이학박사 학위논문

Efficient Fully Homomorphic Encryption over the Integers

(효율적인 정수 기반 동형 암호)

2015년 2월

서울대학교 대학원

수리과학부

김진수

Efficient Fully Homomorphic Encryption over the Integers

(효율적인 정수 기반 동형 암호)

지도교수 천정희

이 논문을 이학박사 학위논문으로 제출함

2014년 11월

서울대학교 대학원

수리과학부

김진수

김진수의 이학박사 학위논문을 인준함

2014년 12월

위 원 장	김 명 환	(인)
부 위 원 장	천 정 희	(인)
위 원	이 인 석	(인)
위 원	오 병 권	(인)
위 원	서 재 홍	(인)

Efficient Fully Homomorphic Encryption over the Integers

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

Jinsu Kim

Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Sciences
Seoul National University

February 2015

Abstract

Fully homomorphic encryption allows a worker to perform additions and multiplications on encrypted plaintext values without decryption. The first construction of a fully homomorphic scheme (FHE) based on ideal lattices was described by Gentry in 2009. Since Gentry's breakthrough result, many improvements have been made, introducing new variants, improving efficiency, and providing new features.

The most FHE schemes still have very large ciphertexts (millions of bits for a single ciphertext). This presents a considerable bottleneck in practical deployments. To improve the efficiency of FHE schemes, especially ciphertext size, we can consider the following two observations. One is to improve the ratio of plaintext and ciphertext by packing many messages in one ciphertext and the other is to reduce the size of FHE-ciphertext by combining FHE with existing public-key encryption.

In the dissertation, we study on construction of efficient FHE over the integers. First, we propose a new variant DGHV fully homomorphic encryption to extend message space. Using Chinese remainder theorem, our scheme reduces the overheads (ratio of ciphertext computation and plaintext computation) from $\tilde{O}(\lambda^4)$ to $\tilde{O}(\lambda)$. We reduce the security of our Somewhat Homomorphic Encryption scheme to a decisional version of Approximate GCD problem (DACD).

To reduce the ciphertext size, we propose a hybrid scheme that combines public key encryption (PKE) and somewhat homomorphic encryption (SHE). In this model, messages are encrypted with a PKE and computations on encrypted data are carried out using SHE or FHE after homomorphic de-

cryption. Our approach is suitable for cloud computing environments since it has small bandwidth, low storage requirement, and supports efficient computing on encrypted data.

We also give alternative approach to reduce the FHE ciphertext size. Some of recent SHE schemes possess two properties, the public key compression and the key switching. By combining them, we propose a hybrid encryption scheme in which a block of messages is encrypted by symmetric version of the SHE and its secret key is encrypted by the (asymmetric) SHE. The ciphertext under the symmetric key encryption is compressed by using the public key compression technique and we convert the ciphertext into asymmetric encryption to enable homomorphic computations using key switching technique.

Key words: fully homomorphic encryption, somewhat homomorphic encryption, hybrid scheme, approximate GCD problem, ciphertext compression
Student Number: 2009-20264

Contents

CONTENTS

Chapter 1

Introduction

In 1978, Rivest, Adleman and Dertouzos introduced the basic concept of privacy homomorphism that allows computation on encrypted data without decryption [RAD78]. It was an interesting work whose idea preceded the recent development of fully homomorphic encryption, although actual example schemes proposed in the paper are all susceptible to simple known-plaintext attacks. After thirty years, Gentry proposed the first fully homomorphic encryption scheme based on ideal lattices which supports arbitrarily many additions and multiplications on encrypted bits [Gen09]. First, one constructs a *somewhat homomorphic encryption* (SHE) scheme, which only supports a limited number of multiplications: ciphertexts contain some noise that becomes larger with successive homomorphic multiplications, and only ciphertexts whose noise size remains below a certain threshold can be decrypted correctly. The second step is to *squash* the decryption procedure associated with an arbitrary ciphertext so that it can be expressed as a low degree polynomial in the secret key bits. Then, Gentry's key idea, called *bootstrapping*, consists in homomorphically evaluating this decryption polynomial on encryptions of the secret key bits, resulting in a different ciphertext associated with the

same plaintext, but with possibly reduced noise. This *refreshed* ciphertext can then be used in subsequent homomorphic operations. By repeatedly refreshing ciphertexts, the number of homomorphic operations becomes unlimited, resulting in a *fully homomorphic encryption* (FHE) scheme. His breakthrough paper drew an explosive interest and lead numerous researches in this area [DGHV10, CMNT11, CNT12, GH11b, SV10, SS10, SS11, GHS12a, BV11, BGV12, Bra12, GSW13].

Brakerski, Gentry and Vaikuntanathan described a different framework, using *modulus switching*, where the ciphertext noise grows only linearly with the multiplicative level instead of exponentially, so that bootstrapping is no longer necessary to obtain a scheme supporting the homomorphic evaluation of any given polynomial size circuit [BGV12]. At Crypto 2012, Brakerski constructed a scale-invariant fully homomorphic encryption scheme based on the LWE problem, in which the same modulus is used throughout the evaluation procedure, instead of a ladder of moduli when doing modulus switching [Bra12]. Recently, Gentry, Sahai, and Waters [GSW13] showed how to achieve an FHE scheme that does not require additional auxiliary information for the homomorphic evaluation. This scheme uses matrices for ciphertexts instead of vectors. In PKC 2014, Coron, Lepoint and Tibouchi described a variant of the van Dijk et. al. FHE scheme over the integers with the same scale-invariant property [CLT14]. Their scheme has a single secret modulus whose size is linear in the multiplicative depth of the circuit to be homomorphically evaluated, instead of exponential.

Even though FHE schemes can support both additions and multiplications on encrypted data infinitely, FHE schemes are still far from being practical because of its large computational cost and large ciphertexts. We refer the implementation of FHE over the integers by Coron et. al [CLT14] for

72 security level as an example. Despite the optimizations for improving the performance and reducing the size of public key, encryption of one bit takes one minute on a a high-end Intel Xeon based server. Furthermore, after every few bit-AND operations, a recryption (convert) operation, which also takes about one minute, must be applied to reduce the noise in the ciphertext to a manageable level. In addition to the computation efficiency, the Coron et.al. scheme requires a ciphertext of more than 15,000,000 bits for encrypting 569 bits. This huge ciphertext size causes bottlenecks on bandwidths required to transfer the ciphertexts.

1.1 A Brief Overview of this Thesis

The goal of this dissertation is to improve the efficiency of fully homomorphic encryption over the integers and move them to practice. To achieve this goal, we propose varieties of ways to improve existing FHE schemes.

Extending Message Space of FHE. We extend the fully homomorphic encryption scheme over the integers of van Dijk et al. (DGHV) into a batch fully homomorphic encryption scheme, i.e. to a scheme that supports encrypting and homomorphically processing a vector of plaintexts as a single ciphertext. We first construct a *symmetric-key* SHE over the integers by exploiting the standard technique to insert an error to a message before encryption and Chinese remainder theorem (CRT). A ciphertext of message vector $\mathbf{m} = (m_1, \ldots, m_k) \in \mathbb{Z}_{Q_1} \times \cdots \times \mathbb{Z}_{Q_k}$ is of the form $c = \mathrm{CRT}_{(q_0,p_1,\ldots,p_k)}(e, m_1 + e_1Q_1, \ldots, m_k + e_kQ_k)$, where $\{p_1, \ldots, p_k\}$ is a secret-key set and e_i's are errors in some range.* We convert this symmetric scheme

*We denote by $\mathrm{CRT}_{(p_0,\ldots,p_k)}(m_0, \ldots, m_k)$ the unique integer in $(-\frac{\prod_i p_i}{2}, \frac{\prod_i p_i}{2}]$ which is congruent to m_i modulo p_i for all i.

to a somewhat homomorphic *public-key* encryption scheme by publishing many encryptions of zero and encryptions of k elementary elements $E_i = \mathrm{CRT}_{(Q_1,\ldots,Q_i,\ldots,Q_k)}(0,\ldots,1,\ldots,0)$.

We reduce the security of our Somewhat Homomorphic Encryption scheme to a decisional version of Approximate GCD problem (DACD). Approximate GCD (ACD) problem is to find p given many multiples of p with some errors (i.e. $x_i = pq_i + e_i$). Note that the ACD assumption was used to prove the security of the DGHV scheme [DGHV10], and another decisional version of the approximate GCD assumption which is slightly different from ours was used to prove the security of a more efficient variant of DGHV by Coron et al. [CNT12].

The ciphertext size of our FHE scheme is $\tilde{O}(\lambda^5)$ as in the DGHV scheme for the security parameter λ. While the plaintext size of the DGHV is $O(\lambda)$, that of ours is $O(\lambda^4)$ for $O(\lambda)$-bit $Q_1,\ldots,Q_k$ with $k = O(\lambda^3)$. Consequently, our scheme reduces the overheads (ratio of ciphertext computation and plaintext computation) from $\tilde{O}(\lambda^4)$ to $\tilde{O}(\lambda)$. For the case that the message space is $\mathbb{Z}_2^k$, the overhead is reduced from $\tilde{O}(\lambda^8)$ to $\tilde{O}(\lambda^5)$ for $k = O(\lambda^3)$.

Our scheme has an advantage over [GHS12a] in applications requiring larger message space. When dealing with arithmetic on $\mathbb{Z}_Q$ for $\log Q = O(\lambda^4)$, our SHE scheme can support $O(\lambda)$ multiplications with many additions. One of the important applications of homomorphic encryption schemes is to securely evaluate a multivariate polynomial over integers. Our scheme is an attractive choice for evaluating a polynomial of degree $O(\lambda)$ with inputs $\Omega(\lambda^2)$. Also our scheme can be used in the applications requiring SIMD style operations in k copies of $\mathbb{Z}_Q$ for $\log Q = \lambda, k = O(\lambda^3)$.

Hybrid Scheme of PKE and SHE The large ciphertext size of existing FHE schemes is another major problem of the FHE schemes. The large

bandwidth will be required when the applications need to transfer ciphertext through the network. To address this issue, we formalized the concept of scheme conversion between different encryption schemes originally mentioned in [NLV11, GHS12b]. In addition, we provide efficient instantiation of the conversion of FHE schemes into other encryption schemes with small ciphertext sizes. We call this combination by *hydrid scheme* of FHE. In this way, we can "compress" the ciphertexts of the FHE schemes and reduce the bandwidth requirement.

When using additive (resp. multiplicative) homomorphic encryption as the underlying encryption scheme for communication, we obtain the additional advantage that additions (resp. multiplications) can be computed without converting to FHE. For multiplicative homomorphic encryptions (MHE) in particular, one can compute $\mathsf{FHE}(f(m_1, \ldots, m_k))$ from $\mathsf{PKE.Enc}(m_1), \ldots,$ $\mathsf{PKE.Enc}(m_k)$ without (expensive) bootstrapping for any multivariate polynomial $f(x_1, \ldots, x_k)$ with polynomially many terms.

One problem when using MHE in the hybrid scheme is that the message space for MHE schemes is not usually closed under addition. For example, the (IND-CPA) ElGamal encryption over a ring R can only take messages with elements in a prime order subgroup, which covers only a small part of R. To resolve this, therefore, we construct a MHE whose message space is $\mathbb{Z}_N^\times$ for an RSA modulus $N = p_1 p_2$. The proposed scheme is constructed by combining ElGamal encryption over $\mathbb{Z}_N^\times$ and Goldwasser-Micali encryption over $\mathbb{Z}_N$, and is secure under the decisional Diffie–Hellman assumption and the quadratic residuosity assumption for common $N = pq$.

Compression of FHE Ciphertexts. We consider different method to reduce the bandwidth when transmitting encrypted data. We start with an asymmetric leveled SHE having a *switch key* $\mathrm{SWK}_{S:S_L}$, with which a con-

version algorithm transforms a ciphertext $\mathrm{SHE}_S(m)$ of a message m with the private key S into $\mathrm{SHE}_{S_L}(m)$ of the same message with the private key S_L of lower level. We have several candidate schemes with such a property [BV11, BGV12, Bra12, CNT12, CLT14].

We consider a public key compression technique in [CNT12] to reduce the SHE-ciphertext size. In the DGHV scheme [DGHV10] and the LWE-based schemes [BV11, Bra12], the public key is a set of encryptions of the zero and so the public key compression techniques is essentially the ciphertext compression in its symmetric version. More precisely, in the DGHV shceme, the $\mathrm{SSHE}_S(m)$ is compressed into a seed se and its correction value $\delta(m)$ such that $\mathrm{PRNG}(\mathsf{se})+\delta(m) = \mathrm{SSHE}_S(m)$. In the LWE-based schemes, the ciphertext is of the form $(\mathbf{b}, \mathbf{A})$ where a matrix $\mathbf{A}$ is generated from $\mathrm{PRNG}(\mathsf{se})$ and can be compressed into a small seed se and its correction value $\delta(m) = \mathbf{b}$. However, this technique can not be applied to its *asymmetric* versions where an encryption of a message m is made from a sparse subset sum of the ciphertexts of the zero instead of choosing a random parts of ciphertext.

Then a hybrid encryption of a message m is composed of the compressed ciphertext $(\mathsf{se}, \delta(m))$ of $\mathrm{SSHE}_S(m)$ along with the switch key $\mathrm{SWK}_{(S:S_L)}$. On receiving a ciphertext $(\mathrm{SWK}_{(S:S_L)}, \mathsf{se}, \delta(m))$, recover $\mathrm{SSHE}_S(m)$ from $(\mathsf{se}, \delta(m))$ and convert it to $\mathrm{SSHE}_{S_L}(m)$ with $\mathrm{SWK}_{(S:S_L)}$. This procedure is possible even when the SSHE has low homomorphic capacity. A conversion is done by a matrix multiplications for LWE-base SHE and inner products for the DGHV scheme and so very fast. In the leveled homomorphic encryption schemes, the switch key $\mathrm{SWK}_{(S:S_L)}$ is made by one who knows both of the private key S and S_L, but in this scenario the secret key S_L is not available to an encryptor. We provide an algorithm to make the switch key $\mathrm{SWK}_{(S:S_L)}$ without knowing the secret key S_L.

Contributions

The thesis contains a joint work with Jung Hee Cheon, Moon Sung Lee and Aaram Yun [KLYC13] which appears in Eurocrypt 2013 as a merged paper [CCK+13] and a work with Jung Hee Cheon [CK15] which is accepted to IEEE Information Forensics and Security . It also includes a prepublication with Jung Hee Cheon and Moon Sung Lee [CKL15].

List of Papers

- [KLYC13] Jinsu Kim, Moon Sung Lee, Aaram Yun and Jung Hee Cheon: CRT-based Fully Homomorphic Encryption over the Integers. IACR Cryptology ePrint Archive 2013: 57.
- [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. EUROCRYPT 2013: 315-335
- [CK15] Jung Hee Cheon and Jinsu Kim: A Hybrid Scheme of Public-key Encryption and Somewhat Homomorphic Encryption. To appear in IEEE Transactions on Information Forensics and Security 2015.
- [CKL15] Jung Hee Cheon, Jinsu Kim and Moon Sung Lee: Hybrid Asymmetric Homomorphic Encryption. In Manuscript.

Chapter 2

CRT-based Fully Homomorphic Encryption over the Integers

In 2009, Gentry [Gen09, Gen10] introduced the first fully homomorphic encryption scheme based on ideal lattices which support arbitrary many additions and multiplications on encrypted bit. His breakthrough paper drew an explosive interest and leads numerous researches in this area [DGHV10, CMNT11, CNT12, GH11b, SV10, SS10, SS11, GHS12a, BV11, Bra12].

The concept of computation on encrypted data without decryption was firstly introduced in 1978 by Rivest, Adleman and Detourzos [RAD78]. They defined a *privacy homomorphism* to be an encryption $\mathsf{Enc} : \mathcal{P} \to \mathcal{C}$ which permits computation of $\mathsf{Enc}(m_1 * m_2)$ from $\mathsf{Enc}(m_1), \mathsf{Enc}(m_2)$ for an algebraic operation $*$ on $\mathcal{P}$, without revealing m_1 and m_2. They presented five examples, but one of them was essentially RSA encryption supporting multiplication only, and the rest of them were insecure against known plaintext attack [BY88].

