이학 박사 학위논문

Homomorphic Encryption Applied to Secure Program Analysis and a New Design

(동형암호와 프로그램 비밀 분석)

2015년 8월

서울대학교 대학원

수리과학부

홍현숙

Homomorphic Encryption Applied to Secure Program Analysis and a New Design

(동형암호와 프로그램 비밀 분석)

지도교수 천정희

이 논문을 이학 박사 학위논문으로 제출함

2015년 4월

서울대학교 대학원

수리과학부

홍현숙

홍현숙의 이학 박사 학위논문을 인준함

2015년 6월

위 원 장	김 명 환	(인)
부 위 원 장	천 정 희	(인)
위 원	이 광 근	(인)
위 원	윤 아 람	(인)
위 원	서 재 홍	(인)

Homomorphic Encryption Applied to Secure Program Analysis and a New Design

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Hyunsook Hong

Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Sciences
Seoul National University

August 2015

Abstract

Homomorphic Encryption Applied to Secure Program Analysis and a New Design

Hyunsook Hong

Department of Mathematical Sciences

The Graduate School

Seoul National University

Homomorphic encryption enables computing certain functions on encrypted data without decryption. Many cloud-based services need efficient homomorphic encryption schemes to provide security to the data in cloud computing.

In this thesis, we focus on applications of homomorphic encryptions for set operation and program analysis, and we suggest a new construction of homomorphic encryption. First, we present a new privacy preserving set union protocol and a secure points-to analysis method as applications of homomorphic encryptions. Our set union protocol is based on the additive homomorphic encryption scheme by Naccache and Stern, whose message space is $\mathbb{Z}_\sigma$ which σ is a product of small primes. We introduce a special polynomial representation such that if a polynomial is represented as this form, then it is factorized uniquely in $\mathbb{Z}_\sigma[X]$. From this representation, we obtain an efficient constant round set union protocol without honest majority assumption.

We adopt a somewhat homomorphic encryption to perform static analysis on encrypted programs. In our method, a somewhat homomorphic encryption scheme of depth $O(\log m)$ is able to evaluate Andersen's pointer analysis with $O(\log m)$ homomorphic matrix multiplications, for the number m of pointer variables when the maximal pointer level is bounded.

Finally, we propose a somewhat homomorphic encryption scheme over the polynomial ring. The security of the proposed scheme is based on the polynomial approximate common divisor problem which can be seen as a polynomial analogous of a base problem of DGHV fully homomorphic encryption and its extension. Our scheme is conceptually simple and does not require a complicated re-linearization process. For this reason, our scheme is more efficient than RLWE-based homomorphic encryption over the polynomial ring when evaluating low degree polynomial of large integers. Furthermore, we convert this scheme to a leveled fully homomorphic encryption scheme, and the resulting scheme has features similar to the variant of van Dijk et al.' s scheme by Coron et al. Our scheme, however, does not use the subset sum, which makes its design much simpler.

Key words: additive homomorphic encryption, somewhat homomorphic encryption, privacy-preserving set union, static analysis in secrecy, pointer analysis, polynomial approximate GCD problem
Student Number: 2009-22898

Contents

Chapter 1

Introduction

Owing to the advancement of information and communication technology, information can be accessed at any time and from any location. The amount of information that is stored and managed by individuals has grown considerably, and cloud services have facilitated easy access to this information. Cloud technology has the advantage that the information stored by users can be accessed through the Internet from anywhere at any time. However, because this information is stored externally, there is a legitimate risk of information leakage. Thus, it is important that the information stored in the cloud is encrypted. Standardly, encrypted information must be decrypted before the information can be accessed, and this exposes the decrypted information to the risk of leakage. To address this problem in cloud environments, security technology using advanced homomorphic encryption is required.

Homomorphic encryption facilitates computation between ciphertexts within the encrypted state. This technology is expected to solve many problems that can occur when private information is stored externally, as it is in the cloud-computing environment.

There are two types of homomorphic encryption considered in this dissertation: additive homomorphic encryption and somewhat homomorphic encryption. (i) An additive homomorphic encryption scheme accepts only homomorphic additions on ciphertexts. (ii) A somewhat homomorphic en-

cryption scheme supports a limited number of homomorphic additions and multiplications over encrypted data.

With somewhat homomorphic encryption, the ciphertext contains a certain amount of noise. The noise increases with successive homomorphic operations, so we use several noise management techniques to handle this noise in ciphertexts.

There are two main techniques used for noise management. The first is the bootstrapping technique, which transfers a ciphertext to a corresponding noise-free ciphertext by homomorphically evaluating the decryption circuit with the encrypted decryption key. The bootstrapping technique facilitates fully homomorphic encryption, supporting multiple arbitrary operations over encrypted data. The other technique is known as modulus switching, and this method reduces noise by scaling it by a factor of the modulus in the ciphertext space. This modulus switching technique provides a leveled fully homomorphic scheme, in which the parameters of a scheme depend polynomially on the depth of the circuits that the scheme can evaluate. The number of times that several processes are performed to adjust the noise is mostly affected by the degree of the circuit under evaluation. Therefore, for encrypted computation using somewhat-homomorphic encryption, the base algorithm for processing data with homomorphic encryption should be expressed as a circuit with a minimum degree.

By contrast, the ciphertexts in an additive homomorphic encryption scheme are not subject to noise. Thus, additive homomorphic encryption does not require a complicated noise-management technique, unlike somewhat homomorphic encryption. Consequently, this type of encryption supports an unlimited number of homomorphic additions on ciphertexts and it is faster than a somewhat homomorphic encryption scheme. However, additive homomorphic encryption does not provide homomorphic multiplication over encrypted data. Therefore, provided that the base algorithm for data processing merely requires addition on ciphertexts or the multiplication of known constants, it is much more effective to use additive homomorphic encryption than some-

what homomorphic encryption.

As a result, it is important to use the appropriate encryption scheme when computing homomorphically encrypted data, and this depends on the necessary type and number of computations on ciphertexts according to the characteristics of the base algorithm that is processed. In this thesis, we focus on applications of homomorphic encryption for set operations and program analysis, and we propose a new design for somewhat homomorphic encryption.

Applying Additive Homomorphic Encryption to a Set-union Protocol We propose the first constant-round privacy-preserving multi-party set union protocol without assuming an honest majority. Our protocol is based on a novel rational function representation of a set over $\mathbb{Z}_\sigma$ and exploits an additive homomorphic encryption scheme over $\mathbb{Z}_\sigma$, where σ is a product of small primes. With our protocol, each user begins with a rational function whose poles consist of elements of a user's own set and ends with a rational function whose poles correspond to the set union. Then, each user computes the denominator of the final rational function using a reconstructed rational function and recovers the set union with a root-finding algorithm. Because $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain, this yields several irrelevant elements. Thus, we introduce a special polynomial representation so that if a polynomial is represented in this form, it is factorized uniquely in $\mathbb{Z}_\sigma[X]$. From this representation, we efficiently recover the set union from the resulting polynomial. Furthermore, our proposed design reduces computational and communicational costs more than previous techniques. We also provide a constant-round privacy-preserving multi-set union protocol by modifying our representation.

Applying Somewhat Homomorphic Encryption to Program Analysis Formerly, program analysis could be performed only after an analyzer or target program was exposed. In other words, program analysis was possible only when the analyzer was provided to the user or when the program

was provided to the user performing the analysis. However, because there are many cases where both the analyzer and the target analysis program contain sensitive information, it is difficult to conduct program analysis when both the analyzer and program cannot be opened. We submit that a somewhat-homomorphic encryption scheme can unleash the potential for the static analysis of encrypted programs. The static analysis of ciphers is desirable as a service, because the program can be encrypted and uploaded to a static-analysis service, and the service provider can analyze the encrypted program without decrypting it. Only the owner of the decryption key (i.e., the program's owner) is able to decrypt the results of the analysis. As a concrete example, we describe how inclusion-based pointer analysis can be performed secretly. With this method, a somewhat-homomorphic encryption scheme of depth $O(\log m)$ is able to evaluate a simple pointer analysis with $O(\log m)$ homomorphic matrix multiplications, for a number m of pointer variables when the maximal pointer level is bounded. We also demonstrate the viability of our method by implementing pointer analysis in secret.

New Design for Fully Homomorphic Encryption We propose a new somewhat homomorphic encryption scheme whose security is based on a new hard problem, called the polynomial approximate common divisor problem. This problem is the polynomial analogue of the approximate integer common-divisor problem. Our scheme is simple and supports homomorphic operations on large integers. The proposed scheme does not require a somewhat complicated re-linearization process, and the operation of ciphertexts is more simplistic than in previous works. When evaluations of a low-degree polynomial of very large integers are required, our scheme is more efficient than the so-called Yet Another Somewhat Homomorphic Encryption (YASHE) scheme proposed by Bos et al. (Cryptography and Coding, 2013) and based on the Ring Learning With Errors (RLWE) problem. In particular, with our scheme, multiplication is ten-times faster that YASHE when evaluating ten-degree polynomials of 1638-bit integers.

We convert this scheme to a leveled fully homomorphic encryption scheme by applying Brakerski's scale-invariant technique, and the resulting scheme has features similar to the variant of van Dijk et al.'s scheme proposed by Coron et al. (PKC, 2014). Our scheme, however, does not use the subset sum, and this simplifies its design.

The base problem is rather new: we performed extensive cryptanalysis, applying various known attacks against problems that are structurally similar. Moreover, we propose a small root-finding algorithm for a multivariate modular-equation system, and we apply it to the proposed problem. Our analyses confirm that the proposed problem is difficult, given the appropriate parameters.

List of Papers This thesis contains results that were obtained jointly with Jung Hee Cheon and Hyung Tae Lee [CHL14], which appears in ICISC 2013, and a work by Woosuk Lee, Kwangkeun Yi, and Jung Hee Cheon [LHYC15] which is accepted to Static Analysis Symposium. It also includes results obtained as a joint effort with a forthcoming article by Jung Hee Cheon, Moon Sung Lee and Hansol Ryu [CHLR14] which has been accepted for publication in Information Sciences.

- [CHL14] Jung Hee Cheon, Hyunsook Hong and Hyung Tae Lee: Invertible Polynomial Representation for Private Set Operations. ICISC 2013: 277-292.

- [LHYC15] Woosuk Lee, Hyunsook Hong, Kwangkeun Yi and Jung Hee Cheon: Static Analysis with Set-closure in Secrecy. To appear in Static Analysis Symposium 2015.

- [CHLR14] Jung Hee Cheon, Hyunsook Hong, Moon Sung Lee and Hansol Ryu: The Polynomial Approximate Common Divisor Problem and its Application to the Fully Homomorphic Encryption. To appear in Information Sciences.

Chapter 2

Private Set Union Protocol

Privacy-preserving set operations (PPSOs) allow a participant to compute set operations between datasets without revealing any data other than the result. There have been many proposals for constructing PPSO protocols using various techniques such as general multi-party computations [GMW87, BOGW88], polynomial representations [FNP04, KS05, Fri07, SS09, HKK$^+$13], pseudo random functions [JL09], and blind RSA signatures [CT10, CKT10]. While the last two techniques are difficult to generalize to multi-party protocols, polynomial representations combined with additive homomorphic encryption (AHE) schemes facilitate the use of multi-party PPSO protocols for various operations including the set intersection [KS05, Fri07, SS09], (over-)threshold set union [KS05], and element reduction [KS05]. Among these constructions, set intersection protocols run in constant rounds, but the others run linear to the number of participants.

We focus on privacy-preserving set union (PPSU) protocols. There are two barriers to constructing constant-round privacy-preserving multi-party set union protocols based on polynomial representations with AHE schemes. First, the set union protocol is conducted by multiplying the polynomials whose roots are the elements of participants' datasets; however, an AHE scheme does not provide multiplication between encrypted data. Second, for the message space $\mathbb{Z}_\sigma$ of the AHE scheme, it is necessary to efficiently find

the roots of the polynomial in $\mathbb{Z}_\sigma$ to recover the union set from the resulting polynomial.

The first barrier is easily overcome by utilizing a rational function representation with the reversed Laurent series in [SCK12]. In this approach, a set of participants is represented by a rational function whose poles are the elements of the set. Then, the denominator of a summation of participants' rational functions corresponds to the union set. Since an AHE scheme supports addition over encrypted date, we can compute the encrypted polynomial, which corresponds to the union set from participants' encrypted rational functions in constant rounds.

Root finding, however, is still an obstacle. There is no appropriate root finding algorithm that works in the message space of AHE schemes. The message space has unknown order [OU98] and is not a unique factorization domain [NS98, Pai99, CS03]. The authors of [SCK12] exploited a secret sharing scheme to provide polynomial multiplication in secrecy. However, this scheme requires computational and communicational costs heavier than the previous and requires an honest majority for security.

Our Results We provide a new polynomial representation that facilitates the unique recovery of roots of a polynomial in $\mathbb{Z}_\sigma[x]$ if the polynomial satisfies certain criteria and if the factorization of $\sigma = \prod_{i=1}^{\bar{\ell}} q_i$ is public for distinct primes q_i.

For a polynomial $f(x) = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$, the number of root candidates of f is usually $d^{\bar{\ell}}$ and is exponential in the size of the modulus σ. To reduce irrelevant candidates, we enforce 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 of the roots of f with negligible failure probability if they are in the image of ι.

As an application of our new representation, we consider the Naccache-Stern (NS) AHE scheme, where the factorization of σ is public, and obtain an efficient constant-round PPSU protocol without an honest majority. In

Table 2.1: Comparison to previous set-union protocols.

HBC	Rounds	Communication Cost	Computational Cost	# of Honest Party
[KS05]	$O(n)$	$O(n^3k\tau_N)$	$O(n^4k^2\tau_N\rho_N)$	≥ 1
[Fri07]	$O(n)$	$O(n^2k\tau_N)$	$O(n^2k^2\tau_N\rho_N)$	≥ 1
[SCK12]	$O(1)$	$O(n^4k^2\tau_{p'})$	$O(n^5k^2\rho_{p'})$	$\geq n/2$
Ours	$O(1)$	$O(n^3k\tau_N)$	$O(n^3k^2\tau_N\rho_N)$	≥ 1
Malicious	Rounds	Communication Cost	Computational Cost	# of Honest Party
[Fri07]	$O(n)$	$O\left((n^2k^2+n^3k)\tau_N\right)$	$O(n^2k^2\tau_N\rho_N)$	≥ 1
[SCK12]	$O(1)$	$O(n^4k^2\tau_p)$	$O(n^5k^2\tau_p\rho_p)$	$\geq n/2$
Ours	$O(1)$	$O(n^3k^2\tau_N)$	$O(n^3k^2\tau_N\rho_N)$	≥ 1

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

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

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

Table 2.1*, we compare our set union protocols to previous results [Fri07, KS05, SCK12].

2.1 Preliminaries

Throughout this chapter, let $\mathcal{U}$ be the universe, n be the number of participants in the protocol, and k be the maximum size of a participant dataset S_i. Furthermore, let d denote the size of set union among participant datasets in the protocol.

2.1.1 Polynomial Representation of a Set

Let R be a commutative ring with unity and S be a subset of R.

*Note that the communication and computational complexities in Table 1 of [SCK12] are for one participant.

Polynomial Representation In many PPSO protocols [FNP04, KS05, Fri07, HKK+13, SCK12], a set S is represented by a polynomial $f_S(x) \in R[x]$ such that the roots of f_S are elements of S; that is,

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

Rational Function Representation In [SCK12], a set S is represented by a rational function representation F_S whose poles are elements of S; that is,

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

Let $F_S(x)$ and $F_{S'}(x)$ be the rational function representations of the sets S and S', respectively.

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

for some polynomial $u(x) \in R[x]$. Hence, the poles of $F_S(x) + F_{S'}(x)$ are exactly the elements of $S \cup S'$ if $u(x)$ and $\mathrm{lcm}(f_S(x), f_{S'}(x))$ have no common roots. To represent a set by a rational function, the authors of [SCK12] exploit a reversed Laurent series, which is an infinite formal power series.

2.1.2 Reversed Laurent Series

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

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

- The RLS representation for a given rational function is unique.

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

1. $F(x) \leftarrow f(x) \cdot x^k$
2. Compute $Q(x)$ and $R(x)$ such that $F(x) = g(x)Q(x) + R(x)$ and $\deg R < \deg g$ using a polynomial division algorithm
3. **Return** $Q(x) \cdot x^{-k}$

Figure 2.1: $\mathsf{RationalToRLS}(f, g, k)$

- Let f and g be polynomials in $\mathbb{Z}_q[x]$ with $\deg f < \deg g$ where $g \neq 0$. Then we can easily compute several higher-order terms of the RLS representation of f/g in $O(\deg g(\deg f + k))$ operations in $\mathbb{Z}_q$ [SCK12]. This algorithm is described in Figure 2.1.

- If $\deg f < \deg g \leq k$ and the bound k on $\deg g$ is given, along with the higher-order $2k$ terms of a rational function f/g, we can recover two polynomials $v(x)$ and $u(x)$ in $\mathbb{Z}_q[x]$ such that $v/u \equiv f/g \pmod q$ and $\gcd(v, u) = 1$ in $O(k^2)$ operations in $\mathbb{Z}_q$ [Sho09, Section 17.5.1].

2.1.3 Additive Homomorphic Encryption

Let R be a commutative ring with unity and G be an R-module, 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 pk. We define a public key encryption for a polynomial $f = \sum_{i=0}^{\deg f} f[i]x^i \in R[x]$ as follows:

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

Assume $\mathsf{Enc}_{\mathsf{pk}}$ has an additive homomorphic property 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 $g = \sum_{i=0}^{\deg g} g[i]x^i$ in $R[x]$, we can induce homomorphic properties of $\mathcal{E}$ as follows:

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

Several efficient AHE schemes already exist [NS98, OU98, Pai99, CS03]. Under the assumption that factoring $N = p^2q$ is difficult, Okamoto and Uchiyama [OU98] 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, the Paillier scheme [Pai99] with $R = \mathbb{Z}_N$ and $G = \mathbb{Z}_{N^2}$ for $N = pq$ was proposed in which the modulus of message spaces is a hard-to-factor composite integer N. Naccache and Stern [NS98] proposed a scheme with $R = \mathbb{Z}_\sigma$ and $G = \mathbb{Z}_N$ under the higher residuosity assumption, where $N = pq$ is a hard-to-factor integer and σ is a product of small primes dividing $\phi(N)$ for Euler's totient function ϕ.

In the above schemes, it is difficult to determine the roots of a polynomial in $R[x]$ without knowing a secret key. Determining a root of the polynomial $f(x) = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_N[x]$ is equivalent to the factor N [Sha93]. However, in the NS scheme, it is possible to compute the roots of a polynomial in $\mathbb{Z}_\sigma[x]$ since the factorization of σ is public. Since $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain (UFD), the number of roots of a polynomial $f \in \mathbb{Z}_\sigma[x]$ can be greater than $\deg f$. In fact, if $f(x) = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$, then the number of roots of the polynomial f is $d^{\bar{\ell}}$, where $\bar{\ell}$ is the number of prime factors of σ. We will use the NS scheme by presenting a method to efficiently recover all of the roots of a polynomial $f \in \mathbb{Z}_\sigma[x]$ satisfying certain criteria.

2.1.4 Root Finding Algorithms

There are several efficient root finding algorithms for polynomials over finite fields. A polynomial of degree d over a field $\mathbb{F}_q$ can be factored in $\widetilde{O}(d^2 \log q)$ field operations using a square-free decomposition and the Cantor-Zassenhaus algorithm [VZGG13, Section 14.4]. Recently, this was improved to $O(d^{1.5+o(1)})$ field operations by Umans [Uma08].

2.2 New Polynomial Representation of a Set

In our set union protocol, each participant performs the following steps:

1. Each participant $\mathcal{P}_i$ represents their own private set S_i as a polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j})) \in \mathbb{Z}_\sigma[x]$ for some encoding function ι, where $\mathbb{Z}_\sigma$ is the message space of the utilized additive homomorphic encryption scheme.

2. Each participant $\mathcal{P}_i$ computes the RLS polynomial

$$F_{S_i}(x) = \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}$$

   and sends the encrypted polynomial $\tilde{F}_{S_i} = \mathcal{E}(F_{S_i})$.

3. After interactions among participants, each participant $\mathcal{P}_i$ obtains the resulting rational function

$$F(x) = \sum_{i=1}^{n} \left(\frac{r_i}{f_{S_i}}\right)_{[k-1,(2n+1)k-2]} = \left(\frac{u}{\operatorname{lcm}(f_{S_1}, \ldots, f_{S_n})}\right)_{[k-1,(2n+1)k-2]}$$

   for random polynomials r_i and u. Then, participant $\mathcal{P}_i$ recovers $f_{\cup S_i}(x) = \operatorname{lcm}(f_{S_1}, \ldots, f_{S_n})$ in $\mathbb{Z}_\sigma[x]$ and finds all roots of $f(x)$ in the image of ι.

In the second step, we execute $\mathsf{RationalToRLS}(1, f_{S_i}, (2n+1)k-1)$ to obtain $(1/f_{S_i}(x))_{[-(2n+1)k+1,-k]}$. However, to perform the third step, there are two important questions that must be addressed:

1. How is $f_{\cup S_i}(x)$ recovered from $F(x)$, and is it unique?

2. How can all roots of $f_{\cup S_i}(x)$ in the image of ι be found efficiently?

For the first question, if F is defined over a field, $f_{\cup S_i}$ can be recovered from F using a rational function reconstruction, and it is unique since the $2nk$ higher-order terms of the RLS representation of $u/\text{lcm}(f_{S_1}, \ldots, f_{S_n})$ are given such that $\deg(\text{lcm}(f_{S_1}, \ldots, f_{S_n})) \leq nk$. For our purposes, F is not defined over a field but over $\mathbb{Z}_\sigma$ for a product σ of small primes. Thus, we apply the rational function reconstruction to each $F(x) \bmod q_j$, where q_j is a divisor of σ.

For the second question, it is difficult to recover the corresponding set $\cup S_i$ from the polynomial $f_{\cup S_i} \in \mathbb{Z}_\sigma[x]$ since $\mathbb{Z}_\sigma$ is not a UFD. Instead, it can be factorized modulo q_j using each divisor q_j of σ, and the Chinese Remainder Theorem (CRT) is applied to determine all roots of $f_{\cup S_i}$. However, this process yields many root candidates.

In this section, we propose a special encoding function ι that enables us to recover the exact corresponding set elements from the polynomial based on our encoding scheme.

Parameter Settings For a given universe $\mathcal{U}$, rational number $0 < \alpha < 1$, and integer τ such that $3\alpha\tau$ and $3(1-\alpha)\tau$ are integers, choose the smallest integer ℓ such that $\mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell}$. Choose distinct primes q_j of $(3\tau+1)$-bit, where $j = 1, \ldots, , \bar{\ell}$, set $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$, and let $\ell' = \bar{\ell} - \ell$.

Naive Approach to Reduce Root Candidates Consider the rule that each root s_i of $f = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$ has the same last bits in $\mathbb{Z}_{q_j}$ for $j = 1, \cdots, \bar{\ell}$. For example, let $h : \{0,1\}^* \to \{0,1\}^{3(1-\alpha)\tau}$ be a uniform hash function. We divide $s_i \in \mathbb{Z}_\sigma$ into $\bar{\ell}$ blocks $s_{i,1}, \ldots, s_{i,\bar{\ell}}$ of $3\alpha\tau$-bit so that $s_i = s_{i,1}||\cdots||s_{i,\bar{\ell}}$. Define the function $\iota' : \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\bar{\ell}} \to \mathbb{Z}_\sigma$ such that $\iota'(s_i)$ is a unique element in $\mathbb{Z}_\sigma$ with $\iota'(s_i) \equiv s_{i,j}||h(s_i) \pmod{q_j}$. Let the polynomial representation of the set S be $f_S(x) = \prod_{s_i \in S}(x - \iota'(s_i)) \in \mathbb{Z}_\sigma[x]$.

The actual roots among the candidates can be determined by checking the hash values $h(s_i)$. However, if there exist s and s' with $h(s) = h(s')$ and $s \neq s'$, there are at least $2^{\bar{\ell}}$ root candidates that are possible combinations of $(s_1, s'_1), \cdots, (s_{\bar{\ell}}, s'_{\bar{\ell}})$. The probability of the event is less than

$$1 - \left(1 - \frac{1}{2^{(1-3\alpha)\tau}}\right) \cdots \left(1 - \frac{d-1}{2^{(1-3\alpha)\tau}}\right) \approx 1 - \exp\left(-\prod_{i=1}^{d-1} \frac{i}{2^{(1-3\alpha)\tau}}\right) \approx \frac{d^2}{2 \cdot 2^{(1-3\alpha)\tau}},$$

which is not negligible even for small α. Moreover, the expected computation cost is $\Omega(2^{\bar{\ell}})$.

2.2.1 New Invertible Polynomial Representation

We present a new polynomial representation to recover a set from a polynomial over $\mathbb{Z}_\sigma$. For optimization, we take $\alpha = \frac{1}{3}$, $\mathcal{U} \subseteq \{0,1\}^{\tau\ell}$. If $\alpha \neq \frac{1}{3}$, the expected computation occurs in polynomial time only when the size of the universe is restricted. Details about the proper size of α can be found in Section 2.2.3.

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

Figure 2.2: Our Encoding Function ι

Let $h : \{0,1\}^* \to \{0,1\}^{2\tau}$ and $h_j : \{0,1\}^* \to \{0,1\}^\tau$ be uniform hash functions for $1 \leq j \leq \ell'$. Parse a message $s_i \in \mathcal{U} \subseteq \{0,1\}^{\tau\ell}$ into ℓ blocks $s_{i,1}, \ldots, s_{i,\ell}$ of τ-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 two blocks $s_{i,\bar{\ell}+1}$ and $s_{i,\bar{\ell}+2}$ of τ-bit. Set $\bar{\ell} = \ell + \ell'$. We define the encoding function $\iota : \mathcal{U} \subseteq \{0,1\}^{\tau\ell} \to \mathbb{Z}_\sigma$, where $\iota(s_i)$ 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 $1 \leq j \leq \bar{\ell}$. Then, a set S can be represented as a polynomial $f_S(x) = \prod_{s_i \in S}(x - \iota(s_i)) \in \mathbb{Z}_\sigma[x]$.

Decoding Phase Let $s_j^{(i)} := \iota(s_i) \bmod q_j$ for each message $s_i = s_{i,1}||\cdots||s_{i,\ell}$. For $1 \le j \le \bar{\ell} - 1$, 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 (2τ)-bit of $s_j^{(i)}$ are equal to the first (2τ)-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.

$$
\begin{aligned}
s_1^{(i_1)} &= s_{i_1,1}||s_{i_1,2}||s_{i_1,3} \\
s_2^{(i_2)} &= \qquad s_{i_2,2}||s_{i_2,3}||s_{i_2,4} \\
s_3^{(i_3)} &= \qquad\qquad s_{i_3,3}||s_{i_3,4}||s_{i_3,5} \\
&\Rightarrow \left(s_1^{(i_1)}, s_2^{(i_2)}, s_3^{(i_3)}\right) \text{ is a linkable pair.}
\end{aligned}
$$

Figure 2.3: Linkable Pair

Let $\iota(s_i)$ and $\iota(s_{i'})$ be the images of elements s_i and $s_{i'}$, respectively, of the function ι with $s_i \neq s_{i'}$. The following properties can easily be confirmed:

- $\left(s_1^{(i)}, \cdots, s_{j+1}^{(i)}\right)$ is always a linkable pair.
- When s_i and $s_{i'}$ are uniformly chosen strings from $\{0,1\}^{\tau\ell}$,

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

  for a fixed $1 \le j \le \bar{\ell}$.

In the decoding phase, when a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota(s_i)) \in \mathbb{Z}_\sigma[x]$ is given, two phases are performed to find the correct d roots of the polynomial $f(x)$. In the first stage, all of the roots $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ are computed over $\mathbb{Z}_{q_j}[x]$ for each j. Sequentially, for each j from 1 to $\bar{\ell}-1$, we determine all 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 (2τ)-bit of $s_j^{(i)}$ and the first (2τ)-bit of $s_{j+1}^{(i')}$ are the same. This can be achieved in d^2 comparisons or $O(d \log d)$ computations using sorting and determining algorithms.

After $\bar{\ell} - 1$ steps, we obtain d' linkable pairs of the $\bar{\ell}$-tuple, which are candidate roots of the polynomial f and elements of the set; this also includes

d elements corresponding to $\iota(s_1), \ldots, \iota(s_d)$. Note that if d' is much larger than d, this can be problematic. However, we show that the expected value of d' is at most $3d$ in Theorem 2.2.1.

