# Invertible Polynomial Representation for Private Set Operations

Hyung Tae Lee, Hyunsook Hong, and Jung Hee Cheon

ISaC & Department of Mathematical Sciences, Seoul National University,
1 Gwanak-ro, Gwanak-gu, Seoul 151-747, Korea
{htsm1138,hongsuk07,jhcheon}@snu.ac.kr

**Abstract.** In many private set operations, a set is represented by a polynomial over a ring $\mathbb{Z}_\sigma$ for a composite integer $\sigma$, where $\mathbb{Z}_\sigma$ is the message space of some additive homomorphic encryption. While it is useful for implementing set operations with polynomial additions and multiplications, a polynomial representation has a limitation due to the hardness of polynomial factorizations over $\mathbb{Z}_\sigma$. That is, it is hard to recover a corresponding set from a resulting polynomial over $\mathbb{Z}_\sigma$ if $\sigma$ is not a prime.

In this paper, we propose a new representation of a set by a polynomial over $\mathbb{Z}_\sigma$, in which $\sigma$ is a composite integer with *known factorization* but a corresponding set can be efficiently recovered from a polynomial except negligible probability. Note that $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain, so a polynomial may be written as a product of linear factors in several ways. To exclude irrelevant linear factors, we introduce a special encoding function which supports early abort strategy. As a result, our representation can be efficiently inverted by computing all the linear factors of a polynomial in $\mathbb{Z}_\sigma[x]$ whose root locates in the image of encoding function.

When we consider group decryption as in most private set operation protocols, inverting polynomial representations should be done without a single party possessing a factorization of $\sigma$. This is very hard for Paillier's encryption whose message space is $\mathbb{Z}_N$ with unknown factorization of $N$. Instead, we detour this problem by using Naccache-Stern encryption with message space $\mathbb{Z}_\sigma$ where $\sigma$ is a smooth integer with public factorization. As an application of our representation, we obtain a constant round privacy-preserving set union protocol. Our construction improves the complexity than the previous without honest majority assumption. It can be also used for constant round multi-set union protocol and private set intersection protocol even when decryptors do not possess a superset of the resulting set.

**Keywords:** Polynomial Representation, Privacy-preserving Set Union, Polynomial Factorization, Root Finding

## 1 Introduction

*Privacy-preserving set operations* (PPSO) are to compute set operations of participants' dataset without revealing any information other than the result. There has been many proposals to construct PPSO protocols with various techniques such as general MPC [11, 1], polynomial representations [8, 15, 9, 22, 12], pseudorandom functions [13], and blind RSA signatures [6, 5]. While the last two techniques are hard to be generalized into multi-party protocols, polynomial representations combining with Additive Homomorphic Encryption (AHE) schemes enable us to have multi-party PPSO protocols for various operations, including set intersection [15, 9, 22], (over-)threshold set union [15], element reduction [15] and so on. Among these constructions, set intersection protocols run in a number of rounds, but others run in linear of the number of participants.

Let us focus on privacy-preserving set union protocols. There are two obstacles to construct constant round privacy-preserving multi-party set union protocols based on polynomial representations with AHEs. First, in the polynomial representations set union corresponds to polynomial multiplication, which is not supported by an AHE in constant rounds. Second, to recover the union set from the resulting polynomial, we need a root finding algorithm of a polynomial over $\mathbb{Z}_\sigma$, where $\mathbb{Z}_\sigma$ is the message space of AHE.

Recently, Seo et al. [23] proposed a constant round set union protocol based on a novel approach in which a set is represented as a rational function using reversed Laurent series. In their protocol, each participant takes part in the protocol with a rational function whose poles consist of the elements of his set and at the end of the protocol he obtains a rational function whose poles correspond to a set union. Then each participant recovers the denominator of the rational function using an extended Euclidean algorithm and finds the roots of the denominator. Since each rational function is summed up to the resulting function after encrypted under AHE, the first obstacle is easily overcome.

However, a root finding is still problematic on the message space $\mathbb{Z}_\sigma$ of an AHE. Since the message space has unknown order [20] or not a unique factorization domain (UFD) [19, 21, 2] in the current *efficient* AHEs, there is no proper polynomial factorization or root finding algorithm working on the message space. To avoid this obstacle, the authors in [23] utilized a secret sharing scheme. However, it requires computational and communicational costs heavier than the previous construction and requires honest majority assumption for security since their protocol exploits secret sharing scheme to support privacy-preserving multiplications.

**Our Contribution:** Let $\sigma = \prod_{i=1}^{\bar{\ell}} q_i$ for distinct primes $q_i$, larger than the size of the universe of dataset. We propose a new representation of a set by a polynomial over $\mathbb{Z}_\sigma$, in which a corresponding set can be efficiently recovered from a polynomial except non-negligible probability when the factorization of $\sigma$ is given. For a given polynomial $f(x) = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$, if the factorization of $\sigma$ is given, one can obtain all roots of $f$ in $\mathbb{Z}_{q_i}$ for each $i$ by exploiting a polynomial factorization algorithm over a finite field $\mathbb{Z}_{q_i}$ [28]. By reassembling the roots of $f$ in $\mathbb{Z}_\sigma$ using the Chinese Remainder Theorem (CRT), we can obtain all the candidates. However, the number amounts to $d^{\bar{\ell}}$, which is exponential in the size of the universe.

We introduce a special encoding function $\iota$ to exclude irrelevant candidates efficiently. For a polynomial $f = \prod_{i=1}^{d}(x - \iota(s_i)) \in \mathbb{Z}_\sigma[x]$, our encoding function abort most irrelevant candidates without $d^{\bar{\ell}}$ CRT computations, by giving a certain relation among roots of $f$ in $\mathbb{Z}_{q_j}[x]$ and roots of $f$ in $\mathbb{Z}_{q_{j+1}}[x]$. As a result, our encoding function enables us to efficiently recover all the roots of $f$ with negligible failure probability if they are in the image of $\iota$.

**Table 1.** Comparison with Previous Set-Union Protocols

| HBC | Rounds | Communication Cost | Computational Cost | # of Honest Party |
|---|---|---|---|---|
| [15] | $O(n)$ | $O(n^3k\tau_N)$ | $O(n^4k^2\tau_N\rho_N)$ | $\geq 1$ |
| [9] | $O(n)$ | $O(n^2k\tau_N)$ | $O(n^2k^2\tau_N\rho_N)$ | $\geq 1$ |
| [23] | $O(1)$ | $O(n^4k^2\tau_{p'})$ | $O(n^5k^2\rho_{p'})$ | $\geq n/2$ |
| Ours | $O(1)$ | $O(n^3k\tau_N)$ | $O(n^3k^2\tau_N\rho_N)$ | $\geq 1$ |
| Malicious | Rounds | Communication Cost | Computational Cost | # of Honest Party |
| [9] | $O(n)$ | $O((n^2k^2 + n^3k)\tau_N)$ | $O(n^2k^2\tau_N\rho_N)$ | $\geq 1$ |
| [23] | $O(1)$ | $O(n^4k^2\tau_p)$ | $O(n^5k^2\tau_p\rho_p)$ | $\geq n/2$ |
| Ours | $O(1)$ | $O(n^3k^2\tau_N)$ | $O(n^3k^2\tau_N\rho_N)$ | $\geq 1$ |

$n$: the number of participants, $k$: the maximum size of sets

$\tau_N, \tau_{p'}, \tau_p$: the size of modulus $N$ for Paillier encryption scheme or NS encryption scheme, the size $p'$ of representing domain, the order $p$ of a cyclic group for Pedersen commitment scheme, respectively

$\rho_N, \rho_{p'}, \rho_p$: modular multiplication cost of modulus $N$ for Paillier encryption scheme or NS encryption scheme, $p'$ for the size of representing domain, $p$ for the order of a cyclic group for Pedersen commitment scheme, respectively

As an application of our new representation, combining with Naccache-Stern (NS) AHE scheme which is the factorization of the size $\sigma$ of message space is public, we obtain an efficient constant round privacy-preserving set union protocol without honest majority assumption. In Table 1[1], we compare our set union protocols with the previous main results.

Remark that we can easily extend our set union protocol to multi-set union protocol by encoding the same elements differently. We describe details in Section 4.4. The resulting multi-set union protocol is little bit slower than a previous result [12], but the public key size of the utilized encryption in our protocol is $O(1)$, while that of previous work is $O(d)$ for the size $d$ multi-set union.

We also consider transforming previous privacy-preserving set intersection protocol in [15] into a protocol even when decryptors do not possess a superset of the resulting set.

**Related Work:** There are a few researches to construct a privacy-preserving multi-party set and multi-set union protocol. Kissner and Song [15] provided a multi-party set and multi-set union protocol in the honest-but-curious case. Frikken [9] and Sand and Shen [22] presented more efficient multi-party set union protocols and multi-set union protocols in the malicious case, respectively. However, these protocols exploit a mix-net protocol [10] instead of polynomial factorization algorithm, which runs in linear of the number of the number of corrupted players, and hence it cannot have constant round complexity.

Recently, Seo et al. [23] proposed constant round multi-party set union protocols by representing elements in a set as poles of a rational function. But, their constructions hire a secret sharing technique for supporting privacy-preserving multiplications, thus requires an honest majority assumption for security and computational and communicational complexity heavier than previous results.

In case of privacy-preserving multi-set union protocols, Hong et al. [12] proposed a protocol based on El Gamal encryption schemes defined over an extension field $\mathbb{F}_{q^\kappa}$ where $\kappa$ is larger than $nk$ where $n$ is the number of participants and $k$ is the size of datasets. But it suffer from the public key size of the utilized encryption, since the extension degree $\kappa$ of extension field has to be larger than $d$ for the size $d$ multi-set union and hence the public key size is $O(d)$.

**Outline of the Paper:** In Section 2 we look into some components of our privacy-preserving set union protocol, polynomial factorization algorithm, polynomial representation, and AHE schemes. We provide our new polynomial representation that enables us to uniquely factorize a polynomial satisfying some criteria in Section 3. Section 4 provides some privacy-preserving set operation protocols based on our polynomial representation.

## 2 Preliminaries

In this section, we look into polynomial representations of a set for PPSO and introduce efficient AHE schemes utilized in PPSO protocols supporting polynomial operations between encrypted polynomials. Then, we briefly look into root finding algorithms over finite fields and message spaces of AHE scheme for applying to recover a set from a polynomial in polynomial representation.

Throughout this paper, for a polynomial $f(x) = \sum_{i=0}^{\deg f} f[i]x^i \in \mathbb{Z}_\sigma[x]$ and $q \mid \sigma$, $f \bmod q$ denotes a polynomial $\sum_{i=0}^{\deg f} (f[i] \bmod q)x^i \in \mathbb{Z}_q[x]$.

[1] Note that the communication and computational complexity in Table 1 of [23] are for one participant.

### 2.1 Polynomial Representation of a Set

Let $R$ be a commutative ring with unity and $S$ be a subset of $R$. We may represent a set $S$ by a polynomial or a rational function over $R$.

