IEICE TRANS. FUNDAMENTALS, VOL.E95–A, NO.8 AUGUST 2012

PAPER

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

Jung Hee CHEON[†], **Stanislaw JARECKI**[††], *Nonmembers*, *and* **Jae Hong SEO**[†††a)], *Member*

SUMMARY 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 a 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, making secure computation of the set intersection only practical for 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. For a fixed security parameter, our protocols require $O(n^2k)$ bits of communication and $\tilde{O}(n^2k)$ group multiplications per player in the malicious adversary setting, where k is the size of each dataset. Our protocol follows the basic idea of the protocol proposed by Kissner and Song, but we gain efficiency by using different representations of the polynomials associated with users' datasets and careful employment of algorithms that interpolate or evaluate polynomials on multiple points more efficiently. Moreover, the proposed protocol is robust. This means that the protocol outputs the desired result even if some corrupted players leave during the execution of the protocol.

key words: *multi-party set intersection, privacy-preserving set operation*

1. Introduction

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

There are general multi-party computations that implement secure computations for any function that can be expressed as a circuit [13], [28], so that they can be solutions for PPSI. In general, however, the overall overhead is often unacceptable since the complexity depends on the size of the circuit and the protocol requires expensive operations such as several oblivious transfers. In particular, the complexity of general multi-party computations is higher in the presence of a malicious adversary (who can behave arbitrarily) than an honest-but-curious adversary (who follows the protocol's directions correctly).

If we restrict the number of protocol participants to two, there are several improvements in PPSI that can be applied to attain better efficiency than general multi-party computations [2], [4]–[6], [9], [14]–[17]. In the multi-party case, the improvement in complexity has gradually been achieved, compared with that in the two-party case. Kissner and Song [18], [19] first addressed PPSI in the multi-party case based on the work by Freedman et al. [9]. Recently, Sang and Shen [23], [24] advanced Kissner and Song's result by reducing complexity in terms of the number of players. In contrast to the two-party case, however, all proposed protocols for PPSI in the multi-party case have quadratic complexity in the size of private datasets, and finding an efficient solution for PPSI with linear complexity in the input size in the multi-party case still remains an open problem (even in the honest-but-curious adversary model).

In multi-party computations against a malicious adversary, robustness has been considered as an important property [3], [8]. When a corrupted player's malicious behavior is detected, a non robust protocol will stop, and then the protocol should restart excluding the 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 study, our contribution is twofold. First, we present a PPSI protocol and show that each player in our PPSI protocol needs to transfer $O(n^2k)$ encryptions and compute $\tilde{O}(n^2\lambda + (n\lambda + n^2)k)$ group multiplications. (k: size of the user's dataset, n: number of users, λ: security parameter) In particular, if we focus on online computations (that is, computations excluding 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, which is linear in k. 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 the protocol, our protocol outputs set intersections among players who have proved that their input datasets are well-formed.

1.1 Technical Overview

Let us explain the key ideas of our construction. When one employs a polynomial to represent his private dataset, multiplications over encrypted polynomials and their zero-knowledge proofs are the most expensive steps in the sys-

Manuscript received December 2, 2011.

[†]The author is with the Department of Mathematical Sciences, Seoul National University, Seoul, 151-747 Korea.

[††]The author is with the School of Information and Computer Sciences, University of California at Irvine, CA92697, USA.

[†††]The author is with the National Institute of Information and Communications Technology, Koganei-shi, 184-8795 Japan.

a) E-mail: jaehong@nict.go.jp (corresponding author)

DOI: 10.1587/transfun.E95.A.1366

tem. In particular, all previous works dealing with PPSI in the multi-party setting that use polynomials to represent datasets consider a polynomial as a tuple of its coefficients, so that a multiplication over encrypted polynomials imposes quadratic computational and communication complexities in the degree of polynomials [18], [19], [22]–[24].

Our approach to reducing complexity is to represent a polynomial $f(x)$ by several points on the curve $y = f(x)$. This has 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 $g(x)$ as $\{(s, f(s))\}_{s\in S}$ and $\{(s, g(s))\}_{s\in S}$, respectively, for a set of integers S, the multiplication of two polynomials is simply performed as $\{(s, f(s) \cdot g(s))\}_{s\in S}$. Hence, the computational complexity of the protocol excluding the input/output data representation conversions becomes linear. Because of the simple computation for polynomial multiplication, we can construct a simple zero-knowledge proof for a multiplication over encrypted polynomials that imposes only linear computational and communication overheads on the protocol.

The goal of a robust PPSI protocol is to compute set intersections among players who have proved their well-formed input even when some subset of corrupted players leave the protocol after proving their well-formed input. One may think that this problem is easily solved using Verifiable Secret Sharing (VSS) schemes, such as Pedersen VSS (Ped-VSS) [21]. It is, however, difficult to deal with committed encryptions. More precisely, it is difficult to construct zero-knowledge proofs on committed encryptions. For example, the Pedersen commitment, which is used in Ped-VSS, puts the opening in an exponent, and Paillier encryption, which is an additive homomorphic encryption, also puts the message in an exponent. Therefore, the message of committed encryption are located in the exponent of the exponent making it difficult to construct zero-knowledge proofs of the message.

We construct a robust protocol in the presence of a malicious adversary by proposing a simple VSS scheme that allows encryption sharing and supports zero-knowledge proofs on shared encryptions. To share the 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 using VSS. To recover $\mathsf{Enc}_{pk}(m)$, all players recover c and then 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., a zero-knowledge proof for knowledge of m. Furthermore, in the security proof, the simulator can equivocate on m because it can equivocate on c if it has a trapdoor of Com'_{ck}. Therefore, we can construct a robust PPSI protocol using such a simple VSS.

Note that our PPSI protocol uses a so-called 'common reference string' since we use the Pedersen commitment as a building block to design the PPSI protocol, and we assume that a broadcast channel exists.

1.2 Related Work of PPSI Protocols

With the exception of the general multi-party computation approach [13], Kissner and Song proposed the first efficient privacy-preserving set intersection protocol that is secure in the honest-but-curious (HBC) adversary model [18] and extended it to be secure in the malicious adversary model using general zero-knowledge proof techniques [19]. Sang and Shen [22]–[24] proposed protocols having better efficiency than those in [19]. However, they only reduced the complexity in terms of the number of players, and their protocols still maintain quadratic complexity in the input size.

In the two-party case of PPSI, there are several notable studies in the asymmetric set intersection model. That is, there are two entities, the server and receiver, along with their sets in the asymmetric set intersection model. At the end of the interactions, the receiver obtains the result of set intersection, and the server obtains nothing.