One of the examples given in [RAD78] is as follows. Let p, q be large primes and $n = pq$. The plaintext space is $\mathbb{Z}_n$ and the ciphertext space is

$\mathbb{Z}_p \times \mathbb{Z}_q$. An encryption of a message $m \in \mathbb{Z}_n$ is $(m \bmod p, m \bmod q)$ and the decryption is done using the Chinese Remainder Theorem (CRT). This cryptosystem supports modular addition and multiplication. Unfortunately, it is shown that this scheme is insecure under the known plaintext attack [BY88]. In fact, we have $p|\gcd(m-c_1, n)$ and $q|\gcd(m-c_2, n)$ when $\mathsf{Enc}(m) = (c_1, c_2)$. Later, Domingo-Ferrer proposed two variants of this scheme using additional secret key elements, but they are also broken under known plaintext attacks [Wag03, CKN06].

In this section, we revisit this particular scheme, and present a secure variant of it. To avoid known plaintext attacks to which previous variants were susceptable, we consider adding small random 'errors' to plaintexts, as in the recent fully homomorphic encryption schemes.

Basic Idea

We denote by $a \bmod p$ the unique integer in $(-\frac{p}{2}, \frac{p}{2}]$ that is congruent to a modulo p, and by $\mathrm{CRT}_{(p_0,\ldots,p_k)}(m_0, \ldots, m_k)$ the unique integer in $(-\frac{\prod_i p_i}{2}, \frac{\prod_i p_i}{2}]$ which is congruent to m_i modulo p_i for all i. Our basic symmetric encryption scheme is as follows:

- **KeyGen**$(\lambda, \{Q_i\})$: Given security parameter λ and relatively small pairwise coprime integers Q_i $(i = 1, \ldots, k)$, choose large pairwise coprime integers p_i $(i = 0, \ldots, k)$ and let $n = \prod_{i=0}^{k} p_i$. Output the secret key $sk = (p_0, \ldots, p_k)$ and the public parameter $pp = (n, Q_1, \ldots, Q_k)$. The message space is $\mathbb{Z}_Q$ for $Q = \prod_{i=1}^{k} Q_i$.

- **Enc**(sk, m): Output $c = \mathrm{CRT}_{(p_0,\ldots,p_k)}(e, m_1 + e_1 Q_1, \ldots, m_k + e_k Q_k)$ where $m_i = m \bmod Q_i$ for all i, e is a random integer in $(-p_0/2, p_0/2]$ and $e_1, \cdots, e_k$ are ρ-bit random integers.

- **Dec**(sk, c): Output

$$m = \mathrm{CRT}_{(Q_1,\ldots,Q_k)}(d_1, \ldots, d_k),$$

where $d_i = (c \bmod p_i) \bmod Q_i$ for all i.

Since the CRT is a ring isomorphism from $\prod_i \mathbb{Z}_{p_i}$ to $\mathbb{Z}_n$ with respect to modular addition and multiplication, Dec is also ring homomorphic. However, to ensure correct decryption of a ciphertext, the size of e_i and Q_i must be sufficiently smaller than that of p_i.

This scheme is a symmetric key encryption scheme which permits bounded number of modular additions and multiplications. We can convert this scheme to a somewhat homomorphic public key encryption scheme by publishing many encryptions of zero and encryptions of k elementary elements $E_i = \mathrm{CRT}_{(Q_1,\ldots,Q_i,\ldots,Q_k)}(0, \ldots, 1, \ldots, 0)$.

We reduce the security of our Somewhat Homomorphic Encryption (SWHE) scheme to a decisional version of Approximate GCD problem (DACD). Approximate GCD (ACD) problem is to find p given many multiples of p with some errors (i.e. $x_i = pq_i + e_i$). Note that the ACD assumption was used to prove the security of the DGHV scheme [DGHV10], and another decisional version of the approximate GCD assumption which is slightly different from ours was used to prove the security of a more efficient variant of DGHV by Coron et al. [CNT12].

In fact, our scheme can be regarded as a generalization of the DGHV scheme, but with larger plaintext space. Moreover, our scheme can be extended to a Fully Homomorphic Encryption (FHE) through bootstrapping and squashing the decryption circuit as in [Gen09, DGHV10], when $Q_1 = \cdots = Q_k = 2$ (see Section 2.4.1). In Section 2.4.2, we also show how we may

do the bootstrapping when Q_i's are sufficiently large.

Let λ be the security parameter. The ciphertext size of our SWHE scheme is $\tilde{O}(\lambda^5)$ as in the DGHV scheme. While the plaintext size of the DGHV is $O(\lambda)$, that of ours is $O(\lambda^4)$ for $O(\lambda)$-bit $Q_1, \ldots, Q_k$ with $k = O(\lambda^3)$. Consequently, our scheme reduces the overheads (ratio of ciphertext computation and plaintext computation) from $\tilde{O}(\lambda^4)$ to $\tilde{O}(\lambda)$. For the case that the message space is $\mathbb{Z}_2^k$, the overhead is reduced from $\tilde{O}(\lambda^8)$ to $\tilde{O}(\lambda^5)$ for $k = O(\lambda^3)$.

Our scheme has an advantage over [GHS12a] in applications requiring larger message space. When dealing with arithmetic on $\mathbb{Z}_Q$ for $\log Q = O(\lambda^4)$, our SWHE scheme can support $O(\lambda)$ multiplications with many additions. One of the important applications of homomorphic encryption schemes is to securely evaluate a multivariate polynomial over integers. Our scheme is an attractive choice for evaluating a polynomial of degree $O(\lambda)$ with inputs $\Omega(\lambda^2)$. Also our scheme can be used in the applications requiring SIMD style operations in k copies of $\mathbb{Z}_Q$ for $\log Q = \lambda, k = O(\lambda^3)$.

Related work

In 2009, Gentry [Gen09, Gen10] introduced the first fully homomorphic encryption scheme based on ideal lattices which supports arbitrarily many additions and multiplications on encrypted bits. His breakthrough paper drew an explosive interest and lead numerous researches in this area [DGHV10, CMNT11, CNT12, GH11b, SV10, SS10, SS11, GHS12a, BV11, Bra12]. Gentry's scheme and its variants [Gen09, Gen10, SV10, SS10] are based on hard problems on ideal lattices. Another class of schemes [DGHV10, CMNT11, CNT12] relies on the approximate GCD problem. The message space of these schemes is $\mathbb{Z}_2$, so the overhead is rather high due to the large ciphertext expansion ratio. Our scheme improves their efficiency. Recent schemes based on

the learning with error (LWE) or the ring-LWE are more efficient and accomplish polylogarithmic overhead for wide enough arithmetic circuits on $\mathbb{Z}_p$ for $p = poly(\lambda)$. For more information on related work, we refer to [GHS12b].

After a previous version of this work was made public on the IACR ePrint archive, Coron et al. proposed a scale-invariant fully homomorphic encryption scheme over integers [CLT14]. In the paper, they showed that the error-free decisional approximate-GCD assumption is equivalent to the error-free approximate GCD assumption, and this allows a reduction proof of the security of our scheme (and as well as other schemes extending DGHV) based only on the error-free approximate GCD assumption.

2.1 Preliminaries

Notation. We use $a \leftarrow A$ to denote the operation of choosing an element a from a set A uniform randomly. When $\mathcal{D}$ is a distribution, the notation $a \leftarrow \mathcal{D}$ means choosing an element a according to the distribution $\mathcal{D}$. We identify elements of $\mathbb{Z}_p$ with those of $\mathbb{Z} \cap (-\frac{p}{2}, \frac{p}{2}]$, and let $x \bmod p$ be the unique number in $\mathbb{Z} \cap (-\frac{p}{2}, \frac{p}{2}]$ which is congruent to x modulo p. Also, for $x \in \mathbb{Q}$, let $x \bmod p$ be the unique number in $\mathbb{Q} \cap (-\frac{p}{2}, \frac{p}{2}]$ whose difference with x is an integral multiple of p. We use the notation $(a_i)^k$ for a vector $(a_1, \ldots, a_k)$. So $\mathcal{D}_\rho(p_1, \ldots, p_k; Q_1, \ldots, Q_k; q_0)$ defined below can be shortened as $\mathcal{D}_\rho\big((p_i)^k; (Q_i)^k; q_0\big)$.

For pairwise coprime integers $p_1, \ldots, p_k$, we define $\mathrm{CRT}_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$ as the unique integer in $(-\frac{x_0}{2}, \frac{x_0}{2}]$ which is congruent to m_i modulo p_i for all $i = 1, \ldots, k$, where $x_0 = \prod_{i=1}^k p_i$. That is, $\mathrm{CRT}_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k) \equiv \sum_{i=1}^k m_i \hat{p}_i \left(\hat{p}_i{}^{-1} \bmod p_i\right) \bmod x_0$, where $\hat{p}_i = \frac{x_0}{p_i} = \frac{\prod_{j=1}^k p_j}{p_i}$.

For η-bit primes $p_1, \ldots, p_k$ and ℓ_Q-bit integers $Q_1, \ldots, Q_k$, we define the

following distributions:

$$\begin{aligned}
\mathcal{D}_{\gamma,\rho}(p) :=& \{\text{choose } q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p), e \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \\
& \text{output } x = pq + e\}, \\
\mathcal{D}_\rho(p_1, \ldots, p_k; q_0) :=& \Big\{\text{choose } e_0 \leftarrow \mathbb{Z} \cap [0, q_0), e_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho),\ \forall i \in \{1, \ldots, k\} : \\
& \text{output } x = \mathrm{CRT}_{(q_0, p_1, \ldots, p_k)}(e_0, \ldots, e_k)\Big\}, \\
\mathcal{D}_\rho(p_1, \ldots, p_k; Q_1, \ldots, Q_k; q_0) :=& \Big\{\text{choose } e_0 \leftarrow \mathbb{Z} \cap [0, q_0), e_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho),\ \forall i \in \{1, \ldots, k\} : \\
& \text{output } x = \mathrm{CRT}_{(q_0, p_1, \ldots, p_k)}(e_0, e_1 Q_1, \ldots, e_k Q_k)\Big\}.
\end{aligned}$$

Remark 2.1.1. When $k = 1$, $\mathcal{D}_\rho(p_1; q_0)$ is identical to $\mathcal{D} := \{\text{choose } q \leftarrow \mathbb{Z} \cap [0, q_0), e \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \text{output } x = p_1 q + e \bmod p_1 q_0\}$. For $x \leftarrow \mathcal{D}_\rho(p_1; q_0)$, we have

$$\begin{aligned}
x &= \mathrm{CRT}_{(q_0, p_1)}(e_0, e_1) \\
&= e_0 p_1 (p_1^{-1} \bmod q_0) + e_1 q_0 (q_0^{-1} \bmod p_1) \bmod q_0 p_1 \\
&= e_0 p_1 \alpha + e_1 (p_1 \beta + 1) \bmod q_0 p_1 \\
&= (e_0 \alpha + e_1 \beta) p_1 + e_1 \bmod q_0 p_1
\end{aligned}$$

for some α and β. If e_0 is chosen from $\mathbb{Z} \cap [0, q_0)$ uniformly, $(e_0\alpha + e_1\beta) \bmod q_0$ is uniform in $\mathbb{Z} \cap [0, q_0)$ when $\gcd(\alpha, q_0) = 1$.

There are two versions of the approximate GCD problem defined by Howgrave-Graham [HG01]. One is the general approximate GCD problem and the other is the partially approximate GCD problem:

General Approximate GCD problem. The (ρ, η, γ)-computational general approximate GCD problem is: for an η-bit prime p, given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$, find p.

Partially Approximate GCD problem. The (ρ, η, γ)-computational partially approximate GCD problem is: for an η-bit prime p, given a γ-bit integer $x_0 = pq_0$ and polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$, find p.

In this section, we use only partially approximate GCD problem, we omit the

term 'partially' throughout the section, and denote it by ACD. The ACD assumption is that the ACD problem is hard for any polynomial time attacker.

2.2 Our Somewhat Homomorphic Encryption Scheme

We propose a homomorphic encryption scheme supporting large integer arithmetic or SIMD operations. The message space is $\prod_{i=1}^{k} \mathbb{Z}_{Q_i}$. If $Q_1, \ldots, Q_k$ are pairwise coprime integers, this message space can be considered as $\mathbb{Z}_Q$ where $Q = \prod_{i=1}^{k} Q_i$. On the other hand, our scheme can support SIMD operations when all Q_i's are the same.

2.2.1 Parameters

Here we describe the parameters used by our scheme:

- λ : the security parameter
- ρ : the bit length of the error
- η : the bit length of the secret primes
- γ : the bit length of a ciphertext
- τ : the number of encryptions of zero in public key
- k : the number of distinct secret primes
- ℓ_Q : the bit length of Q_i for $i = 1, \ldots k$

Roughly speaking, k determines the size of the message space. The parameter ℓ_Q can be an integer from 2 to $\eta/8$ depending on the multiplicative

depth of the scheme. The detailed analysis is given in Section 2.2.3. Some necessary conditions for the choice of parameters of our scheme is as follows:

- $\gamma = \eta^2\omega(\log\lambda)$, to resist Cohn and Heninger's attack [CH11] and the attack using Lagarias algorithm [Lag85] on the approximate GCD problem.
- $\eta = \tilde{\Omega}(\lambda^2 + \rho\cdot\lambda)$, to resist the factoring attack using the elliptic curve method [Len87] and to permit enough multiplicative depth.
- $\rho = \tilde{\mathcal{O}}(\lambda)$, to be secure against Chen-Nguyen's attack [CN12b] and Howgrave-Graham's attack [HG01].
- $\tau = \gamma + \omega(\log\lambda)$, in order to use the leftover hash lemma in the security proof which is given in Section 2.3.

More concretely, we may choose $\gamma = \tilde{\mathcal{O}}(\lambda^5), \eta = \tilde{\mathcal{O}}(\lambda^2), \rho = 2\lambda$, $\tau = \gamma + \lambda$, which is similar to the DGHV's convenient parameter setting [DGHV10].

2.2.2 The Construction

In our construction, we denote $\mathrm{CRT}_{(q_0,p_1,\ldots,p_k)}$ by CRT.

- $\mathsf{KG}(\lambda,\rho,\eta,\gamma,\tau,\ell_Q,k,\{Q_i\})$: Given ℓ_Q-bit integers $Q_1,\ldots,Q_k$ together with other parameters, choose η-bit distinct primes $p_1,\ldots,p_k$ and $q_0 \leftarrow \mathbb{Z}\cap\left[0,\frac{2^\gamma}{\prod_{i=1}^k p_i}\right)$, and set $x_0 := q_0\prod_{i=1}^k p_i$. Repeat this until we have $\gcd(Q_i,x_0)=1$ for all $i=1,\ldots,k$. Output the public key pk as follows:

$$pk = \Big(x_0, \{Q_\ell\}_{\ell=1}^k,\ X := \big\{x_j = \mathrm{CRT}(e_{j0}, e_{j1}Q_1,\ldots,e_{jk}Q_k)\big\}_{j=1}^\tau,$$
$$Y := \big\{y_\ell = \mathrm{CRT}(e'_{\ell 0}, e'_{\ell 1}Q_1+\delta_{\ell 1},\ldots,e'_{\ell k}Q_k+\delta_{\ell k})\big\}_{\ell=1}^k\Big),$$

  where $e_{j0}, e'_{\ell 0} \leftarrow \mathbb{Z}\cap[0,q_0), e_{ji} \leftarrow \mathbb{Z}\cap(-2^\rho,2^\rho)$, $e'_{\ell i} \leftarrow \mathbb{Z}\cap(-2^\rho,2^\rho)$ for $i,\ell\in[1,k], j\in[1,\tau]$, and δ_{ij} is the Kronecker delta. Also, output the secret key $sk=(p_1,\ldots,p_k)$.

- $\mathsf{Enc}(pk, \mathbf{m})$: For any $\mathbf{m} = (m_1, \ldots, m_k)$ with $m_i \in \mathbb{Z}_{Q_i}$, output
$$c = \sum_{i=1}^{k} m_i y_i + \sum_{j \in S} x_j \bmod x_0 \tag{2.2.1}$$
where S is a random subset of $\{1, \ldots \tau\}$.

- $\mathsf{Dec}(sk, c)$: Output $(m_1, \ldots, m_k)$ where $m_i = (c \bmod p_i) \bmod Q_i$.

- $\mathsf{Eval}(pk, C, \mathbf{c} = (c_1, \ldots, c_t))$: Given a public key pk, a permitted circuit C with t inputs defined in Section 2.2.3 and a t-tuple of ciphertexts $\mathbf{c}$, output $C(c_1, \ldots, c_t)$ using Add and Mul gates given below.

- $\mathsf{Add}(pk, c_1, c_2)$: Output $c_1 + c_2 \bmod x_0$.

- $\mathsf{Mul}(pk, c_1, c_2)$: Output $c_1 \times c_2 \bmod x_0$.

