilde{d}_s$.
- **V**: V accepts if (1) $\tilde{f} \in \mathbb{Z}_p^k[x]$, (2) $\tilde{f}(1)=e$, and (3) $\mathsf{Enc}_{pk}(\tilde{f}(s)+\tilde{c}_s;\tilde{r}_s) = C'_s \oplus C_s^e$ for $\forall s \in S$ (4) $\mathsf{Com}'_{ck}(\tilde{c}_s;\tilde{d}_s) = D'_s \oplus D_s^e$ for $\forall s \in S$.

Completeness: It is straightforward. □
(Special) HVZK Simulatability: Given $\{D_s, C_s\}_{s\in S}, e$, the simulator picks a polynomial $\tilde{f}(x) \xleftarrow{\$} \mathbb{Z}_p^k[x]$ passing a point $(1,e)$ and integers $\{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}$ at random, and computes corresponding $\{D'_s, C'_s\}_{s\in S}$ as follows;

$$D'_s = \mathsf{Com}'_{ck}(\tilde{c}_s;\tilde{d}_s)/D_s^e,$$
$$C'_s = \mathsf{Enc}_{pk}(\tilde{f}(s)+\tilde{c}_s;\tilde{r}_s)/C_s^e.$$

D'_s is equal to $\mathsf{Com}_{ck}(\tilde{c}_s - e\cdot c_s; \tilde{d}_s - e\cdot d_s)$, and thus $c'_s = \tilde{c}_s - e\cdot c_s$ and $d'_s = \tilde{d}_s - e\cdot d_s$. Thus, C'_s is equal to $\mathsf{Enc}_{pk}(\tilde{f}(s) - e\cdot f(s) + c'; \tilde{r}_s - e\cdot r_s)$, and thus $f'(s) = \tilde{f}(s) - e\cdot f(s)$ and $r'(s) = \tilde{r}_s - e\cdot r_s$. Since both f and $\tilde{f}$ are in $\mathbb{Z}_p^k[x]$, so f' is, and since $f(1)=1$ and $\tilde{f}(1)=e$, $f'(1)=0$ as desired.

In the simulated distribution of protocol transcripts $(\{D'_s, C'_s, \tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}, \tilde{f})$, $\tilde{f}$ and $\{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}$ are uniformly distributed, but $\{D'_s, C'_s\}_{s\in S}$ is uniquely determined by others.

In real protocol f' and $\{c'_s, d'_s, r'_s\}_{s\in S}$ are chosen at random, and then $\tilde{f}$ and $\{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}$ are computed.

Let us consider the restricted distribution of $\tilde{f}$ and $\{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}$ in the real distribution. Since $f' = (x-1)\cdot h'$ for $h' \xleftarrow{\$} \mathbb{Z}_p^{k-1}$ and $f = (x-1)\cdot h + 1$ for some $h \in \mathbb{Z}_p^{k-1}[x]$, $\tilde{f}$ is equal to $f' + ef = (x-1)(h' + eh) + e$, and hence $\tilde{f}$ is uniformly distributed in $\mathbb{Z}_p^k[x]$ with passing a point $(1, e)$. We can easily check that $\tilde{c}_s, \tilde{d}_s, \tilde{r}_s$ are uniformly distributed in $\mathbb{Z}_p$. Since there exists one-to-one correspondence between $(\tilde{f}, \{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S})$ and $(f', \{c'_s, d'_s, r'_s\}_{s\in S})$. Therefore, there exists unique values of $(f', \{c'_s, d'_s, r'_s\}_{s\in S})$ for each $(\tilde{f}, \{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S})$. For each $(f', \{c'_s, d'_s, r'_s\}_{s\in S})$, there exists unique values for $D' = \mathsf{Com}'_{ck}(c'_s; d'_s)$ and $\mathsf{Enc}_{pk}(f'(s) + c'_s; r'_s)$, so that the real distribution is identical to the simulated distribution. □

(Special) Strong Soundness: We show that if there exist $(e, \tilde{f}, \{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S})$ and $(e^*, \tilde{f}^*, \{\tilde{c}^*_s, \tilde{d}^*_s, \tilde{r}^*_s\}_{s\in S})$ such that
(1). $e \neq e^*$,
(2). $\tilde{f} \in \mathbb{Z}_p^k[x]$ passing $(1, e)$, $\mathsf{Com}'_{ck}(\tilde{c}_s; \tilde{d}_s) = D'_s \oplus D^e_s$, and $\mathsf{Enc}_{pk}(\tilde{f}(s) + \tilde{c}_s; \tilde{r}_s) = C'_s \oplus C^e_s$ for $\forall s \in S$,
(3). $\tilde{f}^* \in \mathbb{Z}_p^k[x]$ passing $(1, e^*)$, $\mathsf{Com}'_{ck}(\tilde{c}^*_s; \tilde{d}^*_s) = D'_s \oplus D^{e^*}_s$, and $\mathsf{Enc}_{pk}(\tilde{f}^*(s) + \tilde{c}^*_s; \tilde{r}^*_s) = C'_s \oplus C^{e^*}_s$ for $\forall s \in S$,
then $\mathsf{Enc}^S_{pk}(f) \in LO_{(k,S)}$. Moreover, polynomial f can be extracted given the above two proof transcripts.