*Polynomial Representation* In some previous works [8, 15, 9, 12, 23], a set $S$ can be represented by a polynomial $f_S(x) \in R[x]$ whose roots are the elements of $S$. That is,

$$f_S(x) := \prod_{s_i \in S} (x - s_i).$$

This representation gives the following relation:

$$f_S(x) + f_{S'}(x) = \gcd(f_S(x), f_{S'}(x)) \cdot u(x)$$

for some polynomial $u(x) \in R[x]$ and hence the roots of a polynomial $f_S(x) + f_{S'}(x)$ are the elements of $S \cap S'$ with overwhelming probability. Also, the roots of $f_S(x) \cdot f_{S'}(x)$ are the elements of $S \cup S'$ as multi-sets.

*Rational function Representation* Recently, Seo et al. [23] introduced a novel representation of a set $S \subset R$ by a rational function $F_S$ over $R$ whose poles consist of the elements of $S$. That is,

$$F_S(x) := \frac{1}{\prod_{s_i \in S}(x - s_i)} = \frac{1}{f_S(x)}.$$

This representation provides the following relation:

$$F_S(x) + F_{S'}(x) = \frac{f_S(x) + f_{S'}(x)}{f_S(x) \cdot f_{S'}(x)} = \frac{\gcd(f_S(x), f_{S'}(x)) \cdot u(x)}{f_S(x) \cdot f_{S'}(x)} = \frac{u(x)}{\mathrm{lcm}(f_S(x), f_{S'}(x))}$$

for some polynomial $u(x) \in R[x]$ which is relatively prime to $\mathrm{lcm}(f_S(x), f_{S'}(x))$. Hence the poles of $F_S(x) + F_{S'}(x)$ are exactly the roots of $\mathrm{lcm}(f_S(x), f_{S'}(x))$, which are the elements of $S \cup S'$ as sets, not multi-sets. This rational function is represented again by an infinite formal power series, so called a *reversed Laurent series* (RLS), in [23].

### 2.2 Additive Homomorphic Encryption

Let us consider a commutative ring $R$ with unity and a $R$-module $G$ where $r \cdot g := g^r$ for $r \in R$ and $g \in G$. Let $\mathsf{Enc}_{\mathsf{pk}} : R \to G$ be a public key encryption under the public key $\mathsf{pk}$. We can define a public key encryption for $f = \sum_{i=0}^{\deg f} f[i]x^i \in R[x]$ as follows:

$$\mathcal{E}_{\mathsf{pk}}(f) := \sum_{i=0}^{\deg f} \mathsf{Enc}_{\mathsf{pk}}(f[i])x^i.$$

Assume $\mathsf{Enc}_{\mathsf{pk}}$ is an AHE such that

$$\mathsf{Enc}_{\mathsf{pk}}(a + b) = \mathsf{Enc}_{\mathsf{pk}}(a)\mathsf{Enc}_{\mathsf{pk}}(b), \quad \mathsf{Enc}_{\mathsf{pk}}(ab) = \mathsf{Enc}_{\mathsf{pk}}(a)^b$$

for $a, b \in R$. Then given two polynomials $f = \sum_{i=0}^{\deg f} f[i]x^i$ and $g = \sum_{i=0}^{\deg g} g[i]x^i$, we can induce homomorphic properties of $\mathcal{E}$ as follows:

- **Polynomial addition**: Given $\mathcal{E}_{\mathsf{pk}}(f)$ and $\mathcal{E}_{\mathsf{pk}}(g)$, it is possible to compute $\mathcal{E}_{\mathsf{pk}}(f+g)$ by calculating $\mathsf{Enc}_{\mathsf{pk}}((f+g)[i]) = \mathsf{Enc}_{\mathsf{pk}}(f[i])\mathsf{Enc}_{\mathsf{pk}}(g[i])$ for all $0 \le i \le \max\{\deg f, \deg g\}$ where $f + g = \sum_{i=0}^{\max\{\deg f, \deg g\}} (f + g)[i]x^i$.
- **Polynomial multiplication**: Given $\mathcal{E}_{\mathsf{pk}}(f)$ and $g$, it is possible to compute $\mathcal{E}_{\mathsf{pk}}(fg)$ by calculating $\mathsf{Enc}_{\mathsf{pk}}((fg)[\ell]) = \prod_{i+j=\ell} \mathsf{Enc}_{\mathsf{pk}}(f[i])^{g[j]}$ for all $0 \le \ell \le \deg f + \deg g$ where $fg = \sum_{\ell=0}^{\deg f + \deg g} (fg)[\ell]x^\ell$.

There are several efficient AHE schemes [19–21, 2]: Under the assumption that factoring $N = p^2q$ is hard, Okamoto and Uchiyama [20] proposed a scheme with $R = \mathbb{Z}_p$ and $G = \mathbb{Z}_N$, in which the order $p$ of the message space $R$ is hidden. With the decisional composite residuosity assumption, Paillier [21] scheme has $R = \mathbb{Z}_N$ and $G = \mathbb{Z}_{N^2}$ for $N = pq$, in which the size of message space is a hard-to-factor composite integer $N$. Naccache and Stern [19] proposed a scheme with $R = \mathbb{Z}_\sigma$ and $G = \mathbb{Z}_N$ under the higher residue assumption, where $N = pq$ is a hard-to-factor integer and $\sigma$ is a product of small primes dividing $\phi(N)$.

In the previous two schemes, it is hard to find the roots of a polynomial in $R[x]$ without knowing a secret key. While, in the Naccache-Stern (NS) scheme, it may be possible to compute some roots of a polynomial in $\mathbb{Z}_\sigma[x]$ since the factorization of $\sigma$ is public. But $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain (UFD) and so the number of roots of $f \in \mathbb{Z}_\sigma[x]$ can be larger than the $\deg f$. In fact, if $f(x) = \prod_{i=1}^{d}(x - s_i) \in R[x]$, then the number of candidates of roots of $f$ is $d^{\bar{\ell}}$ where $\bar{\ell}$ is the number of prime factors of $\sigma$. We will use the NS scheme by introducing a method to efficiently recover all the roots of $f \in \mathbb{Z}_\sigma[x]$ satisfying some criterion.

### 2.3 Root Finding Algorithms

When $R$ is a finite field $\mathbb{F}_q$, we have several efficient root finding algorithms in $R[x]$. Using a square-free decomposition and the Cantor-Zassenhaus algorithm [29, Section 14.4], a polynomial of degree $d$ over a field $\mathbb{F}_q$ is factored in $\widetilde{O}(d^2 \log q)$ field operations. Recently, it has been improved using fast arithmetic into $O(d^{1.5+o(1)})$ field operations by Umans [28]. However, as mentioned above, there is no efficient AHE whose message space is a finite field $\mathbb{F}_q$ with public $q$.

Consider $R = \mathbb{Z}_\sigma$, a message space of NS encryption scheme for $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$ with distinct primes $q_j$'s. Let $f \in \mathbb{Z}_\sigma[x]$ be a polynomial of degree $d$, which is a product of $d$ linear factors $(x - s_i)$'s. Since $\mathbb{Z}_\sigma[x]$ is not a UFD, it is still very hard to recover the exact $(x - s_i)$'s from the polynomial $f$. However, since the factors of $\sigma$ are known, one can apply Umans' polynomial factorization algorithm to find all roots of $f \bmod q_j$ with $\widetilde{O}(d^{1.5})$ field operations in $\mathbb{Z}_{q_j}$ for each $j$. Then one performs the Chinese Remainder Theorem (CRT), which takes $O(\log^2 \sigma)$ bit operations for each candidate of roots. Since each polynomial $f \bmod q_j$ has about $d$ distinct roots, there exist about $d^{\bar{\ell}}$ candidates of roots of $f$ over $\mathbb{Z}_\sigma$. Hence the total complexity becomes $\widetilde{O}(\bar{\ell} d^{1.5} \log^2 q_j + d^{\bar{\ell}} \log^2 \sigma)$ bit operations. However, one can still not determine the exact $s_i$'s since it has no criteria to distinguish the exact $s_i$'s from candidates.

In this paper, we can recover all the exact roots of $f$ satisfying some criteria in $O(\bar{\ell} d^{1.5} \log^2 q_j)$ bit operations using early abort technique. The details are in Section 3.

## 3 Invertible Polynomial Representation

Let $f_S(x) \in \mathbb{Z}_\sigma[x]$ be a polynomial, all of whose roots are the elements in $S \subseteq \mathbb{Z}_\sigma$ for a composite $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$ with distinct primes $q_j$'s. Since $\mathbb{Z}_\sigma[x]$ is not a UFD, it is impossible to recover the exact $S$ from the polynomial $f_S$ in almost all cases. In this section, we provide our new polynomial representation that enables us to efficiently recover the exact set from the polynomial represented by our suggestion.

Throughout this section, let $d_0 = \max\{d, \lceil \log N \rceil\}$, where $N$ is the modulus of NS encryption scheme, and $\tau = \frac{1}{3}(\log d + 2 \log d_0)$. We assume that $q_j$'s are $(3\tau + 1)$-bit distinct primes and let $\mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell}$ be the universe for $\ell \leq \bar{\ell}$ and $0 < \alpha < 1$. The proper size of parameters will be determined later by analyzing complexity of our representation and our set union protocol.

Focus on the fact that the factorization of $\sigma$ is public. Using this fact, given a polynomial $f = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$ for a set $S = \{s_1, \ldots, s_d\}$, one can obtain all roots of $f \bmod q_j$ for each $j$, by applying Umans' polynomial factorization algorithm over a finite field $\mathbb{Z}_{q_j}$. To recover $S$, one can perform CRT computation for obtaining less than $d^{\bar{\ell}}$ candidates of roots of $f$ over $\mathbb{Z}_\sigma$. In general, however, the number of roots of $f$ over $\mathbb{Z}_\sigma$ is larger than the $\deg f$ and there is no criteria to determine the exact set $S$. To remove irrelevant roots which are not in $S$, we give some relations among all roots of polynomials $f \bmod q_j$'s by providing an encoding function.

Before describing our solution, we look into an easy way to give a relation among all roots of polynomial $f \bmod q_j$. However, it is not efficient to recover a set from a polynomial.

*Encoding with a tag.* To give relations among all roots of polynomials $f \bmod q_i$'s, we may consider to insert the same value depending on the element, called a *tag*, into an element part in $\mathbb{Z}_{q_j}$'s. For example, let $h : \{0,1\}^* \to \{0,1\}^{3(1-\alpha)\tau}$ be a collision-resistant hash function for a real number $0 < \alpha < 1$. Parse $s_i$ into $\bar{\ell}$ blocks $s_{i,1}, \ldots, s_{i,\bar{\ell}}$ of $(3\alpha\tau)$-bit so that $s_i = s_{i,1}|| \cdots ||s_{i,\bar{\ell}}$. Consider a function $\iota' : \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\bar{\ell}} \to \mathbb{Z}_\sigma$, in which $\iota'(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota'(s_i) \equiv s_{i,j}||h(s_i) \bmod q_j$ for $1 \le j \le \bar{\ell}$.[2] Then we represent a set $S$ as a polynomial $f_S(x) = \prod_{s_i \in S}(x - \iota'(s_i)) \in \mathbb{Z}_\sigma[x]$.