Remark 2.2.1. $X = \{x_j\}_{j=1}^{\tau}$ is a set of encryptions of the zero vector, and y_ℓ is an encryption of the ℓ-th elementary vector E_ℓ in pk.

Remark 2.2.2. There are $(\tau + k)$ γ-bit integers and k ℓ_Q-bit integers in the public key pk. The public key size is $\tilde{\mathcal{O}}((\tau + k)\gamma + k\ell_Q) = \tilde{\mathcal{O}}(\lambda^{10})$ under the parameter setting in Section 2.2.1.

Remark 2.2.3. If $k = 1, Q_1 = 2$, then our scheme is essentially the same as a noise-free variant of the DGHV [DGHV10].

A ciphertext $c \leftarrow \mathsf{Enc}(pk, \mathbf{m})$ can be written in the form,

$$\begin{aligned}
c &= \sum_{\ell=1}^{k} m_\ell y_\ell + \sum_{j \in S} x_j \bmod x_0 \\
&= \mathrm{CRT}\left(\left(\sum_{\ell=1}^{k} e'_{\ell 0} m_\ell\right), \left(\sum_{\ell=1}^{k} e'_{\ell 1} m_\ell\right) Q_1 + m_1, \ldots, \left(\sum_{\ell=1}^{k} e'_{\ell k} m_\ell\right) Q_k + m_k\right) \\
&+ \mathrm{CRT}\left(\left(\sum_{j \in S} e_{j0}\right), \left(\sum_{j \in S} e_{j1}\right) Q_1, \ldots, \left(\sum_{j \in S} e_{jk}\right) Q_k\right) \\
&= \mathrm{CRT}(e_0, e_1 Q_1 + m_1, \ldots, e_k Q_k + m_k)
\end{aligned}$$

for some $e_0 \in \mathbb{Z} \cap [0, q_0), e_1, \ldots, e_k \in \mathbb{Z} \cap (-2^{\rho'}, 2^{\rho'})$, where $\rho' = \max\{\rho + \log k + \ell_Q, 2\rho + \log \tau\}$.

2.2.3 Correctness

We use the *integer circuits* with Add and Mul gates applied to integers rather than a bit. That is, boolean gates are replaced with integer operations. Now we show that the scheme is correct for any permitted circuit. At first, we define a *permitted circuit* similar to Gentry [Gen10].

Definition 2.2.1 (Permitted Circuit)**.** Let C be an integer circuit with t inputs. We say that C is a *permitted circuit*, if the output of C has absolute value at most $2^{\alpha(\eta-4)}$ whenever the absolute value of each t input is smaller than $2^{\alpha(\rho'+\ell_Q)}$ for any $\alpha \geq 1$.

We denote the set of permitted circuits as $\mathcal{C}_{\mathcal{E}}$. Now we show that our scheme is correct for $\mathcal{C}_{\mathcal{E}}$, that is

$$\mathsf{Dec}(sk, C(c_1, \ldots, c_t)) = C(\mathbf{m}_1, \ldots, \mathbf{m}_t)$$

where $C \in \mathcal{C}_{\mathcal{E}}$, $c_j \leftarrow \mathsf{Enc}(pk, \mathbf{m}_j)$ and $\mathbf{m}_j = (m_{j1}, \ldots, m_{jk})$ for $j = 1, \ldots, t$.

Lemma 2.2.1. *If $c \leftarrow \mathsf{Enc}(pk, \mathbf{m})$ for $\mathbf{m} \in \prod_{i=1}^{k} \mathbb{Z}_{Q_i}$, then $c = p_i a_i + b_i Q_i + m_i$ for some a_i, b_i with $|b_i Q_i + m_i| < 2^{(\rho'+\ell_Q)}$ for all $i = 1, \ldots, k$.*

Proof. The proof is straightforward. If $c \leftarrow \mathsf{Enc}(pk, \mathbf{m})$, then

$$\begin{aligned} c &= \mathrm{CRT}_{(q_0, p_1, \ldots, p_k)}(e_0, e_1 Q_1 + m_1, \ldots, e_k Q_k + m_k) \\ &= p_i a_i + e_i Q_i + m_i \end{aligned}$$

for some a_i and $|e_i Q_i + m_i| < 2^{\rho'+\ell_Q}$ for all $i = 1, \ldots, k$. □

Lemma 2.2.2. *Let $C \in \mathcal{C}_{\mathcal{E}}$ and $c_j \leftarrow \mathsf{Enc}(pk, \mathbf{m}_j)$, where $\mathbf{m}_j = (m_{j1}, \ldots, m_{jk})$ for $j = 1, \ldots, t$. Let $m'_i \leftarrow C(m_{1i}, \ldots, m_{ti})$ and $c \leftarrow \mathsf{Eval}(pk, C, c_1, \ldots, c_t)$. Then $c = p_i a_i + b_i Q_i + m'_i$ for some a_i, b_i with $|b_i Q_i + m'_i| < p_i/8$ for all $i = 1, \ldots, k$.*

Proof. Let f be the multivariate polynomial computed by C. Then

$$\begin{aligned} c \bmod p_i &= f\big(c_1, \ldots, c_t\big) \bmod p_i \\ &= f\big(c_1 \bmod p_i, \ldots, c_t \bmod p_i\big) \bmod p_i. \end{aligned}$$

Since $C \in \mathcal{C}_{\mathcal{E}}$ and $|c_j \bmod p_i| < 2^{\rho'+\ell_Q}$ for all $j = 1, \ldots, t$ by Lemma 2.2.1,

$$\Big| f\big(c_1 \bmod p_i, \ldots, c_t \bmod p_i\big) \Big| < 2^{\eta-4} < p_i/8$$

for all $i = 1, \ldots, k$. Thus $c \bmod p_i = f\big(c_1 \bmod p_i, \ldots, c_t \bmod p_i\big)$. Also,

$$\begin{aligned} (c \bmod p_i) \bmod Q_i &= f\big(c_1 \bmod p_i, \ldots, c_t \bmod p_i\big) \bmod Q_i \\ &= f\Big(\big(c_1 \bmod p_i\big) \bmod Q_i, \ldots, \big(c_t \bmod p_i\big) \bmod Q_i\Big) \bmod Q_i \\ &= f(m_{1i}, \ldots, m_{ti}) \bmod Q_i \\ &= m'_i \bmod Q_i \end{aligned}$$

for all $i = 1, \ldots, k$. □

From Lemmas 2.2.1 and 2.2.2, the correctness follows.

Theorem 2.2.1 (Correctness)**.** *The scheme given in section 2.2.2 is correct for $\mathcal{C}_{\mathcal{E}}$.*

Each noise of $c_1 + c_2$ is increased by at most one bit. But the bit length of each noise for $c_1 \times c_2$ becomes about $2\rho' + 2\ell_Q$ which is two times larger than that of the original ciphertext. As the noise growth by multiplication is more significant than by addition, we focus on the multiplicative depth of a permitted circuit. The following is a simple lemma on permitted circuits.

Lemma 2.2.3. *Let C be an integer circuit and f be the multivariate polynomial computed by C. If $|\overrightarrow{f}| \cdot (2^{\rho'+\ell_Q})^d < 2^{\eta-4}$, then $C \in \mathcal{C}_\mathcal{E}$, where $|\overrightarrow{f}|$ is the ℓ_1 norm of the coefficient vector of f and $d = \deg f$.*

From the above condition, we have

$$d < \frac{\eta - 4 - \log_2 |\overrightarrow{f}|}{\rho' + \ell_Q}$$

which is similar to the DGHV [DGHV10]. Since we want to support polynomials of degree λ, we choose $\eta \geq \rho' \cdot \Theta(\lambda)$, assuming $\log_2 |\overrightarrow{f}|$ is relatively small compared to η, ρ'.

2.3 Security

In this section, we prove the security of our scheme. The security of the DGHV scheme is based on the ACD assumption defined in Section 3.1. The security of our scheme is based on a modified DACD (Decisional Approximate GCD) assumption which says that, for given a distribution $\mathcal{D} = \mathcal{D}_\rho(p; q_0)$ and some integer z, it is hard to determine whether z is chosen from $\mathcal{D}$ or not. Very recently, it is shown that this assumption is equivalent to the ACD assumption [CLT14]. Therefore, we select the parameters of our scheme based on the known attacks on the ACD problem [HG01, CH11, CN12b].

To prove the semantic security of our scheme, we introduce another decisional version of approximate GCD problem.

Definition 2.3.1 (Decisional Approximate GCD Problem: DACD)**.** The (ρ, η, γ)-decisional approximate GCD problem is: for an η-bit prime p, given a γ-bit integer $x_0 = pq_0$ and polynomially many samples from $\mathcal{D}_\rho(p; q_0)$, determine $b \in \{0, 1\}$ from $z = x + r \cdot b \bmod x_0$ where $x \leftarrow \mathcal{D}_\rho(p; q_0)$ and $r \leftarrow \mathbb{Z} \cap [0, x_0)$.

We assume that DACD problem is hard for any polynomial time distinguisher. In the following, we introduce new problems that have a role bridging the gap between DACD problem and our scheme. Overall, our scheme is semantically secure based on the DACD assumption.

Definition 2.3.2. (Decisional Approximate GCD_Q Problem: DACD_Q) The $(\rho, \eta, \gamma, l_Q)$-decisional approximate GCD_Q problem is: for an η-bit prime p and a ℓ_Q-bit integer Q, given a γ-bit integer $x_0 = pq_0$ with $\gcd(x_0, Q) = 1$, and polynomially many samples from $\mathcal{D}_\rho(p; Q; q_0)$, determine $b \in \{0, 1\}$ from $z = x + r \cdot b \bmod x_0$ where $x \leftarrow \mathcal{D}_\rho(p; Q; q_0)$ and $r \leftarrow \mathbb{Z} \cap [0, x_0)$.

Definition 2.3.3. (k-Decisional Approximate GCD_Q Problem: k-DACD_Q) The $(\rho, \eta, \gamma, l_Q)$-$k$-decisional approximate GCD_Q problem is : for η-bit distinct primes $p_1, \ldots, p_k$ and ℓ_Q-bit integers $Q_1, \ldots, Q_k$, given a γ-bit integer $x_0 := q_0 p_1 \cdots p_k$, with $\gcd(x_0, Q_i) = 1$ for $i = 1, \ldots, k$, and polynomially many samples from $\mathcal{D} := \mathcal{D}_\rho\big((p_i)^k; (Q_i)^k; q_0\big)$ and a set $Y := \{y_\ell = \mathrm{CRT}_{(q_0, p_1, \ldots, p_k)}(e_{\ell 0}, e_{\ell 1} Q_1 + \delta_{\ell 1}, \ldots, e_{\ell k} Q_k + \delta_{\ell k}) \big| e_{\ell 0} \leftarrow \mathbb{Z}_{q_0}, e_{\ell i} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ for $\ell, i \in \{1, \ldots, k\}\}$, determine $b \in \{0, 1\}$ from $z = x + r \cdot b \bmod x_0$ where $x \leftarrow \mathcal{D}$ and $r \leftarrow \mathbb{Z} \cap [0, x_0)$.

We say that the DACD assumption holds if no polynomial time distinguisher can solve the DACD problem with non-negligible advantage. The k-DACD_Q assumption is defined similarly.

Now we show that our somewhat homomorphic encryption scheme is semantically secure under the DACD assumption. This is done in three steps. In the following, arrows indicate polynomial time reductions.

Step 1: (ρ, η, γ)-DACD $\longrightarrow$ $(\rho, \eta, \gamma, l_Q)$-$\mathrm{DACD}_Q$ (Lemma 2.3.1)

Step 2: $(\rho, \eta, \gamma, l_Q)$-$\mathrm{DACD}_Q$ $\longrightarrow$ $(\rho, \eta, \gamma + (k-1)\eta, l_Q)$-$k$-$\mathrm{DACD}_Q$ (Lemma 2.3.2)

Step 3: $(\rho, \eta, \gamma + (k-1)\eta, l_Q)$-$k$-DACD$_Q$ $\longrightarrow$ our scheme (Theorem 2.3.1)

The first step is rather easily done by multiplying Q due to the knowledge of the exact multiple of p. In the second step, DACD$_Q$ problem with $x_0 = q_0 p_1$ is converted to k-DACD$_Q$ problem by choosing additional $k-1$ primes $p_2, \ldots, p_k$ and computing necessary terms including $x'_0 = q_0 \prod_{i=1}^{k} p_i$. In the proof, we use a hybrid argument and lose a factor of k in the success probability. Finally, the last step is done by interpreting the input of k-DACD$_Q$ problem as a public key of the scheme.

Lemma 2.3.1. *The* (ρ, η, γ)-DACD *problem is reducible to the* $(\rho, \eta, \gamma, l_Q)$-DACD$_Q$ *problem.*

Proof. Suppose a polynomial time distinguisher $\mathcal{B}$ solves the $(\rho, \eta, \gamma, l_Q)$-DACD$_Q$ problem with an advantage ϵ. We construct a polynomial time distinguisher $\mathcal{A}$ that solves the (ρ, η, γ)-DACD problem with the same advantage. Suppose $\mathcal{A}$ is given γ-bit integer $x_0 = pq_0, z = x + r \cdot b$, and polynomially many samples $X = \{x_i \mid x_i \leftarrow \mathcal{D}_\rho(p; q_0) \text{ for } i = 1, \ldots, \tau\}$. $\mathcal{A}$ works as follows:

1. Choose a ℓ_Q-bit integer Q such that $\gcd(x_0, Q) = 1$.
2. Construct samples $X' := \{x \cdot Q \bmod x_0 \mid x \in X\}$ and $z' := z \cdot Q \bmod x_0$.
3. Give (x_0, Q, X', z') to $\mathcal{B}$.
4. Output b' where b' is $\mathcal{B}$'s answer.

We verify that the statistical distance of $\mathcal{D}' = \{x \leftarrow \mathcal{D}_\rho(p; q_0) : \text{Output } y = x \cdot Q \bmod x_0\}$ and $\mathcal{D}_\rho(p; Q; q_0)$ is negligible when $\gcd(x_0, Q) = 1$. Consider a map $\phi_Q : \mathbb{Z}_{q_0} \to \mathbb{Z}_{q_0}$ defined by $x \mapsto x \cdot Q$. Since $\gcd(x_0, Q) = 1$, ϕ_Q is a ring isomorphism and so $\Delta(\mathcal{D}', \mathcal{D}_\rho(p; Q; q_0)) = 0$. It is easy to see

that z' is uniform in $\mathbb{Z} \cap [0, x_0)$ when z is randomly chosen in $\mathbb{Z} \cap [0, x_0)$. Hence in this case, $\Pr[\mathcal{A}(\mathcal{D}_\rho(p; q_0), z) = 1] = \Pr[\mathcal{B}(\mathcal{D}', z') = 1]$. On the other hand, if z is randomly chosen in $\mathcal{D}_\rho(p; q_0)$, then z' is uniform in $\mathcal{D}'$ and so $\Pr[\mathcal{A}(\mathcal{D}_\rho(p; q_0), z) = 1] = \Pr[\mathcal{B}(\mathcal{D}', z') = 1]$. Thus

$$\mathbf{Adv}(\mathcal{A}) = \Big|\Pr[\mathcal{A}(\mathcal{D}_\rho(p; q_0), z_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}_\rho(p; q_0), z_2) = 1]\Big| = \epsilon$$

by the definition of algorithm $\mathcal{B}$ and the fact $\Delta(\mathcal{D}', \mathcal{D}_\rho(p; Q; q_0)) = 0$ where $z_1 \leftarrow \mathcal{D}_\rho(p; q_0)$ and $z_2 \leftarrow \mathbb{Z} \cap [0, x_0)$. □

Lemma 2.3.2. *The $(\rho, \eta, \gamma_1, l_Q)$-DACD$_Q$ problem is reducible to the $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ problem with the advantage of the latter k times that of the former on average.*

Proof. Suppose a polynomial time distinguisher $\mathcal{B}$ solves the $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ problem. We construct a polynomial time distinguisher $\mathcal{A}$ that solves the $(\rho, \eta, \gamma_1, l_Q)$-DACD$_Q$ problem.