Freedman et al. proposed the two-party set intersection protocol using oblivious polynomial evaluation [9]. However, their schemes are only secure against a malicious client or a malicious server (not both). Dachman-Soled et al. improved security by proposing a protocol secure against both malicious parities [6]. Hazay and Nissim improved the complexity and attained almost linear complexity in the input size [15]. Camenisch and Zaverucha considered the case that no malicious adversaries can arbitrary choose their private inputs [2]. In particular, users should use input sets certified by a Certificate Authority in [2]. Using an oblivious pseudorandom function, Hazay and Lindell [14] provided two-party protocols that are secure in the standard model against covert parties [1]. Jarecki and Liu [16] improved its security to achieve security in the standard model against both malicious parties. Cristofaro and Tsudik [5] presented two-party protocols with linear complexity and proposed the authorized set intersection protocol, which allows the set intersection only for an authorized private set in the honest-but-curious adversary model. Jarecki and Liu [17] proposed an adaptive set intersection protocol in the random oracle model, which allows the receiver to adaptively query an element x in his set, and then he learns whether it is contained in the server's set without revealing x to the server. Cristofaro et al. proposed set intersection and authorized set intersection protocols with linear complexity in the malicious adversary model [4].

1.3 Outline

We give the definitions and tools of cryptography in Sect. 2. In Sect. 3, we propose a new representation, called point representation, to describe private datasets, and possible operations over point representation. We present two pro-

tocols, each for different adversary models, the so-called honest-but-curious adversary and the malicious adversary, in Sect. 4. In Sect. 5, we show how to speed up local computations of our protocols, and we analyze the complexities of the proposed protocols. We give a conclusion in Sect. 6.

2. Definitions and Primitives

2.1 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 of each player[†], $\mathbb{Z}_p$ be a finite field of size 2λ with prime p, $\mathbb{Z}_p[x]$ be a polynomial ring over $\mathbb{Z}_p$ and $\mathbb{Z}_p^d[x]$ be the set of all polynomials of degree at most d in $\mathbb{Z}_p[x]$, where λ is a security parameter. For a set A, $a \xleftarrow{\$} A$ denotes a is uniformly chosen at random from A. $[a, b]$ denotes the set $\{\alpha \in \mathbb{Z} | a \leq \alpha \leq b\}$.

2.2 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 an HBC protocol as an intermediate protocol to obtain a malicious one. In HBC adversary model, the adversary follows the protocol's prescribed directions and he can utilize all information obtained during the protocol procedure to gain information other than the protocol's desired result, e.g., set intersection in the PPSI protocol. In contrast to the HBC adversary model, the malicious adversary may corrupt some subset of less than half of the player and gain full control of the corrupted players. He may not follow the given prescribed actions of the protocol and may behave arbitrarily. We cannot prevent the malicious adversary running the protocol along with arbitrary values as his private input.

Informally, we say that a PPSI protocol is secure in the presence of a 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 strategies in the ideal world and that can be followed in the ideal world. Furthermore, the real execution is computationally indistinguishable from the execution in the ideal world from the viewpoint of the malicious adversary. We refer to [12] for the formal definitions of the HBC adversary model and malicious adversary model.

2.3 Additive Homomorphic Encryption

Our construction utilizes an additive homomorphic encryption scheme. In particular, we describe our protocol using the modified ElGamal encryption scheme widely used in the set intersection protocol [2], [6], [9].

The modified ElGamal encryption consists of three algorithms.

• $\mathsf{Setup}(\lambda)$: A 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 the 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 $pk = g^x$, $sk = x$.
• $\mathsf{Enc}_{pk}(m; r)$: Output $CT = (g^r, pk^r g^m)$.
• $\mathsf{Dec}_{sk}(CT)$: Parse CT as (C_0, C_1) and output $C_1 \cdot C_0^{-sk}$.

This modified ElGamal encryption is also additively homomorphic on messages and randomness:

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

where the $\oplus$ operation denotes componentwise group multiplication. To denote the additions of several ciphertexts CT_i for $i \in [1, \ell]$, we use the notation $\oplus_{i\in[1,\ell]} CT_i$. A scalar multiplication is also easily allowed by repeatedly computing the $\oplus$ 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 integer c, we simply denote the scalar multiplication of CT by c as CT^c. Since the Dec algorithm outputs g^m instead of the plaintext m, it does not support a complete decryption. However, the additive homomorphic property still allows us to enjoy our PPSI protocol without a complete decryption algorithm. Furthermore, the modified ElGamal encryption allows ciphertexts to be rerandomized without a secret key.

2.4 Distributed Key Generation and Threshold Decryption

We use the threshold version of the modified ElGamal encryption for which efficient distributed key generation and threshold decryption protocols exists, which are secure against a malicious adversary. We use the distributed key generation protocol, denoted DKG, in [11]. For the threshold decryption protocol, denoted TDec, we use the protocol in [7], amended using a zero-knowledge proof of correctness of partial decryption. We use the distributed version of this zero-knowledge proof, which is suitable for the multiparty computation setting, as we will explain in Sect. 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 that is explained in that section.

2.5 Verifiable Secret Sharing

We use verifiable secret sharing (VSS) for two reasons: for the distributed zero-knowledge proofs in Sect. 4.2 and for

[†]We assume that all players have same-size private datasets. According to each player's needs, he can add dummy inputs having invalid encoding, e.g., elements not in P, which is the domain of private data.

ensuring the robustness in the presence of a malicious adversary. Specifically, we use Pedersen VSS (Ped-VSS) [21].

3. Data Representation and Operations

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 protocols [2], [6], [9], [18] 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, the product of 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 quadratic in k. Furthermore, the zero-knowledge proof of this computation makes the communication complexity quadratic in k.

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

3.1 Point Representation

Given a set $\mathtt{X}$, and a public set of integers S, which is a set of ℓ distinct elements in $\mathbb{Z}_p$, we define the PR of $\mathtt{X}$ as follows: First, we construct a polynomial $f(x)$ having $\mathtt{X}$ as a set of roots, and then we evaluate $f(x)$ at all elements in S as $\{f(s)\}_{s\in S}$. Then, the resulting set of pairs of a point in S and the corresponding values on $f(x)$, $\{(s, f(s))\}_{s\in S}$ is defined as the PR of $\mathtt{X}$ with S. We denote this conversion from $\mathtt{X}$ and S to PR by

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

where $k < \ell$.

To reduce the probability that a random element accidentally belongs to a private dataset during 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 $\mathtt{P}$ as a subset of $\mathbb{Z}_p$ of all elements of the form $a\|0^\lambda$, where λ is the security parameter. If we set the size of p is equal to 2λ, then the probability that a random element is a member of $\mathtt{P}$ is negligible in λ.

This restriction of $\mathtt{P}$ has some advantages: Let $f_i(x)$ be a private polynomial of a player $\mathcal{P}_i$ and $r_i(x)$ be a random polynomial for each i. Then, the intersection of $\mathtt{P}$ and the set of all roots of the polynomial $\sum_i f_i(x) \cdot r_i(x)$, called the intersection polynomial, is equal to the intersection of all $\mathcal{P}_i$'s inputs with overwhelming probability in λ. Lemma 1 proves the above and shows that $\sum_i f_i(x) \cdot r_i(x)$ leaks no information except for set intersections.