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

$$s_{i,\ell+j} = h_j(s_i) \text{ for all } 1 \leq j \leq \ell', \tag{2.2.1}$$

$$s_{i,\bar{\ell}+1} || \, s_{i,\bar{\ell}+2} = h(s_i). \tag{2.2.2}$$

The linkable pairs of the $\bar{\ell}$-tuple, corresponding to $\iota(s_i)$ for some i, clearly satisfy 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 (2.2.1) is approximately $\frac{1}{2^{\tau\ell'}}$, and the probability that it satisfies (2.2.2) is approximately $\frac{1}{2^{2\tau}}$ under the assumption that h and h_j are uniform hash functions. Hence, the expected number of wrong $\bar{\ell}$-tuples passing both phases is less than $d \times \frac{1}{2^{\tau(2+\ell')}}$, which is less than $2^{-\lambda}$ for a security parameter λ if ℓ' satisfies

$$\ell' > \frac{3(\lambda + \log d)}{\log d + 2 \log d_0} - 2. \tag{2.2.3}$$

For example, when $\lambda = 80$ and $d \approx d_0 \approx 2^{10}$, ℓ' is approximately 8. Therefore, a set can be recovered from the given polynomial represented by our scheme without a negligible failure probability in the security parameter.

Computational Complexity We determine the computational cost of our representation. The encoding phase consists of two steps: (1) the CRT computation per element to obtain a value of the encoding function ι, and (2) the polynomial expansion. The first step requires $O(d \log^2 \sigma)$ bit operations for d elements, and the second step requires $O(d^2)$ multiplications. Hence, the complexity for the encoding phase is $O(d^2)$ multiplications.

The decoding phase can be divided into three steps: (1) finding roots of the polynomial f in $\mathbb{Z}_{q_j}$ for each j, (2) finding all linkable pairs of length $\bar{\ell}$,

and (3) verifying (2.2.1) and (2.2.2). These steps require $O(\bar{\ell}d^{1.5})$ multiplications, $O(\bar{\ell}d\log d)$ bit operations, and $O(\ell' d)$ hash computations, respectively. Hence, the complexity for the decoding phase is dominated by $O(\bar{\ell}d^{1.5})$ multiplications.

2.2.2 The Expected Number of Root Candidates

We analyze the expected number of linkable pairs of an $\bar{\ell}$-tuple when a set is recovered from a polynomial of degree d that is represented according to our scheme.

A set of κ elements $s_j^{(1)}, \cdots, s_j^{(\kappa)} \in \mathbb{Z}_{q_j}$ is called a *κ-collision* if its last 2τ-bit are the same. Since $(s_{j-1}^{(i)}, s_j^{(i)})$ is a trivial linkable pair for $1 \leq i \leq \kappa$, a κ-collision induces at least κ linkable pairs. Assume that $S = \{s_1, \ldots, s_d\}$ is a uniformly and randomly chosen set in the set of subsets of cardinality d of the set $\{0,1\}^{\tau\ell}$. Furthermore, assume that h and h_j utilized in the encoding function ι are uniform hash functions. Then the following observations can easily be obtained:

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 κ-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 [STKT06, Theorem 2].

3. A κ-collision in $\mathbb{Z}_{q_j}$ yields κ^2 more root candidates of the polynomial f, not 2^κ candidates. More precisely, assume that a κ-collision $\{s_j^{(1)}, \cdots, s_j^{(\kappa)}\}$ occurs. Then $s_j^{(1)}$ can be combined with κ candidates $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(\kappa)}\}$; hence, κ^2 linkable pairs are generated.

These observations are evidence that the expected number of linkable pairs of an $\bar{\ell}$-tuple is not large. The expected number of 2-collisions in $\mathbb{Z}_{q_j}$ for all j is approximately $\frac{\bar{\ell}}{2}$, and the expected number of κ-collisions in $\mathbb{Z}_{q_j}$ for $\kappa \geq 3$

is negligible. The following theorem provides a rigorous analysis of the upper bound of the expected number of linkable pairs of an $\bar{\ell}$-tuple.

Theorem 2.2.1. *Let $S = \{s_1, \ldots, s_d\}$ be a uniformly and randomly chosen set in the set of subsets of cardinality d of the set $\{0,1\}^{\tau\ell}$. Define an encoding function $\iota : \{0,1\}^{\tau\ell} \to \mathbb{Z}_\sigma$ so that $\iota(s_i)$ 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 all $1 \le j \le \bar{\ell}$ when $s_i = s_{i,1}|| \ldots ||s_{i,\ell}$ and $s_{i,j}$ is τ-bit. Assume h and h_j utilized in the encoding function ι are uniform hash functions. Then the expected number of linkable pairs of the $\bar{\ell}$-tuple is at most $3d$ for a polynomial $f_S = \prod_{s_i \in S}(x - s_i)$.*

Proof. Let E_j be the expected number of linkable pairs of a j-tuple in $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_j}$ for $j \ge 2$. For $1 \le j \le j' \le \bar{\ell}$, let $\mathsf{S}_{j'-j+1}(i_j, \ldots, i_{j'})$ be the event that $(s_j^{(i_j)}, \ldots, s_{j'}^{(i_{j'})})$ is a linkable pair. Then,

$$\begin{aligned}
\mathsf{E}_2 &= \sum_{i_1,i_2 \in \{1,\ldots,d\}} 1 \cdot \Pr[\mathsf{S}_2(i_1,i_2)] \\
&= \sum_{i_1,i_2 \in \{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_2) \wedge (i_1 = i_2)] + \sum_{i_1,i_2 \in \{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_2) \wedge (i_1 \neq i_2)] \\
&= \sum_{i_1 \in \{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_1)] + \sum_{i_1 \neq i_2 \in \{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_2)] \\
&= d + d(d-1)\frac{1}{2^{2\tau}} = d\left(1 + \frac{d-1}{2^{2\tau}}\right),
\end{aligned}$$

since $\Pr[\mathsf{S}_2(i_1,i_1)] = 1$ for $i_1 \in \{1,\ldots,d\}$ and $\Pr[\mathsf{S}_2(i_1,i_2)] = \frac{1}{2^{2\tau}}$ for distinct $i_1, i_2 \in \{1,\ldots,d\}$ from (2.2.1).

Now, we consider the relation between E_j and E_{j+1}. When $(s_1^{(i_1)}, \ldots, s_j^{(i_j)})$ is a linkable pair, consider the case when $(s_1^{(i_1)}, \ldots, s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ is a linkable pair. This case can be classified into the following three subcases:

1. $i_{j+1} = i_j$,
2. $(i_{j+1} \neq i_j) \ \wedge \ (i_{j+1} = i_{j-1})$,
3. $(i_{j+1} \neq i_j) \ \wedge \ (i_{j+1} \neq i_{j-1})$.

For the first case, if $i_{j+1} = i_j$ and $(s_1^{(i_1)}, \ldots, s_j^{(i_j)})$ is a linkable pair, then $(s_1^{(i_1)}, \ldots, s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ is always a linkable pair. Hence,

$$\begin{aligned} \mathsf{E}_{j+1}^{(1)} &:= \sum_{i_1,\ldots,i_{j+1}} \Pr\left[\mathsf{S}_{j+1}(i_1,\ldots,i_j,i_{j+1}) \wedge (i_{j+1} = i_j)\right] \\ &= \sum_{i_1,\ldots,i_j} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] = \mathsf{E}_j. \end{aligned}$$

For the second case, if $i_{j+1} = i_{j-1} \neq i_j$ and $(s_1^{(i_1)}, \ldots, s_j^{(i_j)})$ form a linkable pair, then the relation $s_{i_{j-1},j+1} = s_{i_j,j+1} = s_{i_{j+1},j+1}$ is satisfied from the encoding rule of ι. Hence,

$$\begin{aligned} \mathsf{E}_{j+1}^{(2)} &:= \sum_{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\}} \Pr[\mathsf{S}_{j+1}(i_1,\ldots,i_j,i_{j+1}) \wedge (i_{j+1} = i_{j-1} \neq i_j)] \\ &= \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-1} \neq i_j}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j) \wedge \mathsf{S}_2(i_j,i_{j+1})\right] \\ &= \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-1} \neq i_j}} \Pr[\mathsf{S}_j(i_1,\ldots,i_j) \wedge (s_{i_j,j+1} || s_{i_j,j+2} = s_{i_{j+1},j+1} || s_{i_{j+1},j+2})] \\ &= \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-1} \neq i_j}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j) \wedge (s_{i_j,j+2} = s_{i_{j+1},j+2})\right] \\ &= \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-1} \neq i_j}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] \Pr\left[s_{i_j,j+2} = s_{i_{j+1},j+2}\right] \\ &= \frac{1}{2^\tau} \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-1} \neq i_j}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] \\ &\leq \frac{1}{2^\tau} \sum_{i_1,\ldots,i_j \in \{1,\ldots,d\}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] = \frac{1}{2^\tau}\mathsf{E}_j. \end{aligned}$$

For the last case, the following result can be obtained:

$$\begin{aligned}
\mathsf{E}_{j+1}^{(3)} &:= \sum_{i_1,\ldots,i_{j+1}\in\{1,\ldots,d\}} \Pr[\mathsf{S}_{j+1}(i_1,\ldots,i_j,i_{j+1}) \wedge ((i_{j+1}\neq i_j) \ \wedge \ (i_{j+1}\neq i_{j-1}))] \\
&= \sum_{\substack{i_1,\ldots,i_{j+1}\in\{1,\ldots,d\} \\ i_{j+1}\notin\{i_{j-1},i_j\}}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j) \wedge \mathsf{S}_2(i_j,i_{j+1})\right] \\
&= \sum_{\substack{i_1,\ldots,i_{j+1}\in\{1,\ldots,d\} \\ i_{j+1}\notin\{i_{j-1},i_j\}}} \Pr[\mathsf{S}_j(i_1,\ldots,i_j) \wedge (s_{i_j,j+1}||s_{i_j,j+2} = s_{i_{j+1},j+1}||s_{i_{j+1},j+2})] \\
&= \sum_{\substack{i_1,\ldots,i_{j+1}\in\{1,\ldots,d\} \\ i_{j+1}\notin\{i_{j-1},i_j\}}} \left(\Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] \times \Pr\left[s_{i_j,j+1}||s_{i_j,j+2} = s_{i_{j+1},j+1}||s_{i_{j+1},j+2}\right]\right) \\
&\leq \frac{d-1}{2^{2\tau}} \sum_{i_1,\ldots,i_j\in\{1,\ldots,d\}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] = \frac{d-1}{2^{2\tau}}\mathsf{E}_j.
\end{aligned}$$

The above results yield the following recurrence formula for E_j:

$$\mathsf{E}_{j+1} = \mathsf{E}_{j+1}^{(1)} + \mathsf{E}_{j+1}^{(2)} + \mathsf{E}_{j+1}^{(3)} \leq \left(1 + \frac{1}{2^\tau} + \frac{d-1}{2^{2\tau}}\right)\mathsf{E}_j$$

for $j \geq 2$; hence

$$\mathsf{E}_{\bar{\ell}} \leq d\left(1 + \frac{1}{2^\tau} + \frac{d-1}{2^{2\tau}}\right)^{\bar{\ell}-1},$$

since $\mathsf{E}_2 = d\left(1 + \frac{d-1}{2^{2\tau}}\right) \leq d\left(1 + \frac{1}{2^\tau} + \frac{d-1}{2^{2\tau}}\right)$.

Now, we show that $\bar{\ell} \leq \frac{2^{2\tau}}{2^\tau + d}$. From the parameter settings, it follows that $\bar{\ell} \leq \min\{d, \frac{\lfloor \log N \rfloor - 2}{3\tau}\}$. When $d_0 \geq 8d$,

$$\min\left\{d, \frac{\lfloor \log N \rfloor - 2}{3\tau}\right\} \leq d \leq \frac{d_0^{1/3} d^{2/3}}{2}.$$

Consider the case when $d_0 < 8d$. Then it is also true that

$$\min\left\{d, \frac{\lfloor \log N \rfloor - 2}{3\tau}\right\} \leq \frac{\lfloor \log N \rfloor - 2}{3\tau} \leq \frac{d_0}{3\tau} \leq \frac{d_0^{1/3} d^{2/3}}{2},$$

since $\tau \geq 3$. Hence,

$$\bar{\ell} \leq \min\left\{d, \frac{\lfloor \log N \rfloor - 2}{3\tau}\right\} \leq \frac{d_0^{1/3} d^{2/3}}{2} \leq \frac{(d_0^2 d)^{2/3}}{2d_0} \leq \frac{2^{2\tau}}{2^\tau + d},$$

since $2d_0 > 2^\tau + d$. Therefore, we obtain the following result:

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

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

2.2.3 The Proper Size of α

We consider the proper size of α. Let $\alpha = \frac{b}{a}$ for relatively prime numbers a and b with $a < b$, and assume ℓ and ℓ' are divisible by b. Let $h : \{0,1\}^* \to \{0,1\}^{3(1-\alpha)\tau}$ and $h_i : \{0,1\}^* \to \{0,1\}^{3\alpha\tau}$ be uniform hash functions for $1 \leq i \leq \ell'$. Note that the prime factor q_j of the message modulus σ on the NS AHE scheme is a $(3\tau+1)$-bit prime, and the outputs of the hash functions h and h_i are less than q_j for all j. Assume that $3\alpha\tau$ and $3(1-\alpha)\tau$ are integers. Parse a message $s_i \in \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell}$ into ℓ 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 $h(s_j)$ into $(a-b)$ blocks $s_{i,\bar{\ell}+1}, \cdots, s_{i,\bar{\ell}+a-b}$ of $3\alpha\tau$-bit. Set $\bar{\ell} = \ell + \ell'$.

We define an encoding function $\iota_\alpha : \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell} \to \mathbb{Z}_\sigma$, where $\iota_\alpha(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota_\alpha(s_i) = s_{i,(j-1)b+1}||\cdots||s_{i,(j-1)b+a} \bmod q_j$ for a composite $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$. Then a set S can be represented as a polynomial $f(x) = \prod_{s_i \in S}(x - \iota_\alpha(s_i)) \in \mathbb{Z}_\sigma[x]$.

For each message $s_i = s_{i,1}||\cdots||s_{i,\ell}$, denote $\iota_\alpha(s_i) \bmod q_j$ by $s_j^{(i)}$. We now generalize the definition of a linkable pair. Specifically, 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 $3(1-\alpha)\tau$-bit of $s_j^{(i)}$ are 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.

Let $\iota_\alpha(s_i)$ and $\iota_\alpha(s_{i'})$ be the images of s_i and $s_{i'}$, respectively, of the encoding function ι_α with $s_i \neq s_{i'}$. Assume that s_i and $s_{i'}$ are uniformly chosen strings from $\{0,1\}^{3\alpha\tau\ell}$. Then,

$$\begin{aligned}
&\Pr[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair}] \\
&= \Pr\left[s_{i,jb+1}||\cdots||s_{i,(j-1)b+a} = s_{i',jb+1}||\cdots||s_{i',(j-1)b+a}\right] \\
&= \frac{1}{2^{3(1-\alpha)\tau}} \qquad (2.2.4)
\end{aligned}$$

for fixed $1 \leq j \leq \bar{\ell}$.

In the decoding phase, when a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota_\alpha(s_i))$ is given, one can compute all roots $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ over $\mathbb{Z}_{q_j}$. Then, sequentially for each j from 1 to $\bar{\ell} - 1$, we determine all linkable pairs between $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ and $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(d)}\}$ and check the hash values to find the d correct s_i values.

Theorem 2.2.2. *Let $S = \{s_1, \ldots, s_d\}$ be a uniformly and randomly chosen set in the set of subsets of cardinality d of the set $\{0,1\}^{3\alpha\tau\ell}$ for $0 < \alpha < 1$. Define an encoding function $\iota_\alpha : \{0,1\}^{3\alpha\tau\ell} \to \mathbb{Z}_\sigma$ so that $\iota_\alpha(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota_\alpha(s_i) \equiv s_{i,(j-1)b+1}||\cdots||s_{i,(j-1)b+a} \bmod q_j$ for all $1 \leq j \leq \bar{\ell}$ when $s_i = s_{i,1}||\cdots||s_{i,\ell}$, where $s_{i,j}$ is $3\alpha\tau$-bit and $\alpha = \frac{b}{a}$ for relatively prime a and b. Assume that ℓ and ℓ' are divisible by b and assume h and h_j utilized in the encoding function ι are uniform hash functions.*

The expected number of linkable pairs of an $\bar{\ell}$-tuple is at most

$$d\left(1 + \frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1},$$

for a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota_\alpha(s_i))$.

Proof. Let E_j be the expected number of linkable pairs of a j-tuple in $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_j}$ for $j \geq 2$. For $1 \leq j \leq j' \leq \bar{\ell}$, let $S_{j'-j+1}(i_j, \ldots, i_{j'})$ be the event

that $(s_j^{(i_j)}, \ldots, s_{j'}^{(i'_j)})$ is a linkable pair. Then,

$$\begin{aligned}
\mathsf{E}_2 &= \sum_{i_1,i_2\in\{1,\ldots,d\}} 1 \cdot \Pr[\mathsf{S}_2(i_1,i_2)] \\
&= \sum_{i_1,i_2\in\{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_2) \wedge (i_1 = i_2)] + \sum_{i_1,i_2\in\{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_2) \wedge (i_1 \neq i_2)] \\
&= \sum_{i_1\in\{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_1)] + \sum_{i_1\neq i_2\in\{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1,i_2)] \\
&= d + d(d-1)\frac{1}{2^{3\alpha\tau}} = d\left(1 + \frac{d-1}{2^{3\alpha\tau}}\right),
\end{aligned}$$

since $\Pr[\mathsf{S}_2(i_1,i_1)] = 1$ for $i_1 \in \{1,\ldots,d\}$ and $\Pr[\mathsf{S}_2(i_1,i_2)] = \frac{1}{2^{3\alpha\tau}}$ for distinct $i_1, i_2 \in \{1,\ldots,d\}$ from (2.2.4).

Now, we consider the relation between S_j and S_{j+1}. When $(s_1^{(i_1)}, \ldots, s_j^{(i_j)})$ is a linkable pair, consider the case when $(s_1^{(i_1)}, \ldots, s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ is a linkable pair. This case can be classified into the following subcases:

1. $i_{j+1} = i_j$,
2. $i_{j+1} = i_{j-k} \notin \{i_{j-k+1}, \ldots, i_j\}$ for $k = 0, \ldots, \lfloor \frac{a-1}{b} \rfloor - 1$,
3. $i_{j+1} = i_{j-k} \notin \{i_{j-k+1}, \ldots, i_j\}$ for $k = \lfloor \frac{a-1}{b} \rfloor, \ldots, j-1$.

For the first case, if $i_{j+1} = i_j$ and $\{s_1^{(i_1)}, \cdots, s_j^{(i_j)}\}$ is a linkable pair, then $\{s_1^{(i_1)}, \cdots, s_j^{(i_j)}, s_{j+1}^{(i_{j+1})}\}$ always forms a linkable pair. Hence,

$$\begin{aligned}
\mathsf{E}_{j+1}^{(1)} &:= \sum_{i_1,\ldots,i_{j+1}\in\{1,\ldots,d\}} \Pr\left[\mathsf{S}_{j+1}(i_1,\ldots,i_j,i_{j+1}) \wedge (i_{j+1} = i_j)\right] \\
&= \sum_{i_1,\ldots,i_j\in\{1,\ldots,d\}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j)\right] = \mathsf{E}_j.
\end{aligned}$$

For the second case, if $0 \leq k \leq \lfloor \frac{a-1}{b} \rfloor - 1$,

$$\begin{aligned}
\mathsf{E}_{j+1}^{(2)} &:= \sum_{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\}} \Pr[\mathsf{S}_{j+1}(i_1,\ldots,i_{j+1})d \wedge (i_{j+1} = i_{j-k} \notin \{i_{j-k+1},\ldots,i_j\})] \\
&= \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-k} \notin \{i_{j-k+1},\ldots,i_j\}}} \Pr\left[\mathsf{S}_j(i_1,\ldots,i_j) \wedge \mathsf{S}'_j(i_j, i_{j+1})\right] \\
&\leq \sum_{k=0}^{\lfloor \frac{a-1}{b} \rfloor - 1} \sum_{i_1,\ldots,i_j \in \{1,\ldots,d\}} \frac{1}{2^{3k\alpha\tau}} \Pr[\mathsf{S}_j(i_1,\ldots,i_j) \leq \frac{2}{2^{3\alpha\tau}} \mathsf{E}_j,
\end{aligned}$$

since the last $\tau(1 - \frac{(k+1)b+1}{a})$-bit of $s_{j-k}^{(i_{j-k})}$ and the first $\tau(1 - \frac{(k+1)b+1}{a})$-bit of $s_{j+1}^{(i_{j+1})}$ are equal from the encoding rule of ι_α when $\mathsf{S}'_j(i_j, i_{j+1})$ is the event that $s_{i_j,(j-k-1)b+a+1} || \cdots || s_{i_j,(j-1)b+a} = s_{i_{j+1},(j-k-1)b+a+1} || \cdots || s_{i_{j+1},(j-1)b+a}$.

For $\lfloor \frac{a-1}{b} \rfloor \leq k < j$, if $i_{j+1} = i_{j-k} \notin \{i_{j-k+1},\ldots,i_j\}$, then

$$\{(j-k)b+1,\ldots,(j-k-1)b+a)\} \cap \{jb+1,\ldots,jb+a\} = \emptyset.$$

Hence,

$$\begin{aligned}
\mathsf{E}_{j+1}^{(3)} &:= \sum_{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\}} \Pr[\mathsf{S}_{j+1}(i_1,\ldots,i_{j+1}) \wedge (i_{j+1} = i_{j-k} \notin \{i_{j-k+1},\ldots,i_j\})] \\
&= \sum_{\substack{i_1,\ldots,i_{j+1} \in \{1,\ldots,d\} \\ i_{j+1} = i_{j-k} \notin \{i_{j-k+1},\ldots,i_j\}}} \Pr[\mathsf{S}_j(i_1,\ldots,i_j)] \cdot \Pr[\mathsf{S}_2(i_j, i_{j+1})] \\
&= \sum_{k=\lceil \frac{a-1}{b} \rceil}^{j-1} \sum_{i_1,\ldots,i_j \in \{1,\ldots,d\}} \frac{1}{2^{3(1-\alpha)\tau}} \Pr[\mathsf{S}_j(i_1,\ldots,i_j)] \leq \frac{d}{2^{3(1-\alpha)\tau}} \mathsf{E}_j,
\end{aligned}$$

since $\sum_{k=\lceil \frac{a-1}{b} \rceil}^{j-1} \frac{1}{2^{3(1-\alpha)\tau}} \leq \frac{\bar{\ell}}{2^{3(1-\alpha)\tau}} \leq \frac{d}{2^{3(1-\alpha)\tau}}$.

The above results yield the following recurrence formula for E_j:

$$\mathsf{E}_{j+1} = \mathsf{E}_{j+1}^{(1)} + \mathsf{E}_{j+1}^{(2)} + \mathsf{E}_{j+1}^{(3)} \leq \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \mathsf{E}_j$$

for $j \geq 2$. Therefore,

$$\begin{aligned}
\mathsf{E}_{\bar{\ell}} &\leq \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \sum_{i_1,\ldots,i_{\bar{\ell}-1} \in \{1,\ldots,d\}} \Pr[\mathsf{S}_{\bar{\ell}}(i_1, \ldots, i_{\bar{\ell}-1})] \\
&\leq \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right)^{\bar{\ell}-2} \sum_{i_1,i_2 \in \{1,\ldots,d\}} \Pr[\mathsf{S}_2(i_1, i_2)] \\
&\leq d\left(1 + \frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}.
\end{aligned}$$

□

If $\bar{\ell} \geq 2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$, then $d\left(1 + \frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}$ exponentially increases. Hence, a restriction on $\bar{\ell}$ is required to ensure $\bar{\ell} < 2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$. However, if $\alpha \neq \frac{1}{3}$, then either 3α or $(2-3\alpha)$ is less than 1; hence, $2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$ less than $\min\left\{d, \frac{\lfloor \log N \rfloor - 2}{3\tau}\right\}$ if d and $\log N$ are increasing. This limits the size of the universe since $|\mathcal{U}| \leq 2^{3\alpha\tau\ell}$ when $\alpha \neq \frac{1}{3}$. Therefore, we fix α so that the universe $\mathcal{U}$ is a subset of $\{0,1\}^{\tau\ell}$, *i.e.,* $\alpha = \frac{1}{3}$.

2.3 New Privacy-preserving Set Union Protocols

In this section, we present set union protocols based on our polynomial representation described in Section 2.2. Our construction exploits the NS AHE scheme to encrypt a rational function whose denominator corresponds to a participant's set. We also explain how to modify our polynomial representation for the construction of a multi-set union protocol. Note that some parts of this section are collaborated with results from Hyung Tae Lee [Lee12].

2.3.1 Application of Our Polynomial Representation

In our protocol, we represent each participant's set S_i using the polynomial representation $f_{S_i} := \prod_{s_j \in S_i}(x - \iota(s_j)) \in \mathbb{Z}_\sigma[x]$, where ι is an encoding

function. Then, we convert the rational function $1/f_{S_i}$ to its RLS over $\mathbb{Z}_\sigma$. Since the RationalToRLS algorithm only requires polynomial divisions and the polynomial f_{S_i} is monic, the conversion algorithm works well on $\mathbb{Z}_\sigma[x]$.

After the interactions among participants conclude, each participant obtains $2nk$ higher-order terms of the RLS representation of the rational function $\frac{u(x)}{U(x)}$, where $U(x) = \mathrm{lcm}(f_{S_1}(x), \ldots, f_{S_n}(x))$. Note that 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, this only requires knowing $U'(x) \bmod q_j$ for each j, which can be obtained from the RLS representation modulo q_j by running the polynomial recovery algorithm on $\mathbb{Z}_{q_j}[x]$.

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

Lemma 2.3.1. *Let f and g be polynomials in $\mathbb{Z}_\sigma[x]$ with $\deg f < \deg g$ and g is the monic polynomial. For each prime q that divides σ and an integer $k > \deg g$,*

$$\mathsf{RationalToRLS}(f \bmod q, g \bmod q, k) = \mathsf{RationalToRLS}(f, g, k) \bmod q.$$

Proof. Let $\mathsf{RationalToRLS}(f, g, k) = Q(x)x^{-k}$, where $x^k f(x) = Q(x)g(x) + R(x)$ in $\mathbb{Z}_\sigma[x]$ with $R = 0$ and $\deg R < \deg g$. 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 remainder in $\mathbb{Z}_q[x]$, $\mathsf{RationalToRLS}(f \bmod q, g \bmod q, k) = Q_q(x)x^{-k} \equiv Q(x)x^{-k} \bmod q$. □

The following lemma provides information on the distribution of $u_j(x) := u(x) \bmod q_j$ in our protocol, which will be used to prove the security of our set union protocol. This lemma guarantees that the distributions of $u_j(x)$ and $u(x)$ are uniformly distributed among the polynomials in the set of polynomials of 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 2.3.2 is given in [SCK12].

Lemma 2.3.2 ([SCK12, 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 $r_1(x), \ldots, r_n(x)$ are polynomials in $\mathbb{Z}_q[x]$ chosen uniformly and independently in the set of polynomials of degree at most $k-1$. Let $u(x)$ be a polynomial such that*

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

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

Finally, to recover the exact $U(x) = \text{lcm}(f_{S_1}, \ldots, f_{S_n})$ from the rational function $\frac{u(x)}{U(x)}$, the relation $\gcd(u(x), U(x)) = 1$ must be satisfied. In our set union protocol, since $u_j(x) := u(x) \bmod q_j$ is uniformly distributed in $\mathbb{Z}_{q_j}[x]$ and the expected number of roots of a random polynomial is one [Leo06], we expect our RLS representation to fail to output all the elements in the set union with probability $\frac{d}{q_j} \approx 2^{-2\tau}$. Furthermore, for a certain j, the probability this occurs is approximately $1 - (1 - \frac{d}{q_j})^{\bar{\ell}} \approx \frac{\bar{\ell} d}{q_j} \leq 2^{-\tau}$; although this value is not negligible, it is still sufficiently small.

2.3.2 Honest-But-Curious Model

Threshold Naccache-Stern Encryption In our protocol, group decryption requires a semantically secure threshold NS AHE scheme. We provide a threshold version of the NS encryption scheme in the full version of this paper [CHL12]. Our construction is based on the technique of Fouque et al. [FPS01], which transforms the original Paillier homomorphic encryption scheme into a threshold version using Shoup's technique [Sho00].

Parameter Settings Let $\mathcal{U}$ be the universe, n be the number of participants, and k be the maximum size of participant datasets. Let d be the possible maximum size of the set union, *i.e.,* , $d = nk$. Let the bit size

be N by considering the security of the threshold NS AHE scheme, which is the modulus of the scheme. Furthermore, let $d_0 = \max\{d, \lceil \log N \rceil\}$ and $\tau = \frac{1}{3}(\log d + 2\log d_0)$. Set ℓ so that $\mathcal{U} \subseteq \{0,1\}^{\tau\ell}$, set a proper size of ℓ' so that (2.2.3) is satisfied, and let $\bar{\ell} = \ell + \ell'$. Note that $\bar{\ell}$ must be smaller than $\min\left\{d, \frac{\lfloor \log N \rfloor - 2}{3\log\log N}\right\}$ since $\tau \geq \log\log N$. Generate the parameters of the threshold NS encryption scheme, including the size of the message space σ, which is a product of $\bar{\ell}$ $(3\tau+1)$-bit distinct primes q_i.