For $x_0 = q_0 p_1$ and $x_0' = q_0 \prod_{i=1}^{k} p_i$, we define $\mathcal{D}_i := \mathcal{D}_\rho\big((p_j)^i; (Q_j)^i; q_0 \prod_{j=i+1}^{k} p_j\big)$ and $\mathcal{D}_0 := \mathbb{Z} \cap [0, x_0')$. Note that the support* of $\mathcal{D}_i$ is included in the support of D_{i-1} for $i \in \{1, \ldots, k\}$. Suppose $\mathcal{B}$ can distinguish z between $\mathcal{D}_0$ and $\mathcal{D}_k$ with advantage ϵ. Then by the standard hybrid argument, $\mathcal{B}$ should distinguish z between $\mathcal{D}_i$ and $\mathcal{D}_{i-1}$ for some $i \in \{1, \ldots, k\}$ with advantage at least ϵ/k. Let us denote this index as i_0.

Let the input of the distinguisher $\mathcal{A}$ be an integer $x_0 = q_0 p_1, Q_1$, polynomially many samples x_i from $\mathcal{D}$ and $z = x + r \cdot b$ where $\mathcal{D} := \mathcal{D}_\rho(p_1; Q_1; q_0), x \leftarrow \mathcal{D}, r \leftarrow \mathbb{Z} \cap [0, x_0)$ and $b \in \{0, 1\}$. We define a set $\mathcal{I}_1 := (x_0, Q_1, \{x_i\}_{i=1}^{\tau})$. Using input $(\mathcal{I}_1, z)$, $\mathcal{A}$ constructs an input $(\mathcal{I}_2, z')$ which will be given to the distinguisher $\mathcal{B}$ as follows:

*The support of a distribution is a set of elements having non-zero probability in the distribution.

1. Choose ℓ_Q-bit integers $Q_2, \ldots, Q_k$, η-bit distinct primes $p_2, \ldots, p_k$ such that $\gcd(Q_i, x_0) = \gcd(p_i, x_0) = 1$ for $i \in \{2, \ldots, k\}$.

2. Let $x'_0 = x_0 \cdot \prod_{i=2}^{k} p_i = q_0 \prod_{i=1}^{k} p_i$.

3. For each sample x_i from $\mathcal{D}$, choose $e_{ij} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ for $j \in \{2, \ldots, k\}$, and construct a sample from the distribution $\mathcal{D}' := \mathcal{D}_\rho\big((p_i)^k; (Q_i)^k; q_0\big)$ by $x'_i = \mathrm{CRT}_{(x_0, p_2, \ldots, p_k)}(x_i, e_{i2}Q_2, \ldots, e_{ik}Q_k)$.

4. To make a set Y, choose $e'_{\ell j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$, $s_\ell \leftarrow \mathcal{D}$ for $\ell \in [1, k], j \in [2, k]$ and construct $y'_\ell = \mathrm{CRT}_{(x_0, p_2, \ldots, p_k)}(s_\ell + \delta_{\ell 1}, e'_{\ell 2}Q_2 + \delta_{\ell 2}, \ldots, e'_{\ell k}Q_k + \delta_{\ell k})$.

5. For $z = x + r \cdot b$, let $z' = \mathrm{CRT}_{(x_0, p_2, \ldots, p_k)}(z, e'_2 Q_2, \ldots, e'_{i_0} Q_{i_0}, e'_{i_0+1}, \ldots, e'_k)$ where $e'_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ for $i \in \{2, \ldots, i_0\}$ and $e'_i \leftarrow \mathbb{Z} \cap [0, p_i)$ for $i \in \{i_0 + 1, \ldots, k\}$.

In Step 3 and 4, since

$$\begin{aligned}
x'_i &= \mathrm{CRT}_{(x_0, p_2, \ldots, p_k)}(x_i, e_{i2}Q_2, \ldots, e_{ik}Q_k) \\
&= \mathrm{CRT}_{(q_0, p_1, p_2, \ldots, p_k)}(e_{i0}, e_{i1}Q_1, e_{i2}Q_2, \ldots, e_{ik}Q_k) \\
y'_\ell &= \mathrm{CRT}_{(x_0, p_2, \ldots, p_k)}(s_\ell + \delta_{\ell 1}, e'_{\ell 2}Q_2 + \delta_{\ell 2}, \ldots, e'_{\ell k}Q_k + \delta_{\ell k}) \\
&= \mathrm{CRT}_{(q_0, p_1, p_2, \ldots, p_k)}(e'_{\ell 0}, e'_{\ell 1}Q_1 + \delta_{\ell 1}, e'_{\ell 2}Q_2 + \delta_{\ell 2}, \ldots, e'_{\ell k}Q_k + \delta_{\ell k})
\end{aligned}$$

for some $e_{i0}, e'_{\ell 0} \in \mathbb{Z} \cap [0, q_0), e_{i1}, e'_{\ell 1} \in \mathbb{Z} \cap (-2^\rho, 2^\rho)$ for $i \in [1, \tau], \ell \in [1, k]$, the set Y given to $\mathcal{B}$ is suitable. The distinguisher $\mathcal{A}$ gives these input $\mathcal{I}_2 = (x'_0, \{Q_i\}_{i=1}^k, \{x'_i\}_{i=1}^\tau, \{y'_\ell\}_{\ell=1}^k)$ and z' to $\mathcal{B}$, and use $\mathcal{B}$'s answer to its answer. Interchanging p_1 and p_{i_0}, we know that z is sampled from D_{i_0} or $D_{i_0 - 1}$. This can be distinguished by $\mathcal{B}$ with advantage ϵ/k, and thus $\mathcal{A}$'s advantage is at least ϵ/k. □

To complete the proof of the semantic security of our scheme, we need two more lemmas. Lemma 2.3.3 shows that the distribution of fake public key is indistinguishable from that of the correct public key. Lemma 2.3.4 implies that an encryption from $\mathcal{A}$ is correct form for the scheme.

Lemma 2.3.3. *For the parameters $(\lambda, \rho, \eta, \gamma, \tau, l_Q, k)$, let $pk = (x_0, \{Q_i\}_{i=1}^k, \{x_j\}_{j=1}^\tau, \{y_\ell\}_{\ell=1}^k)$ and $sk = (p_1, \ldots, p_k)$ be chosen as in the* **KeyGen** *of our scheme. And let us choose x'_j uniformly from $\mathbb{Z}_{x_0} = \mathbb{Z} \cap \left(-\frac{x_0}{2}, \frac{x_0}{2}\right]$ for $j = 1, \ldots, \tau$. Then, pk and pk' are computationally indistinguishable if we define pk' as $(x_0, \{Q_i\}_{i=1}^k, \{x'_j\}_{j=1}^\tau, \{y_\ell\}_{\ell=1}^k)$, under the $(\rho, \eta, \gamma, l_Q)$-k-DACD$_Q$ assumption.*

Proof. Suppose a polynomial time distinguisher $\mathcal{B}$ may distinguish pk from pk' with advantage ϵ. Using $\mathcal{B}$, we construct a polynomial time distinguisher $\mathcal{A}$ that solves the $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ problem. Note that the distinguisher $\mathcal{A}$ has access to the oracle $\mathcal{D} = \mathcal{D}_\rho\big((p_i)^k; (Q_i)^k; q_0\big)$. For $r = 0, \ldots, \tau$, define $pk^{(r)} = (x_0, \{Q_i\}_{i=1}^k, \{x_j^{(r)}\}_{j=1}^\tau, \{y_\ell\}_{\ell=1}^k)$ by setting $x_1^{(r)}, \ldots, x_r^{(r)} \leftarrow \mathbb{Z}_{x_0}$ and $x_{r+1}^{(r)}, \ldots, x_\tau^{(r)} \leftarrow \mathcal{D}$. We see that $pk^{(0)}$ has the same distribution as pk, and $pk^{(\tau)}$ has the same distribution as pk'. For $r = 1, \ldots, \tau$, we define

$$pr_r := \Pr[\mathcal{B}(pk^{(r-1)}) = 1] - \Pr[\mathcal{B}(pk^{(r)}) = 1].$$

(Note that in the above formula we omitted other information $\mathcal{B}$ has: $\lambda, \rho, \eta, \gamma, \tau, l_Q, k$.)

Now we are ready to fully define the algorithm $\mathcal{A}$. It has given a number z, which either is from $\mathcal{D}$, or is uniformly random on $\mathbb{Z}_{x_0}$. The algorithm $\mathcal{A}$ first picks r randomly from $\{1, \ldots, \tau\}$, and selects x_j^* $(j = 1, \ldots, \tau)$ as follows: $x_1^*, \ldots, x_{r-1}^* \leftarrow \mathbb{Z}_{x_0}$, $x_{r+1}^*, \ldots, x_\tau^* \leftarrow \mathcal{D}$ and $x_r^* := z$. Then $\mathcal{A}$ runs $\mathcal{B}$ with input $pk^* := (x_0, \{Q_i\}_{i=1}^k, \{x_j^*\}_{j=1}^\tau, \{y_\ell\}_{\ell=1}^k)$, and echoes the output of $\mathcal{B}$ as its own answer. Clearly, if z is chosen from $\mathcal{D}$, then pk^* has the same

distribution as $pk^{(r-1)}$, and if z is chosen uniformly from $\mathbb{Z}_{x_0}$, then pk^* has the same distribution as $pk^{(r)}$. Now, if z is from $\mathcal{D}$, we have

$$\Pr[\mathcal{A}(z)=1] = \frac{1}{\tau}\sum_{r=1}^{\tau}\Pr[\mathcal{B}(pk^{(r-1)})=1],$$

and if z is from $\mathbb{Z}_{x_0}$, then

$$\Pr[\mathcal{A}(z)=1] = \frac{1}{\tau}\sum_{r=1}^{\tau}\Pr[\mathcal{B}(pk^{(r)})=1].$$

(Again we omit from the notation other information $\mathcal{A}$ has other than z.)

The difference between the two probabilities is equal to

$$\begin{aligned}
&\frac{1}{\tau}\sum_{r=1}^{\tau}\Pr[\mathcal{B}(pk^{(r-1)})=1] - \frac{1}{\tau}\sum_{r=1}^{\tau}\Pr[\mathcal{B}(pk^{(r)})=1] \\
&\quad = \frac{1}{\tau}\left(\Pr[\mathcal{B}(pk^{(0)})=1] - \Pr[\mathcal{B}(pk^{(\tau)})=1]\right) \\
&\quad = \frac{1}{\tau}\left(\Pr[\mathcal{B}(pk)=1] - \Pr[\mathcal{B}(pk')=1]\right) \\
&\quad = \epsilon/\tau.
\end{aligned}$$

Therefore, in this case $\mathcal{A}$ is a distinguisher which solves the (ρ,η,γ,l_Q)-k-DACD$_Q$ problem with advantage ϵ/τ. Under the (ρ,η,γ,l_Q)-k-DACD$_Q$ assumption, we conclude that the distinguisher $\mathcal{B}$ cannot exist. □

Lemma 2.3.4. *For the parameters $(\lambda,\rho,\eta,\gamma,\tau,l_Q,k)$, let $pk=(x_0,\{Q_i\}_{i=1}^k,\{x_j\}_{j=1}^{\tau},\{y_\ell\}_{\ell=1}^k)$ and $sk=(p_1,\ldots,p_k)$ be chosen as in the* **KeyGen** *of our scheme. Let $\mathbf{m}=(m_1,\ldots,m_k)$ where $m_i\in\mathbb{Z}_{Q_i}$. For every $z\in\mathcal{D}_\rho\big((p_i)^k;(Q_i)^k;q_0\big)$, the following distribution*

$$\mathcal{C}_{pk,z}(\mathbf{m}) = \left\{S\subset_R\{1,\ldots,\tau\} : \text{Output}\leftarrow\sum_{i=1}^{k}m_iy_i+\sum_{j\in S}x_j+z \bmod x_0\right\}$$

is computationally close to the distribution $\mathbf{Enc}(pk,\mathbf{m})$*, under the (ρ,η,γ,l_Q)-k-DACD$_Q$ assumption.*

Proof. Since we are making $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ assumption, according to Lemma 2.3.3, instead of normally chosen public key $pk = (x_0, \{Q_i\}_{i=1}^k, \{x_j\}_{j=1}^\tau, \{y_\ell\}_{\ell=1}^k)$, we may use $pk' = (x_0, \{Q_i\}_{i=1}^k, \{x'_j\}_{j=1}^\tau, \{y_\ell\}_{\ell=1}^k)$ with x'_j chosen uniform randomly from $\mathbb{Z}_{x_0}$, since both are computationally indistinguishable from each other.

Hence, we need only to compare $\mathcal{C}_{pk,z}(\mathbf{m})$ and $\mathbf{Enc}(pk, \mathbf{m})$ under the 'false' public key pk'. The output of $\mathcal{C}_{pk,z}(\mathbf{m})$ is $c' = \sum_{i=1}^k m_i y_i + \sum_{j\in S} x'_j + z \bmod x_0$, and the output of $\mathbf{Enc}(pk, \mathbf{m})$ is $c = \sum_{i=1}^k m_i y_i + \sum_{j\in S} x'_j \bmod x_0$. But since x'_j are uniformly chosen modulo x_0, we may use the Leftover Hash Lemma, more specifically Lemma 1 from [DGHV10], to conclude that the distribution of $\sum_{j\in S} x'_j$ is statistically indistinguishable from uniform random distribution on $\mathbb{Z}_{x_0}$, hence both distributions, c' and c, are uniform random on $\mathbb{Z}_{x_0}$. Switched to the correct public key, this implies that two distributions are computationally indistinguishable. □

Now we prove the semantic security of our scheme.

Theorem 2.3.1. *The cryptosystem given in section 2.2 is semantically secure, if the $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ assumption holds.*

Proof. Suppose a polynomial time algorithm $\mathcal{B}$ breaks the semantic security of the scheme with non-negligible advantage. We construct a polynomial time algorithm $\mathcal{A}$ that solves the $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ problem with non-negligible advantage. For η-bit distinct primes $p_1, \ldots, p_k$ and ℓ_Q-bit integers $Q_1, \ldots, Q_k$, the input of $\mathcal{A}$ is $(x_0, (Q_i)^k, \mathcal{D}_\rho((p_i)^k; (Q_i)^k; q_0), Y, z)$ where $x_0 = q_0 \prod_{i=1}^k p_i$ is a γ-bit integer. The algorithm $\mathcal{A}$ works as follows:

1. $\mathcal{A}$ gives tuples $(x_0, (Q_i)^k, X := \{x_j \leftarrow \mathcal{D}_\rho((p_i)^k; (Q_i)^k; q_0)\}_{j=1}^\tau,\ Y := \{y_1, \ldots, y_k\})$ to $\mathcal{B}$ as the public key.

2. $\mathcal{B}$ chooses $\{\mathbf{m}_0 = (m_{01}, \ldots, m_{0k}), \mathbf{m}_1 = (m_{11}, \ldots, m_{1k})\}$ and sends it to $\mathcal{A}$.

3. $\mathcal{A}$ computes $c' = \sum_{\ell=1}^{k} m_{bi} y_\ell + \sum_{j \in J} x_j + z \bmod x_0$ for randomly chosen $b \in \{0, 1\}$ where $J \subset \{1, \ldots, \tau\}$ is a random subset, and give c' to $\mathcal{B}$.

4. $\mathcal{B}$ outputs $b' \in \{0, 1\}$.

5. If $b = b'$, then $\mathcal{A}$ outputs 0. Otherwise, outputs 1.

We see that the public key given to $\mathcal{B}$ is correctly formed and distributed. It is easy to see that c' is uniform in $\mathbb{Z}_{x_0}$ when z is randomly chosen in $\mathbb{Z}_{x_0}$. Hence in this case, the advantage of $\mathcal{A}$ is zero since c' does not reveal any information of $\mathbf{m}_b$ and $\mathcal{B}$'s probability of correct guessing is exactly $1/2$. On the other hand, if z is randomly chosen in $\mathcal{D}_\rho((p_i)^k; (Q_i)^k; q_0)$, then c' is computationally indistinguishable from the correct encryption of $\mathbf{m}_b$ by Lemma 2.3.4 when we choose τ larger than $\gamma + \omega(\log \lambda)$. Thus, in this case, the probability of correct answer for $\mathcal{B}$ is at most negligibly different from that of $\mathcal{B}$. This shows that the advantage of $\mathcal{A}$ is non-negligible, violating the $(\rho, \eta, \gamma, l_Q)$-$k$-DACD$_Q$ assumption. Therefore, the cryptosystem given is section 2.2 is semantically secure. □

2.4 Fully Homomorphic Encryption

The bottleneck of bootstrapping in our construction is to compute $[\cdot]_p \bmod Q$ homomorphically. When all Q_i's are equal to two, our homomorphic encryption can be converted to a fully homomorphic encryption via Gentry's squashing technique based on the sparse subset sum assumption, as is done in DGHV [DGHV10]. However, the same method is difficult to be generalized, when at least one of Q_i's is larger than two.