Lemma 1 ([18]): Let $f(x), h(x) \in \mathbb{Z}_p^\alpha[x]$ with $\gcd(f(x), h(x)) = 1$, and $r(x), s(x) \xleftarrow{\$} \mathbb{Z}_p^\beta[x]$, where $\beta \geq \alpha$. Then, $f(x) \cdot r(x) + h(x) \cdot s(x) = \gcd(f(x), h(x)) \cdot u(x)$, where $u(x)$ is uniformly distributed in $\mathbb{Z}_p^{\alpha+\beta}[x]$[†].

3.2 Arithmetic Operations on Point Representation

If the PRs of two k-degree polynomials $f(x)$ and $h(x)$, i.e., $\{(s, f(s))\}_{s\in S}$ and $\{(s, h(s))\}_{s\in S}$ are given, the addition and multiplication of the two polynomials are simply given 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, making them linear in degree k. This is our key observation in reducing complexity from quadratic to linear during the Online phase. (The Online phase contains all processes in the protocol except the local computation for input/output data conversions.) Moreover, to prove these operations, which are linear in k, zero-knowledge proofs also require only linear-size communications and computations.

3.3 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 set of integers for PR. When $f_i(x)$ is the private polynomial of $\mathcal{P}_i$, we denote the encryption of PR of $f_i(x)$ by $\mathsf{Enc}^S_{pk}(f_i(x))$, which is a set of pairs $\{(s, \mathsf{Enc}_{pk}(f_i(s)))\}_{s\in S}$. We define two operations over encrypted PR:

(1) $\boxed{\oplus_{i\in[1,n]}\mathsf{Enc}^S_{pk}(f_i(x))}$: Given encrypted polynomials PRs $\{\mathsf{Enc}^S_{pk}(f_i(x))\}_{i\in[1,n]}$, we can compute the sum of encrypted PRs as

$$\{(s, \oplus_{i\in[1,n]}\mathsf{Enc}_{pk}(f_i(s)))\}_{s\in S} = \left\{\left(s, \mathsf{Enc}_{pk}\left(\sum_{i\in[1,n]} f_i(s)\right)\right)\right\}_{s\in S}$$

and denote it by $\oplus_{i\in[1,n]}\mathsf{Enc}^S_{pk}(f_i(x))$

(2) $\boxed{h \otimes \mathsf{Enc}^S_{pk}(f(x))}$: Given an unencrypted PR $\{(s, h(s))\}_{s\in S}$ and an encrypted PR $\mathsf{Enc}^S_{pk}(f(x))$, 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 $h \otimes \mathsf{Enc}^S_{pk}(f(x))$.

4. Application to PPSI Protocols

PR is useful for designing PPSI protocols. In this section, we present two protocols for performing the intersection of players' private sets without revealing players' inputs.

4.1 Construction in HBC Adversary Model

We present a PPSI-HBC protocol that is secure against an

[†]The original lemma in [18] stated for arbitrary rings satisfying some conditions that all fields also satisfy. In this paper, we use the field $\mathbb{Z}_p$, so we simply state the lemma for the case of $\mathbb{Z}_p$.

Public Parameters: 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} (modified ElGamal), $\mathbb{G}$ (base group), and $\mathtt{P} \subset \mathbb{Z}_p$ (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 the public key pk of Enc_{pk}. The 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 $\mathsf{Conv}_{\mathtt{S}\to\mathtt{P}}(\mathtt{X_i}, S) = \{(s, f_i(s))\}_{s\in S}$ and encrypt them to $\mathsf{Enc}^S_{pk}(f_i(x))$.
(2) Choose random k-degree polynomials $\{r_{i\ell}(x) \in \mathbb{Z}_p[x]\}_{\ell\in[1,n]}$ and encode them to $\{\{r_{i\ell}(s)\}_{s\in S}\}_{\ell\in[1,n]}$.

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

(1) Send $\mathsf{Enc}^S_{pk}(f_i(x))$ to all players.
(2) For each $\ell \in [1, n]$, compute $C_{i\ell} = r_{i\ell}(x) \otimes \mathsf{Enc}^S_{pk}(f_\ell(x))$ and broadcast them to all players.
(3) Compute $C = \oplus_{i,\ell\in[1,n]} C_{i\ell} = \{(s, \mathsf{Enc}_{pk}(\sum_{i,\ell\in[1,n]} r_{i\ell}(s)f_\ell(s)))\}_{s\in S}$.
(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 $\mathcal{P}_i$ extracts the roots of $I(x)$ from $\{(s, g^{I(s)})\}_{s\in S}$.

Fig. 1 PPSI-HBC protocol against HBC adversary.

HBC adversary in Fig. 1.

Look into Output Data Conversion: After finishing the Online phase, each player $\mathcal{P}_i$ has the pairs $\{(s, g^{I(s)})\}_{s\in S}$, where $I(x)$ is the intersection polynomial of degree $2k$. To extract all the roots of $I(x)$ that are contained $\mathtt{P}$, first player $\mathcal{P}_i$ 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 that 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 states that $I(x) = \sum_{i,\ell\in[1,n]} r_{i\ell}(x) \cdot f_\ell(x)$ is equal to $\gcd(f_1(x), \cdots, f_n(x)) \cdot u(x)$ for some random polynomial $u(x)$. Each root of $u(x)$ will be included in $\mathtt{P}$ with only negligible probability in λ, and hence the 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 the set intersection in the 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 nontrivial information about other player's private inputs during the protocol PPSI-HBC with non-negligible probability. The intersection polynomial $\sum_{i,\ell\in[1,n]} r_{i\ell}(x) \cdot f_\ell(x)$ leaks no information except for the set intersection. This comes from Lemma 1 and the fact that, for $\forall\ell$, $\sum_{i\in[1,n]} r_{i\ell}(x)$ is indistinguishable from a random polynomial in $\mathbb{Z}^k_p[x]$ from the viewpoint of the coalition of fewer than n HBC adversaries. Therefore, no HBC adversary can obtain any information other than the intersection with non-negligible probability.

Complexity: Trivial computations of polynomial interpolation, and evaluations of a k-degree polynomial at k points have quadratic complexity in k; thus, we cannot attain quasi-linear complexity in k. We show how to speed up these polynomial operations in Sect. 5 so that the computational complexity of Input/Output Data Conversion phases becomes quasi-linear in k. In Online phase, each player computes $O(nk)$ exponentiations and $O(n^2)$ multiplications, and broadcasts $O(nk)$ group elements.