From (2) and (3) we know for $s \in S$,

$$\begin{aligned} D'_s \oplus D^e_s &= \mathsf{Com}'_{ck}(\tilde{c}_s; \tilde{d}_s), \\ D'_s \oplus D^{e^*}_s &= \mathsf{Com}'_{ck}(\tilde{c}^*_s; \tilde{d}^*_s). \\ C'_s \oplus C^e_s &= \mathsf{Enc}_{pk}(\tilde{f}(s) + \tilde{c}_s; \tilde{r}_s), \\ C'_s \oplus C^{e^*}_s &= \mathsf{Enc}_{pk}(\tilde{f}^*(s) + \tilde{c}^*_s; \tilde{r}^*_s). \end{aligned}$$

Compute D_s as follows:

$$\begin{aligned} & D_s \\ &= ((D'_s \oplus D^e_s) \oplus (D'_s \oplus D^{e^*}_s)^{-1})^{(e-e^*)^{-1}} \\ &= \mathsf{Com}'_{ck}((e-e^*)^{-1}(\tilde{c}_s - \tilde{c}^*_s); (e-e^*)^{-1}(\tilde{d}_s - \tilde{d}^*_s)) \end{aligned}$$

Then, we can extract c_s and d_s as $c_s = (e-e^*)^{-1}(\tilde{c}_s - \tilde{c}^*_s)$ and $d_s = (e-e^*)^{-1}(\tilde{d}_s - \tilde{d}^*_s)$.

Compute C_s as follows:

$$\begin{aligned} & C_s \\ &= ((C'_s \oplus C^e_s) \oplus (C'_s \oplus C^{e^*}_s)^{-1})^{(e-e^*)^{-1}} \\ &= \mathsf{Enc}_{pk}(\tilde{f}(s) - \tilde{f}^*(s) + c_s; \tilde{r}_s - \tilde{r}^*_s)^{(e-e^*)^{-1}} \end{aligned}$$

Since $\tilde{f}, \tilde{f}^* \in \mathbb{Z}_p^k[x]$, $\tilde{f}(1) = e$ and $\tilde{f}(1)^* = e^*$, $(e-e^*)^{-1}(\tilde{f} - \tilde{f}^*) \in \mathbb{Z}_p^k[x]$ with passing a point $(1, 1)$. Therefore $\{C_s\}_{s\in S}$ is also an encryption of a polynomial in $\mathbb{Z}_p^k[x]$ with passing $(1, 1)$, and we can extract the witness polynomial by computing $(e-e^*)^{-1}(\tilde{f} - \tilde{f}^*)$. □

B Security Proof of Theorem 2

Theorem 2 *(Security of PPSI-MAL): If Enc_{pk} is semantically secure additive homomorphic encryption, protocol PPSI-MAL is a secure computation protocol for computing the PPSI functionality in the presence of any coalition C of t corrupt players such that $2t + 1 \leq n$. Specifically, for any arbitrarily malicious adversarial algorithm $\mathcal{A}$ controlling players in C, there*

exists an efficient simulator $\mathcal{S}$ such that for any set of inputs $\{\mathtt{X_i}\}_{i\notin C}$ to the honest players, the outputs of adversary $\mathcal{A}$ and of the honest players interacting in the PPSI-MAL protocol are computationally indistinguishable from the outputs of $\mathcal{S}$ and of the honest players in the ideal world interacting with the ideal PPSI functionality f_{PPSI}.

Proof. We describe the simulator $\mathcal{S}$, which interacts with adversary $\mathcal{A}$ who controls the set of corrupted players C. Let H be the remaining honest players. Let $\{\mathtt{X_i}\}_{i\in H}$ be the private inputs of the honest players. Let $\mathtt{X} = f_{\mathsf{PPSI}}(\mathtt{X}_1, ..., \mathtt{X}_n)$ denote the multi-party set intersection function, i.e. $\mathtt{X} = \mathtt{X}_1 \cap ... \cap \mathtt{X}_n$. The simulator's goal is two-fold: To extract the effective inputs $\{\mathtt{X}_i\}_{i\in C}$ which the corrupted players enter into the PPSI computation, and to simulate the same view that $\mathcal{A}$ would see when interacting with the honest players on inputs $\{\mathtt{X}_i\}_{i\in H}$ but given only these extracted adversarial inputs $\{\mathtt{X}_i\}_{i\in C}$ and the output $\mathtt{X} = f_{\mathsf{PPSI}}(\mathtt{X}_1, ..., \mathtt{X}_n)$.