In this section, we propose an efficient method to evaluate $[\cdot]_p \bmod Q$ homomorphically for sufficiently large Q using Gentry and Halevi's technique [GH11a]. They propose a new method to construct FHE without squashing, called *chimeric* FHE. The chimeric FHE uses a multiplicative homomorphic encryption (MHE) to bootstrap a SHE without squashing, thereby removinb the assumption on the hardness of the sparse subset sum problem. The idea is to express the decryption circuit of the SHE scheme as a depth-3 ($\sum\prod\sum$) arithmetic circuit. Then temporarily switch to a ciphertext under a MHE, such as ElGamal, to compute $\prod$ part. And then homomorphically evaluate the decryption circuit of MHE to get a ciphertext under SHE. Using this method, SHE only needs to evaluate MHE's decryption circuit, which is of fixed degree, not its own decryption circuit.

2.4.1 Bit Message Space

Our homomorphic encryption can be converted to a fully homomorphic encryption via Gentry's squashing technique as is done in DGHV [DGHV10] when all Q_i's are equal to two, based on the sparse subset sum assumption. In this case, the message space is $\mathbb{Z}_2^k$. Recall that the decryption of message component is done by $m_i \leftarrow (c \bmod p_i) \bmod Q_i = (c \bmod p_i) \bmod 2$. Our sparse subset consists of y_j's such that

$$\frac{1}{p_i} \approx \sum_{j=1}^{\Theta} s_{ij} \cdot y_j,$$

and the secret key s_{ij}'s are included in the public key in an encrypted form $\mathsf{Enc}(pk, \mathbf{s}_j)$, where $\mathbf{s}_j = (s_{1j}, s_{2j}, \ldots, s_{kj})$ for $j = 1, \ldots, \Theta$. Using the same subset $\{y_j\}_{j=1}^{\Theta}$ for every secret prime p_i, parallel computation is possible. In this way, we can lower the multiplicative depth of the decryption to be computed homomorphically.

The remaining question is whether this can be done when Q_i is larger than two. It is unclear, since computing Q_i-ary addition seems to require more complex carry computations than binary addition.

2.4.2 Large Message Space

We describe a bootstrapping method when the message space is $\prod Q_i$ for sufficiently large Q_i's using Gentry and Halevi's technique [GH11a]. If one can evaluate each $[\cdot]_{p_i} \bmod Q_i$ homomorphically, we can bootstrap a ciphertext $c = \mathrm{CRT}_{(q_0,p_1,\ldots,p_k)}(e_0, e_1Q_1 + m_1, \ldots, e_kQ_k + m_k)$ using componentwise evaluation.

At first, we describe a method to evaluate $[\cdot]_{p_i} \bmod Q_i$. We denote p_i, Q_i by p and Q respectively for simplicity. We assume that p is congruent to 1 modulo Q. Then $[c]_p \bmod Q$ can be written in the form as in DGHV [DGHV10],

$$[c]_p \bmod Q = c - \lfloor c/p \rceil \cdot p \bmod Q = c - \lfloor c/p \rceil \bmod Q.$$

In comparison with the DGHV scheme, it is hard to express the division by p by a low degree polynomial over $\mathbb{Z}_Q$ when Q is larger than two. To apply the technique in [GH11a], we first modify the division part using the Gentry's squashing technique [Gen09]. Let us consider two more parameters κ, Θ. We set $\kappa = \gamma\eta/\rho'$ and $\Theta = \omega(\kappa \cdot \log \lambda)$. Given the secret key p and the public parameter Q, set $x_p = \lfloor 2^\kappa/p \rceil$ and choose Θ-bit random binary vector $\mathbf{s} = (s_1, \ldots, s_\Theta)$. Choose random integer $u_i \in \mathbb{Z} \cap [0, Q \cdot 2^\kappa)$ for $i = 1, \ldots, \Theta$ such that $\sum_i s_i u_i = x_p \pmod{Q \cdot 2^\kappa}$. Set $y_i = u_i/2^\kappa$ which is smaller than Q with κ precision after binary point. Also $[\sum_i s_i y_i]_Q = (1/p) - \Delta_p$ for some $|\Delta_p| < 2^{-\kappa}$. To bootstrap a ciphertext c output by a permitted circuit, we firstly compute $z_i \leftarrow [c \cdot y_i]$, keeping only $n = \lceil \log_2 \Theta \rceil + 3$ precision after binary point for $i = 1, \ldots, \Theta$. That is, $[c \cdot y_i]_Q = z_i - \Delta_i$ for some Δ_i with

$|\Delta_i| \leq 1/16\Theta$. We have

$$\begin{aligned}\left[(c/p) - \sum_{i=1}^{\Theta} s_i z_i\right]_Q &= \left[(c/p) - \sum_{i=1}^{\Theta} s_i [c \cdot y_i]_Q - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_Q \\ &= \left[(c/p) - c\left[\sum_{i=1}^{\Theta} s_i \cdot y_i\right]_Q - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_Q \\ &= \left[(c/p) - c \cdot (1/p - \Delta_p) - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_Q \\ &= \left[c \cdot \Delta_p - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_Q\end{aligned}$$

Since the ciphertext c is output by the permitted circuit, the bit length of c is at most $2^{\gamma(\eta-4)/(\rho'+2)} < 2^{\kappa-4}$, thus $c \cdot \Delta_p \leq 1/16$. Also we observe that $|\sum s_i \Delta_i| \leq \Theta \cdot 1/16\Theta = 1/16$. Thus we have

$$[c]_p \bmod Q = c - \lfloor c/p \rceil \bmod Q = c - \left\lfloor \sum s_i z_i \right\rceil \bmod Q. \tag{2.4.2}$$

To apply chimeric technique [GH11a], we convert the above subset sum into a $\sum\prod\sum$ form, defined below.

Definition 2.4.1 (Restricted Depth-3 Circuit)**.** Let $\mathcal{L} = \{L_j(x_1, \ldots, x_n)\}$ be a set of polynomials, all in the same n variables. An arithmetic circuit C is an $\mathcal{L}$-restricted depth-3 circuit over $\mathbf{x} := (x_1, \ldots, x_n)$ if there exists multisets $S_1, \ldots, S_t \subset \mathcal{L}$ and constants $\lambda_0, \lambda_1, \ldots, \lambda_t$ such that

$$C(\mathbf{x}) = \lambda_0 + \sum_{i=1}^{t} \lambda_i \cdot \prod_{L_j \in S_i} L_j(x_1, \ldots, x_n).$$

The degree of C with respect to $\mathcal{L}$ is $d = \max_i |S_i|$.

The equation (3.3.1) can be converted as follows:

$$\begin{aligned}[c]_p \bmod Q &\equiv c - \lfloor c/p \rfloor \bmod Q \\ &\equiv c - \left\lfloor \sum_{i=1}^{\Theta} s_i z_i \right\rceil \bmod Q \\ &\equiv \underbrace{c - \sum_{i=1}^{\Theta} s_i z'_i}_{\text{simple part}} - \underbrace{\left\lfloor 2^{-n} \cdot \sum_{i=1}^{\Theta} s_i z''_i \right\rceil}_{\text{complicated part}} \bmod Q,\end{aligned}$$

where $z_i = z'_i + z''_i \cdot 2^{-n}$ for integers $z'_i \in [0, Q)$ and $z''_i \leq 2^n \leq 8\Theta$. As well as the simple part, the "complicated part" can be also expressed as a $\mathcal{L}_A$-restricted depth-3 circuit C, when we choose Q such that $Q > 8\Theta^2$ by the following Lemmas given in [GH11a].

Lemma 2.4.1 ([GH11a]). *Let Q be a prime with $Q > 8\Theta^2$. Then, there exists an univariate polynomial $f(x)$ of degree $\leq 8\Theta^2$ such that $f(\sum_{i=1}^{\Theta} s_i z''_i) = \lfloor 2^{-n} \sum_{i=1}^{\Theta} s_i z''_i \rceil \bmod Q$.*

Lemma 2.4.2 ([GH11a]). *Let T, Θ be positive integers, and $f(x)$ a univariate polynomial over $\mathbb{Z}_Q$ (for Q prime, $Q \geq T\Theta + 1$). Then there is a multilinear symmetric polynomial M_f on $T\Theta$ variables such that*

$$f(\sum_{i=1}^{\Theta} s_i z''_i) = M_f(\underbrace{s_1, \ldots, s_1}_{z''_1}, \underbrace{0, \ldots, 0}_{T - z''_1}, \ldots, \underbrace{s_\Theta, \ldots, s_\tau}_{z''_\Theta}, \ldots, \underbrace{0, \ldots, 0}_{T - z''_\Theta}), \tag{2.4.3}$$

for all $\mathbf{s} = (s_1, \ldots, s_\Theta) \in \{0,1\}^\Theta$ and $z''_1, \ldots, z''_\Theta \in [0, T]$, .

Lemma 2.4.3 ([GH11a]). *Let $Q \geq \Theta + 1$ be a prime, $A \subset \mathbb{Z}_Q$ have cardinality $\Theta + 1$, and $\mathbf{x} = (x_1, \ldots, x_\Theta)$ be variables. Also, let us define $\mathcal{L}_A = \{a + x_i : a \in A, 1 \leq i \leq \Theta\}$. Then for every multilinear symmetric polynomial $M(\mathbf{x})$ over $\mathbb{Z}_Q$, there is a circuit $C(\mathbf{x})$ such that:*

- *C is a $\mathcal{L}_A$-restricted depth-3 circuit over $\mathbb{Z}_Q$ such that $C(\mathbf{x}) \equiv M(\mathbf{x}) \bmod Q$.*
- *C has $\Theta + 1$ product gates of $\mathcal{L}_A$-degree Θ, one gate for each value $a_j \in A$, with the j-th gate computing the value $\lambda_j \prod_i (a_j + x_i)$ for some constant λ_j.*
- *A description of C can be computed efficiently given the values $M(\mathbf{x})$ at all $\mathbf{x} = 1^i 0^{\Theta - i}$*

Combining these, we obtain the following theorem.

Theorem 2.4.1. *Let p, Q be primes such that $p > Q > 8\Theta^2$. For any $A \subset \mathbb{Z}_Q$ of cardinality at least $8\Theta^2 + 1$, the double modulo reduction $[\cdot]_p \bmod Q$ can be expressed as an $\mathcal{L}_A$-restricted depth-3 circuit C of $\mathcal{L}_A$-degree at most $8\Theta^2$ having at most $8\Theta^2 + \Theta + 1$ product gates.*

Bootstrapping a Ciphertext

Let us describe bootstrapping of a ciphertext c. We expand $1/p_i$ by a subset sum of rational numbers y_j's as follows:

$$\frac{1}{p_i} \approx \sum_{j=1}^{\Theta} s_{ij} \cdot y_j \bmod Q_i.$$

Differently from DGHV [DGHV10], we need not use sparse subset sum to express $1/p_i$. Also we use only one set $\{y_1, \ldots, y_\Theta\}$ to reduce the public key size. The secret key s_{ij}'s are included in the public key in an encrypted form $\mathsf{SHE}(pk, \mathbf{s}_j)$, where $\mathbf{s}_j = (s_{1j}, s_{2j}, \ldots, s_{kj})$ for $j = 1, \ldots, \Theta$. Setting $T = 8\Theta$, we do the following:

1. Generate univariate f_i's such that $f_i(\sum_{j=1}^{\Theta} s_{ij} z''_{ij}) = \lfloor 2^{-n} \sum_{j=1}^{\Theta} s_{ij} z''_{ij} \rceil \bmod Q$, for all $z''_{ij} \in [0, T]$ (**Lemma 2.4.1**).

2. Generate a multilinear symmetric polynomial M_f on $T\Theta$ variables such that

$$f_i(\sum_{j=1}^{\Theta} s_{ij} z''_{ij}) = M_{f_i}(\underbrace{s_{i1},\ldots,s_{i1}}_{z_{i1''}},\underbrace{0,\ldots,0}_{T-z''_{i1}},\ldots,\underbrace{s_{i\Theta},\ldots,s_{i\Theta}}_{z_{i\Theta''}},\ldots,\underbrace{0,\ldots,0}_{T-z_{i\Theta''}}),$$

for all $\mathbf{s} = (s_{i1},\ldots,s_{i\Theta}) \in \{0,1\}^{\Theta}$ and $z_{i1''},\ldots,z_{i\Theta''} \in [0,T]$ (**Lemma 2.4.2**).

3. Generate a L_A-restricted depth-3 circuit $C_i(\mathbf{x}) = \sum_{\ell=1}^{t+1} \lambda_{i\ell} \prod_{j=1}^{t} (a_{i\ell} + x_j)$ for $\lambda_{i\ell}, a_{i\ell} \in \mathbb{Z}_{Q_i}$ and $t = T\Theta$ such that $C_i(\mathbf{x}) = M_{f_i}(\mathbf{x}) \bmod Q_i$ (**Lemma 2.4.3**).

4. Add encryptions of $a_{i\ell}$ and $(a_{i\ell} + s_{ij})$ by a multiplicative homomorphic encryption in the public key.

If Q_i's are not the same, we need to use different multiplicative homomorphic encryption schemes for each $i = 1,\ldots,k$. We denote the i-th multiplicative homomorphic encryptions and our scheme by MHE_i and SHE, respectively. In Step 4, the message space of MHE_i is a multiplicative subgroup of $\mathbb{Z}_{Q_i}^{\times}$, therefore $a_{i\ell}$ and $(a_{i\ell} + s_{ij})$ need to belong to the subgroup for all ℓ and j for respective i. The bootstrapping procedure is as follows:

1. Given a ciphertext c, compute $z_{ij} \leftarrow [c \cdot y_j]_{Q_i}$ keeping only $n = \lceil \log_2 \Theta \rceil + 3$ precision after binary point for $i = 1,\ldots,k$ and $j = 1,\ldots,\Theta$.

2. Let $z_{ij} = z'_{ij} + z''_{ij} \cdot 2^{-n}$ for integers $z'_i \in [0,Q)$ and $z''_i \leq 2^n \leq 8\Theta$. Define vectors $\mathbf{c} := (c \bmod Q_1,\ldots,c \bmod Q_k)$ and $\mathbf{s_j} := (s_{1j},\ldots,s_{kj})$ and $\mathbf{z}'_j = (z'_{1j},\ldots,z'_{kj})$ for $j = 1,\ldots,\Theta$.

3. For the simple part, compute $c_1 = \mathsf{SHE}(\mathbf{c}) - \sum_{j=1}^{\Theta} \mathsf{SHE}(\mathbf{s}_j) \cdot \mathsf{SHE}(\mathbf{z}'_j)$ for $j = 1,\ldots,\Theta$.

4. For the complicated part, choose z''_{ij} copies of $\mathsf{MHE}_i(a_{i\ell} + s_{ij})$ and $(T - z''_{ij})$ copies of $\mathsf{MHE}_i(a_{i\ell})$ and then compute

$$b_{i\ell} = \prod_{j=1}^{\Theta} \left(\prod_{r=1}^{z''_{ij}} \mathsf{MHE}_i(a_{i\ell} + s_{ij}) \cdot \prod_{r=1}^{T-z''_{ij}} \mathsf{MHE}_i(a_{i\ell}) \right)$$

   for $i = 1, \ldots, k$ and $\ell = 1, \ldots, t+1$.

5. Convert $\{b_{i\ell} = \mathsf{MHE}_i(m_{i\ell})\}_{i,\ell}$ to $\mathsf{SHE}(m_{1\ell}, \ldots, m_{k\ell})$ for $\ell = 1, \ldots, t+1$ by evaluating the decryption circuit of MHE_i homomorphically.

6. Compute $c_2 = \sum_{\ell=1}^{t+1} \mathsf{SHE}(\lambda_{1\ell}, \ldots, \lambda_{k\ell}) \cdot \mathsf{SHE}(m_{1\ell}, \ldots, m_{k\ell})$ and output $c_1 - c_2$.

Remark 2.4.1. If we use the same Q_i's, we can use the same MHE_i for each i, which means we can use the same decryption circuit for MHE_i. Therefore, we convert the ciphertexts $\{b_{i\ell} = \mathsf{MHE}_i(m_{i\ell}) : i = 1, \ldots, k\}$ to $\mathsf{SHE}(m_{1\ell}, \ldots, m_{k\ell})$ at once using SIMD operations. Otherwise, we convert $b_{i\ell} = \mathsf{MHE}_i(m_{i\ell})$ to $\mathsf{SHE}(0, \ldots, m_{i\ell} \ldots, 0)$ by evaluating the decryption circuit of each MHE_i, and then we can obtain $\mathsf{SHE}(m_{1\ell}, \ldots, m_{k\ell})$.