4.2 Distributed Zero-Knowledge Proofs

To adapt the PPSI-HBC protocol in the HBC model to the malicious adversary model, we need zero-knowledge proofs of the following statements involving polynomials encrypted by the additive homomorphic encryption Enc_{pk}:
• $\mathsf{P}[\mathsf{Enc}^S_{pk}(f_1(x)), \mathsf{Enc}^S_{pk}(f_2(x)(x)), \mathsf{Enc}^S_{pk}(f_3)]$ holds if $\mathsf{Enc}^S_{pk}(f_3(x)) = f_1(x) \otimes \mathsf{Enc}^S_{pk}(f_2(x))$.
• $\mathsf{D}[\mathsf{Enc}^S_{pk}(f(x))]$ holds if $f(x) \in \mathbb{Z}^k_p[x]$.
• $\mathsf{DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f(x)))]$ holds if $f(x) \in \mathbb{Z}^k_p[x]$ and $f(1) = 1$ (i.e., $f(x) \neq 0$).
Here $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f(x)))$ is a commitment to an encrypted polynomial $\mathsf{Enc}^S_{pk}(f(x))$ defined as

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

where Com'_{ck} is the trapdoor commitment, ck is the public parameter of Com_{ck}, and c_s is a random integer. We use the Pedersen commitment scheme [21] as Com'_{ck} in this paper. To open a committed encryption, for $\forall s \in S$ open c_s, then the receiver can recover $\mathsf{Enc}^S_{pk}(f(x))$ 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 so is Com[†]. Moreover, if Com' is an additive homomorphic

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

Public Parameters: Number of players n, maximum threshold of corrupted players t satisfying $2t+1 \le n$, maximum size of each dataset k, set S of size at least $2k+1$, and descriptions of Enc_{pk} (modified ElGamal), $\mathbb{G}$ (base group), Com_{ck} (commitment scheme using Pedersen commitment scheme), and $\mathtt{P} \subset \mathbb{Z}_p$ (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}. The 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 $\mathsf{Conv}_{\mathtt{S}\to\mathtt{P}}(\mathtt{X_i}, S) = \{(s, f_i(s))\}_{s\in S}$ for the private polynomial $f_i(x) = (\prod_{a_t\in\mathtt{X_i}}(x-a_t)(1-a_t)^{-1})$, and encrypt it to $\mathsf{Enc}^S_{pk}(f_i(x))$.
(2) Choose n random polynomials in $\mathbb{Z}_p^k[x]$, $\{r_{i\ell}(x)\}_{\ell\in[1,n]}$ and encrypt them to $\{\mathsf{Enc}^S_{pk}(r_{i\ell}(x))\}_{\ell\in[1,n]}$.

(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 the set of indices of players that have proved their well-formed input (that is, verifiably secret-shared their input and passed the proof D-ZKPK-DO), and R is the set of indices of players that have passed all zero-knowledge proofs protocols. Next, $\mathcal{P}_i$ decrypts it via a threshold decryption protocol together with other players:

(1) Set $M = [1,n]$ and $R = [1,n]$, commit to $\mathsf{Enc}^S_{pk}(f_i(x))$ to all players, (that is, send $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i(x))) = \{(s, \mathsf{Com}'_{ck}(c_{is}), \mathsf{Enc}_{pk}(f_i(s)+c_{is}))\}_{s\in S}$), and verifiably secret-share $\{c_{is}\}_{s\in S}$. If player $\mathcal{P}_j$ fails the verifiable secret-sharing of his input, then set $M := M\setminus\{j\}$ and $R := R\setminus\{j\}$.
(2) Play the prover part in the proof protocol D-ZKPK-DO[$\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i(x)))$] 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 $\mathsf{Enc}^S_{pk}(f_j(x))$ for $j\in M$ is computed by recovering shared values $\{c_{js}\}_{s\in S}$.
(4) Compute $\{C_{i\ell} = r_{i\ell}(x)\otimes\mathsf{Enc}_{pk}(f_\ell(x))\}_{\ell\in M}$ and broadcast them together with $\{\mathsf{Enc}_{pk}(r_{i\ell}(x))\}_{\ell\in M}$ to all players.
(5) Run D-ZKPK-D[$\mathsf{Enc}^S_{pk}(r_{i\ell}(x))$] and D-ZKPK-P[$\mathsf{Enc}^S_{pk}(r_{i\ell}(x)), \mathsf{Enc}^S_{pk}(f_\ell(x)), C_{i\ell}$] protocols as the prover for $\ell\in M$, and also play the verifier part in D-ZKPK-D[$\mathsf{Enc}^S_{pk}(r_{j\ell}(x))$] and D-ZKPK-P[$\mathsf{Enc}^S_{pk}(r_{j\ell}(x)), \mathsf{Enc}^S_{pk}(f_\ell(x)), C_{j\ell}$] 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[$\mathsf{Enc}^S_{pk}(r_{j\ell}(x))$] or D-ZKPK-P[$\mathsf{Enc}^S_{pk}(r_{j\ell}(x)), \mathsf{Enc}^S_{pk}(f_\ell(x)), C_{j\ell}$], then set $R := R\setminus\{j\}$.
(6) Compute $C = \oplus_{i\in R,\ell\in M} C_{i\ell} = \mathsf{Enc}^S_{pk}(\sum_{i\in R,\ell\in M} r_{i\ell}(x)\cdot f_\ell(x))$.
(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 extracts the roots of $I(x)$ from $\{(s, g^{I(s)})\}_{s\in S}$.

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

commitment, then so is Com, as Enc_{pk} is an additive homomorphic encryption.

For each statement, we construct a so-called sigma protocol, i.e., a three-round public-coin interactive proof with special honest-verifier zero-knowledge and special strong soundness. We denote these sigma protocols as HVZKPK-P, HVZKPK-D, and HVZKPK-DO. We relegate the descriptions of these sigma protocols to Appendix since these proof systems are simple extensions of well-known sigma protocols for proving the knowledge of discrete logarithms, representations, and arithmetic relations between these representations. For example, the sigma protocol for HVZKPK-P[$\mathsf{Enc}_{pk}(f_1(x)), \mathsf{Enc}_{pk}(f_2(x)), \mathsf{Enc}_{pk}(f_3(x))$] is a conjunction of $|S|$ instances, i.e., one for each $s\in S$, of a sigma protocol for proving the knowledge of plaintexts $f_1(s), f_2(s)$, and $f_3(s)$ encrypted in $\mathsf{Enc}_{pk}(f_1(s)), \mathsf{Enc}_{pk}(f_2(s))$, and $\mathsf{Enc}_{pk}(f_3(s))$, which satisfy the relation that $f_1(s)\cdot f_2(s)$ 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 Pedersen VSS [21], which is 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, and D-ZKPK-DO. It is easy to see that the distributed proof protocol has an efficient straight-line simulator (a simulator that controls more than half of the players can steer the distributed coin toss to hit a random challenge of its choice; therefore, it can use a special HVZK simulation of the underlying sigma protocol) and an efficient extraction of the adversary's witnesses (the simulator can run a second execution on the side that produces different challenges and then on the main execution path, and the sigma protocol guarantees witness extraction from successful responses to two different challenges).