Then one can reduce the number of candidates by checking a tag $h(s_i)$ when one gets all roots over $\mathbb{Z}_{q_j}$'s. However, when a collision occurs, say $h(s_i) = h(s_j)$ with $s_i \neq s_j$, one has to check the hash value of $2^{\bar{\ell}}$ elements which are possible combinations $(s_{i,1}, s_{j,1}), \cdots, (s_{i,\bar{\ell}}, s_{j,\bar{\ell}})$. The probability[3] that at least one collision occur among $d$ elements is lower bounded by $\frac{d^2}{2^{1+3\tau(1-\alpha)}}$ which is not negligible even for small $\alpha$. Moreover, the expected computation becomes $\Omega(2^{\bar{\ell}})$, which is not a polynomial in $d$.

### 3.1 Our Polynomial Representation

Now, we present our polynomial representation for supporting to recover a set from the polynomial over $\mathbb{Z}_\sigma$ represented by our suggestion. Take $\alpha = \frac{1}{3}$, i.e., $\mathcal{U} \subseteq \{0,1\}^{\tau\ell}$ for optimization. If $\alpha \neq \frac{1}{3}$, the expected computation is in polynomial time only when the size of the universe is restricted. Details about the proper size of $\alpha$ is in Section 3.3.

*Encoding by repetition.* Let $h : \{0,1\}^* \to \{0,1\}^{2\tau}$ and $h_j : \{0,1\}^* \to \{0,1\}^{\tau}$ be collision-resistant hash functions for $1 \le j \le \ell'$. Parse a message $s_i \in \mathcal{U} \subseteq \{0,1\}^{\tau\ell}$ into $\ell$ blocks $s_{i,1}, \ldots, s_{i,\ell}$ of $\tau$-bit so that $s_i = s_{i,1}|| \cdots ||s_{i,\ell}$. Let $s_{i,\ell+j} = h_j(s_i)$ for $1 \le j \le \ell'$ and parse $h(s_i)$ into two blocks $s_{i,\bar{\ell}+1}$ and $s_{i,\bar{\ell}+2}$ of $\tau$-bit. Let $\bar{\ell} = \ell + \ell'$. We define our encoding function $\iota : \mathcal{U} \subseteq \{0,1\}^{\tau\ell} \to \mathbb{Z}_\sigma$, in which $\iota(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota(s_i) \equiv s_{i,j}||s_{i,j+1}||s_{i,j+2} \bmod q_j$ for $1 \le j \le \bar{\ell}$. Then a set $S$ is represented as a polynomial $f_S(x) = \prod_{s_i \in S}(x - \iota(s_i)) \in \mathbb{Z}_\sigma[x]$.

$$
\begin{array}{cccc}
\overbrace{s_{1,1}||\ s_{1,2}\ ||s_{1,3}}^{s_1} & \overbrace{s_{2,1}||\ s_{2,2}\ ||s_{2,3}}^{s_2} & \cdots & \overbrace{s_{d,1}||\ s_{d,2}\ ||s_{d,3}}^{s_d} \pmod{q_1} \\
s_{1,2}||\ s_{1,3}\ ||s_{1,4} & s_{2,2}||\ s_{2,3}\ ||s_{2,4} & \cdots & s_{d,2}||\ s_{d,3}\ ||s_{d,4} \pmod{q_2} \\
\vdots & \vdots & \ddots & \vdots \\
s_{1,\bar{\ell}}||s_{1,\bar{\ell}+1}||s_{1,\bar{\ell}+2} & s_{2,\bar{\ell}}||s_{2,\bar{\ell}+1}||s_{2,\bar{\ell}+2} & \cdots & s_{d,\bar{\ell}}||s_{d,\bar{\ell}+1}||s_{d,\bar{\ell}+2} \pmod{q_{\bar{\ell}}}
\end{array}
$$

**Fig. 1.** Our Encoding Function $i$

[2] By notation abuse, throughout this paper if necessary, we regard a bit string as the corresponding integer via some converting function.

[3] Pr[At least one collision occur among $d$ hash values $h(s_i)$'s.]
$\le 1 - \left(1 - \frac{1}{2^{3(1-\alpha)\tau}}\right) \cdots \left(1 - \frac{d-1}{2^{3(1-\alpha)\tau}}\right) \approx 1 - \exp\left(-\prod_{i=1}^{d-1} \frac{i}{2^{3(1-\alpha)\tau}}\right) \approx \frac{d^2}{2 \cdot 2^{3(1-\alpha)\tau}}$.

*Decoding phase.* Denote by $s_j^{(i)} := \iota(s_i) \bmod q_j$ for each message $s_i = s_{i,1}||\cdots||s_{i,\ell}$. For $1 \le j \le \bar{\ell} - 1$, we define $(s_j^{(i)}, s_{j+1}^{(i')}) \in \mathbb{Z}_{q_j} \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if the last $(2\tau)$-bit of of $s_j^{(i)}$ is equal to the first $(2\tau)$-bit of $s_{j+1}^{(i')}$ i.e, $s_{i,j+1}||s_{i,j+2} = s_{i',j+1}||s_{i',j+2}$. Inductively, we also define $(s_1^{(i_1)}, \cdots, s_{j+1}^{(i_{j+1})}) \in \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if $(s_1^{(i_1)}, \cdots, s_j^{(i_j)})$ and $(s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ are linkable pairs.

$$
\begin{aligned}
s_1^{(i_1)} &= s_{i_1,1}||s_{i_1,2}||s_{s_1,3} \\
s_2^{(i_2)} &= \quad\; s_{i_2,2}||s_{i_2,3}||s_{s_2,4} \\
s_3^{(i_3)} &= \quad\quad\; s_{i_3,3}||s_{i_3,4}||s_{s_3,5}
\end{aligned}
$$

$\Rightarrow (s_1^{(i_1)}, s_2^{(i_2)}, s_3^{(i_3)})$ is linkable pair.

**Fig. 2.** Linkable Pairs

Let $\iota(s_i)$ and $\iota(s_{i'})$ be an element of the image of $\iota$ with $s_i \neq s_{i'}$. We can easily check the following properties:

- $\left(s_1^{(i)}, \cdots, s_{j+1}^{(i)}\right)$ is always a linkable pair.
- $\Pr\left[\left(s_j^{(i)}, s_{j+1}^{(i')}\right) \text{is a linkable pair}\right] = \Pr\left[s_{i,j+1}||s_{i,j+2} = s_{i',j+1}||s_{i',j+2}\right] = \frac{1}{2^{2\tau}}$.

At decoding phase, when a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota(s_i))$ is given, we perform two phases to find the correct $d$ roots of $f(x)$. In the first stage, one computes all the roots $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ over $\mathbb{Z}_{q_j}[x]$ for each $j$. For each $j$ sequentially from 1 to $\bar{\ell} - 1$, we find all the linkable pairs among $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ and $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(d)}\}$ by checking whether the last $(2\tau)$-bit of $s_j^{(i)}$ and the first $(2\tau)$-bit of $s_{j+1}^{(i)}$ are the same. It can be done by $d^2$ comparisons or $O(d \log d)$ computations using sorting and determining.

After $\bar{\ell}$ steps, we obtain $d'$ number of linkable pairs of $\bar{\ell}$-tuple, which is a candidate of roots of $f$ and an element of a set. It includes the $d$ elements corresponding to $\iota(s_1), \ldots, \iota(s_d)$. If $d'$ is much larger than $d$, it can be a burden. However, we can show that the expected value of $d'$ is at most $3d$ in Theorem 1.

After obtaining $d'$ linkable pairs of $\bar{\ell}$-tuple, in the second phase, we check whether each pair belongs in the image of $\iota$ with the following equalities:

$$s_{i,\ell+j} = h_j(s_i) \qquad \text{for all } 1 \le j \le \ell', \tag{1}$$

$$s_{i,\bar{\ell}+1} \,||\, s_{i,\bar{\ell}+2} = h(s_i). \tag{2}$$

The linkable pairs of $\bar{\ell}$-tuple, corresponding to $\iota(s_i)$ for some $i$ clearly satisfies the above equations. However, for a random $\bar{\ell}$-tuple in $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_{\bar{\ell}}}$, the probability that it satisfies the relation (1) is about $\frac{1}{2^{\tau\ell'}}$ and the probability that it satisfies the relation (2) is about $\frac{1}{2^{2\tau}}$. Hence, the expected number of wrong $\bar{\ell}$-tuples passing both phases is less than $d \times \frac{1}{2^{\tau(2+\ell')}}$. It is less than $2^{-\lambda}$ for a security parameter $\lambda$ if we take the parameter $\ell'$ to satisfy

$$\ell' > \frac{3(\lambda + \log d)}{\log d + 2\log d_0} - 2. \tag{3}$$

For example, when $\lambda = 80$ and $d \approx d_0 \approx 2^{10}$, then $\ell'$ is about 8.

### 3.2 The Expected Number of Linkable Pairs

In this subsection, we analyze the expected number of linkable pairs of $\bar{\ell}$-tuple when we recover a set from a polynomial of degree $d$, represented by our suggestion.

A set of $\kappa$ elements $s_j^{(1)}, \cdots, s_j^{(\kappa)} \in \mathbb{Z}_{q_j}$ is called a *$\kappa$-collision* if their last $2\tau$-bits are the same. Since $(s_{j-1}^{(1)}, s_j^{(1)})$ is a linkable pair, $\kappa$-collision causes at least $\kappa$ linkable pairs. We easily obtain the following observations, which are evidence of the expected number of linkable pairs of $\bar{\ell}$-tuple is not large.

1. The probability that at least one 2-collision occurs in $\mathbb{Z}_{q_j}$ is less than $\frac{1}{2}$ by the birthday paradox.
2. The probability that at least one $\kappa$-collision occurs for $\kappa \geq 3$ in $\mathbb{Z}_{q_j}$ is at most $\frac{1}{4d} \approx \frac{1}{2^{(2+\tau)}}$ of the probability that at least one $(\kappa-1)$-collision occurs from [27, Theorem 2].
3. If $\kappa$-collision occurs in $\mathbb{Z}_{q_j}$, the number of candidates of roots of $f$ is to be $\kappa^2$ from this collision. (Assume that $\kappa$-collision $\{s_j^{(1)}, \cdots, s_j^{(\kappa)}\}$ and $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(\kappa)}\}$ occurs. Then $s_j^{(1)}$ can be combined with $\kappa$ candidates $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(\kappa)}\}$. Hence $\kappa^2$ linkable pairs are generated.)