Simulator Description:
[1]. $\mathcal{S}$ runs, on behalf of players in H, Input Data Conversion by setting $f_i = 1$ for $\forall i \in H$, and steps (1)-(2) of Online phase in the protocol. Let δ_0, δ_1 and δ_2 be the random coins that $\mathcal{S}$ uses in, respectively, Input Data Conversion, and (1)-(2) of Online phase in this execution.
[2]. $\mathcal{S}$ runs until the end of the step (2). Let D be the set of players in C that correctly carry out verifiable secret sharing procedure in the step (1) and pass D-ZKPK-DO proof in the step (2). That is, $D = C \cap M$. $\mathcal{S}$ rewinds $\mathcal{A}$ and runs it repeatedly until that for each player $\mathcal{P}$ in D it finds a corresponding execution such that $\mathcal{P}$ pass, and a challenge on proof is different from original one. In each run, $\mathcal{S}$ performs Input Data Conversion and step (1) of Online phase on coins δ_0 and δ_1, respectively, and step (2) on fresh random coins. (Claim 4 below show that the expected number of such rewindings is polynomial in security parameter λ.) Using the strong soundness property of the D-ZKPK-DO proof, $\mathcal{S}$ extracts the witnesses f_i for players in D. $\mathcal{S}$ factors these polynomials to compute $\{\mathtt{X}_i\}_{i\in D'}$ (Factoring polynomials in $\mathbb{Z}_p[x]$ requires quadratic operations in k, so that it takes polynomial in the security parameter λ.), and sets $\mathtt{X}_i \leftarrow \phi$ for $\forall i \in C \setminus D$, then sends $\{\mathtt{X}_i\}_{i\in C}$ to the ideal PPSI functionality, and receives the output $\mathtt{X} = f_{\mathsf{PPSI}}(\mathtt{X}_1, ..., \mathtt{X}_n)$.
[3]. The simulator $\mathcal{S}$ rewinds $\mathcal{A}$ to the end of step (2) on coins δ_0, δ_1, δ_2, and performs step (3) to reconstruct encrypted polynomials $\{\mathsf{Enc}^S_{pk}(f'_i)\}_{i\in H}$ where each f'_i, $i \in H$, encodes the set $\mathtt{X}$. Pedersen VSS allows $\mathcal{S}$ to equivocate on the reconstructed secret $\{c_{is}\}_{s\in S}$, and hence $\mathsf{Enc}^S_{pk}(f)$.
[4]. The simulator just performs the rest of the protocol on behalf of the players in H using random coins δ_{4-7} and outputs execution, i.e. an execution between $\mathcal{A}$ and players in H using inputs $\{f'_i\}_{i\in H}$ and coins $\delta_0, \delta_1, \delta_2, \delta_{4-7}$.

Claim: The expected number of rewindings in the step [2] in Simulator Description above is polynomial in λ.

Proof. The simulation rewinds $\mathcal{A}$ until that for each $\mathcal{P} \in D$ it finds a corresponding execution such that 1) $\mathcal{P}$ pass, and 2) a challenge on proof is different from original one. Since 2) is satisfied with overwhelming probability if $\mathcal{S}$ choose a random coin independently per each run, we focus on 1).

Let 2^C be a set of power subsets of C, i.e. $2^C = \{\forall\ D \text{ such that } \ D \subset C\}$, the probability that a set of corrupted players $D \subset C$ pass D-ZKPK-DO be ϵ_D where the probability goes over δ_2, and the probability that a corrupted player $\mathcal{P} \in C$ passes D-ZKPK-DO be $\epsilon'_{\mathcal{P}}$ where randomness goes over δ_2, that is, we assume that δ_0 are δ_1 fixed in the probability ϵ_D and $\epsilon'_{\mathcal{P}}$.

Then the expected number of runs for each δ_0, δ_1 is as follows:

$$\begin{aligned}
&\textstyle\sum_{\forall D \in 2^C} \epsilon_D \sum_{\forall \mathcal{P} \in D} (\epsilon'_{\mathcal{P}} \cdot 1 + (1-\epsilon'_{\mathcal{P}})\epsilon'_{\mathcal{P}} \cdot 2 \\
&\qquad\qquad + \cdots + (1-\epsilon'_{\mathcal{P}})^{k-1}\epsilon'_{\mathcal{P}} \cdot k + \cdots) \\
&= \textstyle\sum_{\forall D \in 2^C} \epsilon_D \sum_{\forall \mathcal{P} \in D} \epsilon'_{\mathcal{P}} \sum_{k=1}^{\infty} k(1-\epsilon'_{\mathcal{P}})^{k-1} \\
&= \textstyle\sum_{\forall D \in 2^C} \sum_{\forall \mathcal{P} \in D} (\frac{\epsilon_D}{\epsilon'_{\mathcal{P}}}) \\
&= \textstyle\sum_{\forall D \in 2^C} \epsilon_D \sum_{\forall \mathcal{P} \in D} (\frac{1}{\epsilon'_{\mathcal{P}}}) \\
&= \textstyle\sum_{\forall \mathcal{P} \in C} \sum_{\forall D \ni \mathcal{P}} (\frac{\epsilon_D}{\epsilon'_{\mathcal{P}}}) \\
&= \textstyle\sum_{\forall \mathcal{P} \in C} \frac{1}{\epsilon'_{\mathcal{P}}} \sum_{\forall D \ni \mathcal{P}} \epsilon_D \\
&= \textstyle\sum_{\forall \mathcal{P} \in C} \frac{1}{\epsilon'_{\mathcal{P}}} \epsilon'_{\mathcal{P}} \\
&= \textstyle\sum_{\forall \mathcal{P} \in C} 1 \\
&= |C| = poly(\lambda)
\end{aligned}$$

for some polynomial $poly(\cdot)$. Let us explain why the fourth equality holds. The summation $\sum_{D \in 2^C} \sum_{\mathcal{P} \in D}$ is same to $\sum_{(D,\mathcal{P}) \in \mathsf{S}_0}$ where $\mathsf{S}_0 = \{(D, \mathcal{P}) | \forall D \in 2^C, \text{ and } \forall \mathcal{P} \text{ such that } \mathcal{P} \in D\}$. On the other hand, the summation $\sum_{\mathcal{P} \in C} \sum_{D \ni \mathcal{P}}$ is same to $\sum_{(\mathcal{P},D) \in \mathsf{S}_1}$ where $\mathsf{S}_1 = \{(\mathcal{P}, D) | \forall \mathcal{P} \in C, \text{ and } \forall D \text{ such that } D \ni \mathcal{P}\}$. For every $(D, \mathcal{P}) \in \mathsf{S}_0$, $\mathcal{P}$ is an element of D, so that $(\mathcal{P}, D)$ is also contained S_1. That is, $\mathsf{S}_0 \subset \mathsf{S}_1$. For every $(\mathcal{P}, D) \in \mathsf{S}_1$, D contains $\mathcal{P}$, so that S_0 also has $(D, \mathcal{P})$. That is, $\mathsf{S}_1 \subset \mathsf{S}_0$. Therefore $\mathsf{S}_0 = \mathsf{S}_1$ and the fourth equality holds. Other equalities can be easily verified. Therefore we complete the proof of claim. □

Since simulator's other steps than [2] can be run, the overall running time of $\mathcal{S}$ is polynomial in λ.

Now we argue that in $\mathcal{A}$'s view, $\mathcal{S}$'s simulated transcript is indistinguishable from the real protocol's transcript. First, we consider the random space over which $\mathcal{S}$ uses randomness. In the step [1] of simulation $\mathcal{S}$ chooses $\delta_0, \delta_1, \delta_2$ at random, and runs until the end of step (2). In step [3] of simulation $\mathcal{S}$ rewinds $\mathcal{A}$ to the end of step (2) on coins $\delta_0, \delta_1, \delta_2$, and then it performs the rest of the protocol using randomness δ_{4-7}. Therefore, $\mathcal{S}$ simulation uses all random uniformly.

Next, we show that the simulated transcript is indistinguishable from the real protocol's transcript. In the step [1] $\mathcal{S}$ commits encrypted constant polynomials as honest players' input instead of k-degree polynomials. $\mathcal{A}$, however cannot distinguish because of perfectly hiding property of Com_{ck}. Further, $\mathcal{S}$ can simulate all players in H to pass D-ZKPK-DO in the step [2] since D-ZKPK-DO is straight-line simulatable ZKPK protocol. In the step [3] of simulation is indistinguishable from the real protocol because of trapdoor opening property of Pedersen commitment scheme. Since $\mathcal{S}$ follows the description of the protocol in the step [4], adversarial view is identical in the both simulation and real protocol. At the end of simulation we have the polynomial $I(x)$ that is distributed as a product of a $|\mathtt{X}|$-degree polynomial which encodes $\mathtt{X}$ and a random polynomial of degree $2k - |\mathtt{X}|$, which is the same as the distribution of $I(x)$ in the real protocol between $\mathcal{A}$ and the honest players on inputs $\{\mathtt{X}_i\}_{i \in H}$ by Lemma 1. Therefore the simulated transcript is indistinguishable from the real protocol's transcript.

At the end of protocol the adversary cannot obtain other information about honest players' input than set intersection, and affect the result of the protocol to be wrong. Therefore we complete the proof. □