The most demanding computation is to convert MHE ciphertexts into somewhat homomorphic encryption SHE in Step 5. Since we will use ElGamal encryption as [GH11a] for MHE, the decryption circuit consists of multiplications and exponentiations. One can use Cheon and Kim's technique [CK13], w-ary representation, to lower the depth of MHE decryption circuit. Our scheme becomes fully homomorphic encryption when it can evaluate the decryption circuit of MHE, not its own decryption circuit.

2.5 Discussion

In this section, we discuss application of our scheme to secure large integer arithmetic, and also discuss how to compress the public key.

2.5.1 Secure Large Integer Arithmetic

Secure integer arithmetic is perhaps one of the most important applications of homomorphic encryption schemes. It includes frequently used statistical functions such as mean, standard deviation, logistical regression, and secure evaluation of a multivariate polynomial over integers. Some applications may require very large integer inputs in the computation of these functions. For the homomorphic computation of these functions, one may use FHE supporting homomorphic bit operations. However, the large ciphertext expansion ratio and rather high cost of bootstrapping make this cumbersome and inefficient. In fact, even an addition of two λ-bit integers using bit operations needs computing degree-$O(\lambda)$ polynomial over $\mathbb{Z}_2$ due to carry computation. For this reason, it is useful to construct an efficient somewhat homomorphic scheme supporting large integer arithmetic on encrypted data.

As mentioned earlier, our somewhat homomorphic encryption scheme supports arithmetic operations on $\mathbb{Z}_Q$ with $Q = \prod_{i=1}^{k} Q_i$ when all Q_i's are pairwise coprime. We can freely choose k up to $\tilde{O}(\lambda^3)$ depending on applications. And the advantage of our scheme in the overhead stands out, as the plaintext space gets larger.

2.5.2 Public key compression

As done in [CNT12, CLT13], the public key elements can be compressed. Namely, one can use pseudorandom number generator f and *seed* which are

public to compress the public key elements. Note that the main part of the public key $X = \{x_j\}_{j=1}^{\tau}$ and $Y = \{y_\ell\}_{\ell=1}^{k}$ consists of the elements of the form

$$\mathrm{CRT}_{(q_0,p_1,\ldots,p_k)}(e_0, e_1Q_1 + e'_1, \ldots, e_kQ_k + e'_k) \tag{2.5.4}$$

where $e_0 \leftarrow \mathbb{Z} \cap [0, q_0)$, $e_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$, and $e'_i \in \{0, 1\}$ for $1 \leq i \leq k$. Using f and *seed*, **KeyGen** generates random integers χ_j, χ'_ℓ modulo x_0 and only correcting factors Δ_j, Δ'_ℓ are included in the public key such that $\chi_j - \Delta_j, \chi'_\ell - \Delta'_\ell$ satisfy the form in (2.5.4). Using this technique, the size of the public key element is reduced from $\log_2 x_0$ to $\log_2 (\prod p_i)$ since random part modulo q_0 are generated by f and *seed*.

Chapter 3

A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption

The concept of computation on encrypted data without decryption was first introduced by Rivest, Adleman and Dertouzos in 1978 [RAD78]. Thirty years later, Gentry proposed a fully homomorphic encryption (FHE) based on ideal lattices [Gen09]. This scheme is far from being practical because of its large computational cost and large ciphertexts. Since then, considerable efforts have been made to devise more efficient schemes. However, most FHE schemes still have very large ciphertexts (millions of bits for a single ciphertext). This presents a considerable bottleneck in practical deployments.

We consider the following situation: several users upload data encrypted with a public-key FHE, a server carries out computations on the encrypted data and then sends them to an agency who has a decryption key for the

FHE. This is common in typical FHE scenarios, such as medical and financial applications [NLV11]. In this situation, one approach to reduce the storage requirement is to use AES encryption to encrypt data, and then perform homomorphic computations on ciphertexts after converting to FHE-ciphertexts. This method has a great advantage in storage and communication, because only small AES-ciphertexts are transmitted from user to server, and these are only when their homomorphic computations are required. In an asymmetric setting, we can still use this approach by adding several public-key FHE ciphertexts of a session key. However this approach is not practical when the amount of messages transmitted simultaneously is small compared with the size of one FHE ciphertexts. Moreover, the conversion of AES-ciphertexts into FHE-ciphertexts requires a leveled FHE with multiplicative depth of at least forty [CLT14, CCK$^+$13, GHS12b].

In this paper, we explore an alternative method that encrypts messages with a public key encryption (PKE) and converts them into SHE-ciphertexts for homomorphic computations. In this approach, the ciphertext expansion ratio is only two or three regardless of the message size. Moreover, the decryption circuit is very shallow when the SHE allows large integers as messages. For example, the decryption circuit of ElGamal over $\mathbb{Z}_N$ has a multiplicative depth of ten under a SHE with the message space $\mathbb{Z}_N$ [GH11a]. We can reduce the depth further by representing the secret exponent e as $\log_w e$ binary vectors of length w, which is an improvement over the Gentry- Halevi technique [GH11a].

When using additive (resp. multiplicative) homomorphic encryption as the underlying PKEs, we obtain the additional advantage that additions (resp. multiplications) can be computed without converting to SHE. For multiplicative homomorphic encryptions (MHE) in particular, one can compute

$\mathsf{SHE}(f(m_1,\ldots,m_k))$ from $\mathsf{PKE.Enc}(m_1),\ldots,\mathsf{PKE.Enc}(m_k)$ without (expensive) bootstrapping for any multivariate polynomial $f(x_1,\ldots,x_k)$ with polynomially many terms.

One problem when using MHE in the hybrid scheme is that the message space for MHE schemes is not usually closed under addition. For example, the (IND-CPA) ElGamal encryption over a ring R can only take messages with elements in a prime order subgroup, which covers only a small part of R. To resolve this, therefore, we construct a MHE whose message space is $\mathbb{Z}_N^\times$ for an RSA modulus $N = p_1p_2$. The proposed scheme is constructed by combining ElGamal encryption over $\mathbb{Z}_N^\times$ and Goldwasser-Micali encryption over $\mathbb{Z}_N$, and is secure under the decisional Diffie–Hellman assumption and the quadratic residuosity assumption for common $N = pq$.

We remark that our technique solves the open problem of [KLYC13] when the FHE message space is $\mathbb{Z}_p$ for large p. We convert the double modulo reduction into a depth-3 circuit, and then, we apply the technique of [GH11a]. Our improved technique plays an important role in this method, as the parameters depend heavily on the homomorphic capacity of FHE.

As an independent interest, we also present a generic method for converting from a private-key SHE to a public-key SHE using our hybrid scheme. In this case, the message space is $\mathbb{Z}_p$ for a large integer $p > 2$ and the public key is the encryption of the secret key of a PKE under private-key SHE, which is much smaller than that described in [Rot11].

3.1 Preliminaries

In this section, we introduce some definitions and base problems needed to prove the security of our schemes.

Notation

For $m, n \in \mathbb{N}$, $[m, n]$ and $[m, n)$ denote the sets $\{m, m+1, \ldots, n-1, n\}$ and $\{m, m+1, \ldots, n-2, n-1\}$, respectively. We denote the element in $\mathbb{Z} \cap (-\frac{n}{2}, \frac{n}{2}]$ that is equivalent to a modulo n by $a \bmod n$ or $[a]_n$, and the unique integer in $(-\frac{\prod_i p_i}{2}, \frac{\prod_i p_i}{2}]$ that is congruent to m_i modulo p_i for all i by $CRT_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$. For an integer N, we use sets $\mathsf{J}_N := \{a \in \mathbb{Z}_N^\times | \left(\frac{a}{N}\right) = 1\}$ and $\mathsf{QR}_N := \{a \in \mathbb{Z}_N^\times | a = b^2 \text{ for some } b \in \mathbb{Z}_N^\times\}$. We denote $a^{\phi(N)/p} \bmod N$ by $\left(\frac{a}{N}\right)_p$.

3.1.1 Hard Problems

Definition 3.1.1 (Decisional Diffie–Hellman problem over $\mathbb{G}$)**.** Let $\mathbb{G}$ be a group with a generator g of order q. For a given tuple (g, g^a, g^b, g^c), the *decisional Diffie–Hellman (DDH) problem over* $\mathbb{G}$ is to determine whether $g^{ab} = g^c$.

Definition 3.1.2 (k-Quadratic Residuosity problem over $\mathbb{Z}_N$)**.** Given an odd composite integer $N = pq$ such that $p \equiv q \equiv 1 \pmod{2^k}$ and $a \in \mathsf{J}_N$ where $\mathsf{J}_N := \{a \in \mathbb{Z}_N^\times | \left(\frac{a}{N}\right) = 1\}$, the *$k$-quadratic residuosity problem (k-QR) over* $\mathbb{Z}_N$ is to determine whether a is quadratic residue modulo N.

Definition 3.1.3 (Higher Residuosity problem over $\mathbb{Z}_N$)**.** For a given odd composite integer $N = pq$, an integer d such that $d|(p-1)$, and an integer $a \in \mathbb{Z}_N$, the *higher residuosity problem (HR) over* $\mathbb{Z}_N$ is to determine whether a is the d-th residue modulo N.

We say that the DDH assumption over $\mathbb{G}_q$ holds if no polynomial time distinguisher can solve the DDH problem with non-negligible advantage with

respect to the security parameter λ. The k-QR and HR assumptions over $\mathbb{Z}_N$ are defined similarly.

3.1.2 Homomorphic Encryption Schemes

Definition 3.1.4 (ElGamal Encryption over a Ring)**.** Let R be a ring. The *ElGamal encryption scheme* $\mathsf{EIG} = (\mathsf{EIG.KG}, \mathsf{EIG.Enc}, \mathsf{EIG.Dec})$ consists of the following algorithms:

- $\underline{\mathsf{EIG.KG}(\lambda)}$: Choose a multiplicative cyclic subgroup $\mathbb{G}_q$ of prime order q in R such that the DDH assumption holds with respect to the security parameter λ. Choose a generator g of $\mathbb{G}_q$ and a random $e \in [0, q)$, and compute $y = g^e$. Output a public key $pk_{\mathsf{EIG}} = (R, \mathbb{G}_q, g, y)$ and a secret key $sk_{\mathsf{EIG}} = e$.

- $\underline{\mathsf{EIG.Enc}(pk_{\mathsf{EIG}}, m)}$: Take as input the public key pk_{EIG} and a plaintext $m \in \mathbb{G}_q$. Choose a random $r \in [0, q)$ and compute g^{-r} and $m \cdot y^r$. Output $c = (g^{-r}, m \cdot y^r)$.

- $\underline{\mathsf{EIG.Dec}(sk_{\mathsf{EIG}}, c)}$: Take as input the secret key sk_{EIG} and a ciphertext $c = (v, u) \in R^2$. Output $m = v^e u$.

Definition 3.1.5 (Goldwasser–Micali Encryption)**.** The *Goldwasser-Micali encryption scheme* $\mathsf{GM} = (\mathsf{GM.KG}, \mathsf{GM.Enc}, \mathsf{GM.Dec})$ consists of the following algorithms:

- $\underline{\mathsf{GM.KG}(\lambda)}$: Choose random primes p, q and compute $N = pq$ such that the 1-QR assumption holds with respect to the security parameter λ. Compute the quadratic non–residue y modulo N satisfying $\left(\frac{y}{p}\right) =$

$\left(\frac{y}{q}\right) = -1$. Output a public key $pk_{\mathsf{GM}} = (N, y)$ and a secret key $sk_{\mathsf{GM}} = (p)$.

- $\underline{\mathsf{GM.Enc}(pk_{\mathsf{GM}}, m)}$: For a plaintext $m \in \{0, 1\}$, choose a random $x \in \mathbb{Z}_N^\times$ and output a ciphertext $c = y^m x^2 \bmod N$.

- $\underline{\mathsf{GM.Dec}(sk_{\mathsf{GM}}, c)}$: For a ciphertext $c \in \mathbb{Z}_N^\times$, if $\left(\frac{c}{p}\right)_2 = 1$ output 0. Otherwise output 1.

Definition 3.1.6 (Joye–Libert Encryption)**.** The *Joye-Libert encryption scheme* $\mathsf{JL} = (\mathsf{JL.KG}, \mathsf{JL.Enc}, \mathsf{JL.Dec})$ consists of the following algorithms:

- $\underline{\mathsf{JL.KG}(\lambda)}$: Choose an integer $k \geq 1$ and random primes p and q with $p \equiv q \equiv 1 \pmod{2^k}$, and set $N = pq$ such that the k-QR assumption holds with respect to the security parameter λ. Choose a random $y \in \mathsf{J}_N \backslash \mathsf{QR}_N$. Output a public key $pk_{\mathsf{JL}} = (N, y, k)$ and a secret key $sk_{\mathsf{JL}} = (p)$.

- $\underline{\mathsf{JL.Enc}(pk_{\mathsf{JL}}, m)}$: For a plaintext $0 \leq m < 2^k$, choose a random $x \in \mathbb{Z}_N^\times$ and output a ciphertext $c = y^m x^{2^k} \bmod N$.

- $\underline{\mathsf{JL.Dec}(sk_{\mathsf{JL}}, c)}$: For a ciphertext $c \in \mathbb{Z}_N^\times$, compute $z = \left(\frac{c}{p}\right)_{2^k}$ and then find and output $m \in \{0, 1\}^k$ such that the relation

$$\left[\left(\frac{y}{p}\right)_{2^k}\right]^m = z \pmod{p}$$

holds.

Remark 3.1.1. Note that the case $k = 1$ corresponds to the Goldwasser-Micali cryptosystem.

Definition 3.1.7 (Naccache-Stern Encryption)**.** The *Naccache-Stern encryption scheme* $\mathsf{NS} = (\mathsf{NS.KG}, \mathsf{NS.Enc}, \mathsf{NS.Dec})$ consists of the following algorithms:

- $\underline{\mathsf{NS.KG}(\lambda)}$: Choose random small primes $p_1, \ldots, p_k$, compute $u = \prod_{i=1}^{k/2} p_i$ and $v = \prod_{i=k/2+1}^{k} p_i$ and set $\sigma = uv$. Choose large primes a and b such that $p = 2au + 1$ and $q = 2bv + 1$ are prime and set $N = pq$ such that the HR assumption holds with respect to the security parameter λ. Choose a random $g \bmod N$ of order $\phi(n)/4$. Output a public key $pk_{\mathsf{NS}} = (\sigma, N, g)$ and a secret key $sk_{\mathsf{NS}} = (p)$.

- $\underline{\mathsf{NS.Enc}(pk_{\mathsf{NS}}, m)}$: For a plaintext $m \in \mathbb{Z}_\sigma$, choose $x \in \mathbb{Z}_N$ and output a ciphertext $c = x^\sigma g^m \bmod N$.

- $\underline{\mathsf{NS.Dec}(sk_{\mathsf{NS}}, c)}$: For a ciphertext $c \in \mathbb{Z}_N^\times$, compute $z = c^{\phi(N)/p_i}$ and find m_i such that the relation

$$\left[\left(\frac{g}{N}\right)_{p_i}\right]^{m_i} = z \pmod N$$

holds for each i. Output $m = \mathrm{CRT}_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$.

3.2 Encrypt with PKE and Compute with SHE

In this section, we describe the concept of a hybrid scheme that combines PKE and SHE. A message is encrypted using PKE, and converted to a ciphertext under SHE when homomorphic computations on the message are needed. The ciphertext is decrypted under SHE.

3.2.1 A Hybrid Scheme of PKE and SHE

Suppose that a client who has limited computation capability wants to compute $f(m_1, \ldots, m_k)$ for sensitive messages $\{m_1, \ldots, m_k\}$ and a multivariate polynomial f. The client could outsource the heavy computation to a server that has sufficient computing power. Currently, FHE is a good way of delegating computations, if we ignore the client's bandwidth and the server's storage. However, these are very significant measures in the construction of a cloud environment, as they are directly connected to the real cost .