The expected number of 2-collision in $\mathbb{Z}_{q_j}$ for all $j$ are roughly $\frac{\bar{\ell}}{2}$ and the expected number of $\kappa$-collision in $\mathbb{Z}_{q_j}$ for $\kappa \geq 3$ is negligible. Also, the expected number of linkable pairs of $\bar{\ell}$-tuple is to be less than $\left(\frac{\bar{\ell}}{2}\right)^2$ since the probability that a 2-collision occurs under the same position is at most $\frac{1}{\binom{d}{2}} \approx \frac{1}{2^{2\tau}}$.

Theorem 1 gives a rigorous analysis of the upper bound of the expected number of linked pairs of $\bar{\ell}$-tuple.

**Theorem 1.** *In case of the encoding $\iota$ defined above, when a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota(s_i))$ is given, the expected number of linkable pairs of $\bar{\ell}$-tuple is at most $3d$.*

*Proof.* Let $E_j$ be the expected number of linkable pairs of $j$-tuple in $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_j}$ for $j \geq 2$.

Then,

$$\begin{aligned}
E_2 &= \sum_{i_1, i_2 \in \{1,\cdots,d\}} 1 \cdot \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair}\right] \\
&= \sum \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair and } i_1 = i_2\right] \\
&\quad + \sum \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair and } i_1 \neq i_2\right] \\
&= d + \sum_{i_2 \in \{1,\cdots,d\} \setminus \{i_1\}} \Pr\left[s_{i_1,2} || s_{i_1,3} = s_{i_2,2} || s_{i_2,3}\right] = d\left(1 + \frac{d-1}{2^{2\tau}}\right).
\end{aligned}$$

If $i_{j+1} = i_{j-1}$ and $(s_{j-1}^{(i_{j-1})}, s_j^{(i_j)})$ is a linkable pair, then $s_{i_{j+1},j+1} = s_{i_{j-1},j+1} = s_{i_j,j+1}$. Therefore,

$$\begin{aligned}
&\sum_{i_1, \cdots, i_{j+1} \in \{1,\cdots,d\}} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1} = i_{j-1} \neq i_j\right] \\
&= \sum_{i_{j+1} = i_{j-1} \neq i_j} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair and } s_{i_j,j+1} || s_{i_j,j+2} = s_{i_{j+1},j+1} || s_{i_{j+1},j+2}\right] \\
&= \sum_{i_{j+1} = i_{j-1} \neq i_j} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair and } s_{i_j,j+2} = s_{i_{j+1},j+2}\right] = \frac{1}{2^\tau} E_j.
\end{aligned}$$

Also,

$$\sum_{i_1,\cdots,i_{j+1}\in\{1,\cdots,d\}} \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1}\notin\{i_{j-1},i_j\}\right]$$
$$= \sum_{i_{j+1}\notin\{i_{j-1},i_j\}} \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)}) \text{ is a linkable pair and } s_{i_j,j+1}||s_{i_j,j+2} = s_{i_{j+1},j+1}||s_{i_{j+1},j+2}\right]$$
$$\leq \frac{d-1}{2^{2\tau}}\sum \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)}) \text{ is a linkable pair}\right] = \frac{d-1}{2^{2\tau}}E_j.$$

From the above equations, we obtain the recurrence formula of $E_j$ as follows:

$$\begin{aligned}
E_{j+1} &= \sum_{i_1,\cdots,i_{j+1}\in\{1,\cdots,d\}} 1\cdot\Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair}\right] \\
&= \sum \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j = i_{j+1}.\right] \\
&\quad + \sum \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j \neq i_{j+1}\right] \\
&= E_j + \sum \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j-1} = i_{j+1} \neq i_j\right] \\
&\quad + \sum \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1}\notin\{i_{j-1},i_j\}\right] \\
&\leq \left(1+\frac{1}{2^\tau}+\frac{d-1}{2^{2\tau}}\right)\cdot E_j,
\end{aligned}$$

by induction for $j \geq 2$. Hence, $E_{\bar{\ell}} \leq d\left(1+\frac{1}{2^\tau}+\frac{d-1}{2^{2\tau}}\right)^{\bar{\ell}-1}$. Since $\bar{\ell} \leq \min\{d, \frac{\lceil \log N\rceil}{12\tau}\} \leq \frac{2^{2\tau}}{2^\tau+d}$[4],

$$E_{\bar{\ell}} \leq d\left(1+\frac{1}{2^\tau}+\frac{d-1}{2^{2\tau}}\right)^{\bar{\ell}-1} < ed < 3d$$

where $e \approx 2.718$ is the base of the natural logarithm. From these result, the upper bound of the expected number of linkable pairs of $\bar{\ell}$-tuple is $3d$. □

### 3.3 The proper size of $\alpha$

In this subsection, we explain why the proper size of $\alpha$ is $\frac{1}{3}$. Let $\alpha = \frac{b}{a}$ with relatively prime $a, b$ and suppose that $\ell$ and $\ell'$ are divided by $b$ and let $h : \{0,1\}^* \to \{0,1\}^{3(1-\alpha)\tau}$ and $h_i : \{0,1\}^* \to \{0,1\}^{3\alpha\tau}$ be collision-resistant hash functions $1 \leq i \leq \ell'$ . Parse a message $s_i \in \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell}$ into $\ell$ blocks $s_{i,1},\cdots,s_{i,\ell}$ of $3\alpha\tau$-bit so that $s_i = s_{i,1}||\cdots||s_{i,\ell}$. Let $s_{i,\ell+j} = h_j(s_i)$ for $1 \leq j \leq \ell'$ and parse $h(s_i)$ into $(a-b)$ blocks $s_{i,\bar{\ell}+1},\cdots,s_{i,\bar{\ell}+a-b}$ of $3\alpha\tau$-bit. Let $\bar{\ell} = \ell + \ell'$. We define our encoding function $\iota'' : \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell} \to \mathbb{Z}_\sigma$, in which $\iota''(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota''(s_i) = s_{i,(j-1)b+1}||\cdots||s_{i,(j-1)b+a} \bmod q_j$. Then a set $S$ is represented as a polynomial $f(x) = \prod_{s_i\in S}(x-\iota''(s_i)) \in \mathbb{Z}_\sigma[x]$.

For each message $s_i = s_{i,1}||\cdots||s_{i,\ell}$, denote by $s_j^{(i)} := \iota''(s_i) \bmod q_j$. At decoding phase, when a polynomial $f(x) = \prod_{i=1}^{d}(x-\iota''(s_i))$ is given, one computes all the roots $\{s_j^{(1)},\cdots,s_j^{(d)}\}$ over $\mathbb{Z}_{q_j}$. Then, for each $j$ sequentially from 1 to $\bar{\ell}-1$, we find all linkable pairs among $\{s_j^{(1)},\cdots,s_j^{(d)}\}$ and $\{s_{j+1}^{(1)},\cdots,s_{j+1}^{(d)}\}$ to find the correct $d$ $s_i$'s.

We generalize the definition of linkable pair and look into some properties of linkable elements for the all case of $\alpha$. We define $(s_j^{(i)}, s_{j+1}^{(i')}) \in \mathbb{Z}_{q_j}\times\mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if

[4] We will set the parameter $\bar{\ell}$ satisfies this relation for complexity of our set union protocol. Refer to Section 4.2.

the last $3(1-\alpha)\tau$-bit of of $s_j^{(i)}$ is equal to the first $3(1-\alpha)\tau$-bit of $s_{j+1}^{(i')}$. Inductively, we also define $(s_1^{(i_1)}, \cdots, s_{j+1}^{(i_{j+1})}) \in \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if $(s_1^{(i_1)}, \cdots, s_j^{(i_j)})$ and $(s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ are linkable pairs.

**Lemma 1.** *For $j = 1, \cdots, \ell - 1$,*

1. *$(s_1^{(i)}, s_2^{(i)}, \cdots, s_{j+1}^{(i)})$ is always a linkable pair.*
2. $\Pr\left[(s_j^{(i)}, s_{j+1}^{(i')}) \text{ is a linkable pair}\right] = 1 + \dfrac{d-1}{2^{3(1-\alpha)\tau}}$
3. $\Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}, \cdots, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j \neq i_{j+1}\right]$
   $\leq \left(\frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right]$

*Proof.* 1. It is trivial by the above definition.

2. $\Pr\left[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair}\right]$
   $= \Pr\left[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair and } i = i'\right] + \Pr\left[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair and } i \neq i'\right]$
   $= 1 + \sum_{i' \in \{1,\cdots,d\} \setminus \{i\}} \Pr\left[s_{i,jb+1} || \cdots || s_{i,jb+a-b} = s_{i',jb+1} || \cdots || s_{i',jb+a-b}\right]$
   $= 1 + \dfrac{d-1}{2^{3(1-\alpha)\tau}}.$
3. $\Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}, \cdots, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j \neq i_{j+1}\right]$
   $= \displaystyle\sum_{k=1}^{j} \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}, \cdots, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1} = i_{j-k} \notin \{i_{j-k+1}, \cdots, i_j\}\right]$
   $\leq \displaystyle\sum_{k=1}^{j} \frac{1}{\min\{2^{3k\alpha\tau}, 2^{3(1-\alpha)\tau}\}} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right]$
   $\leq \left(\dfrac{2}{2^{3\alpha\tau}} + \dfrac{d}{2^{3(1-\alpha)\tau}}\right) \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right]$

□

**Theorem 2.** *Given a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota(m_i))$ for the encording funcion $\iota$, the expected number of linkable pairs of $\bar{\ell}$-tuple is at most $d\left(1 + \frac{3}{2^{\min\{3\alpha\tau, (2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}$.*

*Proof.* The expected number of linkable pairs of $\bar{\ell}$-tuple is

$$\begin{aligned}
&\sum_{i_1,\cdots,i_{\bar{\ell}} \in \{1,\ldots,d\}} 1 \cdot \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}}^{(i_{\bar{\ell}})}) \text{ is a linkable pair.}\right] \\
&= \sum \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}}^{(i_{\bar{\ell}})}) \text{ is a linkable pair and } i_{\bar{\ell}} = i_{\bar{\ell}-1}\right] \\
&\quad + \sum \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}}^{(i_{\bar{\ell}})}) \text{ is a linkable pair and } i_{\bar{\ell}} \neq i_{\bar{\ell}-1}\right] \\
&\leq \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \sum \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}-1}^{(i_{\bar{\ell}-1})}) \text{ is a linkable pair}\right] \\
&\leq \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{1}{2^{(2-3\alpha)\tau}}\right) \sum 1 \cdot \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}-1}^{(i_{\bar{\ell}-1})}) \text{ is a linkable pair}\right] \ (\because d < 2^{\tau}) \\
&\leq \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{(2-3\alpha)\tau}}\right)^{\bar{\ell}-2} \cdot \sum_{i_2=1}^{d} \sum_{i_1=1}^{d} \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair}\right] \\
&\leq d\left(1 + \frac{3}{2^{\min\{3\alpha\tau, (2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}
\end{aligned}$$

□

If $\bar{\ell} \geq 2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$, then $d\left(1+\frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}$ is exponentially increased. Hence it requires the limitation of $\bar{\ell}$ so that $\bar{\ell} < 2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$. But, if $\alpha \neq \frac{1}{3}$, then $3\alpha, (2-3\alpha) < 1$ and so $2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}} < \min\{d, \frac{\lceil \log N \rceil}{12\tau}\}$ for increasing of $d$ and $\log N$. It causes the limitation of the size of universe since $|\mathcal{U}| \leq 2^{3\alpha\tau\ell}$ in the case $\alpha \neq \frac{1}{3}$. From these reason, we use the size of $\mathcal{U}$ to $\{0,1\}^{\tau\ell}$ i.e. $\alpha = \frac{1}{3}$.