4.3 Construction in Malicious Adversary Model

We describe a protocol PPSI-MAL that securely computes multi-party set intersections in the presence of a malicious adversary in Fig. 2. PPSI-MAL is based on PPSI-

HBC presented in Sect. 4.1, with the following essential changes made: We add distributed protocols for zero-knowledge proofs of statements $\mathsf{DO}[\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f_i(x)))]$, $\mathsf{D}[\mathsf{Enc}^S_{pk}(r_{i\ell}(x))]$, and $\mathsf{P}[\mathsf{Enc}^S_{pk}(r_{i\ell}(x)), \mathsf{Enc}^S_{pk}(f_\ell(p)), \mathsf{Enc}^S_{pk}(r_{i\ell}(x) \cdot f_\ell(x))]$. We rely on the fact that the distributed key generation and the threshold decryption protocols DKG and TDec of [7], [11] are secure in the malicious model Furthermore, to prevent players from setting their private polynomials to zero polynomials, we add the 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

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

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

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

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

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

Theorem 1: (Security of PPSI-MAL) If Enc_{pk} is a semantically secure additive homomorphic encryption, the 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 the 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 a multi-party set intersection function, i.e., $\mathtt{X} = \mathtt{X}_1 \cap ... \cap \mathtt{X}_n$. The simulator's goal is twofold: to extract the effective inputs $\{\mathtt{X}_i\}_{i \in C}$ that the corrupted players enter into the PPSI computation, and to simulate the same viewpoint as $\mathcal{A}$ when interacting with the honest players on inputs $\{\mathtt{X}_i\}_{i \in H}$ but given only the 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 the 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) and (2) of the Online phase in this execution.

2. $\mathcal{S}$ runs until the end of step (2). Let D be the set of players in C that correctly carry out a verifiable secret-sharing procedure in step (1) and pass the D-ZKPK-DO proof in 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 $\mathcal{S}$ finds a corresponding execution such that $\mathcal{P}$ passes, and the challenge on the proof is different from the original one. In each run, $\mathcal{S}$ performs Input Data Conversion and step (1) of the Online phase on coins δ_0 and δ_1, respectively, and step (2) on fresh random coins. (Claim 4 below shows that the expected number of such rewindings is polynomial in the 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 complexity in k, so that it takes polynomial complexity 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, and δ_2, and performs step (3) to reconstruct the encrypted polynomials $\{\mathsf{Enc}^S_{pk}(f'_i)\}_{i \in H}$, where each f'_i, $i \in H$,

encodes the set $\mathtt{X}$. The Pedersen VSS allows $\mathcal{S}$ to equivocate on the reconstructed secret $\{c_{is}\}_{s\in S}$, and hence on $\mathsf{Enc}^S_{pk}(f)$.

4. The simulator simply performs the rest of the protocol on behalf of the players in H using random coins δ_4-δ_7 and outputs the execution, i.e., the execution between $\mathcal{A}$ and players in H using inputs $\{f'_i\}_{i\in H}$ and coins $\delta_0, \delta_1, \delta_2$, and δ_4-δ_7.

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

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

Let 2^C be the 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:

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

for some polynomial $poly(\cdot)$. Let us explain the second equality. Let $E = \sum_{k=1}^{\infty} k(1-\epsilon'_{\mathcal{P}})^{k-1}$. Then, $(1-\epsilon'_{\mathcal{P}})E = \sum_{k=2}^{\infty}(k-1)(1-\epsilon'_{\mathcal{P}})^{k-1}$. The difference between E and $(1-\epsilon'_{\mathcal{P}})E$ is

$$\epsilon'_{\mathcal{P}}E = 1 + \sum_{k=2}^{\infty}(1-\epsilon'_{\mathcal{P}})^{k-1} = \frac{1}{\epsilon'_{\mathcal{P}}}.$$

Therefore, $E = \frac{1}{\epsilon'^2_{\mathcal{P}}}$ implies that the second equality holds. Next, we argue that the third equality holds. The summation $\sum_{D\in 2^C}\sum_{\mathcal{P}\in D}$ is the same as $\sum_{(D,\mathcal{P})\in \mathsf{S}_0}$, where $\mathsf{S}_0 = \{(D,\mathcal{P})|\forall D \in 2^C$, and $\forall \mathcal{P}$ such that $\mathcal{P} \in D\}$. On the other hand, the summation $\sum_{\mathcal{P}\in C}\sum_{D\ni\mathcal{P}}$ is the same as $\sum_{(\mathcal{P},D)\in \mathsf{S}_1}$, where $\mathsf{S}_1 = \{(\mathcal{P},D)|\forall \mathcal{P} \in C$, and $\forall D$ 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 in 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 the claim. □

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

Now we argue that from $\mathcal{A}$'s viewpoint, $\mathcal{S}$'s simulated transcript is indistinguishable from the real protocol's transcript. First, we consider the random space over which $\mathcal{S}$ uses randomness. In step 1 of the simulation, $\mathcal{S}$ chooses δ_0, δ_1, and δ_2 at random, and runs until the end of step (2) of the protocol. In step 3 of the simulation, $\mathcal{S}$ rewinds $\mathcal{A}$ to the end of step (2) of the protocol on coins δ_0, δ_1, and δ_2, and then it performs the rest of the protocol using randomness δ_4-δ_7. Therefore, simulation uniformly uses all randomness.

Next, we show that the simulated transcript is indistinguishable from the real protocol's transcript. In step 1 of the simulation, $\mathcal{S}$ commits encrypted constant polynomials as the honest players' input instead of k-degree polynomials. $\mathcal{A}$, however cannot distinguish because of the perfectly hiding property of Com_{ck}. Furthermore, $\mathcal{S}$ can simulate all players in H passing D-ZKPK-DO in step 2 of the simulation since D-ZKPK-DO is a straight-line simulatable ZKPK protocol. In step 3 of the simulation, D-ZKPK-DO is indistinguishable from the real protocol because of the trapdoor opening property of the Pedersen commitment scheme. Since $\mathcal{S}$ follows the description of the protocol in step 4, the adversarial viewpoint is identical in both the simulation and the real protocol. At the end of the simulation, we have the polynomial $I(x)$ that is distributed as a product of an $|\mathtt{X}|$-degree polynomial that 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 the protocol, the adversary cannot obtain information about the honest players' input other than the set intersection and makes the result of the protocol wrong. This completes the proof. □