Our Set Union Protocol for the Honest-But-Curious Case Our set union protocol against Honest-But-Curious (HBC) adversaries is described in Figure 2.4. In this set union protocol, each participant computes the $2nk$ higher-order terms 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 the encoding function ι and sends its encryption to all other participants. When the encryptions of F_{S_j} for $1 \leq j \leq n$ are received, each participant $\mathcal{P}_i$ multiplies by a polynomial $r_{i,j}$ using the additive homomorphic property and adds all the resulting polynomials to obtain the encryption of $\phi_i(x) = \sum_{j=1}^{n} F_{S_j} \cdot r_{i,j}$. Note that $r_{i,j}$ is randomly chosen by the participant $\mathcal{P}_i$. Then, the participant sends the encryption of $\phi_i(x)$ to all other participants. After interactions among participants conclude, each participant obtains the $2nk$ higher-order term of the RLS representation of $F(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} \frac{1}{f_{S_j}} \cdot r_{i,j}\right) \in \mathbb{Z}_\sigma[x]$ using group decryption. Furthermore, the participants recover the 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 $\mathbb{Z}_{q_j}[x]$ from these values. Thereafter, each participant extracts all roots of $U_j(x)$ over $\mathbb{Z}_{q_j}$ for each j and recovers all elements based on the criteria of our representation.

Security Analysis We consider the correctness and privacy of our proposed protocol described in Figure 2.4. The following theorems guarantee the correctness and privacy of our construction.

Theorem 2.3.1. *In the protocol described in Figure 2.4, every participant*

learns the set union of private inputs of participating players with high probability.

Proof. After Step 3(b), all participants obtain the $2nk$ higher-order terms of $F(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{1}{f_{S_j}} \cdot r_{i,j} \right) \in \mathbb{Z}_\sigma[x]$; hence, they obtain the $2nk$ higher-order terms of the RLS representation of $F(x) \bmod q_j$. From these values, using a polynomial recovery algorithm, they reconstruct polynomials $u_j(x)$ and $U_j(x)$ so 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 (4) and Lemma 2.3.1, $U_j = (\mathrm{lcm}(f_{S_1}, f_{S_2}, \ldots, f_{S_n}) \bmod q_j)$ with high probability. Since our polynomial representation produces the exact corresponding set with overwhelming probability, it provides $S_1 \cup \ldots \cup S_n$. □

Theorem 2.3.2. *Assume that the utilized AHE scheme is semantically secure. Then in the set union protocol for the HBC case described in Figure 2.4, any adversary* Adv*, colluding fewer than n HBC 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 AHE scheme is semantically secure, each participant learns only $F(x) = \sum_{j=1}^{n} (\sum_{i=1}^{n} r_{i,j}) F_{S_j}$ in $\mathbb{Z}_\sigma[x]$. All players contribute to generate the polynomial $\sum_{i=1}^{n} r_{i,j}$, and the polynomial $\sum_{i=1}^{n} r_{i,j}$ is uniformly distributed and unknown. Moreover, the resulting polynomials u_j are uniformly distributed according to Lemma 2.3.2. Hence, no information can be recovered from the polynomial F, U_j, or u_j, other than that given by revealing the union set. □

Performance Analysis It is clear that our protocol runs in $O(1)$ rounds. We now consider the computational and communicational costs of each participant.

Step 1(a) requires $\tilde{O}(k)$ multiplications in $\mathbb{Z}_\sigma$ for a polynomial expansion of degree k and $O(kd)$ multiplications to run the RationalToRLS algorithm and 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 $\tilde{\phi}_i := \sum_{j=1}^{n} \tilde{F}_{S_j} \cdot r_{i,j}$ using the additive homomorphic property 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 the decryption sharing computation for $2d$ ciphertexts and $O(\bar{\ell}\sqrt{dq_i})$ multiplications for solving d discrete logarithm problems (DLPs) for $\bar{\ell}$ groups of order q_j[†]. The communication cost is $O(nd)$.

Step 4(a) requires $O(d^2)$ multiplications in $\mathbb{Z}_{q_j}$ to recover $U_j(x)$ using an 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.

The computational complexity is dominated by one of the $O(d^2)$ exponentiation terms 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 $\bar{\ell} < \min\left\{d, \frac{\lfloor \log N \rfloor - 2}{3 \log \log N}\right\}$, the computational complexity for each participant is dominated by $O(d^2) = O(n^2k^2)$ 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.

2.3.3 Malicious Model

Zero-knowledge Proofs We exploit the following zero-knowledge proofs for the malicious adversary model. We can efficiently construct the required zero-knowledge proofs for the NS encryption scheme by applying standard techniques [CS97, CDN01]. We briefly introduce how to construct the following zero-knowledge proofs. Let $\mathcal{E}_{\mathsf{pk}}$ be the encryption of a polynomial defined

[†]Note that one must solve $\bar{\ell}$ DLPs over a group of order q_i for one decryption in the NS encryption scheme. In Step 3(b), one must 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 [KS01]; hence, the total complexity of this step is $O(\bar{\ell}\sqrt{dq_j})$ multiplications.

in Section 2.1.3.

- $\mathsf{ZKPK}[g(x)|\mathcal{E}_{\mathsf{pk}}(g(x)), \mathcal{E}_{\mathsf{pk}}(f(x)), \mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))]$: 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 participant knows only $g(x)$, not $f(x)$. We obtain this protocol by generalizing the 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 scheme $\mathsf{Enc}_{\mathsf{pk}}$ with public key 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 Lemma 2 in [SCK12], if $f(x)$ and $g(x)$ satisfy

$$\deg\left(f(x)g(x) - x^{(\deg f + \deg g)}\right) < \deg f,$$

  then $g(x)$ is the RLS representation of a rational function $1/f(x)$. Hence, it suffices to prove that the $\deg(g(x)) + 1$ higher-order coefficients of $f(x)g(x)$ are equal to 1, 0, $\cdots$, 0. To prove this, the participant first gives $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ with the 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 participant 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 participant proves that the encryption of the $(\deg g) + 1$ higher-order coefficients of $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ is an encryption of $1, 0, \ldots, 0$. This 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 [KO04, MY04] so that the simulator in the malicious adversary model can open the envelope to an arbitrary value without being detected by the adversary.

Our Set Union Protocol for the Malicious Case We give a PPSU Malicious protocol that is secure against malicious adversaries in Figure 2.5. The parameters are the same as those of the protocols for the HBC model in Section 2.3.2.

This protocol also runs in $O(1)$ rounds. The complexities are the same as those of the protocol for the HBC model except but for the zero-knowledge proof protocols. However, to give a zero-knowledge proof of polynomial multiplication and an inverse relation, we require $O(nk^2)$ communication costs and $O(nk^2)$ computational costs. In particular, a zero-knowledge proof protocol $\mathsf{ZKPK}[r_{i,j}|\Lambda(r_{i,j}), \mathcal{E}_{\mathsf{pk}}(F_{S_{i,j}}), \mu_{i,j}]$ must execute for all $1 \leq i \neq j \leq n$, and the total communication and computational complexities are $O(n^3k^2)$ $(\log N)$-bit elements and $O(n^3k^2)$ exponentiations, respectively; this is the most expensive part of our malicious protocol.

The correctness is similar to that of the HBC case. The following theorem guarantees the security of the protocol proposed in Figure 2.5.

Theorem 2.3.3. *In the set union protocol for the malicious case described in Figure 2.5, 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* Adv *of colluding players in the real world.*

2.3.4 Extension to the 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 the 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 $\{\eta(s_1), \ldots, \eta(s_k)\}$ 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 η are distinct, a multi-set union can be established.

Consider the probability that there exist at least two same values among the d values of function η. Specifically, this probability is given by

$$1 - \left(1 - \frac{1}{2^{\tau\ell''}}\right) \cdots \left(1 - \frac{d-1}{2^{\tau\ell''}}\right) \approx \frac{d^2}{2^{\tau\ell''}}$$

and is less than $2^{-\lambda}$ if $\ell'' > \dfrac{\lambda + 2\log d - 1}{\log d}$. For example, when $\lambda = 80$ and $d \approx d_0 \approx 2^{10}$, then ℓ'' is approximately 10.

Both the computational and communicational complexities of our multi-set union protocol are the same as those of our set union protocol. It is more complex than the previous best result [HKK+13], which requires $O(n^2k)$ exponentiations in $\mathbb{F}_q$ and $O(n^2k \log q)$ bits, where q is similar to the size of the universe. However, the public key size of our protocol is $O(1)$ elements, while that of the previous result in [HKK+13] is $O(d)$ elements for a multi-set union of size d since this particular construction utilizes ElGamal encryption schemes defined over an extension field $\mathbb{F}_{p^d}$ of extension degree d.

2.4 Conclusion

We provided a new representation of a set using a polynomial over $\mathbb{Z}_\sigma$ that can be efficiently inverted by finding all linear factors of a polynomial whose roots are in the image of our encoding function when the factorization of σ is public. Furthermore, we presented an efficient constant-round set union protocol to transform our representation into a rational function and combined it with the threshold NS AHE scheme. We also extended our set union

protocol to a multi-set union protocol by modifying the rational function representation.

We showed that our encoding function is quite efficient on average; however, it still requires exponential time in the degree of the polynomial to recover a set from a polynomial represented by our encoding function. The probability of the worst-case was sufficiently small. Hence, constructing an encoding function that enables the recovery a set in polynomial time in the worst-case is of particular interest.

Input There are $n \geq 2$ HBC participants $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$, where $|S_i| = k$. The participants share the secret key sk, where pk is the corresponding public key for the threshold NS AHE scheme. Let $\iota : \{0,1\}^* \to \mathbb{Z}_\sigma$ be the encoding function provided in Section 2.2 and set $d = nk$.

Goal Each participant obtains $\cup S_i$ without learning other information.

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

1. (a) Computes the polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j})) \in \mathbb{Z}_\sigma[x]$, executes $\mathsf{RationalToRLS}(1, f_{S_i}, (2n+1)k-1)$ to obtain $\left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]}$, and 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 n random polynomials $r_{i,1}(x), \cdots, r_{i,n}(x) \in \mathbb{Z}_\sigma[x]$ of degree at most k.
   (b) Computes $\tilde{\phi}_i$, the encryption of the polynomial $\phi_i(x) = \sum_{j=1}^{n} F_{S_j} \cdot r_{i,j}$, and sends $\tilde{\phi}_i$ to all other 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 participants to obtain the $2nk$ higher-order terms of $F(x)$.

4. (a) Recovers a polynomial pair $(u_j(x), U_j(x)) \in \mathbb{Z}_{q_j}[x] \times \mathbb{Z}_{q_j}[x]$ 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 $2nk$ higher-order 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 the encoding rule of ι.

Figure 2.4: Set union protocol in the Honest-But-Curious case.

Input There are $n \geq 2$ participants $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$, where $|S_i| = k$. The participants share the secret key sk, where pk is the corresponding public key for the threshold NS AHE scheme. Let $\iota : \{0,1\}^* \to \mathbb{Z}_\sigma$ be the encoding function provided in Section 2.2 and set $d = nk$. We utilize an equivocal commitment scheme and zero-knowledge proof protocols.

Goal Each participant obtains $\cup S_i$ without learning other information.

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

1. (a) Computes the polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j})) \in \mathbb{Z}_\sigma[x]$, runs $\mathsf{RationalToRLS}(1, f_{S_i}, (2n+1)k - 1)$ to obtain $\left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]}$, and 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 $\tilde{f}_{S_i}$ and $\tilde{F}_{S_i}$, the encrypted polynomials of f_{S_i} and F_{S_i}, respectively, and sends them to all other participants with proofs of $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
   (c) Chooses n random polynomials $r_{i,1}(x), \cdots, r_{i,n}(x)$ of degree at most k and sends a commitment of $\Lambda(r_{i,j})$ to all other participants, where $\Lambda(r_{i,j})$ is the encrypted polynomial of $r_{i,j}$.
2. (a) Verifies the zero-knowledge proof $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
   (b) Opens the commitment to $\Lambda(r_{i,j})$ and calculates $\mu_{i,j}$, the encrypted polynomial of $F_{S_j} \times r_{i,j}$ with proofs of correct multiplication $\mathsf{ZKPK}[r_{i,j} | \Lambda(r_{i,j}), \tilde{F}_{S_i}, \mu_{i,j}]$.
   (c) Sends $\{\mu_{i,j}\}_{i,j \in [1,n]}$ with proofs to all other participants.
3. (a) Computes the encrypted polynomial of $F(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} F_{S_j} \times r_{i,j}$ and verifies all attached proofs.
   (b) Performs a group decryption with all other participants to obtain the $2nk$ higher-order terms of $F(x)$.
4. (a) Recovers a polynomial pair $(u_j(x), U_j(x))$ in $\mathbb{Z}_{q_j}[x] \times \mathbb{Z}_{q_j}[x]$ for all $1 \leq j \leq \bar{\ell}$ such that $\left(\frac{u_j(x)}{U_j(x)}\right)_{[-2nk,-1]} \cdot x^{(2n+1)k+1} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]}$ and $\gcd(u_j(x), U_j(x)) = 1$ over $\mathbb{Z}_{q_j}$, using the $2nk$ higher-order 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 the encoding rule of ι.

Figure 2.5: Set union protocol in the malicious case.

Chapter 3

Secure Static Program Analysis

Program analysis to find bugs, error, defects and logical flaws in the program can be performed when analyzer or the target program is published. In other words, it is possible that analyzer is provided to the owner of program or the target program is provided on the side to perform this analysis. However, both analyzer and target program often contain sensitive information. Secure program analysis is desirable to protect the sensitive information in program or analyzer.

We report that the homomorphic encryption scheme can unleash the possibility of static analysis of encrypted programs. In our approach, a target program is provided only in the form of encrypted constraints, and analysis is performed on the data without decryption. Only owner of the decryption key is able to decrypt the analysis result. For more widespread use of our system, we explored a method of performing static analysis on encrypted programs. Figure 3.1 depicts the system.

Our work is based on *homomorphic encryption* (HE). A HE scheme enables computation of arbitrary functions on encrypted data. In other words, a HE scheme provides the functions $f_{\oplus}$ and $f_{\wedge}$ that satisfy the following homomorphic properties for plaintexts $x, y \in \{0, 1\}$ without any secrets:

$$\mathsf{Enc}(x \oplus y) = f_{\oplus}(\mathsf{Enc}(x), \mathsf{Enc}(y)), \qquad \mathsf{Enc}(x \wedge y) = f_{\wedge}(\mathsf{Enc}(x), \mathsf{Enc}(y))$$

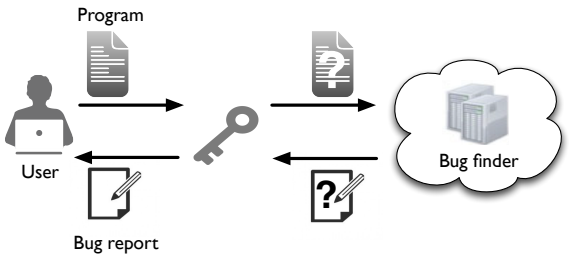

Figure 3.1: Secure static analysis is performed in 3 steps: 1) target program encryption 2) analysis in secrecy, and 3) analysis result decryption

A HE scheme was first shown in the work of Gentry [Gen09]. Since then, although there have been many efforts to improve the efficiency [Bra12, BGV12, CS15, SV14], the cost is still too large for immediate applications into daily computations.

Due to the high complexity of HE operation, practical deployments of HE require *application-specific* techniques. Application-specific techniques are often demonstrated in other fields. Kim et al. [CKL15] introduced an optimization technique to reduce the depth of an arithmetic circuit computing edit distance on encrypted DNA sequences. In addition, methods of bubble sort and insertion sort on encrypted data have been proposed [CKS13]. Also, private database query protocol using somewhat homomorphic encryption has been proposed [BGH$^+$13].

Our Results As a first step, we propose an inclusion-based pointer analysis in secrecy. As many analyses depends on the pointer information, we expect our work to have significant implications along the way to static analysis in secrecy. In our method, a somewhat homomorphic encryption scheme of depth $O(\log m)$ is able to evaluate the pointer analysis with $O(\log m)$ homomorphic matrix multiplications, for the number m of pointer variables when the maximal pointer level is bounded.

Although our interest in this paper is limited to inclusion-based pointer analysis, we expect other analyses in the same family will be performed in a similar manner to our method. Analyses in the family essentially compute

a transitive closure of a graph subject to dynamic changes; new edges may be added during the analysis. Our method computes an encrypted transitive closure of a graph when both edge insertion queries and all the edges are encrypted. Thus, we expect only a few modifications to our method will make other similar analyses (*e.g.* 0-CFA) be in secrecy.

Note that some parts of this chapter were collaborated with results from Woosuk Lee [Lee14]. Woosuk Lee gave the idea that homomorphic encryption probably could be used for secure program analysis. He played an important role in introducing basic construction of pointer analysis in Section 3.2 and in the implementation of the pointer analysis in secrecy in Section 3.4.

3.1 Preliminaries

In this section, we introduce the concept of homomorphic encryption, and describe the security model of our static analysis in secrecy.

3.1.1 Homomorphic Encryption

A homomorphic encryption (HE) scheme $\mathsf{HE}=(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is a quadruple of probabilistic polynomial-time algorithm as follows:

- $(\mathsf{pk}, \mathsf{evk}; \mathsf{sk}) \leftarrow \mathsf{HE.KG}(1^\lambda)$: The algorithm takes the security parameter λ as input and outputs a public encryption key pk, a public evaluation key evk, and a secret decryption key sk.

- $\bar{c} \leftarrow \mathsf{HE.Enc}_{\mathsf{pk}}(\mu, r)$: The algorithm takes the public key pk, a single message $\mu \in \{0,1\}$,* and a randomizer r. It outputs a ciphertext $\bar{c}$. If we have no confusion, we omit the randomizer r.

- $\mu \leftarrow \mathsf{HE.Dec}_{\mathsf{sk}}(\bar{c})$: The algorithm takes the secret key sk and a ciphertext $\bar{c} = \mathsf{HE.Enc}_{\mathsf{pk}}(\mu)$ and outputs a message $\mu \in \{0,1\}$

*For simplicity, we assume that the plaintext space is $\mathbb{Z}_2 = \{0,1\}$ but extension to larger plaintext space is immediate.

- $\bar{c}_f \leftarrow \mathsf{HE.Eval}_{\mathsf{evk}}(f; \bar{c}_1, \ldots, \bar{c}_l)$: The algorithm takes the evaluation key evk, a function $f : \{0,1\}^l \to \{0,1\}$ represented by an arithmetic circuit over $\mathbb{Z}_2 = \{0,1\}$ with the addition and multiplication gates, and a set of l ciphertexts $\{\bar{c}_i = \mathsf{HE.Enc}_{\mathsf{pk}}(\mu_i)\}_{i=1}^{l}$, and outputs a ciphertext $\bar{c}_f = \mathsf{HE.Enc}_{\mathsf{pk}}(f(\mu_1, \cdots, \mu_l))$.

We say that a scheme HE=(KG, Enc, Dec, Eval) is *f-homomorphic* if for any set of inputs $(\mu_1, \cdots, \mu_l)$, and all sufficiently large λ, it holds that

$$\Pr\left[\mathsf{HE.Dec}_{\mathsf{sk}}\left(\mathsf{HE.Eval}_{\mathsf{evk}}(f; \bar{c}_1, \cdots, \bar{c}_l)\right) \neq f(\mu_1, \cdots, \mu_l)\right] = \mathrm{negl}(\lambda),$$

where negl is a negligible function, $(\mathsf{pk}, \mathsf{evk}; \mathsf{sk}) \leftarrow \mathsf{HE.KG}(1^\lambda)$, and $\bar{c}_i \leftarrow \mathsf{HE.Enc}_{\mathsf{pk}}(\mu_i)$.

If a HE scheme can evaluate all functions represented by arithmetic circuits over $\mathbb{Z}_2$ (equivalently, boolean circuits with AND and XOR gates[†]), the HE scheme is called *fully homomorphic*.

To facilitate understanding of HE schemes, we introduce a simple symmetric version of the HE scheme [vDGHV10] based on approximate common divisor problems [HG01]:

- $\mathsf{sk} \leftarrow \mathsf{KG}(1^\lambda)$: Choose an integer p and outputs the secret key $\mathsf{sk} = p$.
- $\bar{c} \leftarrow \mathsf{Enc}(\mu \in \{0,1\})$: Choose a random integer q and a random noise integer r with $|r| \ll |p|$. It outputs $\bar{c} = pq + 2r + \mu$.
- $\mu \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\bar{c})$: Outputs $\mu = ((\bar{c} \bmod p) \bmod 2)$.
- $\bar{c}_{\mathsf{add}} \leftarrow \mathsf{Add}(\bar{c}_1, \bar{c}_2)$: Outputs $\bar{c}_{\mathsf{add}} = \bar{c}_1 + \bar{c}_2$.
- $\bar{c}_{\mathsf{mult}} \leftarrow \mathsf{Mult}(\bar{c}_1, \bar{c}_2)$: Outputs $\bar{c}_{\mathsf{mult}} = \bar{c}_1 \times \bar{c}_2$.

For ciphertexts $\bar{c}_1 \leftarrow \mathsf{Enc}(\mu_1)$ and $\bar{c}_2 \leftarrow \mathsf{Enc}(\mu_2)$, we know each $\bar{c}_i$ is of the form $\bar{c}_i = pq_i + 2r_i + \mu_i$ for some integer q_i and noise r_i. Hence $((\bar{c}_i \bmod p) \bmod 2) =$

[†]AND and XOR gates are sufficient to simulate all binary circuits.

μ_i, if $|2r_i + \mu_i| < p/2$. Then, the following equations hold:

$$\begin{aligned} \bar{c}_1 + \bar{c}_2 &= p(q_1 + q_2) + \underbrace{2(r_1 + r_2) + \mu_1 + \mu_2}_{\text{noise}}, \\ \bar{c}_1 \times \bar{c}_2 &= p(pq_1q_2 + \cdots) + \underbrace{2(2r_1r_2 + r_1\mu_2 + r_2\mu_1) + \mu_1 \cdot \mu_2}_{\text{noise}} \end{aligned}$$

Based on these properties,

$$\mathsf{Dec}_{\mathsf{sk}}(\bar{c}_1 + \bar{c}_2) = \mu_1 + \mu_2 \text{ and } \mathsf{Dec}_{\mathsf{sk}}(\bar{c}_1 \times \bar{c}_2) = \mu_1 \cdot \mu_2$$

if the absolute value of $2(2r_1r_2 + r_1\mu_2 + r_2\mu_1) + \mu_1\mu_2$ is less than $p/2$. The noise in the resulting ciphertext increases during homomorphic addition and multiplication (twice and quadratically as much noise as before respectively). If the noise becomes larger than $p/2$, the decryption result of the above scheme will be spoiled. As long as the noise is managed, the scheme is able to potentially evaluate all boolean circuits as the addition and multiplication in $\mathbb{Z}_2$ corresponds to the XOR and AND operations.

We consider somewhat homomorphic encryption (SWHE) schemes that adopt the modulus-switching [BGV12, BV11a, CNT12, GHS12b] for the noise-management. The modulus-switching reduces the noise by scaling the factor of the modulus in the ciphertext space. SWHE schemes support a limited number of homomorphic operations on each ciphertext, as opposed to fully homomorphic encryption schemes [CCK+13, vDGHV10, Gen09, SV10] which are based on a different noise-management technique. But SWHE schemes are more efficient to support low-degree homomorphic computations.

In this paper, we will measure the efficiency of homomorphic evaluation by the *multiplicative depth* of an underlying circuit. The multiplicative depth is defined as the number of multiplication gates encountered along the longest path from input to output. When it comes to the depth of a circuit computing a function f, we discuss the circuit of the minimal depth among any circuits computing f. For example, if a somewhat homomorphic encryption

scheme can evaluate circuits of depth L, we may maximally perform 2^L multiplications on the ciphertexts maintaining the correctness of the result. We do not consider the number of addition gates in counting the depth of a circuit because the noise increase by additions is negligible compared with the noise increase by multiplications. The multiplicative depth of a circuit is the most important factor in the performance of homomorphic evaluation of the circuit in the view of both the size of ciphertexts and the cost of per-gate homomorphic computation. Thus, minimizing the depth is the most important in performance.

3.1.2 The BGV-type Cryptosystem

Notations. For an integer q, we denote the ring of integers modulo q by $\mathbb{Z}_q$. Let $\Phi(X)$ be an irreducible polynomial over $\mathbb{Z}$. Our implementation is based on the operations in polynomial ring $R = \mathbb{Z}[X]/(\Phi(X))$ which is the set of integer polynomials of degree less than $\deg \Phi$. We identify the quotient ring $R_q := R/qR$ with the set of integer polynomials of degree up to $\deg \Phi - 1$ reduced modulo q for the integer q.

Our underlying HE scheme is a variant of the Brakerski-Gentry-Vaikuntanathan (BGV) cryptosystem [BGV12, GHS12a] using a modulus switching technique. We recall that the BGV cryptosystem [BGV12] based on the hardness of the "ring learning with errors " (RLWE) problem [LPR10]. The RLWE problem is to distinguish pair $(a_i, b_i = a_i \cdot \mathbf{s} + e_i) \in R_q \times R_q$ from uniformly random pairs, where $\mathbf{s} \in R_q$ is a random "secret" polynomial which remains fixed over all pairs, the $a_i \in R_q$ are uniformly random and independent, and the "noise" terms $e_i \in R$ are sampled from a noise distribution that outputs polynomials whose coefficients much "smaller" than q (an example is a discrete Gaussian distribution over R with small standard deviation).

For a polynomial ring $R = \mathbb{Z}[X]/(\Phi(X))$, we set the message space to R_p for some fixed prime $p \geq 2$ and the ciphertext space to $R_q \times R_q$ for an integer q. Then all the ciphertexts are treated as vectors of elements in R_q.

Now, we describe the BGV cryptosystem as follows:

- $((a, b), \mathsf{evk}; \mathbf{s}) \leftarrow \mathsf{BGV.KG}(1^\lambda, w, \sigma, q)$: Chooses a weight w secret key $\mathbf{s}$ and generates a RLWE instance (a, b) relative to the secret key $\mathbf{s}$. Compute a evaluation key for a homomorphic evaluation of ciphertexts. Output the public key $\mathsf{pk} = (a, b)$, the evaluation key evk, and the secret key $\mathsf{sk} = \mathbf{s}$.

- $\bar{\mathbf{c}} \leftarrow \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu)$: To encrypt a message $\mu \in R_t$, choose a random polynomial v whose coefficients are in $\{0, \pm 1\}$ and two noise polynomials e_0, e_1 from a discrete Gaussian distribution over R with standard deviation σ. Outputs the ciphertext $\mathbf{c} = (c_0, c_1) = (bv + pe_0 + \mu, av + pe_1) \bmod q$.

- $\mu \leftarrow \mathsf{BGV.Dec}_{\mathsf{sk}}(\bar{\mathbf{c}})$: Given a ciphertext $\bar{\mathbf{c}} = (c_0, c_1)$, it outputs $\mu = ((c_0 - c_1 \cdot \mathbf{s} \bmod q) \bmod p)$.

- $\bar{\mathbf{c}}_{\mathsf{add}} \leftarrow \mathsf{BGV.Add}_{\mathsf{evk}}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2)$: Given ciphertext $\bar{\mathbf{c}}_1 = \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu_1)$ and $\bar{\mathbf{c}}_2 = \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu_2)$, it outputs the ciphertext $\bar{\mathbf{c}}_{\mathsf{add}} = \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu_1 + \mu_2)$.

- $\bar{\mathbf{c}}_{\mathsf{mult}} \leftarrow \mathsf{BGV.Mult}_{\mathsf{evk}}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2)$: Given ciphertext $\bar{\mathbf{c}}_1 = \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu_1)$ and $\bar{\mathbf{c}}_2 = \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu_2)$, it outputs the ciphertext $\bar{\mathbf{c}}_{\mathsf{mult}} = \mathsf{BGV.Enc}_{\mathsf{pk}}(\mu_1 \cdot \mu_2)$.

In the BGV scheme, homomorphic addition is done by simple componentwise addition of the ciphertexts and homomorphic multiplication is by tensor product over R_q. Since the norm of the noise and the degree of the ciphertext are increased after operations of ciphertexts, modulus and key switching operation should be performed to reduce the norm of the noise and the degree of the ciphertext. For more details to the homomorphic operations on the BGV-type cryptosystem such as the key switching and modulus switching, please refer to [BGV12, GHS12b].

3.1.3 Security Model

We assume that program owners and analyzer servers are semi-honest. In this model, the analyzer runs the protocol exactly as specified, but may try to learn as much as possible about the program information. However, in our

method, since programs are encrypted under the BGV-type cryptosystem which is secure under the hardness of the RLWE problem, analyzers cannot learn no more information than the program size.