## 4 Applications to Set Operations

In this section, we present our set union protocols based on our polynomial representation described in Section 3. Our construction exploits the NS AHE to encrypt a rational function whose denominator polynomial corresponds to a participant's set. For this, we generalize a reversed Laurent Series presented in [23] to work on $\mathbb{Z}_\sigma$ with a composite $\sigma$, which is the domain of the NS scheme.

We also explain how to modify our polynomial representation for applying to construct multi-set union protocols and set intersection protocols in which decryptors are different from set contributors.

### 4.1 Transforming Our Representation into Rational Function using Reversed Laurent Series

To construct constant round privacy-preserving set union protocols, we adopt rational function representations presented in [23]. In the rational function representation, each participant $\mathcal{P}_i$ represents his own set $S_i$ of elements as a rational function $\frac{1}{f_{S_i}}$ where $f_{S_i} := \prod_{s_j \in S}(x - s_j)$. It gives the relation

$$\frac{r_1}{f_{S_1}} + \frac{r_2}{f_{S_2}} + \cdots + \frac{r_n}{f_{S_n}} = \frac{u}{\mathrm{lcm}(f_{S_1}, f_{S_2}, \ldots, f_{S_n})}$$

for random polynomials $r_i$'s and some polynomial $u$. Then each participant tries to recover $f(x) = \mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})$ from a polynomial $F(x) = \frac{u}{\mathrm{lcm}(f_{S_1},\ldots,f_{S_n})}$.

To represent a set as a rational function, the authors in [23] exploit a reversed Laurent series (RLS). We briefly introduce a RLS. (Refer to [23] for details.)

For a positive integer $q$, a *reversed Laurent series* (RLS) over $\mathbb{Z}_q$ is a singly infinite, formal sum of the form $f(x) = \sum_{i=-\infty}^{m} f[i]x^i$ ($f[m] \neq 0$) for an integer $m$ and $f[i] \in \mathbb{Z}_q$. We define the degree of $f$ by $m$ and denoted $\deg f$. For a RLS $f(x)$, given $d_1 \leq d_2 \leq \deg f$, we denote $f(x)_{[d_1,d_2]} = \sum_{i=d_1}^{d_2} f[i]x^i$. For a rational function $f/g$ with $f, g \in \mathbb{Z}_q[x]$ and $g \neq 0$, we define *the RLS representation of a rational function* $f/g$ by a reversed Laurent series of $f/g$.

When $q$ is a prime, the RLS representation has the following property:

- The RLS representation for a given rational function is unique.
- Let $f, g$ be polynomials in $\mathbb{Z}_q[x]$ with $\deg f < \deg g$ and $g \neq 0$. Then there exists an algorithm [23] to compute $k(> \deg g)$ high-order terms of the RLS representation of $f/g$. We describe this algorithm in Figure 3. We denote the output of RationalToRLS algorithm which takes polynomials $f, g$ and integer $k$ by RationalToRLS$(f, g, k)$.
- When $2k$ high-order terms of the RLS representation of a rational function $f/g$ such that $f, g \in \mathbb{Z}_q[x]$, $g \neq 0$, and $\deg f < \deg g \leq k$ is given, there exists an efficient algorithm [26, Section 17.5.1] to recover two polynomials $v(x), u(x)$ in $\mathbb{Z}_q[x]$ such that $\frac{v}{u} = \frac{f}{g}$ in $\mathbb{Z}_q(x)$ and $\gcd(v, u) = 1$.

In our protocol, we will represent each participant's set $S_i$ as our polynomial representation $f_{S_i} := \prod_{s_j \in S_i}(x - \iota(s_j)) \in \mathbb{Z}_\sigma[x]$ with our encoding function $\iota$. Then we convert

**RationalToRLS**$(f, g, k)$
**Input:** $f(x), g(x) \in \mathbb{Z}_q[x]$ with $\deg f < \deg g$, an integer $k > \deg g$
**Output:** $k$ higher-order terms of the RLS representation of a rational function $f/g$

1. Compute $F(x) := f(x) \cdot x^k$.
2. Use polynomial division to compute $Q(x)$ and $R(x)$ with $F(x) = g(x) \cdot Q(x) + R(x)$ and $\deg R < \deg g$.
3. Output $Q(x) \cdot x^{-k}$.

**Fig. 3.** Conversion from a Rational Function to Its RLS

a rational function of $1/f_{S_i}$ to its RLS over $\mathbb{Z}_\sigma$. Since $\mathbb{Z}_\sigma$ is not a Euclidean domain, one may doubt whether the RationalToRLS algorithm works on $\mathbb{Z}_\sigma[x]$. However, in our protocol, since the conversion requires polynomial divisions only by monic polynomials, the RationalToRLS algorithm works well on $\mathbb{Z}_\sigma[x]$.

After the end of interactions among participants in our protocol, each participant obtains the $2nk$ higher-order term of the RLS representation of a rational function $\frac{u(x)}{U(x)}$ where $U(x) = \mathrm{lcm}(f_{S_1}(x), \ldots, f_{S_n}(x))$. There is no algorithm to recover $u'(x)$ and $U'(x)$ in $\mathbb{Z}_\sigma[x]$ such that $\frac{u(x)}{U(x)} = \frac{u'(x)}{U'(x)}$. However, from our polynomial representation, it only requires $U'(x) \bmod q_j$ for each $j$, and we can obtain $U'(x) \bmod q_j$ from the RLS representation modulo $q_j$ by running polynomial recovering algorithm on $\mathbb{Z}_{q_j}[x]$'s.

The following lemma guarantees that, in a polynomial ring $\mathbb{Z}_\sigma[x]$, a modular operation by a prime divisor $q$ of $\sigma$ and RationalToRLS algorithm are commutative. This will be utilized to prove the correctness of our protocol.

**Lemma 2.** *Let $f, g$ be polynomials in $\mathbb{Z}_\sigma[x]$ with $\deg f < \deg g$ and $g \neq 0$. Suppose that $g$ is a monic polynomial in $\mathbb{Z}_\sigma[x]$. For each prime $q$ which divides $\sigma$ and an integer $k > \deg g$, RationalToRLS($f \bmod q, g \bmod q, k$)=RatonalToRLS($f, g, k$) mod$q$.*

*Proof.* Let RationalToRLS$(f, g, k) = Q(x)x^{-k}$ where $x^k f(x) = Q(x)g(x)+R(x)$ in $\mathbb{Z}_\sigma[x]$ with $R = 0$ or $\deg R < \deg g$. For each polynomial $p(x)$ in $\mathbb{Z}_\sigma[x]$, denote $p(x) \bmod q$ by $p_q(x)$. Then $x^k f_q(x) = Q_q(x)g_q(x) + R_q(x)$ in $\mathbb{Z}_q[x]$, where $R_q = 0$ or $\deg R_q \leq \deg R < \deg g = \deg g_q$. Since the division algorithm uniquely outputs the quotient and the remainder in $\mathbb{Z}_q[x]$, RationalToRLS$(f \bmod q, g \bmod q, k) = Q_q(x)x^{-k} \equiv Q(x)x^{-k} \bmod q$. □

The following lemma gives an information on the distribution of $u_j(x) := u(x) \bmod q_j$ in our protocol which is inevitable to prove the security of our set union protocol. It guarantees that the distribution of $u_j(x)$ and $u(x)$ are uniformly distributed among polynomials in the set of polynomials having degree at most $\deg(\mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})) - 1$ in $\mathbb{Z}_{q_j}[x]$ and $\mathbb{Z}_\sigma[x]$, respectively. The proof of Lemma 3 is given in [23].

**Lemma 3 ([23, Lemma 1]).** *Let $f_{S_1}(x), \ldots, f_{S_n}(x) \in \mathbb{Z}_q[x]$ be polynomials of degree $k \geq 1$ for a prime $q$. Suppose $r_1(x)$, …, $r_n(x)$ are polynomials in $\mathbb{Z}_q[x]$, chosen uniformly and randomly in the set of polynomials of degree at most $k-1$. Let $u(x)$ be a polynomial such that*

$$\frac{u(x)}{lcm(f_{S_1}(x), \ldots, f_{S_n}(x))} = \sum_{i=1}^{n} \frac{r_i(x)}{f_{S_i}(x)}. \tag{4}$$

*Then $u(x)$ is uniformly distributed among polynomials in the set of polynomials $\in \mathbb{Z}_q[x]$ having degree at most $\deg(lcm(f_{S_1}(x), \ldots, f_{S_n}(x))) - 1$.*

Finally, to recover the exact $U(x) = \mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})$ from the rational function $\frac{u(x)}{U(x)}$, the relation $\gcd(u(x), U(x)) = 1$ is to be satisfied. In our set union protocol, since

**Input:** There are $n \geq 2$ HBC participants $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$ of cardinality $k$. Set $d = nk$. Let $\mathsf{Enc}_{\mathsf{pk}}$ be a threshold NS encryption scheme. Let $\iota : \{0,1\}^* \to \mathbb{Z}_\sigma$ be our encoding function defined in Section 3.

Each participant $\mathcal{P}_i$,$i = 1, \ldots, n$:

1. (a) constructs the polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j})) \in \mathbb{Z}_\sigma[x]$ and runs RationalToRLS$(1, f_{S_i}, (2n+1)k - 1)$ to obtain $\left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]}$, defines $F_{S_i}(x) = \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}$.
   (b) computes $\tilde{F}_{S_i}$, the encrypted polynomial of $F_{S_i}$ and sends $\tilde{F}_{S_i}$ to all other participants.
2. (a) chooses random polynomials $r_{i,j}(x) \in \mathbb{Z}_\sigma[x]$ of degree at most $k$ for all $1 \leq j \leq n$.
   (b) computes the encryption, $\tilde{\phi}_i$, of the polynomial $\phi_i(x) = \sum_{j=1}^n F_{S_j} \cdot r_{i,j}$ and sends it to all participants.
3. (a) calculates the encryption of the polynomial $F(x) = \sum_{i=1}^n \phi_i(x)$.
   (b) performs a group decryption with all other players to obtain the high-order $2nk$ terms of $F(x)$.
4. (a) recovers a polynomial fair of $u_j(x)$ and $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all $1 \leq j \leq \bar{\ell}$ such th at $\left(\frac{u_j(x)}{U_j(x)}\right)_{[-2nk,-1]} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]} \cdot x^{-(2n+1)k+1}$ and $\gcd(u_j(x), U_j(x)) = 1$ in $\mathbb{Z}_{q_j}[x]$, using the high-order $2nk$ terms of $F(x)$ obtained in Step 3 (b).
   (b) extracts all roots of $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all $j$ using a factorization algorithm.
   (c) determines the set union using our rule of representation.