5. Complexity Analysis

We use the notations exp and mul to respectively denote the numbers of exponentiations and multiplications in $\mathbb{G}$. In the PPSI-MAL protocol excluding the Input/Output Data Conversion phases, each player is required to perform DKG, TDec, verifiable secret sharings, distributed zero-knowledge proofs, and multiplications/additions of encrypted polynomials. Almost all operations are dominated by the distributed zero-knowledge proofs, which 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 the multiplication of n k-degree encrypted polynomials.) Each player also computes $O(n^2k)$ mul in step (6) for the addition of n^2 $2k$-degree encrypted polynomials. If we assume that $O(1)$ exp is equal to $O(\lambda)$ mul, then the overall computational cost excluding 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 the Input Data Conversion phase, and $O(k^2)$ exp for conversions in the Output Data Conversion phase. For the Input/Output Data Conversion phases, we need to carry out polynomial interpolation, and evaluate the polynomial at multiple points, meaning that both computations require quadratic numbers of operations in degree of the polynomial. One of the best-known algorithms for performing polynomial interpolation or evaluation is to use the Fast Fourier Transform (FFT) for which polynomial interpolation/evaluation requires $O(k(\log k)^2)$ field operations for a k-degree polynomial. We refer to [26], [27] for details of the FFT. Note that the FFT can be applied to polynomials even when their values or coefficients are in the exponent since the FFT uses only scalar multiplications and additions. Similarly, polynomial evaluation at multiple points or polynomial interpolation using the FFT can also be applied to polynomials even when their coefficients or values are in the exponent. Therefore, we can utilize the FFT in the Output Data Conversion phase to reduce the computational complexity of the Output Data Conversion phase from $O(k^2)$ exp to $O(k(\log k)^2)$ exp. By applying the FFT to the Input/Output Data Conversion phases, we can reduce the computational overhead to $O(nk(\log k)^2)$ multiplications in $\mathbb{Z}_p$ with $O(nk + k(\log k)^2)$ 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 $\mathbb{Z}_p$. This can be simply written as $\tilde{O}(n^2\lambda + (n\lambda + n^2)k)$ mul.

Estimating the communication overhead is easier than estimating the computational overhead. Each player in PPSI-MAL is required to transfer $O(n^2k)$ encryptions in (4) and (5) of the Online phase. Since the other steps' communication overhead in the Online phase is smaller than the overhead of these steps, the overall communication overhead is $O(n^2k)$. Therefore, we attain linear communicational complexity in k. Since each player performs a linear computation in k in the Online phase, the required zero-knowledge proofs are also linear in k so that we attain linear communication complexity in k.

We compare our PPSI protocol that is secure against a malicious adversary with previous works in Table 1.

To utilize the FFT, we need a restriction on $\mathbb{Z}_p$ such that $\mathbb{Z}_p$ has a primitive 2^mth root of unity, where 2^m is larger than $2k$. The existence of a primitive 2^mth 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 we obtain a prime, we will get such a prime in $O(\log p)$ trials. (That is, the running time to generate a prime p appropriate for the FFT is linear in the security parameter, $O(\lambda)$.) Hence, r must be at least $O(\log p)$, which means that $m < \log p - \log\log p$. This 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. The lower bound of generic attacks on the decisional Diffie-Hellman problem, whose hardness is equivalent to the semantic security of ElGamal encryption, is independent of the format of p [25]. If we use an elliptic curve group without pairing, then there is no known nongeneric attack on 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 the FFT. Furthermore, the multip-arty protocol using Paillier encryption requires us to generate a composite integer that is difficult to factor as part of the public key in the initial step without revealing the corresponding secret key. This process is much costly than generating such a public key and a secret key for ElGamal encryption.

Table 1 Comparison of PPSI in the presence of a malicious adversary.

Protocol	Communication (the number of bits)	Computation (the number of mul)
[19]	$O(n^2k^2\lambda)$	$O(n^2k + n\lambda k^2)$
[22]	$O(c^2k^2\lambda)$	$O(c^2\lambda k^2)$
[23], [24]	$O(nk^2\lambda)$	$O(n\lambda k^2)$
Ours	$O(n^2k\lambda)$	$\tilde{O}(n^2\lambda + (n\lambda + n^2)k)$

c: number of corrupted players

Remark. The most expensive operations in Input/Output Data Conversion are the polynomial arithmetic when values are given in the exponent. If we use Paillier encryption [20] instead of the modified ElGamal encryption as an additive homomorphic encryption, then we do not require the expensive polynomial arithmetic since Paillier encryption support a complete decryption. Then, if we use the Paillier encryption and do not use the FFT so that the polynomial arithmetic is quadratic in k, the polynomial arithmetic will involve field operations dominated by cryptographic operations such as encryptions and exponentiations, so that we may obtain the PPSI protocol with linear complexity. However, in the security proof of the PPSI-MAL protocol, the simulator should factor polynomials to extract their roots. However, 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 [19] has a similar problem to ours if they use Paillier encryption. They suggested two options for proving security without factoring polynomials. One is to prove security in the random oracle model. If every root of each polynomial is forced to be hash-output, then the simulator can extract each root of each polynomial by simulating the random oracle. The other is to prove that each polynomial is a product of degree-1 polynomials in the manner of a zero-knowledge proof with strong soundness. Then, the simulator can extract all roots from the property of the strong soundness of zero-knowledge proofs. However, these zero-knowledge proofs impose $O(k^3)$ complexity.

6. Conclusion and Further Works

In this study, we proposed a privacy-preserving set intersection protocol satisfying (1) linear communication and quasi-

linear computation overheads in the size of the data input, and (2) robustness in the presence of a malicious adversary. The proposed protocol attains linear/quasi-linear complexities; however, it still has some unresolved issues.

One interesting open problem is the construction a PPSI protocol with linear complexity in both the size of the data input and the number of players. Sang and Shen [24] attained linear complexity in the number of players, and our protocol achieves quasi-linear complexity in the size of the data input. There is, however, no PPSI protocol with linear or quasi-linear complexity in both parameters.

Finding practical solutions to other privacy-preserving set operations, such as set union [10] and threshold set union [18], are also interesting open problems.

Acknowledgements

The first author was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0001243)