3.2 A Basic Construction of a Pointer Analysis in Secrecy

In this section, we explain how to perform an inclusion-based pointer analysis in secrecy.

3.2.1 Inclusion-based Pointer Analysis

We begin with a brief review of inclusion-based pointer analysis. We consider flow- and context-insensitive pointer analyses. To simplify our presentation, we consider a tiny language consisting of primitive assignments involving just the operations `*` and `&`. A program P is a finite set of assignments `A`:

$$\texttt{A} \;\rightarrow\; \texttt{x} = \&\texttt{y} \mid \texttt{x} = \texttt{y} \mid *\texttt{x} = \texttt{y} \mid \texttt{x} = *\texttt{y}$$

We present inclusion-based pointer analysis algorithm with simple resolution rules in a similar manner to [HT01]. Given some program P, we construct resolution rules as specified in Table 3.1. In the first rule, the side condition "if $\texttt{x} = \&\texttt{y}$ in P" indicates that there is an instance of this rule for each occurrence of an assignment of the form $\texttt{x} = \&\texttt{y}$ in P. The side conditions in the other rules are similarly interpreted. Intuitively, an edge $\texttt{x} \longrightarrow \&\texttt{y}$ indicates that `x` can point to `y`. An edge $\texttt{x} \longrightarrow \texttt{y}$ indicates that for any variable `v`, if `y` may point to `v` then `x` may point to `v`. The pointer analysis is applying the resolution rules until reaching a fixpoint.

$$\frac{}{\mathtt{x} \longrightarrow \&\mathtt{y}} \ (\text{if } \mathtt{x} = \&\mathtt{y} \text{ in } P) \qquad (\mathsf{New})$$

$$\frac{}{\mathtt{x} \longrightarrow \mathtt{y}} \ (\text{if } \mathtt{x} = \mathtt{y} \text{ in } P) \qquad (\mathsf{Copy})$$

$$\frac{\mathtt{x} \longrightarrow \&\mathtt{z}}{\mathtt{y} \longrightarrow \mathtt{z}} \ (\text{if } \mathtt{y} = *\mathtt{x} \text{ in } P) \qquad (\mathsf{Load})$$

$$\frac{\mathtt{x} \longrightarrow \&\mathtt{z}}{\mathtt{z} \longrightarrow \mathtt{y}} \ (\text{if } *\mathtt{x} = \mathtt{y} \text{ in } P) \qquad (\mathsf{Store})$$

$$\frac{\mathtt{x} \longrightarrow \mathtt{z} \quad \mathtt{z} \longrightarrow \&\mathtt{y}}{\mathtt{x} \longrightarrow \&\mathtt{y}} \qquad (\mathsf{Trans})$$

Table 3.1: Resolution rules for pointer analysis.

3.2.2 The Pointer Analysis in Secrecy

The analysis in secrecy will be performed in the following 3 steps. First, a program owner derives numbers that represent his program and encrypt them under a HE scheme. The encrypted numbers will be given to an analysis server. Next, the server performs homomorphic evaluation of an underlying arithmetic circuit representing the inclusion-based pointer analysis with the inputs from the program owner. Finally, the program owner obtains an encrypted analysis result and recovers a set of points-to relations by decryption.

Before beginning, we define some notations. We assume a program owner assigns a number to every variable using some numbering scheme. In the rest of the paper, we will denote a variable numbered i by $\mathtt{x_i}$. In addition, to express the arithmetic circuit of the pointer analysis algorithm, we define the notations $\delta_{i,j}$ and $\eta_{i,j}$ in $\mathbb{Z}$ for $i, j = 1, \cdots, m$ by

$\delta_{i,j} \neq 0$	iff	An edge $\mathtt{x_i} \longrightarrow \&\mathtt{x_j}$ is derived by the resolution rules.
$\eta_{i,j} \neq 0$	iff	An edge $\mathtt{x_i} \longrightarrow \mathtt{x_j}$ is derived by the resolution rules.

for variables $\mathtt{x_i}$ and $\mathtt{x_j}$, and the number m of pointer variables.

Inputs from Client A client (program owner) derives the following numbers that represent his program P (here, m is the number of variables):

$$\{(\delta_{i,j}, \eta_{i,j}, u_{i,j}, v_{i,j}) \in \mathbb{Z} \times \mathbb{Z} \times \{0,1\} \times \{0,1\} \mid 1 \leq i, j \leq m\}$$

which are initially assigned as follows:

$$\delta_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists \mathtt{x_i} = \&\mathtt{x_j} \\ 0 & \text{otherwise} \end{cases} \qquad \eta_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists \mathtt{x_i} = \mathtt{x_j} \text{ or } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$u_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists \mathtt{x_j} = *\mathtt{x_i} \\ 0 & \text{otherwise} \end{cases} \qquad v_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists *\mathtt{x_j} = \mathtt{x_i} \\ 0 & \text{otherwise} \end{cases}$$

In the assignment of $\delta_{i,j}$, the side condition $\exists \mathtt{x_i} = \&\mathtt{x_j}$ indicates that there is the assignment $\mathtt{x_i} = \&\mathtt{x_j}$ in the program P. The other side conditions are similarly interpreted.

The program owner encrypts the numbers using a HE scheme and provides them to the server. We denote the encryption of $\delta_{i,j}$, $\eta_{i,j}$, $u_{i,j}$, and $v_{i,j}$ by $\bar{\delta}_{i,j}$, $\bar{\eta}_{i,j}$, $\bar{u}_{i,j}$, and $\bar{v}_{i,j}$, respectively. Therefore, the program owner generates $4m^2$ ciphertexts where m is the number of pointer variables.

Server's Analysis Provided the set of the ciphertexts from the program owner, the server homomorphically applies the resolution rules. With a slight abuse of notation, we will denote $+$ and $\cdot$ as homomorphic addition and multiplication respectively to simplify the presentation.

We begin with applying the Trans rule in Table 3.1. For $i, j = 1, \cdots, m$, the server updates $\bar{\delta}_{i,j}$ as follows:

$$\bar{\delta}_{i,j} \leftarrow \textstyle\sum_{k=1}^{m} \bar{\eta}_{i,k} \cdot \bar{\delta}_{k,j}$$

If edges $\mathtt{x_i} \longrightarrow \mathtt{x_k}$ and $\mathtt{x_k} \longrightarrow \&\mathtt{x_j}$ are derived by the resolution rules for some variable $\mathtt{x_k}$, then the edge $\mathtt{x_i} \longrightarrow \&\mathtt{x_j}$ will be derived by the Trans rule and the value $\delta_{i,j}$ will have a positive integer. If there is no variable $\mathtt{x_k}$

that satisfies the conditions for all $k = 1, \cdots, m$, there will be no update on $\delta_{i,j}$ ($\because \eta_{i,i} = 1$).

Next, we describe applying the Load rule.

$$\bar{\eta}_{i,j} \leftarrow \bar{\eta}_{i,j} + \sum_{k=1}^{m} \bar{u}_{i,k} \cdot \bar{\delta}_{k,j}$$

If an edge $\mathtt{x_k} \longrightarrow \&\mathtt{x_j}$ is derived and the program P has a command $\mathtt{x_i} := *\mathtt{x_k}$ and for some integer k, then the edge $\mathtt{x_i} \longrightarrow \mathtt{x_j}$ will be derived and $\eta_{i,j}$ will have a positive value. If none of variables $\mathtt{x_k}$ satisfies the conditions, there will be no update on $\eta_{i,j}$.

Finally, to apply the Store rule, the server performs the following operations:

$$\bar{\eta}_{i,j} \leftarrow \bar{\eta}_{i,j} + \sum_{k=1}^{m} \bar{v}_{j,k} \cdot \bar{\delta}_{k,i}$$

If an edge $\mathtt{x_k} \longrightarrow \&\mathtt{x_i}$ is derived and the program P has a command $*\mathtt{x_k} := \mathtt{x_j}$ for some variable $\mathtt{x_k}$, then an edge $\mathtt{x_i} \longrightarrow \mathtt{x_j}$ will be derived and $\eta_{i,j}$ will have a non-zero value.

Note that the server must repeat applying the rules as if in the worst case since the server cannot know whether a fixpoint is reached during the operations. The server may obtain a fixpoint by repeating the following two steps in turn m^2 times:

1. Applying the Trans rule m times
2. Applying the Load and Store rules

The reason for doing step 1 is that we may have a m-length path through edges as the longest one in the worst case. The reason for repeating the two steps m^2 times is that we may have a new edge by applying the Load and Store rules, and we may have at most m^2 edges at termination of the analysis.

We need $O(m^2 \log m)$ multiplicative depth in total. Because performing the step 1 entails m homomorphic multiplications on each $\bar{\delta}_{i,j}$, and repeating the two steps m^2 times performs about m^{m^2} homomorphic multiplications on each $\bar{\delta}_{i,j}$.

Output Determination The client receives the updated $\{\bar{\delta}_{i,j} \mid 1 \leq i, j \leq m\}$ from the server and recovers a set of points-to relations as follows:

$$\{\mathtt{x_i} \longrightarrow \&\mathtt{x_j} \mid \mathsf{HE.Dec_{sk}}(\bar{\delta}_{i,j}) \neq 0 \text{ and } 1 \leq i, j \leq m\}$$

Why don't we represent the algorithm by a Boolean circuit? One may wonder why we represent the pointer analysis algorithm by an arithmetic circuit rather than a Boolean circuit. As an example of applying the Trans rule, we might update $\delta_{i,j}$ by the following method:

$$\delta_{i,j} \leftarrow \bigvee_{1 \leq k \leq m} \eta_{i,k} \wedge \delta_{k,j}$$

However, this representation causes more multiplicative depth than our current approach. The OR operation consists of the XOR and AND operations as follows:

$$x \vee y \stackrel{\mathtt{def}}{=} (x \wedge y) \oplus x \oplus y$$

Note that the addition and multiplication in $\mathbb{Z}_2$ correspond to the XOR and AND operations, respectively. Since the OR operation requires a single multiplication over ciphertexts, this method requires m more multiplications than our current method to update $\delta_{i,j}$ once.

3.3 Improvement of the Pointer Analysis in Secrecy

In this section, we present three techniques to reduce the cost of the basic approach described in the section 3.2.2. We begin with problems of the basic approach followed by our solutions.

3.3.1 Problems of the Basic Approach

The basic scheme has the following problems that make the scheme impractical.

- Huge # of homomorphic multiplications: The scheme described in the section 3.2.2 can be implemented with a SWHE scheme of the depth $O(m^2 \log m)$. Homomorphic evaluation of a circuit over the hundreds depth is regarded unrealistic in usual. The depth of the arithmetic circuit described in the section 3.2.2 exceeds 300 even if a program has only 10 variables.

- Huge # of ciphertexts: The basic approach requires $4m^2$ ciphertexts, where m is the number of pointer variables. When a program has 1000 variables, 4 million ciphertexts are necessary. For instance, the size of a single ciphertext in the BGV cryptosystem is about 2MB when the depth is 20. In this case, the scheme requires 7.6 TB memory space for all the ciphertexts.

- Decryption error may happen: In our underlying HE scheme, the message space is the polynomial ring over modulus p. During the operations, $\delta_{i,j}$ and $\eta_{i,j}$ increase and may become p which is congruent to 0 modulo p. Since we are interested in whether each value is zero or not, incorrect results may be derived if the values become congruent to 0 modulo p by accident.

3.3.2 Overview of Improvement

For the number m of pointer variables and the maximal pointer level n, the followings are our solutions.

- **Level-by-level Analysis**: We analyze pointers of the same level together from the highest to lowest in order to decrease the depth of the arithmetic circuit described in the section 3.2.2. To apply the technique, program owners are required to reveal an upper bound of the

maximal pointer level. By this compromise, the depth of the arithmetic circuit significantly decreases: from $O(m^2 \log m)$ to $O(n \log m)$. We expect this information leak is not much compromise because the maximal pointer level is well known to be a small number in usual cases.

- **Ciphertext Packing**: We adopt ciphertext packing not only for performance boost but also for decreasing the huge number of ciphertexts required for the basic scheme. The technique makes total ciphertext sizes be linear to the number of variables.
- **Randomization of Ciphertexts**: We randomize ciphertexts to balance the probability of incorrect results and ciphertext size. We may obtain correct results with the probability of $\left(1 - \frac{1}{p-1}\right)^{n(\lceil \log m \rceil + 3)}$.

The following table summarizes the improvement.

	Multiplicative depth	# Ctxt
Basic	$O(m^2 \log m)$	$4m^2$
Improved	$O(n \log m)$	$(2n+2)m$

m : the number of pointer variables in the target program
n : the maximum level of pointer in the program, which does not exceed 5 in usual

Table 3.2: The comparison between the basic and the improved scheme

3.3.3 Level-by-level Analysis

We significantly decrease the multiplicative depth by doing the analysis in a level by level manner in terms of *level of pointers*. The level of a pointer is the maximum level of possible indirect accesses from the pointer, *e.g.* the pointer level of `p` in the definition "`int** p`" is 2. From this point, we denote the level of a pointer variable `x` by `ptl(x)`.

We assume that type-casting a pointer value to a lower or higher-level pointer is absent in programs. For example, we do not consider a program that has type-casting from `void*` to `int**` because the pointer level increases from 1 to 2. On the assumption, we analyze the pointers of the same level together from the highest to lowest. The correctness is guaranteed because lower-level pointers cannot affect pointer values of higher-level pointers during the analysis. For example, pointer values of x initialized by assignments of the form $\mathtt{x} = \&\mathtt{y}$ may change by assignments of the form $\mathtt{x} = \mathtt{y}$, $\mathtt{x} = *\mathtt{y}$, or $*\mathtt{p} = \mathtt{y}$ ($\because \mathtt{p}$ may point to $\mathtt{x}$) during the analysis.

The following table presents pointer levels of involved variables in the assignments that affects pointer values of $\mathtt{x}$. Note that all the variables affect pointer values of $\mathtt{x}$ have higher or equal pointer level compared to $\mathtt{x}$.

Assignment	Levels
$\mathtt{x} = \mathtt{y}$	$\mathsf{ptl}(\mathtt{x}) = \mathsf{ptl}(\mathtt{y})$
$\mathtt{x} = *\mathtt{y}$	$\mathsf{ptl}(\mathtt{y}) = \mathsf{ptl}(\mathtt{x}) + 1$
$*\mathtt{p} = \mathtt{y}$	$\mathsf{ptl}(\mathtt{p}) = \mathsf{ptl}(\mathtt{x}) + 1 \wedge \mathsf{ptl}(\mathtt{y}) = \mathsf{ptl}(\mathtt{x})$

Now we describe the level-by-level analysis in secrecy similarly to the basic scheme. Before beginning, we define the notations $\delta_{i,j}^{(\ell)}$ and $\eta_{i,j}^{(\ell)}$ in $\mathbb{Z}$ for $i, j = 1, \cdots, m$ by

$$\begin{array}{lll} \delta_{i,j}^{(\ell)} \neq 0 & \text{iff} & \text{An edge } \mathtt{x_i} \longrightarrow \&\mathtt{x_j} \text{ is derived and } \mathsf{ptl}(\mathtt{x_i}) = \ell \\ \eta_{i,j}^{(\ell)} \neq 0 & \text{iff} & \text{An edge } \mathtt{x_i} \longrightarrow \mathtt{x_j} \text{ is derived and } \mathsf{ptl}(\mathtt{x_i}) = \ell. \end{array}$$

Inputs from Client For the level-by-level analysis, a program owner derives the following numbers that represent his program P (n is the maximal level of pointer in the program):

$$\{(\delta_{i,j}^{(\ell)}, \eta_{i,j}^{(\ell)}) \mid 1 \leq i, j \leq m, 1 \leq \ell \leq n\} \cup \{(u_{i,j}, v_{i,j}) \mid 1 \leq i, j \leq m\}$$

where $\delta_{i,j}^{(\ell)}$ and $\eta_{i,j}^{(\ell)}$ are defined as follows.

$$\delta_{i,j}^{(\ell)} = \begin{cases} 1 & \text{if } \exists \mathtt{x_i} = \&\mathtt{x_j} \text{ and } \mathsf{ptl}(\mathtt{x_i}) = \ell \\ 0 & \text{o.w.} \end{cases}$$

$$\eta_{i,j}^{(\ell)} = \begin{cases} 1 & \text{if } (\exists \mathtt{x_i} = \mathtt{x_j} \text{ or } i = j) \text{ and } \mathsf{ptl}(\mathtt{x_i}) = \ell \\ 0 & \text{o.w.} \end{cases}$$

The definitions of $u_{i,j}$ and $v_{i,j}$ are the same as in the section 3.2.2. We denote the encryption of $\delta_{i,j}^{(\ell)}$ and $\eta_{i,j}^{(\ell)}$ by $\bar{\delta}_{i,j}^{(\ell)}$, $\bar{\eta}_{i,j}^{(\ell)}$, respectively.

Server's Analysis Server's analysis begins with propagating pointer values of the maximal level n by applying the Trans rule as much as possible. In other words, for $i, j = 1, \cdots, m$, the server repeats the following update m times:

$$\bar{\delta}_{i,j}^{(n)} \leftarrow \sum_{k=1}^{m} \bar{\eta}_{i,k}^{(n)} \cdot \bar{\delta}_{k,j}^{(n)}$$

Next, from the level $n-1$ down to 1, the analysis at a level ℓ is carried out in the following steps:

1. applying the Load rule

$$\bar{\eta}_{i,j}^{(\ell)} \leftarrow \bar{\eta}_{i,j}^{(\ell)} + \sum_{k=1}^{m} \bar{u}_{i,k} \cdot \bar{\delta}_{k,j}^{(\ell+1)}$$

2. applying the Store rule

$$\bar{\eta}_{i,j}^{(\ell)} \leftarrow \bar{\eta}_{i,j}^{(\ell)} + \sum_{k=1}^{m} \bar{v}_{j,k} \cdot \bar{\delta}_{k,i}^{(\ell+1)}$$

3. applying the Trans rule: repeating the following update m times

$$\bar{\delta}_{i,j}^{(\ell)} \leftarrow \sum_{k=1}^{m} \bar{\eta}_{i,k}^{(\ell)} \cdot \bar{\delta}_{k,j}^{(\ell)}$$

Through step 1 and 2, edges of the form $x_i \longrightarrow x_j$ are derived where either x_i

or x_j is determined by pointer values of the immediate higher level $\ell + 1$. In step 3, pointer values of a current level ℓ are propagated as much as possible.

We need $O(n \log m)$ multiplicative depth in total because repeating the above 3 steps n times entails maximally m^n homomorphic multiplications on a single ciphertext.

Output Determination The client receives the updated $\{\bar{\delta}_{i,j}^{(\ell)} \mid 1 \leq i, j \leq m, 1 \leq \ell \leq n\}$ from the server and recovers a set of points-to relations as follows:

$$\{\mathtt{x_i} \longrightarrow \&\mathtt{x_j} \mid \mathsf{HE.Dec_{sk}}(\bar{\delta}_{i,j}^{(\ell)}) \neq 0,\ 1 \leq i, j \leq m, \text{ and } 1 \leq \ell \leq n\}$$

3.3.4 Ciphertext Packing

Our use of ciphertext packing aims to decrease total ciphertext size by using fewer ciphertexts than the basic scheme. Thanks to ciphertext packing, a single ciphertext can hold multiple plaintexts rather than a single value. For given a vector of plaintexts $(\mu_1, \cdots, \mu_m)$, the BGV cryptosystem allows to obtain a ciphertext $\bar{\boldsymbol{c}} \leftarrow \mathsf{BGV.Enc}(\mu_1, \cdots, \mu_m)$.

As each ciphertext holds a vector of multiple plaintexts, homomorphic operations between such ciphertexts are performed component-wise. For given ciphetexts $\bar{\mathbf{c}}_1 = \mathsf{BGV.Enc}(\mu_{1,1}, \cdots, \mu_{1,m})$ and $\bar{\mathbf{c}}_2 = \mathsf{BGV.Enc}(\mu_{2,1}, \cdots, \mu_{2,m})$, the homomorphic addition and multiplication in the BGV scheme satisfy the following properties:

$$\mathsf{BGV.Add}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2) \text{ returns a ciphertext } \mathsf{BGV.Enc}(\mu_{1,1} + \mu_{2,1}, \cdots, \mu_{1,m} + \mu_{2,m})$$

$$\mathsf{BGV.Mult}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2) \text{ returns a ciphertext } \mathsf{BGV.Enc}(\mu_{1,1} \cdot \mu_{2,1}, \cdots, \mu_{1,m} \cdot \mu_{2,m})$$

The BGV scheme provides other homomorphic operations such as cyclic rotation. For example, we can perform cyclic rotation of vector by any amount on ciphertexts (*e.g.* $\mathsf{BGV.Enc}(\mu_m, \mu_1, \cdots, \mu_{m-1})$ from $\mathsf{BGV.Enc}(\mu_1, \mu_2, \cdots, \mu_m)$). Using the homomorphic addition, multiplication, and other operations, we

can perform the matrix addition, multiplication and transposition operations on encrypted matrices.

In this subsection, we describe ciphertext packing and the homomorphic matrix operations in more detail.

Principle of Ciphertext Packing We begin with some notations. For an integer q, $\mathbb{Z}_q \stackrel{\mathsf{def}}{=} [-q/2, q/2) \cap \mathbb{Z}$ and $x \bmod q$ denotes a number in $[-q/2, q/2) \cap \mathbb{Z}$ which is equivalent to x modulo q. Recall that the message space of the BGV cryptosystem is $R_p = \mathbb{Z}[X]/(p, \Phi(X))$ for a prime p and an irreducible polynomial $\Phi(X)$. We identify the polynomial ring R_p with $\{a_0 + a_1 X + \cdots + a_{\deg\Phi-1} X^{\deg\Phi-1} \mid a_i \in \mathbb{Z}_p \text{ and } 0 \leq i < \deg\Phi\}$.

In the basic approach, although the message space of the BGV scheme is the polynomial ring R_p, we have used only constant polynomials (*i.e.,* numbers) for plaintexts. Thus, if a vector of plaintexts is represented as a single non-constant polynomial, a single ciphertext can hold multiple plaintexts rather than a single value. Therefore we can save the total memory space by using fewer ciphertexts than the basic scheme.

Suppose the factorization of $\Phi(X)$ modulo p is $\Phi(X) = \prod_{i=1}^{m} F_i(X) \bmod p$ where each F_i is an irreducible polynomial in $\mathbb{Z}_p[X]$. Then a polynomial $\mu(X) \in R_p$ can be viewed as a vector of m different small polynomials, $(\mu_1(X), \cdots, \mu_m(X))$ such that $\mu_i(X) = (\mu(X) \text{ modulo } F_i(X))$ for $i = 1, \cdots, m$.

From this observation, we can encrypt a vector $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_m)$ of plaintexts in $\prod_{i=1}^{m} \mathbb{Z}_p$ into a single ciphertext by the following transitions:

$$\begin{array}{ccccccc} \mathbb{Z}_p \times \cdots \times \mathbb{Z}_p & \longrightarrow & \prod_{i=1}^{m} \mathbb{Z}_p[X]/(F_i(X)) & \longrightarrow & \mathbb{Z}_p[X]/(\Phi(X)) & \longrightarrow & R_q \\ (\mu_1, \cdots, \mu_m) & \stackrel{\mathsf{id}}{\longmapsto} & (\mu_1, \cdots, \mu_m) & \stackrel{\mathrm{CRT}}{\longmapsto} & \mu(X) & \stackrel{\mathsf{BGV.Enc}}{\longmapsto} & \bar{\mathbf{c}} \end{array}$$

First, we view a component μ_i in a vector $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_m)$ as a constant polynomial $\mu_i \in \mathbb{Z}_p[X]/(F_i(X))$ for $i = 1, \cdots, m$. Then, we can compute the unique polynomial $\mu(X) \in R_p$ satisfying $\mu(X) = \mu_i \bmod (p, F_i(X))$ for $i = 1, \cdots, m$ by the Chinese Remainder Theorem (CRT) of polynomials. Finally,

to encrypt a vector $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_m)$ in $\prod_{i=1}^{m} \mathbb{Z}_p$, we encrypt the polynomial $\mu(X) \in R_p$ into a ciphertext $\bar{\mathbf{c}}$ which is denoted by $\mathsf{BGV.Enc}\,(\mu_1, \cdots, \mu_m)$.

Homomorphic Matrix Operations Applying the resolution rules in the level-by-level analysis in the section 3.3.3 can be re-written in a matrix form as shown in Table 3.3. In Table 3.3, $\Delta_\ell = [\delta_{i,j}^{(\ell)}], H_\ell = [\eta_{i,j}^{(\ell)}], U = [u_{i,j}]$, and $V = [v_{i,j}]$ are $m \times m$ integer matrices. Let the i-th row of Δ_ℓ and H_ℓ be $\boldsymbol{\delta}_i^{(\ell)}$ and $\boldsymbol{\eta}_i^{(\ell)}$ respectively. And we denote the encryptions as $\bar{\boldsymbol{\delta}}_i^{(\ell)} = \mathsf{BGV.Enc}(\boldsymbol{\delta}_i^{(\ell)})$ and $\bar{\boldsymbol{\eta}}_i^{(\ell)} = \mathsf{BGV.Enc}(\boldsymbol{\eta}_i^{(\ell)})$.

Rule	Integer form	Matrix form
Trans	$\delta_{i,j}^{(\ell)} \leftarrow \sum_{k=1}^{m} \eta_{i,k}^{(\ell)} \cdot \delta_{k,j}^{(\ell)}$	$\Delta_\ell \leftarrow H_\ell \cdot \Delta_\ell$
Load	$\eta_{i,j}^{(\ell)} \leftarrow \eta_{i,j}^{(\ell)} + \sum_{k=1}^{m} u_{i,k} \cdot \delta_{k,j}^{(\ell+1)}$	$H_\ell \leftarrow H_\ell + U \cdot \Delta_{\ell+1}$
Store	$\eta_{i,j}^{(\ell)} \leftarrow \eta_{i,j}^{(\ell)} + \sum_{k=1}^{m} v_{j,k} \cdot \delta_{k,i}^{(\ell+1)}$	$H_\ell \leftarrow H_\ell + (V \cdot \Delta_{\ell+1})^T$

Table 3.3: Circuit expression of the level-by-level analysis

We follow the methods in [HS14] to perform multiplication between encrypted matrices. We use the Replicate homomorphic operation supported by the BGV scheme [HS14]. For a given ciphertext $\bar{\mathbf{c}} = \mathsf{BGV.Enc}(\mu_1, \cdots, \mu_m)$, the operation $\mathsf{Replicate}(\bar{\mathbf{c}}, i)$ generates a ciphertext $\mathsf{BGV.Enc}(\mu_i, \cdots, \mu_i)$ for $i = 1, \cdots, m$. Using the operation, we can generate an encryption of the i-th row of $(H_\ell \cdot \Delta_\ell)$ as follows:

$$\mathsf{BGV.Mult}\left(\mathsf{Replicate}(\bar{\boldsymbol{\eta}}_i^{(\ell)}, 1), \bar{\boldsymbol{\delta}}_1^{(\ell)}\right) + \cdots + \mathsf{BGV.Mult}\left(\mathsf{Replicate}(\bar{\boldsymbol{\eta}}_i^{(\ell)}, m), \bar{\boldsymbol{\delta}}_m^{(\ell)}\right).$$

Note that this method does not affect the asymptotic notation of the multiplicative depth since the operation Replicate entails only a single multiplication.

To compute a transpose of an encrypted matrix, we use the masking and cyclic rotation techniques described in [HS14]. Algorithms for the homomorphic operations on encrypted matrices are described in Figure 3.3–3.5 at the end of this chapter.

3.3.5 Randomization of Ciphertexts

During the matrix multiplications, components of resulting matrices may become p by coincidence, which is congruent to 0 in $\mathbb{Z}_p$. In this case, incorrect results may happen. We randomize intermediate results to decrease the failure probability.

To multiply the matrices $H_\ell = [\eta_{i,j}^{(\ell)}]$ and $\Delta_\ell = [\delta_{i,j}^{(\ell)}]$, we choose non-zero random elements $\{r_{i,j}\}$ in $\mathbb{Z}_p$ for $i, j = 1, \cdots, m$ and compute $H'_\ell = [r_{i,j} \cdot \eta_{i,j}^{(\ell)}]$. Then, each component of a resulting matrix of the matrix multiplication $(H'_\ell \cdot \Delta_\ell)$ is almost uniformly distributed over $\mathbb{Z}_p$.

Thanks to the randomization, the probability for each component of $H' \cdot \Delta$ of being congruent to zero modulo p is in inverse proportion to p. We may obtain a correct component with the probability of $(1 - \frac{1}{p-1})$. Because we perform in total $n(\lceil \log m \rceil + 3) - 2$ matrix multiplications for the analysis, the probability for a component of being correct is greater than $(1 - \frac{1}{p-1})^{n(\lceil \log m \rceil + 3)}$. For example, in the case where $n = 2, m = 1000$ and $p = 503$, the success probability for a component is about 95%.

Putting up altogether, we present the final protocol in Figure 3.2.