**Fig. 4.** PPSU-HBC protocol in the HBC case

$u_j(x) := u(x) \bmod q_j$ is uniformly distributed in $\mathbb{Z}_{q_j}[x]$ and the expected number of roots of a random polynomial is one [17], we may expect that our RLS representation fails to output all the elements in the set union with probability $\frac{d}{q_j} \approx 2^{-2\tau}$. Furthermore, it can be detected if this happens for a certain $j$, whose probability is about $1 - (1 - \frac{d}{q_j})^{\bar{\ell}} \approx \frac{\bar{\ell} d}{q_j} \leq 2^{-\tau}$. It is not negligible, but small enough.

### 4.2 Set Union for Honest-But-Curious Case

**Threshold Naccache-Stern Encryption** For a group decryption, it requires a semantically secure, threshold NS additive homomorphic encryption scheme in our protocol. We provide a threshold version of the NS encryption scheme in Appendix A. Our construction is based on the technique of Fouque et al. [7], which transforms the original Paillier homomorphic encryption scheme into a threshold version working from Shoup's technique [25].

**Parameter Setting** Let $\mathcal{U}$ be the universe, $n$ be the number of participants, and $k$ be the maximum size of participants' datasets. Let $d$ be the maximum size of the set union, i.e., $d = nk$. Take $N$ so that $\log N > 12 \log |\mathcal{U}|$, which is the modulus of threshold NS encryption scheme. Put $d_0 = \max\{d, \lceil \log N \rceil\}$ and $\tau = \frac{1}{3}(\log d + 2 \log d_0)$. Set $\ell$ so that $\ell > \log |\mathcal{U}|/\tau$ and a proper $\ell'$ so that satisfies the relation (3). Let $\bar{\ell} = \ell + \ell'$. Note that $\bar{\ell}$ is to be smaller than $\min\{d, \frac{\log N}{4 \log \log N}\}$. Generate the parameters of the threshold NS encryption scheme, including the size of message space $\sigma$, which is a product of $\bar{\ell}$ $(3\tau + 1)$-bit distinct primes $q_i$'s.

**Our Set Union Protocol for Honest-But-Curious Case** Our set union protocol against honest-but-curious adversaries is described in Figure 4. In our set union protocol, each participant computes the higher-order $2nk$ term of the RLS representation of $F_{S_i} = \frac{1}{f_{S_i}} = \frac{1}{\prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j}))} \in \mathbb{Z}_\sigma[x]$ for our encoding function $\iota$ and sends its encryption to all others. After interactions among participants, each participant obtains the

high-order $2nk$ term of the RLS representation of $F(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{n} \frac{1}{f_{S_j}} \cdot r_{i,j} \right) \in \mathbb{Z}_\sigma[x]$ for some random polynomials $r_{i,j}$'s chosen by each participant $\mathcal{P}_i$. Then each participant obtains the high-order $2nk$ terms of the RLS representation of $F$ in $\mathbb{Z}_\sigma[x]$ with group decryption and using this values, recover polynomials $u_j(x)$ and $U_j(x)$ such that $\left(\frac{u_j(x)}{U_j(x)}\right)_{[-2nk,-1]} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]} \cdot x^{-(2n+1)k+1}$ and $\gcd(u_j(x), U_j(x)) = 1$ in $\mathbb{Z}_{q_j}[x]$. Thereafter, each participant extracts all roots of $U_j(x)$ over $\mathbb{Z}_{q_j}$ for each $j$ and recover all elements based on criteria of our representation.

*Analysis* Now, we consider the correctness and privacy of our proposed protocol describe in Figure 4. The following theorems guarantee the correctness and privacy of our construction in Figure 4.

**Theorem 3.** *In the protocol described in Figure 4, every participants learn the set union of private inputs participating players, with high probability.*

*Proof.* After Step 3 (b), all participants obtain the high order $2nk$ terms of $F(x)$ and hence they obtain the high order $2nk$ terms of the RLS representation of $F(x) \bmod q_j$. From these values, using polynomial recovering algorithm, they reconstruct polynomials $u_j(x)$ and $U_j(x)$ such that $\frac{u_j(x)}{U_j(x)} \equiv F_j(x) \bmod q_j$, and $\gcd(u_j(x), U_j(x)) = 1$. From the equation (4) and Lemma 2, $U_j = (\mathrm{lcm}(f_{S_1}, f_{S_2}, \ldots, f_{S_n}) \mod q_j)$ with high probability. Since our polynomial representation can give the exact corresponded set with overwhelming probability, it gives $S_1 \cup \ldots \cup S_n$. □

**Theorem 4.** *Assume that the utilized additive homomorphic encryption scheme is semantically secure. Then, in our set union protocol for the HBC case described in Figure 4, any adversaries $\mathcal{A}$ of colluding fewer than n honest-but-curious participants learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

*Proof.* Since the utilized additive homomorphic encryption scheme is semantically secure, each participant learns only $F(x) = \sum_{j=1}^{n} (\sum_{i=1}^{n} r_{i,j}(x)) F_{S_j}$ in $\mathbb{Z}_\sigma[x]$. All players contribute to generate the polynomial $(\sum_{i=1}^{n} r_{i,j}(x))$ and the polynomial $\sum_{i=1}^{n} r_{i,j}(x)$ is uniformly distributed and unknown. Moreover, the resulting polynomials $u_j(x)$ are uniformly distributed by Lemma 3. Hence, no information can be recovered from the polynomial $F$, $U_j$'s and $u_j$'s, other than that given by revealing the union set. □

*Performance Analysis* It is clear that our protocol runs in $O(1)$ rounds. Let us count the computational and communicational costs for each participant.

- Step 1 (a): requires $\tilde{O}(k)$ multiplications in $\mathbb{Z}_\sigma$ for polynomial expansion of degree $k$ and $O(kd)$ multiplications to compute $F_{S_i}$.
- Step 1 (b): requires $O(d)$ exponentiations for $2d$ encryptions and $O(nd)$ communication costs.
- Step 2 (b): requires $O(d^2)$ exponentiations for computing the encryption $\tilde{\phi_i} := \sum_{j=1}^{n} \tilde{F}_{S_j} *_h r_{i,j}$ and $O(nd)$ communication costs.
- Step 3 (a): requires $O(nd)$ multiplications for computing $\sum_{i=1}^{n} \tilde{\phi_i}$.
- Step 3 (b): requires $O(d)$ exponentiations for decryption share computation for $2d$ ciphertexts and $O(\bar{\ell}\sqrt{dq_i})$ multiplications for solving $d$ DLPs for $\bar{\ell}$ groups of order $q_j$'s.[5] The communication cost is $O(nd)$.

[5] Note that one has to solve $\bar{\ell}$ DLPs over a group of order $q_i$ for one decryption in NS encryption scheme. In Step 3 (b), one has to solve $2d = 2nk$ DLPs over a group of order $q_i$ for each $q_i$. It requires $O(\sqrt{dq_i})$ multiplications to solve $d$ DLPs over a group of order $q_i$ [16] and hence total complexity of this step is $O(\bar{\ell}\sqrt{dq_j})$ multiplications.

Step 4 (a): requires $O(d^2)$ multiplications in $\mathbb{Z}_{q_j}$ to recover $U_j(x)$ using extended Euclidean algorithm for each $j$.
Step 4 (b): requires $O(d^{1.5+o(1)})$ multiplications in $\mathbb{Z}_{q_j}$ for each $j$ to factor a polynomial of degree $d$.
Step 4 (c): requires $O(\bar{\ell} d \log d \log q_j)$ bit operations for sorting and $O(d)$ hash computations.

Then the computational complexity is dominated by one of terms $O(d^2)$ exponentiations in Step 2 (b) and $O(\bar{\ell}\sqrt{dq_i})$ multiplications in Step 3 (b). Since one modular exponentiation for a modulus $N$ requires $O(\log N)$ multiplications and $\bar{\ell} < \min\{d, \frac{\log N}{4 \log\log N}\}$, the computational complexity for each participant is $O(d^2) = O(n^2k^2)$ exponentiations in $\mathbb{Z}_N$ and the total complexity is $O(n^3k^2)$ exponentiations in $\mathbb{Z}_N$. The total communication cost for our protocol is $O(n^2 d) = O(n^3 k)$ $(\log N)$-bit elements.

### 4.3 Set Union for Malicious Case

*Zero-knowledge Proofs* We exploit the following zero-knowledge proofs for the malicious adversary model. We can efficiently construct the above zero-knowledge proofs for the NS encryption scheme by applying some standard techniques [3, 4]. We briefly introduce how to construct the following zero-knowledge proofs.

- $\mathsf{ZKPK}[g(x)|\mathcal{E}_{\mathsf{pk}}(g(x)), \mathcal{E}_{\mathsf{pk}}(f(x)), \mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))]$: this is a zero-knowledge proof that $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ is an encryption of $f(x)g(x)$ when polynomial encryptions $\mathcal{E}_{\mathsf{pk}}(g(x))$, $\mathcal{E}_{\mathsf{pk}}(f(x))$ and $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ are given. In this case, the prover knows only $g(x)$, not $f(x)$. We obtain this protocol by generalizing zero-knowledge proof of correct multiplication which proves $\mathsf{Enc}_{\mathsf{pk}}(c)$ is an encryption of $ab$, when $\mathsf{Enc}_{\mathsf{pk}}(a)$ and $\mathsf{Enc}_{\mathsf{pk}}(b)$ are given for an AHE $\mathsf{Enc}_{\mathsf{pk}}$. This protocol requires $O(nk^2)$ exponentiations for computation and $O(nk^2)$ $(\log N)$-bit elements for communication when $f(x)$ is a polynomial of degree $2nk$ and $g(x)$ is a polynomial of degree $k$.
- $\mathsf{ZKPK}[f(x), g(x)]$: this is a zero-knowledge proof that $g(x)$ is the RLS representation of $1/f(x)$ when encryptions of $f(x)$ and $g(x)$ are given. By the Lemma 2 in [23], if $f(x)$ and $g(x)$ satisfy $\deg(f(x)g(x) - x^{(\deg f + \deg g)}) < \deg f$, then $g(x)$ is the RLS representation of a rational function $1/f(x)$. Hence it is enough to prove that the higher-order $\deg(g(x)) + 1$ coefficient of $f(x)g(x)$ are equal to 1, 0, ..., 0. To prove this, the prover first gives $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ with zero-knowledge proof $\mathsf{ZKPK}[g(x)|\mathcal{E}_{\mathsf{pk}}(g(x)), \mathcal{E}_{\mathsf{pk}}(f(x)), \mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))]$. (In this case, the prover also knows $f(x)$, but the protocol is the same.) Then, using zero-knowledge protocols that a ciphertext is an encryption of 0 and a ciphertext is an encryption of 1, the prover proves that the encryption of the higher-order $\deg g + 1$ of $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ are encryption of $1, 0, \ldots, 0$. It requires $O(nk^2)$ exponentiations and $O(nk^2)$ $(\log N)$-bit elements for communications when $f(x)$ is a polynomial of degree $k$ and $g(x)$ is a polynomial of degree $2nk$.