The bandwidth and storage requirements can be reduced by combining PKE with a small ciphertext size and SHE that can evaluate the decryption circuit of the PKE, rather than its own decryption circuit. We propose a hybrid scheme to improve the efficiency of SHE when used in a cloud computing environment. Let $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE and $\mathsf{SHE} = (\mathsf{SHE.KG}, \mathsf{SHE.Enc}, \mathsf{SHE.Dec}, \mathsf{SHE.Eval})$ be a SHE. Suppose that there exists a circuit f_{Dec} such that $f_{\mathsf{Dec}}(sk, c) = m$ for all ciphertexts $c = \mathsf{PKE.Enc}(m)$ and secret key sk. We also assume that f_{Dec} can be evaluated homomorphically under SHE. The *Hybrid scheme* of PKE and SHE consists of the following five algorithms $\mathsf{Hyb} = (\mathsf{Hyb.KG}, \mathsf{Hyb.Enc}, \mathsf{Hyb.Conv}, \mathsf{Hyb.Eval}, \mathsf{Hyb.Dec})$:

- $\underline{\mathsf{Hyb.KG}(\lambda, \mathsf{PKE.KG}, \mathsf{SHE.KG})}$: Run $\mathsf{PKE.KG}$ and $\mathsf{SHE.KG}$ to get $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}, pk_{\mathsf{SHE}}, sk_{\mathsf{SHE}})$. Output

$$\begin{aligned} pk_{\mathsf{Hyb}} &= (pk_{\mathsf{PKE}}, pk_{\mathsf{SHE}}, \mathsf{SHE.Enc}(sk_{\mathsf{PKE}})) \\ sk_{\mathsf{Hyb}} &= (sk_{\mathsf{SHE}}), \end{aligned}$$

  where $\mathsf{SHE.Enc}(sk_{\mathsf{PKE}})$ is an encryption of sk_{PKE} under SHE.

- $\underline{\mathsf{Hyb.Enc}(pk_{\mathsf{Hyb}}, m)}$: For a plaintext $m \in \mathsf{M}_{pk_{\mathsf{PKE}}}$, output

$$c = \mathsf{PKE.Enc}(pk_{\mathsf{PKE}}, m).$$

- $\underline{\mathsf{Hyb.Conv}(pk_{\mathsf{Hyb}}, c)}$: Evaluate the decryption circuit $\mathsf{PKE.Dec}$ with pk_{Hyb}. That is, compute and output a ciphertext C, where

$$C = f_{\mathsf{Dec}}\big(\mathsf{SHE.Enc}(sk_{\mathsf{PKE}}), \mathsf{SHE.Enc}(c)\big).$$

- $\underline{\mathsf{Hyb.Eval}(pk_{\mathsf{Hyb}}, f, C_1, \ldots, C_t)}$: For a given circuit f and t ciphertexts $C_1, \ldots, C_t$ under SHE, output

$$C = \mathsf{SHE.Eval}(pk_{\mathsf{SHE}}, f, C_1, \ldots, C_t).$$

- $\underline{\mathsf{Hyb.Dec}(sk_{\mathsf{Hyb}}, c)}$: For a ciphertext $C \in \mathcal{C}_{pk_{\mathsf{SHE}}}$, output $m = \mathsf{SHE.Dec}(sk_{\mathsf{SHE}}, C)$.

Remark 3.2.1. In the $\mathsf{Hyb.Conv}$ algorithm, note that

$$\begin{aligned} C &= f_{\mathsf{Dec}}(\mathsf{SHE.Enc}(sk_{\mathsf{PKE}}), \mathsf{SHE.Enc}(c)) \\ &= \mathsf{SHE.Enc}(f_{\mathsf{Dec}}(sk_{\mathsf{PKE}}, c)) = \mathsf{SHE.Enc}(m), \end{aligned}$$

since SHE can evaluate the decryption circuit of PKE, f_{Dec}.

Remark 3.2.2. In our hybrid scheme, the homomorphic capacity of SHE at least exceeds the degree of the decryption circuit f_{Dec} of PKE.

Theorem 3.2.1 (Semantic Security of Hybrid Scheme)**.** *If both the* PKE *and the* SHE *are semantically secure, then so is the hybrid scheme.*

Proof. Suppose a polynomial time adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ breaks the semantic security of the scheme with non-negligible advantage. We define the advantage of adversary $\mathcal{B}$ by

$$\begin{aligned}
&\mathbf{Adv}_{\mathcal{B}}(\lambda) := \Big|\Pr\big[\mathcal{B}_2(x_0, x_1, pk_{\mathsf{Hyb}}, y) = b\big| \\
&(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KG}, (pk_{\mathsf{SHE}}, sk_{\mathsf{SHE}}) \leftarrow \mathsf{SHE.KG}, \\
&(x_0, x_1, pk_{\mathsf{Hyb}}) \leftarrow \mathcal{B}_1(pk_{\mathsf{PKE}}, pk_{\mathsf{SHE}}, \mathsf{SHE}(sk_{\mathsf{PKE}})), \\
&b \leftarrow \{0,1\}, y \leftarrow \mathsf{PKE.Enc}(x_b)\big] - 0.5\Big|,
\end{aligned}$$

and denote this by $\mathbf{Adv}_{\mathcal{B}}$. We will show that we can use $\mathcal{B}$ to break either PKE or SHE. We now set up two games for PKE and SHE.

In the game for PKE, the adversary $\mathcal{A}_{\mathsf{PKE}}$ only has access to pk_{PKE}. $\mathcal{A}_{\mathsf{PKE}}$ prepares the following game for $\mathcal{B}$.

1. $\mathcal{A}_{\mathsf{PKE}}$ runs PKE.KG and SHE.KG to obtain $(pk'_{\mathsf{PKE}}, sk'_{\mathsf{PKE}})$ and $(pk_{\mathsf{SHE}}, sk_{\mathsf{SHE}})$.
2. $\mathcal{A}_{\mathsf{PKE}}$ sends $(pk_{\mathsf{PKE}}, pk_{\mathsf{SHE}}, \mathsf{SHE.Enc}(sk'_{\mathsf{PKE}}))$ to $\mathcal{B}_1$ as a public key of Hyb.
3. $\mathcal{B}_1$ chooses x_0, x_1 and sends them to $\mathcal{A}_{\mathsf{PKE}}$.
4. $\mathcal{A}_{\mathsf{PKE}}$ selects $b \leftarrow \{0,1\}$ and sends $y \leftarrow \mathsf{PKE.Enc}(x_b)$ to $\mathcal{B}_2$.
5. $\mathcal{B}_2$ outputs $b' \in \{0,1\}$ to $\mathcal{A}_{\mathsf{PKE}}$ as an answer.
6. $\mathcal{A}_{\mathsf{PKE}}$ outputs $b = b'$.

Steps $2) - 6)$ of this game are identical to the standard game for Hyb with an unmatched pair of pk_{Hyb} and pk_{PKE}. Therefore $\mathbf{Adv}_{\mathcal{A}_{\mathsf{PKE}}} \leq \mathbf{Adv}_{\mathcal{B}}$.

In the game for SHE, the adversary $\mathcal{A}_{\mathsf{SHE}}$ only has access to pk_{SHE}. $\mathcal{A}_{\mathsf{SHE}}$ prepares the following game for $\mathcal{B}$.

1. $\mathcal{A}_{\mathsf{SHE}}$ runs SHE.KG and SHE.KG to obtain $(pk_{\mathsf{PKE}}, sk_{\mathsf{PKE}})$ and $(pk'_{\mathsf{SHE}}, sk'_{\mathsf{SHE}})$.

2. $\mathcal{A}_{\mathsf{SHE}}$ runs $\mathsf{PKE.KG}$ again to get another secret key sk'_{PKE}.
3. $\mathcal{A}_{\mathsf{SHE}}$ sets $x_0 = sk_{\mathsf{PKE}}, x_1 = sk'_{\mathsf{PKE}}$ and is given $y \leftarrow \mathsf{SHE.Enc}(x_b)$ for some randomly chosen $b \in \{0, 1\}$.
4. $\mathcal{A}_{\mathsf{SHE}}$ sends $(pk_{\mathsf{PKE}}, pk_{\mathsf{SHE}}, y)$ to $\mathcal{B}_1$ as a public key of Hyb.
5. $\mathcal{B}_1$ chooses x'_0, x'_1 and sends them to $\mathcal{A}_{\mathsf{SHE}}$.
6. $\mathcal{A}_{\mathsf{SHE}}$ selects $b' \leftarrow \{0, 1\}$ and sends $y' \leftarrow \mathsf{PKE.Enc}(x'_{b'})$ to $\mathcal{B}_2$.
7. $\mathcal{B}_2$ outputs $b'' \in \{0, 1\}$ to $\mathcal{A}_{\mathsf{SHE}}$ as an answer.
8. If $b'' = b'$, $\mathcal{A}_{\mathsf{SHE}}$ outputs $b = 0$, otherwise $b = 1$ is output.

In this game, if $b = 0$ then y is a valid encryption of sk_{PKE} that matches the public key of PKE contained in pk_{Hyb}. Steps $4) - 7)$ are identical to the standard game for Hyb. Then, $\mathcal{B}$ will have the advantage $\mathbf{Adv}_{\mathcal{B}}$. If $b = 1$, then y is an encryption of sk'_{PKE} that does not match the public key pk_{PKE}. In this case, the whole process is identical to the game we designed for PKE. Thus, the advantage of $\mathcal{A}_{\mathsf{SHE}}$ is

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{A}_{\mathsf{SHE}}} &= \Pr[\text{Correct guess}] - 0.5 \\
&= \Pr[b = 0|\text{Guess } 0] + \Pr[b = 1|\text{Guess } 1] - 0.5 \\
&= 0.5 \cdot (0.5 + \mathbf{Adv}_{\mathcal{B}}) + 0.5 \cdot (0.5 - \mathbf{Adv}_{\mathcal{A}_{\mathsf{PKE}}}) - 0.5 \\
&= 0.5\mathbf{Adv}_{\mathcal{B}} - 0.5\mathbf{Adv}_{\mathcal{A}_{\mathsf{PKE}}}
\end{aligned}$$

Therefore, if $\mathbf{Adv}_{\mathcal{B}}$ is non-negligible, either $\mathbf{Adv}_{\mathcal{A}_{\mathsf{PKE}}}$ or $\mathbf{Adv}_{\mathcal{A}_{\mathsf{SHE}}}$ is non-negligible. □

3.2.2 Additive Homomorphic Encryptions for PKE in the Hybrid Scheme

In constructing a hybrid scheme, candidate encryptions for an additive homomorphic IND-CPA PKE include Goldwasser–Micali [GM84], Paillier [Pai99], Okamoto–Uchiyama [OU98], Naccache–Stern [NS98] and Joye–Libert [JL] encryptions. The decryption circuit of each system requires additional circuits besides exponentiation, the Chinese remainder algorithm for the Goldwasser–Micali and Naccache–Stern encryptions, and integer division for the Paillier and Okamoto–Uchiyama encryptions. Because it is difficult to evaluate the integer division part efficiently, the latter encryptions are unsuitable for the construction of our hybrid scheme. Thus we only consider the Goldwasser–Micali, Joye–Libert, and Naccache–Stern encryptions for PKE in the hybrid scheme.

Goldwasser-Micali Encryption

The decryption circuit of Goldwasser-Micali encryption is as follows: for a given ciphertext $c \in \mathbb{Z}_N$, output 0 if $\left(\frac{c}{p}\right)_2 = 1$ and $\left(\frac{c}{q}\right)_2 = 1$ and output 1 otherwise. We can modify the decryption circuit as follows:

1. First, compute
$$m' = \mathrm{CRT}_{(p,q)}(c^{(p-1)/2}, c^{(q-1)/2}).$$
2. Output $m = (1 - m')/2$.

Using the homomorphic evaluation of secret exponentiation and the Chinese remainder algorithm, it is possible to evaluate the decryption circuit of Goldwasser-Micali encryption.

We describe a homomorphic decryption method briefly: Suppose that the Goldwasser–Micali encryption scheme over $\mathbb{Z}_N$ and a SHE scheme with message space $\mathbb{Z}_N$ for $N = pq$ are given. To construct hybrid scheme, we use the following two facts:

1. For binary representation of $e = \sum_i e_i 2^i$,
$$v^e = v^{\sum_i e_i 2^i} = \prod_i v^{e_i 2^i} = \prod_i \left(e_i v^{2^i} + (1 - e_i) v^0 \right)$$
2. $\mathrm{CRT}_{(p,q)}(a, b) = (a \cdot q(q^{-1} \bmod p) + b \cdot p(p^{-1} \bmod q)) \bmod N$

In $\mathsf{Hyb.KG}$ of hybrid scheme, we add encryptions of p_i, q_i for $i = 0, \ldots, \max\{\lceil \log p \rceil, \lceil \log q \rceil\}$ to the pk_{Hyb}, where $(p-1)/2 = \sum_i p_i 2^i$ and $(q-1)/2 = \sum_i q_i 2^i$. Since p and q are secret, the encryptions of $q(q^{-1} \bmod p)$ and $p(p^{-1} \bmod q)$ also need to be added to the public key of the hybrid scheme.

Now, we run $\mathsf{Hyb.Conv}$ algorithm to evaluate the decryption circuit of GM. Let us consider a GM ciphertext c. Below, we denote $\mathsf{SHE.Enc}$ by Enc and homomorphic multiplication and addition of SHE by $\cdot$ and $+$, respectively.

1. To evaluate exponentiation with secret exponent, compute
$$\begin{aligned} c_1 &= \prod_i \left(\mathsf{Enc}(p_i) \cdot c^{2^i} + (1 - \mathsf{Enc}(p_i)) \right) \\ c_2 &= \prod_i \left(\mathsf{Enc}(q_i) \cdot c^{2^i} + (1 - \mathsf{Enc}(q_i)) \right) \end{aligned}$$
2. To evaluate the Chinese remaindering algorithm, compute
$$c_3 = c_1 \cdot \mathsf{Enc}\left(q(q^{-1} \bmod p)\right) + c_2 \cdot \mathsf{Enc}\left(p(p^{-1} \bmod q)\right)$$
3. Finally, we compute $c_4 = (1 - c_3) \cdot (2^{-1} \bmod N)$

In the above algorithm, we can verify that the degree of decryption circuit is approximately $2 \log p$.

Naccache-Stern Encryption

The decryption of the Naccache–Stern encryption computes $z = (\frac{c}{N})_{p_i} = c^{\phi(N)/p_i}$ and find m_i by comparing it with $[(\frac{g}{N})_{p_i}]^j$ for all $j = 0, 1, \ldots, (p_i - 1)$. The message m is recovered by computing $m = \mathrm{CRT}_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$.

The only difference from GM scheme is to fine m_i by searching z in the set $\{[(\frac{g}{N})_{p_i}]^j\}_{0 \le j \le (p_i - 1)}$. We use polynomial interpolation to obtain an encryption of m_i. That is, we construct a polynomial $f_i(x)$ of degree $p_i - 1$ such that

$$f_i\left(\left[\left(\frac{g}{N}\right)_{p_i}\right]^k\right) = k \pmod{N},$$

for all $k \in \mathbb{Z}_{p_i}$. We use the same algorithms when evaluating $z = (\frac{c}{N})_{p_i} = c^{\phi(N)/p_i}$ and Chinese remaindering. The degree of the decryption circuit of Naccache–Stern is approximately $\log \phi(N) \cdot \max_i\{p_i - 1\}$.

Joye–Libert Encryption

The homomorphic decryption of Joye–Libert encryption is the same as the first part of Naccache–Stern decryption if we replace the parameters N, p_i, and g by $p, 2^k$, and y, respectively. In this case, we need to use SHE with a $\mathbb{Z}_Q$ message space where $Q = 2^k N$. The degree of the decryption circuit is approximately $\log p \cdot (2^k - 1)$.

Remark 3.2.3. Although the elliptic curve ElGamal cryptosystem [MV93] has a small ciphertext, we do not consider it in this paper, since it is difficult to evaluate the inverse map of the message encoding and the addition of points on the elliptic curve.

3.2.3 Multiplicative Homomorphic Encryptions for PKE in the Hybrid Scheme

We may consider ElGamal encryption [ElG84] as a candidate for a multiplicative PKE in constructing a hybrid scheme. An ordinary ElGamal encryption over a ring R has a message space $\mathbb{G}_q \subset R$ of prime order q. In other words, elements not in the prime subgroup cannot be securely encrypted under ElGamal encryption. It is possible to take all nonzero element in R as a message only when $R = \mathbb{F}_{2^n}$ for n such that $2^n - 1$ is prime, but we cannot use the ElGamal encryption over $\mathbb{F}_{2^n}$ for PKE. This is because the DLP in an extension field with a small characteristic is no longer considered a hard problem [Jou13, BGJT13]. Unlike the extension field case, it is impossible to construct an ElGamal encryption whose message space is a full domain over an integer ring.