3.4 Experimental Result

In this section, we demonstrate the performance of the pointer analysis in secrecy. In our experiment, we use HElib library [HS14], an implementation of the BGV cryptosystem. We test on 4 small C example programs including tiny Linux packages. The experiment was done on a Linux 3.13 system running on 8 cores of Intel 3.2 GHz box with 24GB of main memory. Our implementation runs in parallel on 8 cores using shared memory.

Table 3.4 shows the result. We set the security parameter 72 which is usually considered large enough. It means a ciphertext can be broken in a worst case time proportional to 2^{72}. In all the programs, the maximum pointer level is 2.

Table 3.4: Experimental Result

Program	# Var	Enc	Propagation	Edge addition	Total
toy	9	26s	28m 49s	5m 58s	35m 13s
buthead-1.0	17	1m 26s	5h 41m 36s	56m 19s	6h 39m 21s
wysihtml-0.13	32	2m 59s	18h 11m 50s	2h 59m 38s	21h 14m 27s
cd-discid-1.1	41	3m 49s	32h 22m 33s	5h 22m 35s	37h 48m 57s

Enc : time for program encryption
Propagation : time for homomorphic applications of the Trans rule
Edge addition : time for homomorphic applications of the Load and Store rules

3.5 Discussions

Why "Basic" Algorithm? Many optimization techniques to scale inclusion-based pointer analysis to larger programs [FFSA98, FS98, HL07, HT01, PKH03] cannot be applied into our setting without exposing much information of the program. Two key optimizations for inclusion-based pointer analysis are the cycle elimination and the difference propagation. But neither method is applicable. The cycle elimination [FFSA98, HL07, HT01, PKH03] aims to prevent redundant computation of transitive closure by collapsing each cycle's components into a single node. The method cannot be applied into our setting because cycles cannot be detected and collapsed as all the program information and intermediate analysis results are encrypted. The other technique, difference propagation [FS98, PKH03], only propagates new reachability facts. Also, we cannot consider the technique because analysis server cannot determine which reachability fact is new as intermediate analysis results are encrypted.

Privacy Preserving App Reviews Our method may be used for app store review systems. App review systems (*e.g.* Apple App Store, Samsung Apps) aim to filter malicious apps before deployments. In app review systems, a server-side analysis in secrecy may help for the following reasons:

- Analysis cannot be performed on the client-side because they may tamper with the analysis results.

- Revealed analysis mechanism may be used to avoid the detection.
- App source codes often require privacy for copyright protection.

A prerequisite for the realization of this scenario is a threshold cryptosystem. In threshold cryptosystems, two parties must cooperate in the decryption protocol. In our setting, the secret key is shared between analysis server and program owner. This decryption mechanism is for preventing the program owner from doing the decryption by himself and tampering with the result. Another prerequisite is a zero-knowledge protocol by which that the program owner did not maliciously change his original program is proved.

Conclusion We report that the homomorphic encryption scheme can unleash the possibility of static analysis of encrypted programs. As a representative example, we have described an inclusion-based pointer analysis in secrecy. In our method, a somewhat homomorphic encryption scheme of depth $O(\log m)$ is able to evaluate the pointer analysis with $O(\log m)$ homomorphic matrix multiplications.

We also show the viability of our work by implementing the pointer analysis in secrecy. We expect our method will scale to larger programs thanks to new developments and advances in HE that are constantly being made.

Main Protocol

Client Input: There m pointer variables in the client's program with the maximal pointer level n. The sets $\{(\delta_{i,j}^{(\ell)}, \eta_{i,j}^{(\ell)}) \mid 1 \leq i,j \leq m, 1 \leq \ell \leq n\}$ and $\{(u_{i,j}, v_{i,j}) \mid 1 \leq i,j \leq m\}$ are initialized by the manner in the section 3.2.2 and 3.3.3. For a security parameter λ, the client generates the parameters $(\mathsf{pk}, \mathsf{evk}; \mathsf{sk}) \leftarrow \mathsf{BGV.KG}(1^\lambda)$ of the BGV scheme.

Sub-algorithms: In this protocol, we use sub-algorithms in Figure 3.3–3.5.

– **Program Encryption** (Client's work)

1. **for** $\ell = 1$ to n and **for** $i = 1$ to m **do**
2. $\quad \bar{\boldsymbol{\delta}}_i^{(\ell)} \leftarrow \mathsf{BGV.Enc}(\delta_{i,1}^{(\ell)}, \cdots, \delta_{i,m}^{(\ell)})$, $\bar{\boldsymbol{\eta}}_i^{(\ell)} \leftarrow \mathsf{BGV.Enc}(\eta_{i,1}^{(\ell)}, \cdots, \eta_{i,m}^{(\ell)})$
3. $\quad \bar{\boldsymbol{u}}_i \leftarrow \mathsf{BGV.Enc}(u_{i,1}, \cdots, u_{i,m})$, $\bar{\boldsymbol{v}}_i \leftarrow \mathsf{BGV.Enc}(v_{i,1}, \cdots, v_{i,m})$
4. **for** $\ell = 1$ to n **do**
5. $\quad \bar{\Delta}_\ell \leftarrow \left\langle \bar{\boldsymbol{\delta}}_1^{(\ell)} | \cdots | \bar{\boldsymbol{\delta}}_m^{(\ell)} \right\rangle^T$, $\bar{H}_\ell \leftarrow \left\langle \bar{\boldsymbol{\eta}}_1^{(\ell)} | \cdots | \bar{\boldsymbol{\eta}}_m^{(\ell)} \right\rangle^T$ // the i-th row of $\bar{\Delta}_\ell$ is $\bar{\boldsymbol{\delta}}_i^{(\ell)}$.
6. $\bar{U} \leftarrow \langle \bar{\mathbf{u}}_1 | \cdots | \bar{\mathbf{u}}_m \rangle^T$, $\bar{V} \leftarrow \langle \bar{\mathbf{v}}_1 | \cdots | \bar{\mathbf{v}}_m \rangle^T$ // the i-th row of $\bar{U}$ is $\bar{\mathbf{u}}_i$.
7. Client sends the sets $\{(\bar{\Delta}_\ell, \bar{H}_\ell) \mid 1 \leq \ell \leq n\}$ and $\{(\bar{U}, \bar{V})\}$ to server.

– **Analysis in Secrecy** (Server's work)

1. $\bar{\Delta}_n \leftarrow \mathsf{HE.MatMult}\left(\mathsf{HE.MatPower}(\bar{H}_n, m), \bar{\Delta}_n\right)$
2. **for** $\ell = n - 1$ to 1 **do**
3. $\quad \bar{A} \leftarrow \mathsf{HE.MatMult}(\bar{U}, \bar{\Delta}_{\ell+1})$, $\bar{B} \leftarrow \mathsf{HE.MatTrans}\left(\mathsf{HE.MatMult}(\bar{V}, \bar{\Delta}_{\ell+1})\right)$
4. $\quad \bar{H}_\ell \leftarrow \mathsf{HE.MatAdd}\left(\mathsf{HE.MatAdd}(\bar{H}_\ell, \bar{A}), \bar{B}\right)$ // apply Load and Store rules
5. $\quad \bar{\Delta}_\ell \leftarrow \mathsf{HE.MatMult}(\mathsf{HE.MatPower}(\bar{H}_\ell, m), \bar{\Delta}_\ell)$ // apply Trans rule
6. Server sends the ciphertext set $\{\bar{\boldsymbol{\delta}}_i^{(\ell)} \mid 1 \leq \ell \leq n, 1 \leq i \leq m\}$ to client.

– **Output Determination** (Client's work)

1. **for** $i = 1$ to m and **for** $\ell = 1$ to n **do**
2. $\quad$ Client computes $(\delta_{i,1}^{(\ell)}, \cdots, \delta_{i,m}^{(\ell)}) \leftarrow \mathsf{BGV.Dec}(\bar{\boldsymbol{\delta}}_i^{(\ell)})$.
3. Client determines the set $\{\mathtt{x_i} \longrightarrow \&\mathtt{x_j} \mid \delta_{i,j}^{(\ell)} \neq 0, 1 \leq i,j \leq m, 1 \leq \ell \leq n\}$.

Figure 3.2: The Pointer Analysis in Secrecy

// We assume that m is the same as the number of plaintext slots in the BGV scheme.
// A prime p is the modulus of message space in the BGV-type cryptosystem.
// We denote the encryption of the matrix $A = [a_{i,j}] \in \mathbb{Z}_p^{m \times m}$ by $\bar{A}$.
// The i-th row $\bar{\mathbf{a}}_i$ of $\bar{A}$ is the ciphertext $\mathsf{BGV.Enc}(a_{i,1}, \cdots, a_{i,m})$ for $i = 1, \cdots, m$.
// For ciphertexts $\bar{\mathbf{c}}_1, \cdots, \bar{\mathbf{c}}_m$, we denote the matrix whose rows are $\bar{\mathbf{c}}_i$ by $\langle \bar{\mathbf{c}}_1 | \cdots | \bar{\mathbf{c}}_m \rangle^T$.

$\underline{\mathsf{HE.MatAdd}}(\bar{A}, \bar{B})$
// **Input**: $\bar{A}, \bar{B}$ are encryptions of $A = [a_{i,j}], B = [b_{i,j}]$.
// **Output**: $\overline{A + B}$ is an encryption of $A + B = [a_{i,j} + b_{i,j}]$.
1 **for** $i = 1$ to m **do**
2 $\quad \bar{\mathbf{z}}_i \leftarrow \mathsf{BGV.Add}(\bar{\mathbf{a}}_i, \bar{\mathbf{b}}_j)$
3 **return** $\bar{Z} \leftarrow \langle \bar{\mathbf{z}}_1 | \bar{\mathbf{z}}_2 | \cdots | \bar{\mathbf{z}}_m \rangle^T$ // the i-th row of $\bar{Z}$ is $\bar{\mathbf{z}}_i$

$\underline{\mathsf{HE.MatMult}}(\bar{A}, \bar{B})$
// **Input**: $\bar{A}, \bar{B}$ are encryptions of $A = [a_{i,j}], B = [b_{i,j}]$.
// **Output**: $\overline{R_A \cdot B}$ is an encryption of $R_A \cdot B = [\sum_{k=1}^{m} r_{i,k} \cdot (a_{i,k} b_{k,j})]$,
// where $r_{i,j} \xleftarrow{\$} [-p/2, p/2) \cap \mathbb{Z}$ with $r_{i,j} \neq 0$.
1 $\bar{R} \leftarrow \mathsf{HE.MatRandomize}(\bar{A})$
2 **for** $i = 1$ to m **do**
3 $\quad \bar{\mathbf{z}}_i \leftarrow \sum_{j=1}^{m} \mathsf{BGV.Mult}\left(\mathsf{HE.Replicate}(\bar{\mathbf{r}}_i, j), \bar{\mathbf{b}}_j\right)$// ciphertext additions
4 **return** $\bar{Z} \leftarrow \langle \bar{\mathbf{z}}_1 | \bar{\mathbf{z}}_2 | \cdots | \bar{\mathbf{z}}_m \rangle^T$ // the i-th row of $\bar{Z}$ is $\bar{\mathbf{z}}_i$

$\underline{\mathsf{HE.MatPower}}(\bar{A}, k)$
// **Input**: $\bar{A}$ is an encryption of A.
// **Output**: $\overline{A^w}$ is an encryption of A^w, where $w = 2^{\lceil \log k \rceil}$.
1 $\bar{Z} \leftarrow \bar{A}$
2 **for** $i = 1$ to $\lceil \log k \rceil$ **do**
3 $\quad \bar{Z} \leftarrow \mathsf{HE.MatrixMult}\left(\bar{Z}, \bar{Z}\right)$
4 **return** $\bar{Z}$

Figure 3.3: Pseudocode for the Homomorphic Matrix Operations I

$\underline{\mathsf{HE.MatTrans}}(\bar{A})$
// **Input**: $\bar{A}$ is an encryption of $A = [a_{i,j}]$.
// **Output**: $\overline{A^T}$ is an encryption of $A^T = [a_{j,i}]$.
1 **for** $i = 1$ to m **do**
2 **for** $j = 1$ to m **do**
3 $\bar{\mathbf{z}}_{i,j} \leftarrow \mathsf{HE.Masking}(\bar{\mathbf{a}}_j, i)$
4 $\bar{\mathbf{z}}_i \leftarrow \sum_{j=1}^{i-1} \mathsf{HE.Rotate}(\bar{\mathbf{z}}_{i,j}, j - i + m) + \sum_{j=i}^{m} \mathsf{HE.Rotate}(\bar{\mathbf{z}}_{i,j}, j - i)$
// $\triangle$ ciphertext additions
5 **return** $\bar{Z} \leftarrow \langle \bar{\mathbf{z}}_1 | \bar{\mathbf{z}}_2 | \cdots | \bar{\mathbf{z}}_m \rangle^T$ // the i-th row of $\bar{Z}$ is $\bar{\mathbf{z}}_i$

$\underline{\mathsf{HE.MatRandomize}}(\bar{A})$
// **Input**: $\bar{A}$ is an encryption of $A = [a_{i,j}]$.
// **Output**: $\overline{R_A}$ is an encryption of $R_A = [r_{i,j} \cdot a_{i,j}]$, where $r_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ with $r_{i,j} \neq 0$.
1 **for** $i = 1$ to m **do**
2 Choose a vector $\mathbf{r}_i = (r_{i,1}, \cdots, r_{i,m}) \xleftarrow{\$} \mathbb{Z}_p^m$ with $r_{i,j} \neq 0 \bmod p$.
3 $\bar{\mathbf{z}}_i \leftarrow \mathsf{BGV.multByConst}(\mathbf{r}_i, \bar{\mathbf{a}}_i)$
4 **return** $\bar{Z} \leftarrow \langle \bar{\mathbf{z}}_1 | \bar{\mathbf{z}}_2 | \cdots | \bar{\mathbf{z}}_m \rangle^T$ // the i-th row of $\bar{Z}$ is $\bar{\mathbf{z}}_i$

Figure 3.4: Pseudocode for the Homomorphic Matrix Operations II

// The following algorithms are in the library HElib.
// Here, we only give preview of the algorithms.

HE.Replicate($\bar{\mathbf{c}}, k$)
// The ciphertext $\bar{\mathbf{c}}$ is the encryption of $(\mu_1, \cdots, \mu_m)$
return the ciphertext BGV.Enc$(\mu_k, \cdots, \mu_k)$

HE.Masking($\bar{\mathbf{c}}, k$)
// The ciphertext $\bar{\mathbf{c}}$ is the encryption of $(\mu_1, \cdots, \mu_m)$
return the ciphertext BGV.Enc$(0, \cdots, 0, \mu_k, 0 \cdots, 0)$ // k-th of plaintext slots is μ_k

HE.Rotate($\bar{\mathbf{c}}, k$)
// The ciphertext $\bar{\mathbf{c}}$ is the encryption of $(\mu_1, \cdots, \mu_m)$
// This operation is the right rotation as a linear array
return the ciphertext BGV.Enc$(\mu_{m-k+2}, \cdots, \mu_m, \mu_1, \cdots, \mu_{m-k+1})$

BGV.multByConst($\mathbf{r}, \mathbf{c}$)
// The operation of the multiply-by-constant induces "moderate" noise-growth,
// while a multiplication of ciphertexts induces "expensive" noise-growth.
// The constant vector $\mathbf{r} = (r_1, \cdots, r_m) \in \mathbb{Z}_p \times \cdots \times \mathbb{Z}_p$
// The ciphertext $\bar{\mathbf{c}}$ is the encryption of $(\mu_1, \cdots, \mu_m)$
return the ciphertext BGV.Enc$(r_1\mu_1, \cdots, r_m\mu_m)$

Figure 3.5: Pseudocode for Some Homomorphic Algorithms

Chapter 4

New Fully Homomorphic Encryption

Homomorphic encryption enables computing certain functions over encrypted data. This property would be very useful in applications like secure cloud computing. The first fully homomorphic encryption (FHE) scheme which can compute arbitrary functions was proposed by Gentry [Gen09] based on ideal lattices. After that, many follow-up works appeared and the performances of FHE schemes are improved dramatically, whereas only few base problems are known. To date, there are three main families of the FHE schemes:

1. Gentry's scheme [Gen09] based on ideal lattices and early follow-up works [SV10, GH11]. Ideal coset problem is used to prove the semantic security.

2. van Dijk et al.'s scheme (DGHV scheme) [vDGHV10] based on the approximate common divisor (ACD) problem introduced by Howgrave-Graham in [HG01]. It's efficiency is much improved, implemented [CMNT11, CNT12], and batch variants are proposed [CCK$^+$13].

3. Brakerski and Vaikuntanathan's schemes based on the (Ring) Learning with Errors ((R)LWE) problem [BV11a, BV11b]. Follow-up works include [BGV12] and the NTRU-variant [LATV12].

In this chapter, we propose a FHE scheme whose security is based on a new hard problem called polynomial Approximate Common Divisor (polyACD) problem, which is a polynomial analogue of the ACD problem. The symmetric version of our scheme is as follows: Let $R = \mathbb{Z}_N[x]/(p(x))$ be polynomial ring where N is hard-to-factor integer and $p(x)$ is a $(d+1)$-degree polynomial in $\mathbb{Z}_N[x]$ with linear factors. One of the roots of $p(x)$ is a secret key, and the ciphertext is a polynomial of degree d in R whose evaluation at the secret key is a small integer. Note that the message space of our basic scheme is an integer ring $\mathbb{Z}_Q$ with $Q < N$. More precisely, encryption and decryption with secret key α is as follows:

> $\mathbf{Enc}_\alpha(m)$: Choose a random noise integer e with $|e| \ll N$ and a random polynomial $q(x)$ of degree $\leq d-1$ in $\mathbb{Z}_N[x]$. Output $c(x) = q(x)(x-\alpha) + eQ + m \bmod N$.
>
> $\mathbf{Dec}_\alpha(c(x))$: Output $m' = ((c(\alpha) \bmod N) \bmod Q)$.

This conceptually simple scheme is a natural analogue of the DGHV scheme. Similar to the batch DGHV scheme [CCK+13], our scheme can be extended to encrypt multiple messages using other roots of $p(x)$. The resulting scheme can support not only SIMD style operations but also large integer arithmetic. Using multi-point evaluation of a ciphertext polynomial, decryption of multiple messages can be done efficiently. Unlike other homomorphic schemes [BV11b, BGV12, LATV12] based on polynomial rings, our scheme needs only single polynomial $p(x)$ for homomorphic evaluations. Using $p(x)$, it is easy to see that we can homomorphically add and multiply ciphertexts in the ring R in a natural way. For this reason, the multiplication timing of ciphertexts of our scheme is faster than RLWE-based homomorphic encryption using polynomial rings, YASHE by Bos et al.[BLLN13].

Since our base problem is rather new, we extensively cryptanalyze the polyACD problem by applying all known attacks such as Chen-Nguyen's attack, Coppersmith's attack, and distinguishing attack. Through this analysis, we get more confidence on the hardness of the base problem.

Our scheme is yet another homomorphic encryption using *polynomials*. It is very simple and gives useful polynomial analogue of van Dijk et al.'s scheme. As Brakerski writes in [Bra13], it is important to diversify and base FHE on different assumptions. This study enriches FHE and we believe that it is helpful to the future development of homomorphic encryption.

Comparison to other FHEs It is interesting to compare the proposed scheme with other FHE schemes. Unlike the RLWE-based FHE [BLLN13], our scheme performs simple and efficient homomorphic multiplications. Moreover, our scheme can use different message spaces in each slot, which is similar to the ACD-based FHEs [CCK$^+$13, CLT14]. Our scheme naturally supports both SIMD style operations and large integer arithmetic. Compared to the ACD-based leveled-FHEs [CNT12, CLT14], our scheme does not use the subset sum of the secret key since division with a secret key is not required.

4.1 Preliminaries

Notations. Let $a \leftarrow A$ denote choosing a uniform random element a from a set A. When $\mathcal{D}$ is a probability distribution, the notation $a \leftarrow \mathcal{D}$ refers to choosing a random element a according to the distribution $\mathcal{D}$. For arbitrary numbers x and y, we denote by $[x, y]$ the set of integers contained within the interval and use $a \leftarrow I$ to denote randomly choosing an integer a from the interval I.

We adopt the convention that $\mathbb{Z}_N$ is a ring of integers modulo N. Given $x \in \mathbb{Z}$, let $x \bmod N$ or $[x]_N$ denote the unique number in $\mathbb{Z} \cap \left(-\frac{N}{2}, \frac{N}{2}\right]$, which is congruent to x modulo N. Lowercase bold letters are used for vectors whereas uppercase letters are used for matrices. The transpose of a vector $\mathbf{v}$ or matrix A is denoted by $\mathbf{v}^T$ or A^T, respectively.

For an integer d, we define the set of polynomials in $\mathbb{Z}_N[x]$ of degree less than or equal to d as $\mathbb{Z}_N^d[x] := \{f(x) \in \mathbb{Z}_N[x] \mid \deg f(x) \leq d\}$. We use $\mathcal{U}_d(N)$ to represent a uniform distribution over $\mathbb{Z}_N^d[x]$. For a polynomial $f(x) =$

$\sum_{i=0}^{\deg f} f_i x^i$ in $\mathbb{Z}[x]$, let $f(x) \bmod N$ denote the polynomial $\sum_{i=0}^{\deg f} [f_i]_N \, x^i$ in $\mathbb{Z}_N[x]$. We say $p(x) \in \mathbb{Z}_N[x]$ *splits completely* if there exist $\alpha_1, \cdots, \alpha_d \in \mathbb{Z}$ such that

$$p(x) = \prod_{i=1}^{d} (x - \alpha_i) \bmod N$$

and $\gcd(\alpha_i - \alpha_j, N = 1)$ for $1 \leq i \neq j \leq d$. We denote by $S_{(N,d)}$ the set of such polynomials.

4.1.1 Lattices

A *lattice* Λ is a discrete additive subgroup of $\mathbb{R}^m$. For a linearly independent set $\{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ in $\mathbb{R}^m$ $(m \geq n)$, the lattice Λ generated by $\{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ is the set of integer linear combinations of $\mathbf{b}_i$,

$$\Lambda = \mathcal{L}(B) := \mathbb{Z}^n \cdot B = \left\{ \sum_{i=1}^{n} l_i \mathbf{b}_i \;\middle|\; l_i \in \mathbb{Z} \right\},$$

where B is an $n \times m$ matrix whose ith row is $\mathbf{b}_i$. The set of vectors $\{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ is called a *lattice basis*, and B is called a *basis matrix* of a lattice Λ. The *rank* and *dimension* of a lattice Λ is n and m, respectively. If $n = m$, then Λ is said to be a *full rank* lattice. The *determinant* (or *volume*) of a lattice Λ is given by $\sqrt{\det(BB^T)}$.

For an integer $q \geq 2$, a *q-ary lattice* Λ of dimension n is a lattice in $\mathbb{Z}^n$ that contains $q\mathbb{Z}^n$ as a sublattice. For example, if B is an integer matrix in $\mathbb{Z}^{n \times m}$, the following is a q-ary lattice of dimension n:

$$\Lambda_q^{\perp}(B) = \{\mathbf{y} \in \mathbb{Z}^n \mid \mathbf{y} \cdot B \equiv \mathbf{0} \pmod{q}\}.$$

Note that a vector $\mathbf{x} \in \mathbb{Z}^n$ is in the q-ary lattice Λ if and only if $\mathbf{x} \bmod q$ is also in Λ.

4.1.2 Chinese Remaindering for Polynomials over Composite Modulus

Let N be a composite modulus and $m(x)$, $n(x)$ be nonzero polynomials in $\mathbb{Z}[x]$. We say $m(x)$ and $n(x)$ are *relatively prime modulo* N if there exist polynomials $\bar{m}(x)$ and $\bar{n}(x)$ in $\mathbb{Z}[x]$ such that $m\bar{m} + n\bar{n} \equiv 1 \pmod{N}$.

For $i = 1, \cdots, k$ and $j = 1, \cdots, l$, let m_i and n_j be polynomials in $\mathbb{Z}[x]$ which are pairwise relatively prime modulo N. Then, for arbitrary polynomials $f_1, \cdots, f_k, g_1, \cdots, g_l$ in $\mathbb{Z}[x]$, there exists a polynomial h in $\mathbb{Z}[x]$ of degree less than $\deg\left(\prod_i m_i \cdot \prod_j n_j\right)$ such that

$$h \equiv f_i \pmod{(m_i, N)} \text{ for } 1 \le i \le k \text{ and } h \equiv g_j \pmod{(n_j, N)} \text{ for } 1 \le j \le l,$$

by the Chinese remainder theorem for commutative rings [Lan02]. Furthermore, h is uniquely determined up to congruence modulo N. Now, let us introduce the following notation to simplify h instead of writing in words:

$$h := \mathsf{polyCRT}_{N,\left((m_i)_{i=1}^k,(n_j)_{j=1}^l\right)}\left((f_i)_{i=1}^k, (g_j)_{j=1}^l\right)$$

whose coefficients belong to $\left(-\frac{N}{2}, \frac{N}{2}\right]$. If m_i and n_j are polynomials of degree one, then $\mathsf{polyCRT}$ is identical to the polynomial interpolation that takes values f_i and g_j at the roots of m_i and n_j, respectively, for all i and j.

Throughout the paper, the modulus N in $\mathsf{polyCRT}$ may be omitted when there is no confusion.

4.1.3 Distributions

Given two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, and a probabilistic polynomial time distinguisher $\mathcal{A}$, the advantage of $\mathcal{A}$ is defined by

$$Adv(\mathcal{A}) = \left|\Pr[\mathcal{A}(\mathcal{D}_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}_2) = 1]\right|,$$

where $\mathcal{A}(\mathcal{D}_i)$ is given by $g_i \leftarrow \mathcal{D}_i$ for $i = 1, 2$. Note that a function f is *negligible* if for every positive polynomial $p(\cdot)$, there exists some $N \in \mathbb{N}$ such that $f(n) < 1/|p(n)|$ for all $n > N$.

If $Adv(\mathcal{A})$ is negligible for any polynomial time distinguisher $\mathcal{A}$, then we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable and write $\mathcal{D}_1 \approx_c \mathcal{D}_2$. When the statistical distance of $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible, we say distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically close and write $\mathcal{D}_1 \approx_s \mathcal{D}_2$.

Let ρ, γ, and d be integers. For a γ-bit integer N and $\alpha, \alpha_1, \cdots, \alpha_l \leftarrow \mathbb{Z}_N$ with $\gcd(\alpha_i - \alpha_j, N) = 1$ for $1 \le i \ne j \le l$, we define the following distributions:

$$\mathcal{D}_{\rho,d}(N;\alpha) := \left\{ \begin{array}{l} \mathsf{Choose}\ q(x) \leftarrow \mathbb{Z}_N^{d-1}[x], e \leftarrow (-2^\rho, 2^\rho) : \\ \mathsf{Output}\ f(x) = (x-\alpha)q(x) + e \bmod N \end{array} \right\}$$

$$\mathcal{D}_{\rho,d}(N;(\alpha_i)_{i=1}^l) := \left\{ \begin{array}{c} \mathsf{Choose}\ e_i \leftarrow (-2^\rho, 2^\rho), i = 1, \cdots, l,\ q(x) \leftarrow \mathbb{Z}_N^{d-1}[x] : \\ \mathsf{Output}\ f(x) = q(x)\prod_{i=1}^l (x-\alpha_i) + \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}\left((e_i)_{i=1}^l\right) \end{array} \right\}.$$

4.2 Our Fully Homomorphic Encryption Scheme

We propose a new homomorphic encryption scheme over the polynomial ring $\mathbb{Z}_N[x]$ for a hard-to-factor integer N. Let $p(x)$ be a $S_{(N,d+1)}$-polynomial for a positive integer d. Our homomorphic encryption scheme is based on the polynomial operations in ring $\mathbb{Z}[x]/(p(x))$. The message space of the basic scheme is $\mathbb{Z}_Q$ for an integer $Q \ge 2$. This can be extended to $\mathbb{Z}_{Q_1} \times \cdots \times \mathbb{Z}_{Q_l}$ for integers $Q_i \ge 2$ using other roots of $p(x)$. After describing its basic parameters, we present a somewhat homomorphic encryption scheme in Section 4.2.2. The leveled FHE scheme is briefly described in Section 4.2.3.

4.2.1 Basic Parameters

The parameters of the proposed scheme are as follows:

λ: the security parameter

ρ: the bit length of the errors
γ: the bit length of the composite modulus N
τ: the number of encryptions of zero in the public key
μ: the bit length of linear combination coefficients during encryption
ℓ_Q: the bit length of the message

The parameter of our scheme should satisfy the following constraints:

– $\gamma = \Omega(\lambda^3)$, to resist integer factoring algorithms.
– $d = \Theta(\gamma)$, to avoid a distinguishing attack (see Section 4.4.1).
– $\rho \geq \max\{2\lambda - 2\log d - 2\log\gamma,\ \gamma/d + \lambda\}$, to avoid Chen-Nguyen's attack [CN12] (see Section 4.4.2) and Coppersmith's attack [Cop96] (see Section 4.4.3).
– $\mu\tau > d\gamma + 2\lambda$, to apply the leftover hash lemma (see Section 4.3.2). Note that μ should be less than the bit length of the smallest factor of N.