*Commitment Scheme* We also exploit some equivocal commitment schemes [14, 18] so that the simulator in the malicious adversary model can open the envelope to arbitrary value without being detected by the adversary.

**Our Set Union Protocol for Malicious Case** We give a PPSU-MAL protocol which is secure against malicious adversaries in Figure 5. The parameters are the same with that of protocols for the HBC adversaries model in Section 4.2.

This protocol also runs in $O(1)$ round. The complexities are the same with the protocol for Honest-But-Curious adversary model as a big-O notation except those

**Input:** There are $n \geq 2$ participants $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$ of cardinality $k$. Set $d = nk$. Let $\mathsf{Enc}_{\mathsf{pk}}$ be a threshold NS encryption scheme. Let $\iota : \{0,1\}^* \to \mathbb{Z}_\sigma$ be our encoding function defined in Section 3.We utilize an equivocal commitment scheme and zero-knowledge proofs protocols.

Each participant $\mathcal{P}_i$, $i = 1, \ldots, n$:

1. (a) constructs the polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j})) \in \mathbb{Z}_\sigma[x]$ and runs RationalToRLS$(1, f_{S_i}, (2n+1)k - 1)$ to obtain $\left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]}$, defines $F_{S_i}(x) = \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}$.
   (b) computes $\tilde{f_{S_i}}$, $\tilde{F}_{S_i}$, the encrypted polynomial of $f_{S_i}, F_{S_i}$ and send it to all other participants with proofs of $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
   (c) chooses random polynomials $r_{i,j}(x)$ of degree at most $k$ for all $1 \leq j \leq n$, and sends a commitment of $\Lambda(r_{i,j})$ to all parties, where $\Lambda(r_{i,j}) = \mathcal{E}_{pk}(r_{i,j})$.
2. (a) opens the commitment to $\Lambda(r_{i,j})$.
   (b) verify zero-knowledge proofs $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
   (c) sets the leading coefficient to a known encryption of 1
   (d) calculates $\mu_{i,j}$, the encrypted polynomial of $F_{S_j} \times r_{i,j}$ with proofs of correct multiplication $\mathsf{ZKPK}[r_{i,j}|\Lambda(r_{i,j}), \mathcal{E}_{\mathsf{pk}}(F_{S_{i,j}}), \mu_{i,j}]$ and sends them all other participants.
3. (a) calculates the encrypted polynomial of $F(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} F_{S_j} \times r_{i,j}$ and verified all attached proofs.
   (b) performs a group decryption with all other players to obtain the high-order $2nk$ terms of $F(x)$.
4. (a) recovers a polynomial fair of $u_j(x)$ and $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all $1 \leq j \leq \bar{\ell}$ such that $\left(\frac{u_j(x)}{U_j(x)}\right)_{[-2nk,-1]} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]} \cdot x^{-(2n+1)k+1}$ and $\gcd(u_j(x), U_j(x)) = 1$ in $\mathbb{Z}_{q_j}[x]$, using the high-order $2nk$ terms of $F(x)$ obtained in Step 3 (b).
   (b) extracts all roots of $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all $j$ using a factorization algorithm.
   (c) determines the set union using our rule of representation.

**Fig. 5.** PPSU-MAL protocol in the malicious case

of running zero-knowledge proof protocols. However, to give a zero-knowledge proof of polynomial multiplication and inverse relation, we need to $O(nk^2)$ communication cost and $O(nk^2)$ computational cost. In particular, a zero-knowledge proof protocol $\mathsf{ZKPK}[r_{i,j}|\Lambda(r_{i,j}), \mathcal{E}_{\mathsf{pk}}(F_{S_{i,j}}), \mu_{i,j}]$ has to run for all $1 \leq i \neq j \leq n$, the total communication complexity and computational complexity are $O(n^3k^2)$ $(\log N)$-bit elements and $O(n^3k^2)$ exponentiation and this is the most expensive part in our malicious protocol.

**Theorem 5.** *Assume that the specified zero-knowledge proofs and proofs of correct decryption for the utilized threshold NS encryption scheme cannot be forged. Then, in our set union protocol for the malicious case described in Figure 5, there is a simulator $\mathcal{S}$ for a player (or a group of players) operating in the ideal model, such that the view of the players in the ideal model is computationally indistinguishable from the view of the honest players and any adversaries $\mathcal{A}$ of colluding players in the real world.*

*Proof.* Let $\mathcal{S}$ be a simulator in ideal world, communicating with malicious adversaries $\mathcal{A}$ in the real world, who are able to collude among malicious adversaries. We want to show that malicious adversaries $\mathcal{A}$ in the real world cannot distinguish that $\mathcal{S}$ does not play in the real world.

$\mathcal{S}$ operates the following steps:

1. For each honest player $i$, a simulator $\mathcal{S}$
   (a) chooses monic polynomial $f_i$ such that each such polynomial is relatively prime.
   (b) chooses polynomials $r_{i,1}, \ldots, r_{i,n}$ and creates encryptions $\Lambda(r_{i,j})$ from them.
2. The simulator $\mathcal{S}$ performs Step 1 (a), (b), (c) of our set union protocol:
   (a) calculates $F_{S_i}(x) := \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}$.

(b) computes $\tilde{f_{S_i}}$, $\tilde{F}_{S_i}$, the encrypted polynomial of $f_{S_i}, F_{S_i}$ and send it to all other participants with proofs of $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
(c) chooses random polynomials $r_{i,j}(x)$ of degree at most $k$ for all $1 \leq j \leq n$, and sends trapdoor commitment of $\Lambda(r_{i,j})$ to all parties, where $\Lambda(r_{i,j}) = \mathcal{E}_{pk}(r_{i,j})$.
(d) receives from each malicious player $\alpha \in \mathcal{A}$ the followings: $\tilde{f_{S_\alpha}}$, $\tilde{F}_{S_\alpha}$, $\mathsf{ZKPK}[f_{S_\alpha}, F_{S_\alpha}]$ and trapdoor commitment of $\Lambda(r_{\alpha,j})$ for $1 \leq j \leq n$.

3. The simulator $\mathcal{S}$ extracts $f_{S_\alpha}$ from $\mathsf{ZKPK}[f_{S_\alpha}, F_{S_\alpha}]$ and trapdoor commitments to $\Lambda(r_{\alpha,j})$ to obtain a polynomial $F_\alpha$ and polynomials $r_{\alpha,j}$ for all $\alpha \in \mathcal{A}$.
4. The simulator $\mathcal{S}$ submits all roots to TTP and return the set union.
5. The simulator prepare to reveal the set union:
(a) computes $U(x) =: \prod_{s_j \in U}(x - \iota(s_j)) \in \mathbb{Z}_\sigma[x]$ and compute the high-order $2nk$ RLS representation $F(x)$ of a rational function $\frac{u(x)}{U(x)}$ over $\mathbb{Z}_\sigma$ for a randomly chosen $u(x)$ such that $\gcd(u_j(x), U_j(x)) = 1$ where $\gcd(u_j(x), U_j(x)) = 1$, $\deg u < \deg U$.
(b) computes $U_j(x) := \frac{u_j(x)}{F_j(x)} \bmod q_j$ where $U_j(x) := \prod_{s_j \in U}(x - s_j) \bmod q_j$.
(c) chooses a set of polynomial $r_{i,j}$'s such that $F(x) = \sum_{i=1}^{n} F_{S_j}(x)(\sum_{j=1}^{n} r_{i,j}) \in \mathbb{Z}_\sigma[x]$.
6. The simulator $\mathcal{S}$ follows the rest of the protocol as described and he opens the trapdoor commitment to reveal an appropriate $\Lambda(r_{i,j})$ for the new chosen polynomial $r_{i,j}$. Then the participants calculate an encryption of the polynomial $U$ chosen by the simulator $\mathcal{S}$, and then decrypt it and hence learn the union set.

Note that the colluding players $\mathcal{A}$ cannot distinguish that the simulator $\mathcal{S}$ is in the real world or not and all players obtain the correct answer in both the real world and ideal world. □

### 4.4 Extend to Multi-set Union Protocol

We can easily extend our set union protocol to a multi-set union protocol by modifying our encoding function. Assume that each participant $\mathcal{P}_i$ has a multi-set $S_i \subseteq \mathcal{U}$ for the known universe $\mathcal{U} \subseteq \{0,1\}^{\tau\ell}$. Define a function $\eta : \mathcal{U} \to \mathcal{U}'' \subseteq \{0,1\}^{\tau(\ell+\ell'')}$ by $\eta(s) = s||r$ where $r$ is a randomly chosen element in $\{0,1\}^{\tau\ell''}$. Then each participant takes part in our set union protocol with a set $\{\eta(s_1), \ldots, \eta(s_k)\}$ as his set instead of $\{s_1, \ldots, s_k\}$. For the same messages $s_1$ and $s_2$, if $\eta(s_1)$ is different from $\eta(s_2)$, one can obtain $\eta(s_1)$ and $\eta(s_2)$ as a part of a set union, so the frequency of $s_1$ in the union can be revealed. Hence, if all values of $\eta$ are distinct, we can learn multi-set union.

Consider the probability that there exist at least two same values among $d$ values of function $\eta$. Then this probability is $1 - \left(1 - \frac{1}{2^{\tau\ell''}}\right) \cdots \left(1 - \frac{d-1}{2^{\tau\ell''}}\right) \approx \frac{d^2}{2^{\tau\ell''}}$ and it is less than $2^{-\lambda}$ if $\ell'' > \frac{\lambda + 2\log d - 1}{\log d}$. For example, when $\lambda = 80$ and $d \approx d_0 \approx 2^{10}$, then $\ell''$ is about 10.

Both computational and communicational complexities of our multi-set union protocol are the same with those of our set union protocol as big-O notation. However, it is more heavy than the previous best result [12], which requires $O(n^2k)$ exponentiations in $\mathbb{F}_q$ and $O(n^2k \log q)$ bits where $q$ is the similar size of the size of the universe. However, the public key size of our protocol is $O(1)$ elements, while that of previous result is $O(d)$ elements for the size $d$ of multi-set union since their construction utilized El Gamal encryption schemes defined over an extension field $\mathbb{F}_{p^d}$ of extension degree $d$.