References

[1] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," TCC 2007, LNCS, pp.137–156, Springer-Verlag, 2007.
[2] J. Camenisch and G.M. Zaverucha, "Private intersection of certified sets," Financial Cryptography 2009, LNCS, pp.108–127, Springer-Verlag, 2009.
[3] J. Cohen and M. Fischer, "A robust and verifiable cryptographically secure election scheme," FOCS, pp.372–382, 1985.
[4] E.D. Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in malicious model," ASIACRYPT, vol.6477 of LNCS, pp.213–231, Springer, 2010.
[5] E.D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," Financial Cryptography, LNCS, pp.143–159, Springer, 2010.
[6] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," ACNS 2009, vol.5536 of LNCS, pp.126–142, Springer-Verlag, 2009.
[7] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," CRYPTO, vol.435 of LNCS, pp.307–315, Springer, 1989.
[8] P. Feldman, "A practival scheme for non-interactive verifiable secure sharing," IEEE Annual Symposium on Foundations of Computer Science, pp.427–437, 1987.
[9] M. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set-intersection," Advances in Cryptology-EuroCrypt'04, vol.3027 of LNCS, pp.1–19, Springer-Verlag, 2004.
[10] K.B. Frikken, "Privacy-preserving set union," J. Katz and M. Yung, ed., ACNS, vol.4521 of LNCS, pp.237–252, Springer, 2007.
[11] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," J. Cryptology, vol.20, no.1, pp.51–83, Springer, New York, 2007.
[12] O. Goldreich, The Foundations of Cryptography, vol.2, Cambridge University Press, 2004.
[13] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," ACM STOC, pp.218–229, 1987.
[14] C. Hazay and Y. Lindell, Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries, TCC 2008, LNCS, pp.155–175, Springer-Verlag, 2008.
[15] C. Hazay and K. Nissim, Efficient set operations in the presence of malicious adversaries, In Public Key Cryptography, LNCS, pp.312–331, Springer, 2010.
[16] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," O. Reingold, ed., TCC 2009, vol.5444 of LNCS, pp.577–594, Springer-Verlag, 2009.
[17] S. Jarecki and X. Liu, Fast secure computation of set intersection, Security and Cryptography for Networks, vol.6280 of LNCS, pp.418–435, Springer, 2010.
[18] L. Kissner and D. Song, "Privacy-preserving set operations," V. Shoup, ed., Advances in Cryptology-Crypto'05, vol.3621 of LNCS, pp.241–257, Springer-Verlag, 2005.
[19] L. Kissner and D. Song, Privacy-preserving set operations, Technical Report CMU-CS-05-133, Carnegie Mellon University, 2006.
[20] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," J. Stern, ed., Advances in Cryptology-EuroCrypt'99, vol.1592 of LNCS, pp.223–238, Springer-Verlag, 1999.
[21] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," Advances in Cryptology-Crypto'91, vol.576 of LNCS, pp.129–140, Springer-Verlag, 1991.
[22] Y. Sang and H. Shen, "Privacy preserving set intersection protocol secure against malicious behaviors," Proc. Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies, pp.461–468, Washington, DC, USA, 2007, IEEE Computer Society.
[23] Y. Sang and H. Shen, "Privacy preserving set intersection based on bilinear groups," Proc. Thirty-First Australasian Conference on Computer Science, vol.74, pp.47–54, Darlinghurst, Australia, 2008, Australian Computer Society.
[24] Y. Sang and H. Shen, "Efficient and secure protocols for privacy-preserving set operations," ACM Trans. Information and Systems Security, vol.13, 2009.
[25] V. Shoup, "Lower bounds for discrete logarithms and related problems," EUROCRYPT, pp.256–266, Springer, 1997.
[26] V. Shoup, A Conputational Introduction to Number Theory and Algebra, 2nd ed., Cambridge University Press, 2008.
[27] J. von zur Gathen and J. Gerhard, Modern Computer Algebra, Cambridge University Press, 2003.
[28] A.C.-C. Yao, "Protocols for secure computations," FOCS, pp.160–164, 1982.

Appendix: Honest-Verifier Zero-Knowledge Proofs

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

We use the notation $\mathsf{Enc}_{pk}(\cdot;\cdot)$ and $\mathsf{Com}'_{ck}(\cdot;\cdot)$ instead of $\mathsf{Enc}_{pk}(\cdot)$ and $\mathsf{Com}'_{ck}(\cdot)$ to indicate randomizers in addition to messages in an encryption scheme and a commitment scheme, respectively. We assume that both Enc_{pk} and

Com'_{ck} have an additively homomorphic property on both message and randomness, for example, the modified ElGamal encryption and Pedersen commitment scheme, respectively.

(1) Sigma Protocol for Product Relation HVZKPK-P

We show a sigma protocol for the relation HVZKPK-P between the prover P and the verifier V to prove that the equality $\mathsf{Enc}^S_{pk}(f_3(x)) = f_1(x) \otimes \mathsf{Enc}^S_{pk}(f_2(x))$ for the given encrypted polynomials $\mathsf{Enc}^S_{pk}(f_1(x))$, $\mathsf{Enc}^S_{pk}(f_2(x))$, and $\mathsf{Enc}^S_{pk}(f_3(x))$, is simply a conjunction of $|S|$ proof systems, one for each of three ciphertexts $\mathsf{Enc}_{pk}(f_1(s)), \mathsf{Enc}_{pk}(f_2(s))$, and $\mathsf{Enc}_{pk}(f_3(s))$ 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: It is 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 the 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 other values are uniquely determined; in particular, there is only one corresponding (a', r'_a, r', A', C') for each $(\tilde{a}, \tilde{r}_a, \tilde{r})$. Let us consider the real distribution $(a', r'_a, r', A', C', \tilde{a}, \tilde{r}_a, \tilde{r})$ generated by the prover. In the real distribution, a', r'_a, and r' are uniformly distributed, and other values are calculated using a', r'_a, and r'. If we consider the restricted distribution $(\tilde{a}, \tilde{r}_a, \tilde{r})$, then $(\tilde{a}, \tilde{r}_a, \tilde{r})$ is uniformly distributed. Furthermore, there exists a one-to-one correspondence between (a', r'_a, r') and $(\tilde{a}, \tilde{r}_a, \tilde{r})$. Therefore, the simulated distribution is exactly the same as 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 but 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 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 randomness $ar_b + (e - e^*)^{-1}(\tilde{r} - \tilde{r}^*)$.

(2) Sigma Protocol for Product Relation HVZKPK-D

Let $L_{(k,S)}$={ $\mathsf{Enc}^S_{pk}(f(x)) \mid f(x) \in \mathbb{Z}^k_p[x]$} 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 a set of integers such that $|S| \geq k + 1$, $f(x)$ is a k-degree polynomial, 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 $\mathsf{Enc}^S_{pk}(f(x)) \in L_{(k,S)}$.

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

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

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 $f(x)$ and $\tilde{f}(x)$ have degrees less than or equal to k, so does $f'(x)$. In the simulated distribution of protocol transcripts $(\{C'_s\}_{s\in S}, \tilde{f}(x), \{\tilde{r}_s\}_{s\in S})$, $\tilde{f}(x)$ 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 the real protocol, $f'(x)$ and $\{r'_s\}_{s\in S}$ are chosen at random, and then $\tilde{f}(x)$ and $\{\tilde{r}_s\}_{s\in S}$ are computed. If we consider only the distributions of $\tilde{f}(x)$ and $\{\tilde{r}_s\}_{s\in S}$, they are uniformly distributed, and there exists a one-to-one correspondence between $(\tilde{f}(x), \{\tilde{r}_s\}_{s\in S})$ and $(f'(x), \{r'_s\}_{s\in S})$. Therefore, there exist unique values of $(f(x), \{r'_s\}_{s\in S}, \{C'_s\}_{s\in S})$ for each $(\tilde{f}(x), \{\tilde{r}_s\}_{s\in S})$, so that the simulated distribution is identical to the real distribution.