4.2.2 The Somewhat Homomorphic Encryption Scheme

We describe our somewhat homomorphic encryption scheme.

- $\mathsf{KeyGen}(\lambda)$: Choose a γ-bit hard-to-factor integer N and a polynomial $p(x) = (x-\alpha)\prod_{j=1}^{d}(x-\beta_j) \bmod N$ from $S_{(N,d+1)}$. Choose ℓ_Q-bit integer Q such that $\gcd(N, Q) = 1$. Choose polynomials

$$f_k(x) \ = \ \mathsf{polyCRT}_{\left(x-\alpha,(x-\beta_j)_{j=1}^{d}\right)}\left(Qe_k, (r_{kj})_{j=1}^{d}\right) \quad \text{for } 1 \leq k \leq \tau,$$

  where $e_k \leftarrow (-2^\rho, 2^\rho)$ and $r_{kj} \leftarrow \mathbb{Z}_N$. Output the public parameter $\mathsf{pp} = (N, Q)$, the public key $\mathsf{pk} = (\{f_k(x)\}_{k=1}^{\tau})$, the evaluation key $\mathsf{evk} = p(x)$, and the secret key $\mathsf{sk} = \alpha$.

- $\mathsf{Enc}_{\mathsf{pk}}(m)$: To encrypt a message $m \in \mathbb{Z}_Q$, choose a random vector $\mathbf{b} = (b_1, \cdots, b_\tau) \in [0, 2^\mu)^\tau$ and output $c(x) = m + \sum_{i=1}^{\tau} b_i f_i(x) \bmod N$.

- $\mathsf{Dec}_{\mathsf{sk}}(c(x))$: Output $m' = [c(\alpha)]_N \bmod Q$.

- $\mathsf{Add}(c_1(x), c_2(x))$: Output $c_{\text{add}}(x) = (c_1(x) + c_2(x) \bmod N)$.
- $\mathsf{Mult}_{\mathsf{evk}}(c_1(x), c_2(x))$: Output $c_{\text{mult}}(x) = (c_1(x) \times c_2(x) \bmod (N, p(x)))$.

Correctness For a ciphertext $c(x) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$ with $m \in \mathbb{Z}_Q$, we know $c(x)$ is of the form $c(x) = (x-\alpha)q(x) + Qe + m$ for some polynomial $q(x)$ in $\mathbb{Z}_N^{d-1}[x]$ and integer e with $|Qe + m| < \tau 2^{\ell_Q+\mu+\rho+1}$. Hence, $[c(\alpha)]_N \bmod Q = m$, if $\tau 2^{\ell_Q+\mu+\rho+1} < N/2$.

For ciphertexts $c_1(x) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1)$ and $c_2(x) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_2)$, the following equations hold:

$$\begin{aligned} c_1(\alpha) + c_2(\alpha) &\equiv (m_1 + m_2) + Q(e_1 + e_2) \pmod N, \\ c_1(\alpha) \times c_2(\alpha) &\equiv m_1 \cdot m_2 + Q(m_1 e_2 + m_2 e_1 + Q e_1 e_2) \pmod N. \end{aligned}$$

for some noise integers e_1 and e_2. Based on these properties,

$$\mathsf{Dec}_{\mathsf{sk}}(c_1 + c_2) = m_1 + m_2 \text{ and } \mathsf{Dec}_{\mathsf{sk}}(c_1 \times c_2) = m_1 \cdot m_2,$$

if the absolute value of $Q(m_1 e_2 + m_2 e_1 + Q e_1 e_2)$ is less than $N/2$. In general, decryption works correctly after evaluating a polynomial of degree up to $D = \lfloor \frac{\gamma - 1}{\ell_Q + \rho + \mu + \log \tau + 1} \rfloor$.

Remark 4.2.1. One may consider the polyACD problem without an exact multiple of $x - \alpha$. In that case, it is not necessary to use a hard-to-factor integer as a modulus. The multiplication process needs to be modified similar to [vDGHV10], adding the set of polynomials of degree from $d+1$ to $2d$, which are noisy polynomial multiples of $x - \alpha$, to the evaluation key. To control the noise increase, $d\lceil \log N/\mu \rceil$ polynomials are needed. Moreover, the modulus should be large enough to provide a reasonable number of multiplications. So, the modulus in this case is not necessarily smaller than a hard-to-factor integer. For example, the modulus should be larger than 1000-bit for degree-10 evaluation. Thus, there is no clear benefit to remove $p(x)$ in the definition of the polyACD problem.

Remark 4.2.2. For a batch variant, we choose a polynomial $p(x) = \prod_{i=1}^{l}(x-\alpha_i)\prod_{j=1}^{d}(x-\beta_j)$ from $S_{(N,l+d)}$, and ℓ_Q-bit integers $Q_1, \cdots, Q_l$ with $\gcd(N, Q_i) = 1$ for $i = 1, \cdots, l$. Furthermore, choose the following polynomials:

$$\begin{aligned} f_k(x) &= \mathsf{polyCRT}_{N,((x-\alpha_i)_{i=1}^{l},(x-\beta_j)_{j=1}^{d})}((Q_i e_{ki})_{i=1}^{l}, (r_{kj})_{j=1}^{d}), \text{for } 1 \le k \le \tau, \\ h_k(x) &= \mathsf{polyCRT}_{N,((x-\alpha_i)_{i=1}^{l},(x-\beta_j)_{j=1}^{d})}((Q_i e'_{ki} + \delta_{ki})_{i=1}^{l}, (r'_{kj})_{j=1}^{d}), \text{for } 1 \le k \le l, \end{aligned}$$

where $e_{ki}, e'_{ki} \leftarrow (-2^{\rho}, 2^{\rho})$ and $r_{kj}, r'_{kj} \leftarrow \mathbb{Z}_N$. The public key consists of these polynomials: $\mathsf{pk} = \big(\{f_k(x)\}_{k=1}^{\tau}, \{h_k(x)\}_{k=1}^{l}\big)$.

4.2.3 Leveled Fully Homomorphic Encryption Scheme

In this subsection, we propose a leveled FHE by applying the scale-invariant technique in [Bra12, CLT14]. A leveled FHE is defined as follows:

Definition 4.2.1 (Leveled Fully Homomorphic Encryption [BGV12])**.** We say that a family of homomorphic encryption schemes $\{\mathcal{E}^{(L)} : L \in \mathbb{Z}^+\}$ is leveled fully homomorphic if, for all $L \in \mathbb{Z}^+$, all encryption schemes use the same decryption circuit. $\mathcal{E}^{(L)}$ compactly evaluates all circuits of depth at most L, which use some specified complete set of gates. The computational complexity of $\mathcal{E}^{(L)}$'s algorithms is polynomial (the same polynomial for all L) in the security parameter, L and (in the case of the evaluation algorithm) the size of the circuit.

Let N be a hard-to-factor integer and $p(x)$ be a polynomial in $S_{(N^2,l+d)}$ where l and d are positive integers. The ciphertext space of our leveled FHE is a polynomial ring $\mathbb{Z}_{N^2}[x]/(p(x))$ and the message space space is $\prod_{i=1}^{l} Q_i$. When Q_i's are pairwise coprime, the message space can be considered as $\mathbb{Z}_Q$ where $Q = \prod_{i=1}^{l} Q_i$. Hence our scheme can support not only SIMD style operations, but also large integers arithmetic.

- $\mathsf{LHE.KeyGen}(\lambda)$: Choose a $\left(\frac{\gamma}{2}\right)$-bit hard-to-factor integer N and a polynomial $p(x) = \prod_{i=1}^{l}(x-\alpha_i)\prod_{j=1}^{d}(x-\beta_j)$ from $S_{(N^2,l+d)}$. Choose ℓ_Q-bit integers $Q_1, \cdots, Q_l$ with $\gcd(N, Q_i) = 1$ for $i = 1, \cdots, l$. Furthermore,

choose the following polynomials: for $1 \le k \le \tau$ and $1 \le t \le l$,

$$\begin{aligned} f_k(x) &= \mathsf{polyCRT}_{N^2,\left((x-\alpha_i)_{i=1}^l,(x-\beta_j)_{j=1}^d\right)}\left((e_{ki})_{i=1}^l,(r_{kj})_{j=1}^d\right), \\ h_t(x) &= \mathsf{polyCRT}_{N^2,\left((x-\alpha_i)_{i=1}^l,(x-\beta_j)_{j=1}^d\right)}\left((e'_{ti}+\delta_{ti}\cdot\lfloor\tfrac{N}{Q_i}\rfloor)_{i=1}^l,(r'_{tj})_{j=1}^d\right), \end{aligned}$$

where $e_{ki}, e'_{ti} \leftarrow (-2^\rho, 2^\rho), r_{kj}, r'_{tj} \leftarrow \mathbb{Z}_{N^2}$, and δ_{ti} is the Kronecker delta.

For each $\nu = 0, \cdots, \lfloor 2\log N \rfloor$, $k = 0, \cdots, d+l-1$, choose integers $\tilde{e}_{\nu,k,i} \leftarrow (-2^\rho, 2^\rho)$ for $i = 1, \cdots, l$, $\tilde{r}_{\nu,k,j} \leftarrow \mathbb{Z}_{N^2}$ for $j = 1, \cdots, d$, and compute polynomials

$$\tilde{f}_{\nu,k}(x) = \mathsf{polyCRT}_{N^2,((x-\alpha_i)_{i=1}^l,(x-\beta_j)_{j=1}^d)}\left((\tilde{e}_{\nu,k,i}+\left\lfloor\tfrac{Q_i}{N}[2^\nu\alpha_i^k]_{N^2}\right\rceil)_{i=1}^l,(\tilde{r}_{\nu,k,j})_{j=1}^d\right).$$

Output the the secret key $\mathsf{sk} = (\{\alpha_i\}_{i=1}^l)$, the public parameter $\mathsf{pp} = \left(N, \{Q_i\}_{i=1}^l\right)$, the public key $\mathsf{pk} = \left(\{f_k(x)\}_{k=1}^\tau, \{h_t(x)\}_{t=1}^l\right)$, and the evaluation key $\mathsf{evk} = \left(p(x), \{\tilde{f}_{\nu,k}\}_{\nu\in[0,\lceil 2\log N\rceil),k\in[0,l+d)}\right)$.

- $\mathsf{LHE.Enc}_{\mathsf{pk}}(\mathbf{m})$: To encrypt a message vector $\mathbf{m} = (m_1, \cdots, m_l) \in \prod_{i=1}^l \mathbb{Z}_{Q_i}$, choose a random vector $\mathbf{b} = (b_1, \cdots, b_\tau) \in [0, 2^\mu)^\tau$ and output $c(x) = \sum_{i=1}^l m_i h_i(x) + \sum_{i=1}^\tau b_i f_i(x) \bmod N^2$.
- $\mathsf{LHE.Dec}_{\mathsf{sk}}(c(x))$: Compute $m'_i = \left(\left\lfloor [c(\alpha_i)]_{N^2}\frac{Q_i}{N}\right\rceil \bmod Q_i\right)$ for $i = 1, \cdots, l$ and output $\mathbf{m}' = (m'_1, \cdots, m'_l)$.
- $\mathsf{LHE.Add}(c_1(x), c_2(x))$: Output $c_{\text{add}}(x) = (c_1(x) + c_2(x) \bmod N^2)$.
- $\mathsf{LHE.Mult}_{\mathsf{evk}}(c_1(x), c_2(x))$: To homomorphically multiply ciphertexts $c_1(x)$ and $c_2(x)$, compute the polynomial $\tilde{c}_{\text{mult}}(x) := c_1(x) \times c_2(x)$ in $\mathbb{Z}_{N^2}[x]/(p(x))$. Write $\tilde{c}_{\text{mult}}(x) = \sum_{\nu=0}^{\lfloor 2\log N\rfloor} 2^\nu \tilde{c}_\nu(x)$ where all coefficients of $\tilde{c}_\nu(x)$ are numbers in $\{0, 1\}$. Output $c_{\text{mult}}(x) = \sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1} \tilde{c}_{\nu,k} \cdot \tilde{f}_{\nu,k}(x)$ in $\mathbb{Z}_{N^2}[x]/(p(x))$ where $\tilde{c}_{\nu,k}$ is the coefficient of x^k in $\tilde{c}_\nu(x)$.

One can write the polynomial $\tilde{c}_{\text{mult}}(x)$ in 2^w-base representation so that $\tilde{c}_{\text{mult}}(x) = \sum_{\nu=0}^{\lfloor 2\log N\rfloor} 2^{w\nu}\tilde{c}_\nu(x)$ where all coefficients of $\tilde{c}_\nu(x)$ are numbers in $(-2^{w-1}, 2^{w-1}]$ instead of $\{0, 1\}$. In this case, $\tilde{f}_{\nu,k}(x)$ is modified to be the encryption of $\lfloor\frac{Q_i}{N}[2^{w\nu}\alpha_i^k]_{N^2}\rceil$.

Note that the above scheme has features similar to the variant of van Dijk et al.'s scheme by Coron et al [CLT14]. However, the leveled FHE scheme presented by Coron et al. needs an additional assumption, the so called subset sum assumption, to provide efficient homomorphic divisions of the secret key. On the other hand, our scheme does not need the hardness of subset sum assumption since the division with a secret key is not required in the decryption process.

Correctness For a ciphertext $c(x) \leftarrow \mathsf{LHE.Enc}_{\mathsf{pk}}(\mathbf{m})$ with $\mathbf{m} = (m_1, \cdots, m_l) \in \mathbb{Z}_{Q_1} \times \cdots \times \mathbb{Z}_{Q_l}$, it is in the form of

$$c(\alpha_i) = q_i \cdot N^2 + (m_i + Q_i^2 e_i^*) \lfloor \tfrac{N}{Q_i} \rfloor + e_i,$$

for some integers q_i, noises $e_i^* \in (-2^{\rho^*}, 2^{\rho^*})$, and $e_i \in (-2^{\rho'}, 2^{\rho'})$ with $\rho' \leq \rho + \mu + \log \tau$ for all $i = 1, \cdots, l$. Therefore, $\left\lfloor [c(\alpha_i)]_{N^2} \cdot \frac{Q_i}{N} \right\rceil \bmod Q_i = m_i$ for $i = 1, \cdots, l$ if $|e_i^*|, |e_i| < N/2Q_i$. For the ciphertexts $c_1(x) \leftarrow \mathsf{LHE.Enc}_{\mathsf{pk}}(\mathbf{m_1})$ and $c_2(x) \leftarrow \mathsf{LHE.Enc}_{\mathsf{pk}}(\mathbf{m_2})$, we know that

$$c_1(\alpha_i) + c_2(\alpha_i) = q_{\text{add},i} \cdot N^2 + (m_{1,i} + m_{2,i} + Q_i^2 e_{\text{add},i}^*) \lfloor \tfrac{N}{Q_i} \rfloor + e_{\text{add},i},$$

for some integers $q_{\text{add},i}, e_{\text{add},i}^*$, and $e_{\text{add},i}$ with $|e_{\text{add},i}^*| < 2^{\rho^*+1}$ and $|e_{\text{add},i}| < 2^{\rho'+1}$ where $\mathbf{m}_1 = (m_{1,1}, \cdots, m_{1,l})$ and $\mathbf{m}_2 = (m_{2,1}, \cdots, m_{2,l})$. Next, to perform homomorphic multiplication, we compute $\tilde{c}_{\text{mult}}(x) = c_1(x) \cdot c_2(x)$ in $\mathbb{Z}_{N^2}[x] / (p(x))$. Then, we have

$$\tilde{c}_{\text{mult}}(\alpha_i) = c_1(\alpha_i) \cdot c_2(\alpha_i) = q_{\text{mult},i} \cdot N^2 + (m_{1,i} m_{2,i}) \lfloor \tfrac{N}{Q_i} \rfloor^2 + e_{\text{mult},i},$$

for some integers $q_{\text{mult},i}$ and $e_{\text{mult},i}$ with $|e_{\text{mult},i}| < Q_i N 2^{\rho'+\rho^*+2}$, assuming $2 \log Q_i + \rho^* < \rho'$.

The following lemma gives the homomorphic property of our scheme.

Lemma 4.2.1. *Let* $\mathbf{m}_1 = (m_{1,1}, \cdots, m_{1,l})$ *and* $\mathbf{m}_2 = (m_{2,1}, \cdots, m_{2,l})$ *be message vectors and let* $c_1(x)$ *and* $c_2(x)$ *be encryptions of* $\mathbf{m}_1$ *and* $\mathbf{m}_2$ *respectively.*

Define $c_{\mathrm{mult}}(x) \leftarrow \mathsf{LHE.Mult}_{\mathsf{evk}}(c_1(x), c_2(x))$. *Then, for* $i = 1, \cdots, l$,

$$[c_{\mathrm{mult}}(\alpha_i)]_{N^2} = (m_i + Q_i^2 e^*_{\mathrm{mult},i})\lfloor \tfrac{N}{Q_i} \rfloor + e_{\mathrm{mult},i}$$

for some integers $e^*_{\mathrm{mult},i}$ *and* $e_{\mathrm{mult},i}$ *with* $|e^*_{\mathrm{mult},i}| < (d+l)\lceil 2\log N \rceil$ *and* $|e_{\mathrm{mult},i}| < Q_i^2(d+l)\lceil 2\log N\rceil 2^{\rho'+3}$, *where* $m_i = m_{1,i}m_{2,i}$.

Proof. Let $\tilde{c}(x) = c_1(x) \times c_2(x)$ in $\mathbb{Z}_{N^2}[x]/\left(p(x)\right)$ and rewrite $\tilde{c}(x)$ such that $\tilde{c}(x) = \sum_{\nu=0}^{\lfloor 2\log N\rfloor} 2^\nu \tilde{c}_\nu(x)$ where all coefficients of $\tilde{c}_\nu(x)$ are numbers in $\{0,1\}$. For $k = 0, \cdots, d+l-1$, let $\tilde{c}_{\nu,k}$ is the k-th coefficient of $\tilde{c}_\nu(x)$. Then, for each $i = 1, \cdots, l$,

$$\begin{aligned}
c_{\mathrm{mult}}(\alpha_i) &\equiv_{N^2} \sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1} \tilde{c}_{\nu,k}\cdot \tilde{f}_{\nu,k}(\alpha_i) \\
&\equiv_{N^2} \sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1} \tilde{c}_{\nu,k}\left(\tilde{e}_{\nu,k,i} + \left\lfloor \tfrac{Q_i}{N}[2^\nu \alpha_i^k]_{N^2}\right\rceil\right) \\
&\equiv_{N^2} \sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1} \tilde{c}_{\nu,k}\left(\tfrac{Q_i}{N}[2^\nu \alpha_i^k]_{N^2}\right) + \underbrace{\sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1} \tilde{c}_{\nu,k}\left(\tilde{e}_{\nu,k,i} + \tilde{w}_{\nu,k,i}\right)}_{:=\delta_{1,i}},
\end{aligned}$$

where $\tilde{w}_{\nu,k,i} = \left\lfloor \tfrac{Q_i}{N}[2^\nu\alpha_i^k]_{N^2}\right\rceil - \tfrac{Q_i}{N}[2^\nu\alpha_i^k]_{N^2}$ with $|\tilde{w}_{\nu,k,i}| \le 1/2$. Moreover,

$$\begin{aligned}
\sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1} \tilde{c}_{\nu,k}\left(\tfrac{Q_i}{N}[2^\nu\alpha_i^k]_{N^2}\right) &= \tfrac{Q_i}{N}\sum_{\nu=0}^{\lfloor 2\log N\rfloor}\sum_{k=0}^{d+l-1}\left(\tilde{c}_{\nu,k}2^\nu\alpha_i^k + q_{\nu,k,i}N^2\right) \\
&= \tfrac{Q_i}{N}\left([\tilde{c}(\alpha_i)]_{N^2} + e^*_{\mathrm{mult},i}N^2\right) \\
&= \tfrac{Q_i}{N}\left(m_{1,i}m_{2,i}\lfloor\tfrac{N}{Q_i}\rfloor^2 + e_{\mathrm{mult},i} + e^*_{\mathrm{mult},i}N^2\right) \\
&= (m_{\mathrm{mult},i} + Q_i^2 e^*_{\mathrm{mult},i})\lfloor\tfrac{N}{Q_i}\rfloor + \tfrac{Q_i}{N}e_{\mathrm{mult},i} + \delta_{2,i},
\end{aligned}$$

for some $q_{\nu,k,i} \in \mathbb{Z}$, $|e^*_{\mathrm{mult},i}| < (d+l)\lceil 2\log N\rceil$ and $|\delta_{2,i}| < 2Q_i(d+l)\lceil 2\log N\rceil$. Therefore,

$$[c_{\mathrm{mult}}(\alpha_i)]_{N^2} = (m_{\mathrm{mult},i} + Q_i^2 e^*_{\mathrm{mult},i})\lfloor\tfrac{N}{Q_i}\rfloor + \delta_{3,i} + \delta_{2,i} + \delta_{1,i}$$

where $\delta_{3,i} = \tfrac{Q_i}{N}e_{\mathrm{mult},i}$ and $\delta_{3,i} + \delta_{2,i} + \delta_{1,i} \in \mathbb{Z}$ with $|\delta_{3,i} + \delta_{2,i} + \delta_{1,i}| <$

$Q_i^2(d+l)\lceil 2\log N\rceil 2^{\rho'+3}$. □

When evaluated at the secret key, the fresh ciphertexts have noise about ρ'-bit for $\rho' < \rho+\mu+\log\tau$. After LHE.Mult of two fresh ciphertexts, the noise of the resulting ciphertext increase $8\Theta Q_i^2$ times, where $\Theta = (d+l)\lceil 2\log N\rceil$. If all Q_i's are the same, the value of $\tilde{c}_{\text{mult}}$ in LHE.Mult should be $\tilde{c}_{\text{mult}}(x) = Q \cdot c_1(x) \times c_2(x)$ and $\tilde{f}_{\nu,k}$ in LHE.KeyGen is modified to encrypt $\left\lfloor \frac{1}{N}[2^\nu \alpha_i^k]_{N^2} \right\rceil$ instead of $\left\lfloor \frac{Q_i}{N}[2^\nu \alpha_i^k]_{N^2} \right\rceil$. In this case, the noise after the multiplication increases only $8\Theta Q$ times. The following theorem states that our scheme is a leveled FHE scheme.

Theorem 4.2.1. *The proposed homomorphic encryption scheme with parameters (ρ', N, d) can correctly evaluate circuits of multiplicative depth L with*

$$N/2^{\rho'} \geq (O(d\log N))^{L+O(1)}.$$

Therefore, the proposed scheme is an L-leveled FHE.

If $Q_i = 2$ for all i, our scheme can be transformed into an FHE scheme, which can compute arbitrary functions on encrypted data, using Gentry's bootstrapping technique [Gen09] assuming circular security.

Note that the decryption process can be accelerated by the multi-point evaluation method [VZGG13]. For example, the decryption process that applies fast, multi-point evaluation takes 1.84 seconds while the original basic decryption takes 119.608 seconds when $\gamma = 1024$, $d = 17833$, and $l = 2048$. This test was conducted on an ordinary laptop with an Intel Core i7 processor running at 1.7 GHz with 4 GB RAM.

4.3 Security

We prove the security of our scheme. The security of our scheme is based on the hardness of a polynomial ACD problem, which is an analogue of the ACD problem [HG01] on which the DGHV scheme [vDGHV10] and its variants [CMNT11, CNT12, CCK+13] are based.

In ACD problem, only the approximate multiple of p are given in the form of $x_i = q_i \cdot p + e_i$, where e_i's are small noise integers. Viewing p as a polynomial $x - \alpha$ for a secret $\alpha \in \mathbb{Z}_N$, and q_i as a random polynomial in some polynomial ring $\mathbb{Z}_N[x]$, we get the following problem: Given many approximate polynomial multiples of $x - \alpha$, which is of the form $f_i(x) = q_i(x)(x - \alpha) + e_i \bmod N$ where "noise" integers e_i are uniformly sampled from $[-B, B] \cap \mathbb{Z}$ with B is much "smaller" than N, find α. If $f_i(x) = q_i(x)(x - \alpha) + e_i$ over $\mathbb{Z}$, then the problem can be reduced to the ACD problem. However, this is not the case; efficient reduction between the ACD problem and the polynomial ACD problem does not seem to exist since they equal modulo N.

Note that some parts of this section were collaborated with results from Hansol Ryu and Moon Sung Lee [CHLR14]. Hansol Ryu played a key role in providing the relations of the polynomial ACD problem and its variants. Moon Sung Lee played a important role in proving the semantic security of our scheme based on the hardness of the polynomial ACD problem.

4.3.1 The Polynomial ACD Problems

Definition 4.3.1 (**Computational Polynomial Approximate Common Divisor Problem: polyACD).** The (ρ, γ, d)-polyACD problem is defined as follows: Given a γ-bit integer N and $\alpha \leftarrow \mathbb{Z}_N$, and a polynomial $p(x) = (x - \alpha)\tilde{p}(x)$, where $\tilde{p}(x)$ is randomly chosen from $S_{(N,d)}$ such that $\gcd(\tilde{p}(\alpha), N) = 1$ and polynomially many samples from $\mathcal{D}_{\rho,d}(N; \alpha)$, find α.

Definition 4.3.2 (**Decisional Polynomial Approximate Common Divisor Problem: DpolyACD).** The (ρ, γ, d)-DpolyACD problem is defined as follows: Given a γ-bit integer N and $\alpha \leftarrow \mathbb{Z}_N$, $g(x) = f(x) + r(x) \cdot b \bmod N$ where $f(x) \leftarrow \mathcal{D}_{\rho,d}(N; \alpha)$, $r(x) \leftarrow \mathbb{Z}_N^d[x]$ and $b \leftarrow \{0, 1\}$, and a polynomial $p(x) = (x - \alpha)\tilde{p}(x)$, where $\tilde{p}(x)$ is randomly chosen from $S_{(N,d)}$ such that $\gcd(\tilde{p}(\alpha), N) = 1$ and polynomially many samples from $\mathcal{D}_{\rho,d}(N; \alpha)$, determine b.

Since $p(x)$ is also given, which is an exact polynomial multiple of $x - \alpha$, the polynomial factorization of $p(x)$ should be difficult. For this reason, we use a polynomial ring $\mathbb{Z}_N[x]$ where N is a hard-to-factor integer. Applying the method in [CLT14], we can show that polyACD problem is reducible to the DpolyACD problem. Therefore, we obtain the equivalence between the polyACD problem and DpolyACD problem.

To proof the security of our leveled FHE scheme, we define the extended polyACD problem.

Definition 4.3.3 (**l-Decisional Polynomial Approximate Common Divisor Problem: l-DpolyACD).** The (ρ, γ, d)-l-DpolyACD problem is as follows: For a γ-bit integer N, let $p(x) = \prod_{i=1}^{l}(x - \alpha_i)\prod_{j=1}^{d}(x - \beta_j)$ be a polynomial from $S_{(N,l+d)}$. Given $p(x)$, polynomially many samples from $\mathcal{D}_{\rho,d}(N; (\alpha_i)_{i=1}^{l})$, and $g(x) = f(x)+r(x)\cdot b \bmod N$, where $f(x) \leftarrow \mathcal{D}_{\rho,d}(N; (\alpha_i)_{i=1}^{l})$, $r(x) \leftarrow \mathbb{Z}_N^{l+d-1}[x]$ and $b \leftarrow \{0, 1\}$, determine b.

Intuitively, the l-DpolyACD problem seems to be easier than the DpolyACD problem since sampled polynomials have more specific structure. The following lemma show that the l-DpolyACD problem is not easier than the DpolyACD problem. We provide the proof of the following theorem in [CHLR14].

Lemma 4.3.1. *The (ρ, γ, d)-DpolyACD problem is reducible to the (ρ, γ, d)-l-DpolyACD problem.*

4.3.2 Security Proof

To prove the security of the proposed scheme, we recall the definition of lossy encryption schemes [BHY09, HLOV11]. As is noted in [HLOV11], semantic security is implied by the properties of lossy encryption schemes. To prove that our encryption scheme is lossy, we recall the leftover hash lemma.

Leftover Hash Lemma A family $\mathcal{H}$ of hash functions from finite sets X to Y is said to be 2-universal if for all distinct $x, x' \in X$, $\Pr_{h\leftarrow\mathcal{H}}[h(x) =$

$h(x')] = 1/|Y|$. A distribution $\mathcal{D}$ is ϵ-uniform if its statistical distance from the uniform distribution is at most ϵ. We now state the simplified leftover hash lemma in [vDGHV10].

Lemma 4.3.2 (Simplified Leftover Hash Lemma)**.** *Let $\mathcal{H}$ be a family of 2-universal hash functions from X to Y. Suppose $h \leftarrow \mathcal{H}$ and $x \leftarrow X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $\frac{1}{2}\sqrt{|Y|/|X|}$-uniform over $\mathcal{H} \times Y$.*

This lemma is applied to our construction.