### 4.5 Set Intersection Protocol

In most private set intersection protocols based on polynomial representation and AHE schemes, it is hard to factorize the resulting polynomial corresponding to the intersection of datasets. Hence, it is assumed that the decryptors who want to obtain the set

intersection possess a set having the resulting set as a subset and evaluate at elements of his set to check whether each element is a root of the resulting polynomial. Our polynomial representation provides a private set intersection protocol in which the decryptors efficiently recover the resulting set without a superset of the resulting set.

## 5 Conclusion

In this paper, we provide a new representation of a set by a polynomial over $\mathbb{Z}_\sigma$, which can be efficiently inverted by finding all the linear factors of a polynomial whose root locates in the image of our encoding function, when the factorization of $\sigma$ is public. Then we presented an efficient constant-round set union protocols, transforming our representation into a rational function and then combining it with threshold NS AHE scheme. We also extend our set union protocol to a multi-set union protocol by modifying our representation and consider the set intersection protocol in the case that decryptors do not posses a superset of the resulting set.

## References


1. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In J. Simon, editor, *ACM Symposium on Theory of Computing (STOC) 1988*, pages 1–10. ACM, 1988.
2. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.
3. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report, No. 260, March 1997, 1997.
4. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, 2001.
5. E. D. Cristofaro, J. Kim, and G. Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, 2010.
6. E. D. Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In R. Sion, editor, *Financial Cryptography (FC) 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, 2010.
7. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In Y. Frankel, editor, *Financial Cryptography (FC) 2000*, volume 1962 of *LNCS*, pages 90–104. Springer, 2001.
8. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.
9. K. B. Frikken. Privacy-preserving set union. In J. Katz and M. Yung, editors, *Applied Cryptography and Network Security (ACNS) 2007*, volume 4521 of *LNCS*, pages 237–252. Springer, 2007.
10. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, 2001.
11. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In A. V. Aho, editor, *ACM Symposium on Theory of Computing (STOC) 1987*, pages 218–229. ACM, 1987.
12. J. Hong, J. W. Kim, J. Kim, K. Park, and J. H. Cheon. Constant-round privacy preserving multiset union. IACR Cryptology ePrint Archive, 2011. Available at http://eprint.iacr.org/2011/138.
13. S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In O. Reingold, editor, *Theory of Cryptography (TCC) 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, 2009.
14. J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, 2004.
15. L. Kissner and D. Song. Privacy-preserving set operations. In V. Shoup, editor, *Advances in Cryptology-CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257, 2005.
16. F. Kuhn and R. Struik. Random walks revisited: Extensions of pollard's rho algorithm for computing multiple discrete logarithms. In S. Vaudenay and A. M. Youssef, editors, *Selected Areas in Cryptography (SAC) 2001*, volume 2259 of *LNCS*, pages 212–229. Springer, 2001.

17. V. K. Leont'ev. Roots of random polynomials over a finite field. *Matematicheskie Zametki*, 80(2):313–316, 2006.
18. P. D. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 382–400. Springer, 2004.
19. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In L. Gong and M. K. Reiter, editors, *ACM Conference on Computer and Communications Security (ACM CCS) 1998*, pages 59–66. ACM, 1998.
20. T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 308–318. Springer, 1998.
21. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.
22. Y. Sang and H. Shen. Efficient and secure protocols for privacy-preserving set operations. *ACM Transactions on Information and System Security*, 13(1), 2009.
23. J. H. Seo, J. H. Cheon, and J. Katz. Constant-round multi-party private set union using reversed laurent series. In M. Fischlin, editor, *Public Key Cryptography (PKC) 2012*, LNCS. Springer, 2012. To appear.
24. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
25. V. Shoup. Practical threshold signatures. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000.
26. V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2nd edition, 2009.
27. K. Suzuki, D. Tonien, K. Kurosawa, and K. Toyota. Birthday paradox for multi-collisions. In M. S. Rhee and B. Lee, editors, *Information Security and Cryptography (ICISC) 2006*, volume 4296 of *LNCS*, pages 29–40. Springer, 2006.
28. C. Umans. Fast polynomial factorization and modular composition in small characteristic. In C. Dwork, editor, *ACM Symposium on Theory of Computing (STOC) 2008*, pages 481–490. ACM, 2008.
29. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition, 1999.

## A Threshold Naccache-Stern Cryptosystem

In this section, we provide a threshold version of NS additive homomorphic encryption scheme. Our threshold version is based on Fouque et al.'s work [7], which transforms the original Paillier homomorphic encryption scheme into a threshold version working from Shoup's technique [25]. The security proof of our construction is similar with that of a threshold version of Paillier encryption scheme under the assumption that the original NS additive encryption scheme is semantically secure. We omit the detail of the security proof.

*Threshold Naccache-Stern Encryption Scheme* Our threshold version of the NS encryption scheme consists of the following four algorithms:

- $\mathsf{KeyGen}(\lambda, n, t)$: this algorithm is executed by a dealer. It takes as an input a security parameter $\lambda$, the number $n$ of participating users and the threshold parameter $t$. Then the dealer runs the following steps:
  1. Generate $\bar{\ell}$ $(\tau(\lambda)+1)$-bit primes $q_i$'s where $\tau(\lambda)$ is a polynomial in $\lambda$. We simply write $\tau(\lambda)$ by $\tau$.Let $u = \prod_{i=1}^{\bar{\ell}/2} q_i$ and $v = \prod_{i=\bar{\ell}/2+1}^{\bar{\ell}} q_i$.
  2. Generate two $\kappa(\lambda)$-bit large primes $p'_1, p'_2$.
  3. Compute $p_1 = 2up'_1 + 1$ and $p_2 = 2vp'_2 + 1$. If at least one of $p_1$ and $p_2$ are not prime, go to the first step. Otherwise, let $N = p_1p_2$ and $\sigma = uv$.
  4. The secret key $\phi(N)/\sigma$ is shared with the Shamir's secret sharing scheme [24]: Let $f(x) = \sum_{i=0}^{t} a_i x^i$ where $a_0 = \phi(N)/\sigma$ and $a_i$'s are randomly chosen element in $\mathbb{Z}_{\phi(N)}$. Then the secret share $\mathsf{sk}_i$ for a user $\mathcal{P}_i$ is $f(i) \bmod \phi(N)/\sigma$.
  5. Choose a random element $g$ of order $\phi(N)/4$ in $\mathbb{Z}_N^*$.

6. Choose $\mathsf{vk}$ randomly which generates of the cyclic group of squares in $\mathbb{Z}_N^*$ and let $\mathsf{vk}_i = \mathsf{vk}^{\Delta \mathsf{sk}_i} \bmod N$ where $\Delta = n!$.

It outputs a public key $\mathsf{pk} = (N, g, \sigma, \mathsf{vk}, \{\mathsf{vk}_i\}_{i=1}^n, \Delta)$ and a secret key $\mathsf{sk}_i$ for each user $\mathcal{P}_i$.

- $\mathsf{Enc}(\mathsf{pk}, m)$: this algorithm takes as input a public key $\mathsf{pk}$ and a message $m$. It chooses a random element $x$ in $\mathbb{Z}_N$ and outputs $c = g^m x^\sigma \bmod N$.
- $\mathsf{ShareDec}(\mathsf{pk}, i, \mathsf{sk}_i, c)$: this algorithm takes as an input a public key $\mathsf{pk}$, an index $1 \leq i \leq n$, a secret share $\mathsf{sk}_i$ and a ciphertext $c$. Then it computes $c_i = c^{2\Delta \mathsf{sk}_i}$ along with its proof $\pi_{c_i}$ of the equality of the discrete logarithm between $(c^4, c_i^2)$ and $(\mathsf{vk}, \mathsf{vk}^{\Delta \mathsf{sk}_i})$ and computes $g_i = g^{\Delta \mathsf{sk}_i}$ along with its proof $\pi_{g_i}$ of the equality of the discrete logarithm between $(g, g^{\Delta \mathsf{sk}_i})$ and $(\mathsf{vk}, \mathsf{vk}^{\Delta \mathsf{sk}_i})$. The non-interactive version of the proof of the equality of the discrete logarithm can be easily obtained from [25]. It outputs $(c_i, \pi_{c_i}, g_i, \pi_{g_i})$.
- $\mathsf{Combine}(\mathsf{pk}, S, c, \{c_i\}_{i\in S}, \{\pi_{c_i}\}_{i\in S}, \{g_i\}_{i\in S}, \{\pi_{g_i}\}_{i\in S})$: this algorithm is executed by the combiner. It takes as an input a public key $\mathsf{pk}$, an index set $S$, a ciphertext $c$, its decryption shares $\{c_i\}_{i\in S}$, corresponding proofs $\{\pi_{c_i}\}_{i\in S}$, generator shares $\{g_i\}_{i\in S}$, and corresponding proofs $\{\pi_{g_i}\}_{i\in S}$. Then it runs the following steps:
  1. If $|S| \leq t$, this algorithm returns $\perp$.
  2. Otherwise, it verifies all proofs $\pi_{c_i}$'s and $\pi_{g_i}$'s. If at least one of proofs are failed to verify, return $\perp$.
  3. For a set $S$, compute $\mu_{0,j}^S := \Delta \prod_{j'\in S\setminus\{j\}} \frac{j'}{j'-j} \in \mathbb{Z}$. Then compute $\bar{c} = \prod_{j\in S} c_j^{2\mu_{0,j}^S} \bmod N$ and $\bar{g} = \prod_{j\in S} g_j^{2\mu_{0,j}^S} \bmod N$
  4. For all $q_i | \sigma$, compute $m_i' = \log_{\bar{g}^{\hat{q}_i}} \bar{c}^{\hat{q}_i}$ where $\hat{q}_i = \frac{\sigma}{q_i}$. Then compute $m'$ such that $m' \equiv m_i' \bmod q_i$ for all $i$ using CRT. It outputs $m'$.

*Correctness of Our Construction* By the Lagrange interpolation formula, we obtain

$$\bar{c} = \prod_{j\in S} c_j^{2\mu_{0,j}^S} = \prod_{j\in S} c^{4\Delta \mathsf{sk}_i \mu_{0,j}^S} = c^{4\Delta \sum_{j\in S} \mathsf{sk}_j \mu_{0,j}^S} = c^{4\Delta^2 f(0)}$$

and similarly $\bar{g} = g^{4\Delta^2 f(0)}$. Hence,

$$\begin{aligned}\log_{\bar{g}^{\hat{q}_i}} \bar{c}^{\hat{q}_i} &= \log_{g^{4\Delta^2 \frac{\phi(N)}{q_i}}} c^{4\Delta^2 \frac{\phi(N)}{q_i}} = \log_{g^{4\Delta^2 \frac{\phi(N)}{q_i}}} (g^m x^\sigma)^{4\Delta^2 \frac{\phi(N)}{q_i}} \\ &= \log_{g^{4\Delta^2 \frac{\phi(N)}{q_i}}} (g^m)^{4\Delta^2 \frac{\phi(N)}{q_i}} \equiv (m \bmod q_i),\end{aligned}$$

it gives the correctness of our construction.