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

From (2) and (3), we know that, 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

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

Since $\tilde{f}(x)$ and $\tilde{f}^*(x) \in \mathbb{Z}_p^k[x]$, $(e - e^*)^{-1}(\tilde{f}(x) - \tilde{f}^*(x)) \in \mathbb{Z}_p^k[x]$. 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 $(e - e^*)^{-1}(\tilde{f}(x) - \tilde{f}^*(x))$.

(3) Sigma Protocol for Product Relation HVZKPK-DO

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

We let $LO_{(k,S)}$=$\{ \mathsf{Enc}^S_{pk}(f(x)) \mid f(x) \in \mathbb{Z}_p^k[x]$ 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 the randomizer d_s as well as the 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 a set of integers such that $|S| \geq k + 1$, f is a k-degree polynomial $f(x)$, and r_s is chosen at random $\mathbb{Z}_p$.
Prover Input: $f(x)$, $\{(c_s, d_s, r_s)\}_{s\in S}$.
Goal: $\mathsf{Enc}^S_{pk}(f(x)) \in LO_{(k,S)}$.

P $\rightarrow$ **V**: P chooses $f'(x) \xleftarrow{\$} \mathbb{Z}_p^k[x]$ passing a point $(1, 0)$ and $r'_s, c'_s \xleftarrow{\$} \mathbb{Z}_p$ for $s \in S$, and then computes $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f'(x))) = \{(s, D'_s, C'_s)\}_{s\in S}$, 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 $\mathsf{Com}_{ck}(\mathsf{Enc}^S_{pk}(f'(x)))$ to V.

V $\rightarrow$ **P**: V chooses $e \xleftarrow{\$} \mathbb{Z}_p$, and then sends e to P.

P $\rightarrow$ **V**: P computes $\tilde{f}(x) := f'(x) + e \cdot f(x)$, $\tilde{r}_s := r'_s + e \cdot r_s$, $\tilde{c}_s := c'_s + e \cdot c_s$, and $\tilde{d}_s := d'_s + e \cdot d_s$ for $\forall s \in S$, and then, sends V $\tilde{f}(x), \tilde{r}_s, \tilde{c}_s, \tilde{d}_s$.

V: V accepts if (1) $\tilde{f}(x) \in \mathbb{Z}_p^k[x]$, (2) $\tilde{f}(1) = e$, (3) $\mathsf{Enc}_{pk}(\tilde{f}(s) + \tilde{c}_s; \tilde{r}_s) = C'_s \oplus C_s^e$ for $\forall s \in S$ and (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 the corresponding $\{D'_s, C'_s\}_{s\in S}$ as

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

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 $f(x)$ and $\tilde{f}(x)$ are in $\mathbb{Z}_p^k[x]$, so is $f'(x)$, and since $f(1) = 1$ and $\tilde{f}(1) = e$, $f'(1) = 0$ as desired.

In the simulated distribution of the protocol transcripts $(\{D'_s, C'_s, \tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}, \tilde{f}(x))$, $\tilde{f}(x)$, 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 the real protocol, $f'(x)$ and $\{c'_s, d'_s, r'_s\}_{s\in S}$ are chosen at random, and then $\tilde{f}(x)$ and $\{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}$ are computed.

Let us consider the restricted distribution of $\tilde{f}(x)$ and $\{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S}$ in the real distribution. Since $f'(x) = (x - 1) \cdot h'(x)$ for $h'(x) \xleftarrow{\$} \mathbb{Z}_p^{k-1}[x]$ and $f(x) = (x-1) \cdot h(x) + 1$ for some $h(x) \in \mathbb{Z}_p^{k-1}[x]$, $\tilde{f}$ is equal to $f' + ef = (x-1)(h'(x) + eh(x)) + e$, and hence $\tilde{f}(x)$ is uniformly distributed in $\mathbb{Z}_p^k[x]$ while passing the point $(1, e)$. We can easily check that $\tilde{c}_s, \tilde{d}_s$, and $\tilde{r}_s$ are uniformly distributed in $\mathbb{Z}_p$, since there exist one-to-one correspondence between $(\tilde{f}(x), \{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S})$ and $(f'(x), \{c'_s, d'_s, r'_s\}_{s\in S})$. Therefore, there exists unique values of $(f'(x), \{c'_s, d'_s, r'_s\}_{s\in S})$ for each $(\tilde{f}(x), \{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S})$. For each $(f'(x), \{c'_s, d'_s, r'_s\}_{s\in S})$, there exist 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 $(e, \tilde{f}(x), \{\tilde{c}_s, \tilde{d}_s, \tilde{r}_s\}_{s\in S})$ and $(e^*, \tilde{f}^*(x), \{\tilde{c}^*_s, \tilde{d}^*_s, \tilde{r}^*_s\}_{s\in S})$ such that
(1) $e \neq e^*$,
(2) $\tilde{f}(x) \in \mathbb{Z}_p^k[x]$ and passes $(1, e)$, $\mathsf{Com}'_{ck}(\tilde{c}_s; \tilde{d}_s) = D'_s \oplus D^e_s$, 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) $\tilde{f}^*(x) \in \mathbb{Z}_p^k[x]$ and passes $(1, e^*)$, $\mathsf{Com}'_{ck}(\tilde{c}^*_s; \tilde{d}^*_s) = D'_s \oplus D^{e^*}_s$, 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 $\mathsf{Enc}^S_{pk}(f(x)) \in LO_{(k,S)}$. Moreover, the polynomial f can be extracted given the above two proof transcripts.

From (2) and (3), we know that, 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

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

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

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

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

Jung Hee Cheon is a professor in the department of mathematical sciences of Seoul national university (SNU). He received his B.S. and Ph.D. degrees in mathematics from KAIST in 1991 and 1997, respectively. After working as a senior researcher in ETRI and a visiting scientist in Brown university, he was an assistant professor in Information and Communications University (ICU) for two years. His research interests include computational number theory, cryptography and information security. He has been on program committees of many international conferences including Crypto, Eurocrypt, and Asiacrypt. He received the best paper award in Asiacrypt 2008.

Stanislaw Jarecki received a Ph.D. from the Massachusetts Institute of Technology in 2001. After a year at Intertrust's StarLab research group and a year of postdoctoral studies at Stanford, he joined the faculty of the University of California at Irvine. Stanislaw's research concerns cryptography, distributed algorithms, and computer security, with the focus on efficient algorithms for multi-party secure computation tasks. His recent research projects include threshold cryptosystems, multi-signatures, private authentication schemes, covert authentication and computation, password-based authentication, secure computation of set intersection, and privacy-preserving database access.

Jae Hong Seo is a researcher of National Institute of Information and Communications Technology, Japan. He received the B.S. degree in mathematics from Korea University, Korea, in 2004. He received his Ph.D. degree from Seoul National University, Korea, in 2011. His research interests include computational number theory and public-key cryptography.