Lemma 4.3.3. *Let N be an integer whose smallest prime factor is p, and let μ be a positive integer such that 2^μ is less than p. Pick $\mathbf{x}_1, \cdots, \mathbf{x}_\tau \leftarrow \mathbb{Z}_N^{d+1}$ uniformly and independently, choose $\mathbf{s} = (s_1, \ldots, s_\tau) \leftarrow [0, 2^\mu)^\tau$, and set $\mathbf{x}_{\tau+1} = \sum_{i=1}^{\tau} s_i \mathbf{x}_i \bmod N$. Then, $(\mathbf{x}_1, \cdots, \mathbf{x}_\tau, \mathbf{x}_{\tau+1})$ is $\frac{1}{2}\sqrt{N^{d+1}/2^{\mu\tau}}$-uniform over $\mathbb{Z}_N^{(d+1)(\tau+1)}$.*

Proof. Define a family of hash functions $\mathcal{H}$ from $[0, 2^\mu)^\tau$ to $\mathbb{Z}_N^{d+1}$ as follows: A member $h \in \mathcal{H}$ is defined by elements $\mathbf{h}_1, \cdots, \mathbf{h}_\tau \in \mathbb{Z}_N^{d+1}$ such that for any $\mathbf{s} \in [0, 2^\mu)^\tau$, $h(\mathbf{s}) = \sum_{i=1}^{\tau} s_i \mathbf{h}_i \bmod N$. For any distinct $\mathbf{s}$ and $\mathbf{s}'$, there exists i such that $s_i \neq s_i'$ with $\gcd(s_i - s_i', N) = 1$; this is based on the fact that $s_i - s_i'$ is less than any prime factor of N. This family is 2-universal since

$$\begin{aligned}
\Pr_{h \leftarrow \mathcal{H}}[h(\mathbf{s}) = h(\mathbf{s}')] &= \Pr_{h \leftarrow \mathcal{H}}[h(\mathbf{s}) - h(\mathbf{s}') = \mathbf{0}] \\
&= \Pr_{h \leftarrow \mathcal{H}}\left[\sum_{i=1}^{\tau} (s_i - s_i')\mathbf{h}_i \bmod N = \mathbf{0}\right] \\
&= 1/N^{d+1}.
\end{aligned}$$

Therefore, by the leftover hash lemma, $(h, h(x))$ is $\frac{1}{2}\epsilon$-uniform over $\mathbb{Z}_N^{(d+1)(\tau+1)}$ where $\epsilon = \sqrt{N^{d+1}/2^{\mu\tau}}$. And, it is negligible when $\mu\tau > (d+1)\log N + 2\lambda$. □

Lossy Encryption In the lossy encryption scheme [HLOV11], a key generation algorithm outputs two kinds of keys, injective keys and lossy keys based on the input parameter and the following properties*. First, under

*We do not consider *openablility* since it is implied by the other properties.

injective keys, traditional correctness is satisfied. Second, injective public keys pk_{inj} and lossy public keys pk_{lossy} are computationally indistinguishable. Third, under lossy public keys, the distributions of any two messages are statistically close.

Using properties of lossy encryption, it is easy to show that a lossy encryption scheme is semantically secure under the chosen plaintext attack [HLOV11]. Namely, for messages m_0 and m_1 of the lossy encryption scheme, the second and third properties of lossy encryptions imply

$$\mathcal{E}(\mathsf{pk}_{inj}, m_0) \approx_c \mathcal{E}(\mathsf{pk}_{lossy}, m_0) \approx_s \mathcal{E}(\mathsf{pk}_{lossy}, m_1) \approx_c \mathcal{E}(\mathsf{pk}_{inj}, m_1),$$

where $\mathcal{E}(\mathsf{pk}, m)$ is the distribution of encryptions of m under the public key pk.

We are now ready to prove the semantic security of our scheme based on the hardness of the polynomial ACD problem.

Theorem 4.3.1. *Our scheme is semantically secure based on the hardness of the polynomial ACD problem.*

Proof. Since polynomial ACD problem is reducible to DpolyACD problem, it is sufficient to prove security based on the hardness of the DpolyACD problem. We first view our basic scheme in Section 4.2.2 as a lossy encryption scheme. Injective keys are generated by KeyGen step. Correctness under the injective keys is easily verified. For lossy keys, the key generation algorithm is identical to the one used for injective keys with the exception that $f_k(x)$ are chosen uniformly and randomly from $\mathcal{U}_d(N)$ rather than $\mathcal{D}_{\rho,d}(N;\alpha)$. Using the DpolyACD assumption and a standard hybrid argument, it is easy to see that lossy keys and injective keys are computationally indistinguishable. Lastly, by the leftover hash lemma under lossy keys, when $\mu\tau > (d+1)\gamma+2\lambda$, encryptions of m_0 and m_1 are statistically close for any two messages m_0 and m_1.

By using the l-DpolyACD assumption instead of the DpolyACD assump-

tion, the semantic security of leveled FHE scheme in Section 4.2.3 can be proved similarly. From the equivalence in [CHLR14], the DpolyACD problem is reducible to the l-DpolyACD problem and so the polyACD problem is also reducible to the l-DpolyACD problem. Thus, our scheme is semantically secure under the hardness of the polynomial ACD problem. □

4.4 Analysis of the Polynomial ACD Problems

In this section we provide an extensive cryptanalysis of the polyACD problem and consider all known attacks applicable to the polyACD problem. We first consider a distinguishing attack based on lattices using the similarities to the LWE problem. Next, we apply attacks against the ACD problem to the polyACD problem in Sections 4.4.2 and 4.4.3. In Section 4.4.4, we present a method for finding common solutions of multivariate polynomials over composite modulus when solutions are small. We combine Cohn and Heninger's analysis [CH13] with ours to determine secure parameters for the polyACD assumption.

Section 4.4.1 is an analysis of the DpolyACD problem and the remaining subsections pertain to the polyACD problem.

4.4.1 Distinguishing Attack

Due to its structural similarity, a distinguishing attack on the LWE problem [MR09] can be applied to the polyACD problem. This attack is described as follows.

Let $f_i(x) = \sum_{j=0}^{d} f_{i,j}x^j$ be samples from $\mathcal{D}_{\rho,d}(N;\alpha)$ with $f_i(\alpha) \bmod N = e_i$, where $e_i \in (-2^\rho, 2^\rho)$ for $1 \le i \le m$. Given such samples and $g(x) = \sum_{i=0}^{d} g_i x^i = f(x) + r(x) \cdot b$ where $f(x) \leftarrow \mathcal{D}_{\rho,d}(N;\alpha)$ and $r(x) \leftarrow \mathbb{Z}_N^d[x]$ with $g(\alpha) \equiv e_0 + r(\alpha) \cdot b \pmod N$, our goal is to distinguish whether $b = 0$ or $b = 1$; that is, whether $g(x)$ is from $\mathcal{D}_{\rho,d}(N;\alpha)$ or not.

Let $\mathbf{f}_i = (f_{i,j})_{1\le j\le d}$, $\mathbf{g} = (g_j)_{1\le j\le d}$, $\mathbf{s} = (\alpha^j \bmod N)_{1\le j\le d}$, and $\mathbf{v} = (v_i)_{1\le i\le m+1} = (f_{1,0}, \cdots, f_{m,0}, g_0)$. The inner product of $\mathbf{f}_i$ and $\mathbf{s}$ is given by

$$\langle \mathbf{f}_i, \mathbf{s}\rangle = \sum_{j=1}^{d} f_{i,j}s_j = \sum_{j=1}^{d} f_{i,j}(\alpha^j \bmod N) \equiv f_i(\alpha) - f_{i,0} \equiv e_i - v_i \pmod N,$$

for $i = 1, \cdots, m$. Let A be the $(m+1)\times d$ matrix whose rows are $\mathbf{f}_i$ and $\mathbf{g}$. By the previous calculation, $\mathbf{e}^T - A\mathbf{s}^T \equiv \mathbf{v}^T \pmod N$ where $\mathbf{e} = (e_1, \cdots, e_m, e_0 + r(\alpha)b)$.

Consider the following lattice Λ,

$$\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+1} \mid \mathbf{x}A \equiv \mathbf{0} \pmod N\}.$$

If $\mathbf{w} \in \Lambda$ is a vector of norm smaller than $\frac{\epsilon}{2}N/(2^\rho\sqrt{m+1})$ where $0 < \epsilon < 1$, then

$$\langle \mathbf{w}, \mathbf{v}\rangle \bmod N = \langle \mathbf{w}, \mathbf{e}\rangle \bmod N \le \|\mathbf{w}\|\|\mathbf{e}\| < \frac{\epsilon}{2}N/(2^\rho\sqrt{m+1})\|\mathbf{e}\|.$$

When $b = 0$, this value is smaller than $\epsilon\frac{N}{2}$; hence, we can distinguish whether $b = 0$ or not with high probability.

To thwart this attack, we set the parameters such that finding short vectors of norm less than $N/(2^\rho\sqrt{m+1})$ in Λ is difficult. Since A is a random matrix modulo M, we follow the estimates in [MR09]. The length of the shortest vector that can be obtained is close to

$$\min\{N, (\det(\Lambda))^{\frac{1}{m+1}} \cdot \delta^{m+1}\} = \min\{N, N^{\frac{d}{m+1}}\delta^{m+1}\},$$

where δ is a root-Hermite factor that depends on lattice reduction algorithms. The value $N^{\frac{d}{m+1}}\delta^{m+1}$ is minimized whenever $m+1 = \sqrt{d\log N/\log\delta}$. By using polynomial time lattice reduction algorithms, the shortest vector one can find in Λ is at least $\min\{N, 2^{2\sqrt{d\log N\log\delta}}\}$; thus, we arrive at the following condition:

$$N/(2^{\rho}\sqrt{m+1}) \ll 2^{2\sqrt{d \log N \log \delta}}.$$

Therefore, if

$$d > \frac{\log N}{4 \log \delta},$$

the distinguishing attack is not effective. We refer to [LP11, LN14] for the minimal achievable value of δ.

4.4.2 Chen-Nguyen's Attack

The basic attack against the polyACD problem involves a simple exhaustive search. Specifically, for every possible noise e, check whether $\gcd(p(x), f(x) - e)$ yields a non-trivial factor of $p(x)$. More efficient method was proposed by Chen and Nguyen [CN12] for the integer ACD problem. Their algorithm has $\tilde{O}(2^{\rho/2})$ time complexity, which is essentially "square root" of that of the exhaustive search. In this subsection, we adapt their algorithm to the polyACD problem.

Let $f(x) = (x - \alpha)q(x) + e$ be a polynomial from $\mathcal{D}_{\rho,d}(N; \alpha)$ and $p(x) = (x - \alpha)\tilde{p}(x)$ in $S_{(N,d+1)}$. For simplicity, we only consider $e \in [0, 2^{\rho})$ rather than $(-2^{\rho}, 2^{\rho})$. With high probability, we have

$$x - \alpha = \gcd\left(p(x), \prod_{i=0}^{2^{\rho}-1} (f(x) - i) \bmod p(x)\right).$$

To compute this polynomial efficiently, we first construct a bivariate polynomial $F_{2^{\rho'}}(x, t) = \prod_{i=0}^{2^{\rho'}-1}(f(x) - (t+i)) \bmod p(x)$, where $\rho' = \lfloor \frac{\rho}{2} \rfloor$ and evaluate the polynomial at $2^{\rho' + (\rho \bmod 2)}$ points using the multi-point evaluation algo-

rithm. We observe that

$$\begin{aligned}\prod_{i=0}^{2^{\rho}-1}(f(x)-i) &\equiv \prod_{k=0}^{2^{\rho'+(\rho \bmod 2)}-1}\left(\prod_{i=0}^{2^{\rho}-1}\left(f(x)-(2^{\rho'}k+i)\right)\right)\\ &\equiv \prod_{k=0}^{2^{\rho'+(\rho \bmod 2)}-1} F_{2^{\rho'}}(x,2^{\rho'}k) \pmod{p(x)}.\end{aligned}$$

Note that polynomials in $\mathbb{Z}_N[x,t]$, of degree less than d_1 in x and less than d_2 in t can be multiplied using $\tilde{O}(d_1 d_2)$ operations in $\mathbb{Z}_N$ [VZGG13]. Thus, we can construct a bivariate polynomial $F_{2^{\rho'}}(x,t)$ in $\tilde{O}(2^{\rho'}d)$ multiplications in $\mathbb{Z}_N$ using the product tree method. On the other hand, dividing a polynomial over $\mathbb{Z}_N$ of degree d_1 by a polynomial over $\mathbb{Z}_N$ of degree d_2 has complexity $\tilde{O}(d_1)$ when using Newton's method. Overall, using multipoint evaluation, we can evaluate the polynomial at $2 \times 2^{\rho'+(\rho \bmod 2)}$ points in $\tilde{O}(2^{\rho'}d)$ multiplications in $\mathbb{Z}_N$. Therefore, the total complexity of this method is $\tilde{O}(2^{\rho/2}d)$ operations in $\mathbb{Z}_N$.

4.4.3 Coppersmith's Attack

Finding roots of a polynomial over $\mathbb{Z}_N$ is difficult since the factorization of N is unknown. However, if the polynomial has small roots, Coppersmith's algorithm [Cop96] gives such solutions. Let $p(x)$ be a polynomial in $S_{(N,d+1)}$ for integers N, d, and let $f(x)$ be polynomial in $\mathcal{D}_{\rho,d}(N;\alpha)$ for a root α of $p(x)$. Since the error is quite smaller than the modulus in the polyACD problem, we may apply Coppersmith's algorithm if we know a polynomial which has $f(\alpha)$ as a root.

We first recall Coppersmith's result.

Lemma 4.4.1. (Coppersmith method [Cop96]) *Let N be a positive integer and let $f(x)$ be a monic polynomial in $\mathbb{Z}_N[x]$ of degree d. Then one can find all solutions x_0 to the equation*

$$f(x_0) \equiv 0 \pmod{N} \text{ with } |x_0| < N^{\frac{1}{d}},$$

in time $O\left(d^5 \log^9 N\right)$.

To apply this lemma, we first prove the following theorem, which is related to the construction of a polynomial that has $f(\alpha)$ as a root.

Theorem 4.4.1. *Given an instance* $(p(x), f(x))$ *of the* (ρ, γ, d)*-polyACD problem, one can find* $g(x)$ *in* $\mathbb{Z}_N[x]$ *of degree* $\leq d+1$ *such that* $g\left(f(\alpha)\right) \equiv 0 \pmod N$ *in* $O(d^3)$ *operations in* $\mathbb{Z}_N$ *where* α *is a root of* $p(x)$.

Proof. Let $f_i(x) = (f(x))^i \bmod p(x)$ be a polynomial in $\mathbb{Z}_N[x]$ for $i = 1, \cdots, d+1$. Since $p(\alpha) = 0$, $f_i(\alpha)$ is equal to $f(\alpha)^i \bmod N$. Define bivariate polynomials $F_i(x, y) = y^i - f_i(x) \in \mathbb{Z}_N[x, y]$ for $i = 1, \cdots, d+1$. Using the Gaussian elimination, one can find $a_1, \cdots, a_{d+1} \in \mathbb{Z}_N$ such that $\sum_{i=1}^{d+1} a_i f_i(x)$ is a constant and $a_1, \cdots, a_{d+1} \in \mathbb{Z}_N$ are not all zero. Hence, $\sum_{i=1}^{d+1} a_i f_i(x) = \sum_{i=1}^{d+1} a_i f_i(\alpha)$.

By defining $g(y) = \sum_{i=1}^{d+1} a_i F_i(x, y)$, we can observe that

$$g(y) = \sum_{i=1}^{d+1} a_i (y^i - f_i(x)) = \sum_{i=1}^{d+1} a_i (y^i - f_i(\alpha)),$$

$$g(f(\alpha)) = \sum_{i=1}^{d+1} a_i (f(\alpha)^i - f_i(\alpha)) \equiv 0 \pmod N.$$

Hence $g(y)$ is a univariate polynomial of degree $(d+1)$ and has $f(\alpha)$ as a root modulo N.

Computing $f_i(x)$ for $1 \leq i \leq d+1$ requires $O\left(d^3\right)$ operations in $\mathbb{Z}_N$, and the cost of Gaussian elimination is $O(d^3)$ operations in $\mathbb{Z}_N$. Thus, the total complexity is $O(d^3)$ operations in $\mathbb{Z}_N$. □

Combining with the Coppersmith method, one can compute $f(\alpha)$ in time $O(d^5 \log^9 N)$ if $|f(\alpha)| < N^{\frac{1}{d+1}}$. Therefore, if we choose

$$\rho \geq \frac{1}{d+1} \log N + \lambda,$$

then the time complexity of this attack is at least 2^λ.

4.4.4 Extension of Cohn-Heninger's Attack

In [CH13], Cohn and Heninger give an analysis of the polynomial approximate common divisor problem over fields. It can be seen as an extension of Section 4.4.3 to the multi-instance case. Since we use a polynomial ring over $\mathbb{Z}_M$, which is not a field, their analysis can not be applied directly. However, their strategy can be used to obtain several multivariate polynomials which have a common root. These polynomials are used to analyze the polyACD problem.

Let $f_1(x), \cdots, f_m(x)$ be polynomials from $\mathcal{D}_{\rho,d}(N;\alpha)$ and let $p(x)$ be the corresponding polynomial in $S_{(N,d+1)}$. Similar to Section 4.4.3, we first construct m variable polynomials $Q(z_1, \cdots, z_m)$ such that $Q(f_1(\alpha), \cdots, f_m(\alpha)) = 0 \bmod N$ and consider the bound of roots one can find in polynomial time. We define a lattice Λ generated by coefficient (in $\mathbb{Z}_N[x]$) vectors of the polynomials

$$(z_1 - f_1(x))^{i_1} \cdots (z_m - f_m(x))^{i_m} p(x)^{\max(s-\sum_j i_j, 0)}$$

with $i_1 + \cdots + i_m \leq t$ for proper parameters t and s. The dimension of the lattice is $\binom{t+m}{m}$ and $\deg_x(\det \Lambda) = (d+1)\binom{s+m}{m}\frac{s}{m+1}$. By the polynomial analogue of lattice basis reduction [Kai80], we obtain a reduced basis $v_1, \cdots, v_n$ with corresponding polynomials $Q_1, \cdots, Q_n$ for Λ such that

$$\deg_x(Q_1) + \cdots + \deg_x(Q_n) = \deg_x(\det \Lambda)$$

and $\deg_x(Q_1) \leq \cdots \leq \deg_x(Q_n)$, where $n = \dim \Lambda = \binom{t+m}{m}$. As a result,

$$\deg_x(Q_i) \leq \frac{\deg_x \det(\Lambda)}{n - (i-1)}.$$

By the construction of $Q_i(z_1, \cdots, z_m)$, $Q_i(f_1(\alpha), \cdots, f_m(\alpha))$ is divided by $(x-\alpha)^s$. Hence we choose

$$\frac{\deg_x \det(\Lambda)}{\dim(\Lambda) - (k-1)} < s \tag{4.4.1}$$

so that $Q_i(f_1(\alpha), \cdots, f_m(\alpha)) \equiv 0 \bmod N$ for $i = 1, \cdots, k$ and $k \leq m$. Therefore, we have $Q_1(z_1, \cdots, z_m), \cdots, Q_k(z_1, \cdots, z_m)$ in $\mathbb{Z}_N[z_1, \cdots, z_m]$ of total degree t such that

$$Q_1(f_1(\alpha), \cdots, f_m(\alpha)) \equiv \cdots \equiv Q_k(f_1(\alpha), \cdots, f_m(\alpha)) \equiv 0 \bmod N,$$

for proper parameters t, s, and k with $k \leq m$.

In general, finding solutions of modular multivariate polynomials is difficult if factorizations of the modulus are unknown. However, such problems may be solved in polynomial time if the solutions are quite small. In [JM06], Jochemsz and May describe a strategy for finding small modular roots of a multivariate polynomial. They use the Coppersmith technique to obtain the bound of small roots when m-variable polynomial of total degree t is given. That is,

$$X_1 \cdots X_m < N^{\frac{1}{t}(1+o(1))},$$

where X_i is the bound of the root of the ith variable.

As an extension, we now present a method for finding small solutions of multivariate polynomials over $\mathbb{Z}_M$ with multiple instances using the lattice-based Coppersmith technique. This technique achieves better bounds since we used two or more polynomials whereas only single polynomial is used in [JM06]. Combining the following theorem with the results of Cohn-Heninger's, we examine the hardness of the polyACD problem.

Theorem 4.4.2. *Let N be a positive integer and $Q_1, \cdots, Q_k$ be m variable polynomials over $\mathbb{Z}_N$ of total degree t such that*

$$Q_1(e_1, \cdots, e_m) \equiv \cdots \equiv Q_k(e_1, \cdots, e_m) \equiv 0 \bmod N, \tag{4.4.2}$$

for some $e_1, \cdots, e_m \in \mathbb{Z}$ and $k \leq m$. Moreover, let bounds $X_i \in \mathbb{N}$ with $|e_i| \leq X_i$ for $i = 1, \cdots, m$. Then nontrivial polynomials $R(x_1, \cdots, x_m)$ can be found with $R(e_1, \cdots, e_m) = 0$ for all tuples $(e_1, \cdots, e_m) \in \mathbb{Z}^m$ satisfying

(4.4.2) and $|e_i| \leq X_i$ *in polynomial time in* $\log M$*,* m*, and* t*, provided that*

$$X_1 \cdots X_m < \frac{1}{2}\, N^{\frac{k}{t} - \frac{m+1}{4th \log N}\binom{th+m}{m}}$$

for some integer $h \geq m$*.*

Proof. After several polynomial operations, we obtain polynomials Q_i such that Q_i has no monomial x_j^t for $j < i$. We assume that the coefficients of x_i^t of $Q_i(x_1, \cdots, x_m)$ are one since we can easily multiply $Q_i(x_1, \cdots, x_m)$ by the inverse of the coefficients of x_i^t. If the inverse does not exist for some i, we factorize N.

Let $h \geq m$ be an integer. Consider the collection of polynomials that have a common root modulo N^h:

$$x_1^{i_1} \cdots x_m^{i_m} Q_1^{\delta_1}(x_1, \cdots, x_m) \cdots Q_k^{\delta_k}(x_1, \cdots, x_m) N^{(h - \sum_j \delta_j)},$$

where $0 \leq i_1 + \cdots + i_m + t(\delta_1 + \cdots + \delta_k) \leq th,\ 0 \leq \delta_1 + \cdots + \delta_k \leq h$, and $0 \leq i_1, \cdots, i_k < t$. Define the lattice Λ that is spanned by the coefficient vectors of the above polynomials at $(x_1 X_1, \cdots, x_m X_m)$. The dimension of Λ is

$$n = \dim \Lambda = \binom{th+m}{m}.$$

Since Λ can be represented by a lower triangular basis matrix, the volume of Λ is simply the product of all entries on the diagonal. Specifically, $\det \Lambda = (X_1 \cdots X_m)^{s_X} N^{s_N}$ where

$$\begin{aligned}
s_X &= \sum_{\substack{j_1 + \cdots + j_m \leq th \\ 0 \leq j_1, \cdots, j_m \leq th}} j_1 = \binom{th+m}{m+1} = n \cdot \frac{th}{m+1}, \\
s_N &= \sum_{\substack{\delta_1 + \cdots + \delta_k \leq h \\ 0 \leq \delta_1, \cdots, \delta_m \leq h}} \ \sum_{\substack{0 \leq i_1 + \cdots + i_m \leq t(h - (\delta_1 + \cdots + \delta_k)) \\ 0 \leq i_1, \cdots, i_k < t}} (h - (\delta_1 + \cdots + \delta_k)) \\
&\approx h\binom{th+m}{m} - \frac{k}{t}\binom{th+m}{m+1} = n\left(h - \frac{kh}{m+1}\right).
\end{aligned}$$

Running the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm on this basis results in an LLL-reduced basis with the m shortest vectors $v_1, \cdots, v_m$ satisfying

$$|v_1| \leq \cdots \leq |v_m| \leq 2^{\dim \Lambda/4} (\det \Lambda)^{1/(\dim \Lambda + 1 - m)}.$$

Therefore the corresponding polynomials $R_i(x_1, \cdots, x_m)$ satisfy

$$|R_i(e_1, \cdots, e_m)| \leq \sqrt{\dim \Lambda}\ 2^{\dim \Lambda/4} (\det \Lambda)^{1/(\dim \Lambda + 1 - m)},$$

for $i = 1, \cdots, m$, by the Cauchy-Schwarz inequality. It remains to show that

$$\sqrt{\dim \Lambda}\ 2^{\dim \Lambda/4} (\det \Lambda)^{1/(\dim \Lambda + 1 - m)} < N^h \tag{4.4.3}$$

in order for $R_i(e_1, \cdots, e_m) = 0$ over $\mathbb{Z}$ for $i = 1, \cdots, m$. The following is a condition on the bound of $X_1, \cdots, X_m$:

$$X_1 \cdots X_m \leq n^{-\frac{(n+1-k)(m+1)}{2nth}}\ 2^{-\frac{(m+1)(n+1-m)}{4th}}\ N^{\frac{k}{t} - \frac{m^2-1}{nt}}.$$

Note that $n^{-\frac{(n+1-m)(m+1)}{2nth}} \geq 2^{-\frac{m+1}{2th}\log n} \geq 2^{-1}$ for $h \geq m$ and $2^{-\frac{(m+1)(n+1-m)}{4th}} \geq 2^{-\frac{(m+1)n}{4th}}$. Therefore, if

$$X_1 \cdots X_m \leq \frac{1}{2}\ N^{\frac{k}{t} - \frac{m+1}{4th \log N}\binom{th+m}{m}},$$

inequality (4.4.3) holds. □

For a sufficiently large modulus N, we apply Theorem 4.4.2 to the poly-ACD problem. Parameter h is set according to the following conditions:

$$h \leq \begin{cases} \left(\frac{4km! \log N}{t^{m+1}(m+1)}\right)^{1/(m-1)}, & \text{if } t \geq m \\ \left(\frac{4km! \log N}{t^m (m+1)}\right)^{1/(m)}, & \text{if } t < m. \end{cases}$$

With this in hand, we can find all tuples $(e_1, \cdots, e_m) \in \mathbb{Z}^m$ satisfying

$$Q_1(e_1, \cdots, e_m) \equiv \cdots \equiv Q_k(e_1, \cdots, e_m) \equiv 0 \bmod N \text{ with } |e_i| \leq X_i$$

and

$$X_1 \cdots X_m < N^{\frac{k}{t}(1+o(1))},$$

as long as Q_i's are algebraically independent.

Recall that it is possible to obtain m variable polynomials $Q_1, \cdots, Q_k$ over $\mathbb{Z}_N$ of total degree t when $(d+1)\binom{s+m}{m}\frac{1}{m+1} < \binom{t+m}{m} - (k-1)$ such that

$$Q_1(f_1(\alpha), \cdots, f_m(\alpha)) \equiv \cdots \equiv Q_k(f_1(\alpha), \cdots, f_m(\alpha)) \equiv 0 \bmod N,$$

for some integer t, s, and k with $k \leq m$, and $|f_i(\alpha)| < 2^\rho$. From the attacker's perspective, we set $s = 1$ to minimize t and m and to maximize the bound of Theorem 4.4.2. As a result, inequality in (4.4.1) holds when $\binom{t+m}{m} > d + m$. However, if

$$\rho \geq \max_{k,h} \left\{ \frac{k}{mt} - \frac{m+1}{4mth \log N} \binom{th+m}{m} \right\} \log N = \left(\frac{1}{t} - \frac{m+1}{4m^2 t \log M} \binom{tm+m}{m} \right) \log N,$$

then this attack does not work. Thus, the polyACD problem cannot be solved by the above attack on the condition that $\rho > \frac{1}{t} \log N \left(1 - \frac{d+m}{4 \log N}\right) + \lambda$ for positive integers t and m when $\binom{t+m}{m} > d + m$. On the other hand, we recommended choosing $d > 4 \log M$ to avoid the distinguishing attack in Section 4.4.1. This results in a weaker condition upon ρ in Theorem 4.4.2 than in Section 4.4.2.

4.5 Implementation

In this section, we describe the implementation results when large integers are encrypted for the public evaluation of low degree polynomial. We implemented batch variant of our somewhat homomorphic encryption scheme in Section 4.2.2. In Section 4.5.1, we first describe the compression method of the public key. Then, our implementation results are described and com-

pared with RLWE-based scheme in Section 4.5.2.

4.5.1 Public Key Compression

Since our public key consists of many encryptions of zero, we compress the encryptions with a pseudo-random number generator using a technique introduced in [CNT12]. In brief, a random polynomial is generated by a pseudo-random number generator, and it is adjusted to the proper form using secret keys. More precisely, after α_i, β_j, and $p(x)$ are generated, a pseudo-random number generator h with a random seed se is used to generate $f_k(x)$, which are encryptions of zero in the public key for $1 \leq k \leq \tau$. The process begins by generating random polynomials $f'_k(x)$ in $\mathbb{Z}_N^{l+d-1}[x]$ using h and se. Next, $f''_k(x) = \mathsf{polyCRT}_{N,(x-\alpha_i)_{i=1}^l}(f'_k(\alpha_i) - Q_i e_{ki})_{i=1}^l$ are computed where $e_{ki} \leftarrow (-2^\rho, 2^\rho)$. Moreover, set $f_k(x) = f'_k(x) - f''_k(x) \bmod N$. Since $f_k(\alpha_i) = f'_k(\alpha_i) - f''_k(\alpha_i) = Q_i e_{ki}$ in $\mathbb{Z}_N$, it is an encryption of zero. Furthermore, $f_k(x)$ can be reconstructed from h, se, and $f''_k(x)$. Thus, instead of publishing $f_k(x)$, we publish only $f''_k(x)$ with h and se; this significantly compresses the size of the public key since the degree of $f''_k(x)$ is $l-1$ whereas the degree of $f_k(x)$ is $l+d-1$. Note that $h_k(x)$ in the public key can be compressed similarly. Assuming polynomials generated by the pseudo-random number generator are uniformly random in $\mathbb{Z}_N^{l+d-1}[x]$, the public key generated by this process is indistinguishable from the one generated by the usual key generation process. Semantic security of the scheme follows from Lemma 4.5.1 below in the random oracle model.

Lemma 4.5.1. *Given N and a polynomial $p(x) = \prod_{i=1}^{l}(x - \alpha_i)\prod_{j=1}^{d}(x -$*

$\beta_j) \in S_{(N,l+d)}$, *the followings distributions are the same:*

$$\begin{aligned}
\mathcal{D} &= \left\{ \begin{array}{c} (f(x), f'(x), f''(x)) \mid f'(x) \leftarrow \mathbb{Z}_N^{l+d-1}[x], e_j \leftarrow (-2^\rho, 2^\rho), \\ f''(x) = \mathsf{polyCRT}_{N,(x-\alpha_j)_{j=1}^l}(f'(\alpha_j) - Q_j e_j), \\ f(x) = f'(x) - f''(x) \bmod N \end{array} \right\} \\
\mathcal{D}' &= \left\{ \begin{array}{c} (f(x), f'(x), f''(x)) \mid e_j \leftarrow (-2^\rho, 2^\rho),\ r_j \leftarrow \mathbb{Z}_N,\ f''(x) \leftarrow \mathbb{Z}_N^{l-1}[x], \\ f(x) = \mathsf{polyCRT}_{N,(x-\alpha_j)_{j=1}^l,(x-\beta_j)_{j=1}^d}((Q_j e_j)_{j=1}^l, (r_j)_{j=1}^d), \\ f'(x) = f(x) + f''(x) \bmod N \end{array} \right\}.
\end{aligned}$$

Proof. Let $p_1(x) = \prod_{i=1}^l (x - \alpha_i) \bmod N$. Then one can easily verify that the below two distributions are the same:

$$\begin{aligned}
\mathcal{D}_0 &= \{f(x) \mid f(x) \leftarrow \mathbb{Z}_N^{l+d-1}[x]\}, \\
\mathcal{D}_1 &= \{q(x)p_1(x) + r(x) \mid q(x) \leftarrow \mathbb{Z}_N^{d-1}[x], r(x) \leftarrow \mathbb{Z}_N^{l-1}[x]\}.
\end{aligned}$$

Now let $(f(x), f'(x), f''(x))$ be sampled from $\mathcal{D}'$. Then $f(x)$ can be written as $f(x) = q(x)p_1(x)+r(x)$, where $q(x) \leftarrow \mathbb{Z}_N^{d-1}[x], r(x) = \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i), e_i \in (-2^\rho, 2^\rho)$. Since $(f(x), f'(x), f''(x))$ is determined by $(q(x), r(x), f''(x))$, we only consider the following distribution:

$$\begin{aligned}
\bar{\mathcal{D}}' &= \{(q(x), r(x), f''(x)) \mid (f(x), f'(x), f''(x)) \leftarrow \mathcal{D}', f(x) = q(x)p_1(x) + r(x)\} \\
&= \left\{ \begin{array}{c} (q(x), r(x), f''(x)) \mid q(x) \leftarrow \mathbb{Z}_N^{d-1}[x], e_i \leftarrow (-2^\rho, 2^\rho), \\ r(x) = \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i), f''(x) \leftarrow \mathbb{Z}_N^{l-1}[x] \end{array} \right\}
\end{aligned}$$

On the other hand, if $(f(x), f'(x), f''(x))$ is sampled from $\mathcal{D}$, then $f'(x) = q'(x)p_1(x) + r'(x)$ and $f'(\alpha_i) = r'(\alpha_i)$ where $q'(x) \leftarrow \mathbb{Z}_N^{d-1}[x]$ and $r'(x) \leftarrow \mathbb{Z}_N^{l-1}[x]$. This leads to the following equations:

$$
\begin{aligned}
f(x) &= f'(x) - f''(x) \\
&= q'(x)p_1(x) + \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(r'(\alpha_i)) - \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(f'(\alpha_i) - Q_i e_i) \\
&= q'(x)p_1(x) + \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i), \\
f''(x) &= \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(f'(\alpha_i) - Q_i e_i) \\
&= \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(r'(\alpha_i)) - \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i)) \\
&= r'(x) - \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i)).
\end{aligned}
$$

Thus, the following holds:

$$
\begin{aligned}
\bar{\mathcal{D}} &= \{(q(x), r(x), f''(x)) \mid (f(x), f'(x), f''(x)) \leftarrow \mathcal{D}, f(x) = q(x)p_1(x) + r(x)\} \\
&= \left\{ \begin{array}{c} (q(x), r(x), f''(x)) \mid q(x) \leftarrow \mathbb{Z}_N^{d-1}[x], e_i' \leftarrow (-2^\rho, 2^\rho), r'(x) \leftarrow \mathbb{Z}_N^{l-1}[x] \\ r(x) = \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i), e_i \leftarrow (-2^\rho, 2^\rho), \\ f''(x) = r'(x) - \mathsf{polyCRT}_{(x-\alpha_i)_{i=1}^l}(Q_i e_i')), \end{array} \right\} \\
&= \bar{\mathcal{D}}',
\end{aligned}
$$

and the lemma is proved. □

4.5.2 Implementation Results

As is noted before, the message space of this batch variant is $\mathbb{Z}_Q$ where $Q = \prod_{i=1}^l Q_i$ for relatively prime Q_i's. To evaluate the degree-3 polynomial of 170-bit integers, we need to choose Q larger than 2^{510}.

We provide the run times for key generation, encryption, decryption, and homomorphic addition and multiplication for various values of the maximal degree (i.e., for the polynomial whose homomorphic evaluation supported) and the bit-length of message space, $\log Q$ with a security parameter $\lambda = 80$. The experiments are conducted on an Intel Core i7 processor running at 3.4 GHz with 16 GB RAM. We used the number theory library (NTL) [Sho] running over the GNU multiple precision arithmetic library (GMP) [Gt] to provide the required big integer and polynomial arithmetics.

As is shown in Table 4.1, the time for encryption, addition, and multiplication is not significantly affected by the size of the message. In contrast, the

time for key generation and decryption increase moderately as the message size grows. For example, when the maximal degree is three, encryption and multiplication take 7 to 10 seconds and less than one seconds, respectively, for all parameters. In contrast, the key generation time increases from 65 seconds to 496 seconds when the message size grows from 512-bit to 16384-bit. The reason is that, we mainly used larger l to increase the message space, and this affects the key generation time mostly. Other timings are only slightly affected.

We compare the result to the timings of the recent RLWE-based scheme, YASHE [BLLN13], which has a publicly available implementation [LN14, Lep14]. To encrypt the large integer using YASHE, we combine the approaches taken in [NLV11] and [BLLN13]. Recall that the message and ciphertext of YASHE scheme are polynomials in R/tR and R/qR, respectively, for the polynomial ring R. Note that one can choose $R = \mathbb{Z}[x]/(x^n+1)$ where n is a power of two. To encrypt an integer m, we take the binary representation of $m = m_{k-1}2^{k-1} + \cdots + m_1 2 + m_0$, and convert it into a polynomial $m(x) = m_{k-1}x^{k-1} + \cdots + m_1 x + m_0$ with binary coefficients as in [NLV11]. For efficiency, we also use the CRT approach in [BLLN13] on the coefficients of $m(x)$. With this combined method, we can evaluate the degree-10 polynomial of 819-bit integers encoded in three different ciphertexts of which 31-bit t's are used, and it is more efficient than using only one 93-bit t. Experiments are conducted on the same processor as before, using the implementation in [Lep14].

Table 4.2 shows the experimental results where the maximal degree is ten. To evaluate degree-10 polynomial of 51-bit integers, we choose n and $\prod t_j$ larger than $510\,(= 10 * 51 < 512)$ and $128 \cdot 10^9$, respectively, and the timings are shown in the first row in the table. As it shows, our scheme becomes more efficient as the size of message space grows. When evaluating polynomials of 819-bit integers, our scheme is slightly faster in decryption, addition, and multiplication. For the homomorphic evaluation of 1638-bit integers, our scheme is about ten times faster in multiplication.

Table 4.1: Timings of our batch somewhat homomorphic encryption scheme

Deg.	$\log Q$	ct size	pk size	KeyGen(s)	Enc(s)	Dec(s)	Add(s)	Mult(s)
3	512	2.18 MB	3.27 MB	65.052	7.699	0.037	0.0004	0.983
	1024	2.18 MB	4.36 MB	110.099	7.844	0.056	0.0006	0.984
	2048	2.18 MB	6.56 MB	192.928	8.275	0.095	0.0006	0.983
	4096	2.18 MB	11.01 MB	375.752	8.083	0.175	0.0006	0.984
	8192	2.18 MB	20.09 MB	760.219	8.318	0.330	0.0006	0.986
	16384	2.18 MB	39.01 MB	492.175	9.371	0.645	0.0006	0.984
5	512	2.18 MB	4.36 MB	114.451	8.496	0.057	0.0004	1.182
	1024	2.18 MB	6.56 MB	197.536	7.936	0.096	0.0008	1.177
	2048	2.18 MB	11.01 MB	366.199	8.389	0.171	0.0004	1.172
	4096	2.19 MB	20.26 MB	820.299	9.690	0.362	0.0008	1.195
	8192	2.21 MB	39.88 MB	528.961	10.120	0.718	0.0008	1.220
	16384	2.23 MB	82.68 MB	671.042	11.632	0.430	0.0008	1.231
7	512	2.18 MB	6.56 MB	198.544	7.861	0.096	0.0005	1.252
	1024	2.18 MB	11.01 MB	380.736	8.981	0.174	0.0005	1.249
	2048	2.18 MB	20.14 MB	744.439	8.283	0.330	0.0005	1.264
	4096	2.18 MB	39.11 MB	500.811	10.316	0.683	0.0008	1.277
	8192	2.18 MB	80.04 MB	632.576	11.521	0.411	0.0008	1.274
	16384	2.20 MB	176.15 MB	1185.399	14.290	0.471	0.0005	1.300
10	512	3.51 MB	21.48 MB	998.586	21.421	0.341	0.0010	1.967
	1024	3.51 MB	39.77 MB	2334.649	21.933	0.649	0.0010	1.922
	2048	3.51 MB	77.30 MB	1319.139	23.173	0.668	0.0010	1.908
	4096	3.51 MB	156.17 MB	1598.169	25.448	0.695	0.0010	1.901
	8192	4.39 MB	215.16 MB	2497.199	35.413	0.891	0.0012	2.310
	16384	4.39 MB	448.65 MB	3959.849	39.236	0.969	0.0012	2.288

(*) All timings are in seconds.

Table 4.2: Timings of YASHE for the evaluation of degree-10 polynomial

Deg.	$\log m$	ct size	pk size	KeyGen(s)	Enc(s)	Dec(s)	Add(s)	Mult(s)
10	51	1.15 MB	15.45 MB	82.363	0.192	0.183	0.0062	0.998
	102	1.22 MB	17.43 MB	84.419	0.218	0.204	0.0074	1.099
	204	1.53 MB	23.24 MB	109.818	0.252	0.244	0.0082	1.330
	409	2.43 MB	54.35 MB	401.271	0.750	0.736	0.0120	3.038
	819	3.12 MB	87.08 MB	571.914	0.933	0.917	0.0141	4.737
	1638	9.71 MB	405.71MB	4183.168	3.041	3.001	0.0400	20.294

(*) In this table, m is a maximum size of the input integers.
(*) This table corresponds to the degree-10 part of Table 4.1.
(*) All timings are in seconds.

We believe that the results in this section shows that our scheme is efficient when evaluating low degree polynomial of large integers.

4.6 Conclusion

In this chapter, we examined the polynomial analogue of the ACD problem– the polynomial approximate common divisor (polyACD) problem and provided extensive cryptanalysis on this new problem. The structure of the polyACD problem is very similar to several problems (e.g., (R)LWE, ACD); however, its properties are slightly different. Because of its similarity, we apply several attacks to the polyACD problem, such as Chen-Nguyen's attack, Coppersmith's attack and the distinguishing attack. We also proposed a method for finding small solutions of modular multivariate polynomials with multiple instances.

We presented a somewhat homomorphic encryption scheme whose security is based on the polyACD problem as an application of our new problem. Our scheme is conceptually simple and does not require a complicated re-linearization process. For this reason, the multiplication timing of ciphertexts of our scheme is faster than RLWE-based homomorphic encryption using polynomial rings. Our scheme can be easily converted into a leveled FHE scheme using the technique in [Bra12, CLT14]. The resulting scheme has features similar to the variant of van Dijk et al.'s scheme by Coron et al. Our scheme, however, did not use the hardness of subset sum assumption, while the leveled FHE scheme presented by Coron et al. needs an additional subset sum assumption, to provide homomorphic divisions of the secret key.

Providing a reduction between polyACD problem and other cryptographic hard problems remains as an interesting open problem.

Chapter 5

Conclusions

This paper dealt with application and development for the homomorphic encryption. In the applied research, an additive homomorphic encryption scheme, proposed by Naccache and Stern, was used for the privacy-preserving set union protocol, and a RLWE-based BGV homomorphic encryption scheme was used for the program static-analysis in secrecy.

To support an effective set union operation, the special encoding function that expressed set's elements of the participants was proposed, which was used to develop the technology that enabled effective recovery of the root of polynomials even if the polynomials are in the unique factorization domain. Applying this encoding function to polynomial representation for sets, we obtained an efficient constant round threshold set union protocol without assuming an honest majority.

As the next step, in the secure program analysis, we applied homomorphic encryption to propose a secure pointer analysis. By exposing type information of variables in the target program, we could dramatically reduce the cost of homomorphic operations. The level-by-level fashioned analysis decreased the depth of the arithmetic circuit for the pointer analysis from $O(m^2 \log m)$ to $O(\log m)$, where m is the number of pointer variables in the program. Without this technique, even a tiny program of 10 variables cannot be analyzed in secrecy. Through this, it became possible to analyze whether the

pointer variables in the program can point to the intended variable or storage location during execution from the encrypted program information.

Finally, we proposed the polynomial approximate common divisor problem, a new cryptographic hard problem, along with a new leveled fully homomorphic encryption based on this problem. We examined the relations of the variants for the proposed problem and cryptanalyze the proposed problems by applying all known attacks such as Chen-Nguyen's attack, Coppersmith's attack, and distinguishing attack. The proposed homomorphic encryption scheme support not only SIMD style operations, but also large integer arithmetic. Unlike other homomorphic schemes over the polynomial rings, the proposed scheme needs only single polynomial for homomorphic evaluations, and so does not require a complicated re-linearization process. For this reason, the proposed scheme is more efficient than RLWE-based homomorphic encryption over polynomial ring when evaluation low degree polynomial of large integers.

In general, the ciphertext of somewhat homomorphic encryption contains a certain noise, which increases with successive homomorphic operations, and the noise increasing during homomorphic evaluation should be managed for correctness. The leveled fully homomorphic encryptions over the integers exploit the subset sum of the secret key to provide the homomorphic operations for dividing the secret key while applying the noise-management technique. In contrast, the proposed homomorphic encryption does not require dividing of secret information in the decryption process; hence, it is not necessary to use the subset sum of the secret key.

Bibliography

[BGH+13] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and DavidJ. Wu. Private Database Queries Using Somewhat Homomorphic Encryption. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, volume 7954 of *LNCS*, pages 102–118. Springer Berlin Heidelberg, 2013.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325. ACM, 2012.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and Impossibility Results for Encryption and Commitment Secure under Selective Opening. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, 2009.

[BLLN13] JoppeW. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer Berlin Heidelberg, 2013.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant dis-

tributed computation (extended abstract). In Janos Simon, editor, *ACM Symposium on Theory of Computing (STOC) 1988*, pages 1–10. ACM, 1988.

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer Berlin Heidelberg, 2012.

[Bra13] Zvika Brakerski. When Homomorphism Becomes a Liability. In Amit Sahai, editor, *Theory of Cryptography Conference – TCC 2013*, volume 7785 of *LNCS*, pages 143–161. Springer, 2013.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, Foundations of Computer Science – FOCS 2011, pages 97–106, Washington, DC, USA, 2011. IEEE Computer Society.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In Phillip Rogaway, editor, *Advances in Cryptology –- CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer Berlin Heidelberg, 2011.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch Fully Homomorphic Encryption over the Integers. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 315–335. Springer, 2013.

BIBLIOGRAPHY

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, 2001.

[CH13] Henry Cohn and Nadia Heninger. Approximate common divisors via lattices. In *Proccedings of the Tenth Algorithmic Number Theory Symposium - ANTS X*. Mathematical Sciences Publishers, 2013.

[CHL12] Jung Hee Cheon, Hyunsook Hong, and Hyung Tae Lee. Invertible polynomial representation for set operations. Cryptology ePrint Archive, Report 2012/526, 2012. Available at http://eprint.iacr.org/2012/526.

[CHL14] JungHee Cheon, Hyunsook Hong, and HyungTae Lee. Invertible Polynomial Representation for Private Set Operations. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology – ICISC 2013*, volume 8565 of *LNCS*, pages 277–292. Springer International Publishing, 2014.

[CHLR14] Jung Hee Cheon, Hyunsook Hong, Moon Sung Lee, and Hansol Ryu. The Polynomial Approximate Common Divisor Problem and its Application to the Fully Homomorphic Encryption. 2014. To appear in Information Sciences.

[CKL15] Jung Hee Cheon, Miran Kim, and Kristin Lauter. Homomorphic Computation of Edit Distance. *IACR Cryptology ePrint Archive*, 2015:132, 2015. To appear in WAHC 2015.

[CKS13] Ayantika Chatterjee, Manish Kaushal, and Indranil Sengupta. Accelerating Sorting of Fully Homomorphic Encrypted Data. In Goutam Paul and Serge Vaudenay, editors, *Progress in Cryptology - INDOCRYPT 2013*, volume 8250 of *LNCS*, pages 262–273. Springer International Publishing, 2013.

BIBLIOGRAPHY

[CKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, 2010.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-Invariant Fully Homomorphic Encryption over the Integers. In Hugo Krawczyk, editor, *Public Key Cryptography – PKC 2014*, volume 8383 of *LNCS*, pages 311–328. Springer, 2014.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *LNCS*, pages 487–504. Springer, 2011.

[CN12] Yuanmi Chen and Phong Q. Nguyen. Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 502–519. Springer, 2012.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 446–464. Springer, 2012.

[Cop96] Don Coppersmith. Finding a Small Root of a Univariate Modular Equation. In Ueli Maurer, editor, *Advances in Cryptology*

– *EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 155–165. Springer, 1996.

[CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report, No. 260, March 1997, 1997.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.

[CS15] JungHee Cheon and Damien Stehlé. Fully Homomophic Encryption over the Integers Revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 513–536. Springer Berlin Heidelberg, 2015.

[CT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *Financial Cryptography (FC) 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, 2010.

[FFSA98] Manuel Fähndrich, Jeffrey S. Foster, Zhendong Su, and Alexander Aiken. Partial online cycle elimination in inclusion constraint graphs. In *Proceedings of the ACM SIGPLAN 1998 Conference on Programming Language Design and Implementation*, PLDI '98, pages 85–96, New York, NY, USA, 1998. ACM.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.

[FPS01] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In Yair Frankel, editor, *Financial Cryptography (FC) 2000*, volume 1962 of *LNCS*, pages 90–104. Springer, 2001.

[Fri07] Keith B. Frikken. Privacy-preserving set union. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security (ACNS) 2007*, volume 4521 of *LNCS*, pages 237–252. Springer, 2007.

[FS98] Christian Fecht and Helmut Seidl. Propagating differences: An efficient new fixpoint algorithm for distributive constraint systems. *Nord. J. Comput.*, 5(4):304–329, 1998.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 129–148. Springer, 2011.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully Homomorphic Encryption with Polylog Overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. In *CRYPTO 2012*, 2012.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *ACM Symposium*

on Theory of Computing (STOC) 1987, pages 218–229. ACM, 1987.

[Gt] Torbjörn Granlund and the GMP development team. GMP: The GNU Multiple Precision Arithmetic Library (version 6.0.0). `https://gmplib.org/`.

[HG01] Nick Howgrave-Graham. Approximate Integer Common Divisors. In JosephH. Silverman, editor, *Cryptography and Lattices – CaLC 2001*, volume 2146 of *LNCS*, pages 51–66. Springer Berlin Heidelberg, 2001.

[HKK+13] Jeongdae Hong, Jung Woo Kim, Jihye Kim, Kunsoo Park, and Jung Hee Cheon. Constant-round privacy preserving multiset union. *Bulletin of the Korean Mathematical Society*, 50(6):1799–1816, 2013.

[HL07] Ben Hardekopf and Calvin Lin. The ant and the grasshopper: Fast and accurate pointer analysis for millions of lines of code. *SIGPLAN Not.*, 42(6):290–299, June 2007.

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy Encryption: Constructions from General Assumptions and Efficient Selective Opening Chosen Ciphertext Security. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88. Springer, 2011.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. Cryptology ePrint Archive, Report 2014/106, 2014. `http://eprint.iacr.org/`.

[HT01] Nevin Heintze and Olivier Tardieu. Ultra-fast aliasing analysis using cla: A million lines of c code in a second. In *PLDI '01*, 2001.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In Omer Reingold, editor, *Theory of Cryptography – TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, 2009.

[JM06] Ellen Jochemsz and Alexander May. A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 267–282. Springer, 2006.

[Kai80] Thomas Kailath. *Linear Systems*. Prentice-Hall, Inc, 1 edition, January 1980.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer Berlin Heidelberg, 2004.

[KS01] Fabian Kuhn and René Struik. Random walks revisited: Extensions of pollard's rho algorithm for computing multiple discrete logarithms. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography (SAC) 2001*, volume 2259 of *LNCS*, pages 212–229. Springer, 2001.

[KS05] Lea Kissner and Dawn Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257, 2005.

[Lan02] Serge Lang. Algebra. *Graduate Texts in Mathematics*, 211, 2002.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-Fly Multiparty Computation on the Cloud via Multikey

Fully Homomorphic Encryption. In *STOC '12 Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234. ACM, 2012.

[Lee12] Hyung Tae Lee. Private communication. 2012.

[Lee14] Woosuk Lee. Private communication. 2014.

[Leo06] Roots of random polynomials over a finite field. *Mathematical Notes*, 80(1-2):300–304, 2006.

[Lep14] Tancrede Lepoint. A proof-of-concept implementation of the homomorphic evaluation of SIMON using FV and YASHE leveled homomorphic cryptosystems. Available at https://github.com/tlepoint/homomorphic-simon, 2014.

[LHYC15] Woosuk Lee, Hyunsook Hong, Kwangkeun Yi Yi, and Jung Hee Cheon. Static Analysis with Set-closure in Secrecy. 2015. To appear in SAS 2015.

[LN14] Tancrède Lepoint and Michael Naehrig. A Comparison of the Homomorphic Encryption Schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 318–335. Springer International Publishing, 2014.

[LP11] Richard Lindner and Chris Peikert. Better Key Sizes (and Attacks) for Lwe-Based Encryption. In Joseph H. Silverman, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer Berlin Heidelberg, 2010.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.

[MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 382–400. Springer, 2004.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.

[NS98] David Naccache and Jacques Stern. A New Public Key Cryptosystem Based on Higher Residues. In Li Gong and Michael K. Reiter, editors, *ACM Conference on Computer and Communications Security (ACM CCS) 1998*, pages 59–66. ACM, 1998.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 308–318. Springer, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[PKH03] D.J. Pearce, P.H.J. Kelly, and C. Hankin. Online cycle detection and difference propagation for pointer analysis. In *Source Code Analysis and Manipulation, 2003. Proceedings. Third IEEE International Workshop on*, pages 3–12, 2003.

[SCK12] JaeHong Seo, JungHee Cheon, and Jonathan Katz. Constant-Round Multi-party Private Set Union Using Reversed Laurent Series. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012*, volume 7293 of *LNCS*, pages 398–412. Springer, 2012.

[Sha93] Adi Shamir. On the Generation of Multivariate Polynomials which are Hard to Factor. In *STOC'93 Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 796–804. ACM, 1993.

[Sho] Victor Shoup. NTL: A Library for doing Number Theory (version 6.1.0). `http://www.shoup.net/ntl/`.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000.

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2nd edition, 2009.

[SS09] Yingpeng Sang and Hong Shen. Efficient and secure protocols for privacy-preserving set operations. *ACM Transactions on Information and System Security*, 13(1), 2009.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *Information Security and Cryptography (ICISC) 2006*, volume 4296 of *LNCS*, pages 29–40. Springer, 2006.

[SV10] N. P. Smart and F. Vercauteren. Fully homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptog-*

raphy – PKC 2010, volume 6056 of *LNCS*, pages 420–443. Springer, 2010.

[SV14] N.P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[Uma08] Christopher Umans. Fast polynomial factorization and modular composition in small characteristic. In Cynthia Dwork, editor, *ACM Symposium on Theory of Computing (STOC) 2008*, pages 481–490. ACM, 2008.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, 2010.

[VZGG13] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, June 2013.

국문초록

동형 암호는 복호화 과정을 거치지 않고 암호화 된 상태에서 암호문끼리 연산을 통해 데이터의 자료 처리를 가능하게 하는 암호 기술로 최근 많이 사용되고 있는 클라우드 서비스 환경에서 발생 할 수 있는 보안 문제들을 해결 할 수 있는 암호시스템으로 주목 받고 있다.

본 학위 논문에서는 동형 암호 응용 기술 연구와 함께 새로운 동형암호 알고리즘 개발에 대해 연구한다. 응용기술 연구에서는 Naccache-Stern 덧셈 동형 암호를 이용하여 프라이버시를 보존하는 합집합 연산 프로토콜과 RLWE기반 BGV 동형암호를 이용하여 비밀 프로그램 정적 분석 방법을 제안한다.

효율적인 합집합 연산을 지원하기 위해, 참여자의 집합원소들을 표현하는 특별한 인코딩 함수 제안하고, 제안한 인코딩 함수를 적용하여 유일 인수 분해 정역(unique factorization domain)이 아닌 공간에서도 다항식들의 근을 효율적으로 복구 할 수 있는 방법을 제안한다. 이를 바탕으로, 현존하는 가장 효율적인 상수라운드의 합집합 연산 프로토콜을 제안한다. 프로그램 비밀 분석에서는 동형암호를 이용하여 비밀 포인터 분석방법을 제시한다. 프로그램 변수의 타입 정보를 이용하여, 동형암호 연산시 필요한 곱 연산의 횟수를 $O(m^2 \log m)$ 에서 $O(\log m)$ 로 획기적으로 줄일 수 있는 방법을 제시하고, 이를 바탕으로 실제 생활에 이용 가능한 수준의 프로그램 비밀 분석 방법을 제안한다. 이를 통해 분석가는 암호화된 프로그램 정보를 이용하여 프로그램에 있는 포인터 변수가 실행 중 어느 변수 혹은 저장 장소를 가리킬 수 있는 지에 대한 분석이 가능해진다.

마지막으로 새로운 암호학적 난제인 다항식 근사공약수 문제를 제안하고, 이 문제에 기반하는 새로운 동형암호를 제안한다. 제안한 동형암호는 Djik 등이 제안한 동형암호의 다항식 버전으로 볼 수 있으며, 이에 따라 데이터 병렬처리뿐만 아니라 큰 정수 연산 지원하는 특징을 가지고 있다. Djik 등이 제안한 동형암호계열의 완전동형암호들은 비밀키를 나누는 연산을 제공하기 위해 부분합 문제가 어렵다는 가정을 사용하는 반면, 제안한 동형암호는 복호화 과정에서 비밀 정보를 나누는 과정이 필요 없기 때문에 부분합 문제의 가정을 필요로 하지 않는다.

주요어휘 : 덧셈동형암호, 준동형 암호, 프라이버시를 보존하는 합집합, 비밀 프로그램 정적 분석, 포인터분석, 다항식 근사 최대공약수 문제

학번 : 2009-22898