We propose a new multiplicative homomorphic encryption whose message space is $\mathbb{Z}_N^\times$, which covers almost all nonzero elements of $\mathbb{Z}_N$. Our scheme is a combination of the ElGamal scheme over $\mathbb{Z}_N$ [ElG84] and the Goldwasser-Micali encryption scheme over $\mathbb{Z}_N$ [GM84] for a common $N = p_1 p_2$, and the ciphertext consists of three elements in $\mathbb{Z}_N^\times$. We call this the *EGM* encryption scheme.

First, we need the following Lemma to construct EGM encryption.

Lemma 3.2.1. *Let $N = p_1 p_2$, where $p_1 = 2q_1 + 1$ and $p_2 = 4q_2 + 1$ for distinct primes p_1, p_2, q_1, q_2 and $q_1, q_2 \neq 2$. Let $\mathsf{J}_N := \{a \in \mathbb{Z}_N^\times | \left(\frac{a}{N}\right) = 1\}$. Then, J_N is a cyclic subgroup in $\mathbb{Z}_N^\times$ of order $4q_1q_2$.*

Proof. The order of $\mathbb{Z}_N^\times$ is $\phi(N) = 8q_1q_2$, and the order of J_N is $\phi(N)/2 = 4q_1q_2$. A subgroup of order $4q_1q_2$ in $\mathbb{Z}_N^\times$ is isomorphic to $\mathbb{Z}_2 \oplus \mathbb{Z}_{2q_1q_2}$ or $\mathbb{Z}_{4q_1q_2}$, since the group $\mathbb{Z}_N^\times$ is isomorphic to $\mathbb{Z}_{p_1}^\times \oplus \mathbb{Z}_{p_2}^\times \cong \mathbb{Z}_{2q_1} \oplus \mathbb{Z}_{4q_2}$. If there is an element $\alpha \in \mathbb{Z}_N^\times$ of order 4, then $\mathsf{J}_N \cong \mathbb{Z}_{4q_1q_2}$ and so J_N is a cyclic group of order $4q_1q_2$.

Let us consider generators g_1 of order $2q_1$ in $\mathbb{Z}_{p_1}^\times$ and g_2 of order $4q_2$ in $\mathbb{Z}_{p_2}^\times$ and the map

$$\mathrm{CRT}_{(p_1,p_2)} : \mathbb{Z}_{p_1}^\times \times \mathbb{Z}_{p_2}^\times \to \mathbb{Z}_N^\times.$$

Let $\alpha := CRT_{(p_1,p_2)}(g_1^{q_1}, g_2^{q_2})$. Then we can easily verify that the order of α is 4. In addition,

$$\left(\frac{\alpha}{N}\right) = \left(\frac{\alpha}{p_1}\right)\left(\frac{\alpha}{p_2}\right) = \left(\frac{g_1^{q_1}}{p_1}\right)\left(\frac{g_2^{q_2}}{p_2}\right) = (-1)\cdot(-1) = 1.$$

Therefore $\mathsf{J}_N \cong \mathbb{Z}_{4q_1q_2}$ if $q_1, q_2 \neq 2$. □

We use the parameters N, p_1, p_1, q_1, q_2 and J_N as defined in Lemma 3.2.1. We remark that the Jacobi symbol of -1 is $\left(\frac{-1}{N}\right) = \left(\frac{-1}{p_1}\right)\left(\frac{-1}{p_2}\right) = -1$. Take an element $\sigma \in \mathbb{Z}_N^\times$ with $\left(\frac{\sigma}{p_1}\right) = \left(\frac{\sigma}{p_2}\right) = -1$. We define a bijective map $\iota : \mathbb{Z}_N^\times \to \mathsf{J}_N \times \{0,1\}$ by

$$m \mapsto (\hat{m}, \check{m}) = \left(m \cdot \left(\frac{m}{N}\right), \left(1 - \left(\frac{m}{N}\right)\right)/2\right).$$

The $\mathsf{EGM} = (\mathsf{EGM.KG}, \mathsf{EGM.Enc}, \mathsf{EGM.Dec})$ encryption is as follows:

- $\underline{\mathsf{EGM.KG}(\lambda)}$: Choose a generator g of J_N with order $\phi(N)/2$ in $\mathbb{Z}_N^\times$ and a random $e \in [0, 4q_1q_2)$, and compute $y \equiv g^e \bmod N$. Output a public key $pk_{\mathsf{EGM}} = (N, g, y, \sigma)$ and a secret key $sk_{\mathsf{EGM}} = (e, p_1, p_2)$.

- $\underline{\mathsf{EGM.Enc}(pk_{\mathsf{EGM}}, m)}$: For a plaintext $m \in \mathbb{Z}_N^\times$, compute $\iota(m) = (\hat{m}, \check{m})$ and choose a random $r \in [0, N^2)^*$ and a random $h \in \mathbb{Z}_N$. Output a ciphertext $c = (c_1, c_2, c_3) = (g^{-r}, \hat{m}y^r, \sigma^{\check{m}}h^2)$.

- $\underline{\mathsf{EGM.Dec}(sk_{\mathsf{EGM}}, c)}$: Take as input the secret key sk_{EGM} and a ciphertext $c = (c_1, c_2, c_3) \in (\mathbb{Z}_N^\times)^3$. Compute and output a message $m \equiv c_1^e c_2 \cdot \mathrm{CRT}_{(p_1,p_2)}(c_3^{q_1}, c_3^{2q_2}) \bmod N$.

*The statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is at most $1/N$, where $\mathcal{D}_1 := \{\text{choose } r \leftarrow [0, N^2) : \text{output } r \bmod \phi(N)\}$ and $\mathcal{D}_2 := \{\text{choose } r \leftarrow [0, \phi(N)) : \text{output r}\}$. In fact, we can choose $N^{1+\epsilon}$ for some $\epsilon > 0$ instead of N^2.

The EGM encryption is multiplicatively homomorphic over $\mathbb{Z}_N^{\times}$, which covers almost all nonzero element of $\mathbb{Z}_N$. Note that $\mathrm{CRT}_{(p_1,p_2)}(c_3^{q_1}, c_3^{2q_2}) = 1$ if $\check{m}$ is even, and $\mathrm{CRT}_{(p_1,p_2)}(c_3^{q_1}, c_3^{2q_2}) = -1$ otherwise. From this, the EGM encryption is correct for an unlimited number of multiplications on encrypted data. Additionally, the EGM encryption is semantically secure under the DDH assumption over J_N and the QR assumption over $\mathbb{Z}_N$ for common $N = p_1 p_2$.

Theorem 3.2.2 (Multiplicative Homomorphism)**.** *For any positive integer k, suppose that $c_i = \mathsf{EGM.Enc}(pk_{\mathsf{EGM}}, m_i)$ for all $i \in [1, k]$. Then,*

$$\mathsf{EGM.Dec}(sk_{\mathsf{EGM}}, \prod_{i=1}^{k} c_i) = \prod_{i=1}^{k} m_i,$$

where the multiplication of two ciphertexts is defined as the componentwise product. That is, EGM encryption is multiplicatively homomorphic.

Proof. Suppose that $c_i := (c_{i1}, c_{i2}, c_{i3}) = (g^{r_i}, \hat{m}_i y^{r_i}, \sigma^{\check{m}_i} h_i^2)$ for $i \in [1, k]$. Then

$$\begin{aligned} C &= (C_1, C_2, C_3) := \prod_{i=1}^{k} c_i \\ &= (g^{-\sum_{i=1}^{k} r_i}, \prod_{i=1}^{k} \hat{m}_i y^{\sum_{i=1}^{k} r_i}, \sigma^{\sum_{i=1}^{k} \check{m}_i} \prod_{i=1}^{k} h_i^2) \end{aligned}$$

We remark that if $\sum_{i=1}^{k} \check{m}_i$ is even, then $CRT_{(p_1,p_2)}(C_3^{q_1}, C_3^{2q_2}) = 1$ and $\prod_{i=1}^{k}(\frac{m_i}{N}) = 1$. Otherwise, $CRT_{(p_1,p_2)}(C_3^{q_1}, C_3^{2q_2}) = -1$ and $\prod_{i=1}^{k}(\frac{m_i}{N}) = -1$. Thus, we obtain

$$\begin{aligned} & C_1^e C_2 \cdot \mathrm{CRT}_{(p_1,p_2)}(C_3^{q_1}, C_3^{2q_2}) \\ = & \Big(\prod_{i=1}^{k} \hat{m}_i\Big) \cdot \mathrm{CRT}_{(p_1,p_2)}(C_3^{q_1}, C_3^{2q_2}) \\ = & \Big(\prod_{i=1}^{k} m_i \left(\frac{m_i}{N}\right)\Big) \cdot \mathrm{CRT}_{(p_1,p_2)}(C_3^{q_1}, C_3^{2q_2}) \\ = & \prod_{i=1}^{k} m_i \pmod N. \end{aligned}$$

□

Theorem 3.2.3 (Semantic Security)**.** *The* EGM *scheme is semantically secure under the DDH assumption over* J_N *and the QR assumption over* $\mathbb{Z}_N$*.*

Proof of Theorem 3.2.3. Under the attacker scenario, the attacker first receives a public key of the encryption scheme, and outputs a message $m_0, m_1 \in \mathbb{Z}_N$. The challenger returns an encryption of m_b for a randomly chosen bit b. Finally, the attacker outputs a guess b' and succeeds if $b = b'$. We use hybrid argument to prove semantic security. We use a sequence of games and denote S_i the event that the attacker succeeds in Game_i.

Game$_0$: this is the original attack scenario. That is, we simulate the challenger by running EGM.KG to obtain a public key $pk_0 = (N, g, y, \sigma)$ and a secret key $sk_0 = (e, p_1, p_2)$.

Game$_1$: this game is the same as Game_0, except for the following modification to the key generation. Instead of choosing $\sigma \in \mathbb{Z}_N^\times$ with $(\frac{\sigma}{p_1}) = (\frac{\sigma}{p_2}) = -1$, we choose it as $\sigma' = t^2$ for a randomly chosen t from $\mathbb{Z}_N$. That is, $pk_1 = (N, g, y, \sigma')$.

It is clear that any significant difference between $\Pr[S_0]$ and $\Pr[S_1]$ leads immediately to an effective statistical test for solving the QR problem over $\mathbb{Z}_N$. Thus we obtain

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathbf{Adv}_{\mathsf{QR}},$$

where $\mathbf{Adv}_{\mathsf{QR}}$ denotes the advantage in solving QR problem over $\mathbb{Z}_N$.

Game$_2$: this game is the same as Game_1, except for the following modification to the encryption of m_b. Instead of encrypting a m_b as $c = (g^{-r}, \hat{m}_b y^r, \sigma'^{\check{m}_b} h^2)$, we compute $c = (g^{-r}, u, \sigma'^{\check{m}_b} h^2)$ for randomly chosen $r \in [0, N^2)$, $h \in \mathbb{Z}_N$ and $u \in \mathsf{J}_N$ and send c to the attacker as a challenge ciphertext.

It is also clear that any significant difference between $\Pr[S_1]$ and $\Pr[S_2]$ leads immediately to an effective statistical test for solving the DDH problem over J_N. Thus

$$|\Pr[S_1] - \Pr[S_2]| \leq \mathbf{Adv}_{\mathsf{DDH}},$$

where $\mathbf{Adv}_{\mathsf{DDH}}$ denotes the advantage in solving DDH problem over J_N.

Since the challenge ciphertext c in Game_2 is independent from message m_b, $\Pr[S_2]$ is $1/2$. Thus we obtain that

$$\begin{aligned}\left|\Pr[S_0] - \frac{1}{2}\right| &= |\Pr[S_0] - \Pr[S_2]| \\ &\leq |\Pr[S_0] - \Pr[S_1]| + |\Pr[S_1] - \Pr[S_2]| \\ &\leq \mathbf{Adv}_{\mathsf{QR}} + \mathbf{Adv}_{\mathsf{DDH}}.\end{aligned}$$

Thus the advantage of the attacker in Game_0 is negligible under DDH assumption over J_N and QR assumption over $\mathbb{Z}_N$.

□

Encryption of Zero

Since the EGM scheme has the message space $\mathbb{Z}_N^\times$, we cannot encrypt zero or any multiples of p_1 or p_2 in $\mathbb{Z}_N$. As the probability of an encryptor choosing the multiples of p_1 or p_2 is negligible, we are only concerned with zero message.

Borrowing an idea in [SYY99], we can modify the scheme by appending λ Goldwasser–Micali encryptions of an encoding defined by $0 \mapsto \mathbf{r} \in_R \{0,1\}^\lambda$ and $1 \mapsto 0^\lambda \in \{0,1\}^\lambda$ for 2^λ security. The ciphertext of the modified EGM scheme is of the form $(g^{-r}, \hat{m}y^r, h^2\sigma^{\check{m}}, \mathsf{GM.Enc}(r_1), \ldots, \mathsf{GM.Enc}(r_\lambda))$, where $(r_1, \ldots, r_\lambda) = (0, \ldots, 0)$ for a nonzero message $m \in \mathbb{Z}_N^\times$ and a random λ-bit element for the zero element. Note that $\hat{m}$ and $\check{m}$ can be taken arbitrary from $\mathbb{Z}_N^\times$ when $m = 0$. The random $(r_1, \ldots, r_\lambda)$ in the appended ciphertext is preserved under multiplications with $1 - 2^{-\lambda}$ probability. The decryption algorithm is similar to the original EGM using a polynomial $f(r_1, \ldots, r_\lambda) = \frac{(-1)^\lambda}{\lambda!} \prod_{i=1}^{\lambda}(r_1 + \cdots + r_\lambda - i)$.

EGM Encryption

The decryption circuit of EGM encryption consists of three secret exponentiations, two multiplications and one Chinese remaindering algorithm. Similar to the Goldwasser–Micali encryption, we can evaluate the decryption circuit of degree $\log e + \log p_1$ ($\approx \lambda^2$). The message space of EGM is $\mathbb{Z}_N^\times$ and the ciphertext space is $(\mathbb{Z}_N^\times)^3$. Thus we choose the message space of SHE to be $\mathbb{Z}_N$ to construct the hybrid scheme.

3.3 Homomorphic Evaluation of Exponentiation

To enhance the performance of the hybrid scheme, we must efficiently evaluate a modular exponentiation by a secret exponent efficiently; this is related to the decryption circuit of Goldwasser–Micali, Naccache–Stern, Joye–Libert and EGM encryptions. Actually, this problem has been dealt with by Gentry and Halevi [GH11a] in evaluating the depth-3 decryption circuit of the form $\Sigma\Pi\Sigma$. Their idea is to express the secret key e of the ElGamal encryption as a binary representation, and then convert the exponentiation into a multivariate polynomial. In their approach, the ElGaml decryption circuit is represented by a 4λ–degree polynomial that is too large to evaluate efficiently, where λ is the security parameter. We present an improved algorithm to evaluate an exponentiation with a small degree multivariate polynomial.

3.3.1 Improved Exponentiation using Vector Decomposition

Gentry and Halevi [GH11a] first proposed a method to homomorphically evaluate an exponentiation using a secret exponent. They expand the secret key e of

ElGamal encryption as a binary representation $e = \sum_i e_i 2^i$, with $e_i \in \{0, 1\}$, and compute v^e as follows:

$$v^e = v^{\sum_i e_i 2^i} = \prod_i v^{e_i 2^i}.$$

They use the Lagrange interpolation to compute $v^{e_i 2^i}$, that is, $v^{e_i 2^i} = e_i v^{2^i} + (1 - e_i)v^0$. The degree of their exponentiation circuit is approximately $2 \log e$ ($\approx 4\lambda$), where λ is the security parameter. Reducing the degree of exponentiation of secret exponent is a meaningful approach, since the degree of the decryption circuit is directly related to the selection of parameters for SHE. We consider a w-ary ($w > 2$) representation of the secret e to reduce the degree of the exponentiation circuits. Using this w-ary representation, there are $\log_w e$ individual terms $v^{e_i w^i}$, which is fewer than $\log_2 e$. However, a$(w-1)$-degree polynomial is required to express each $v^{e_i w^i}$ for $e_i \in [0, w-1]$ when using Lagrange interpolation. Indeed, this increases the degree of exponentiation by e from $2 \log e$ to $w \log_w e$.

Instead of Lagrange interpolation, we use a vector representation of e_i to reduce the degree in computing $v^{e_i w^i}$. First, we expand the secret e in a w-ary representation, $e = \sum_{\ell=0}^{\lfloor \log_w e \rfloor} e_\ell w^\ell$. Then, v^e can be written in the form $v^e = v^{\sum_{\ell=0}^{n} e_\ell w^\ell} = \prod_{\ell=0}^{n} v^{e_\ell w^\ell}$, where $e_\ell \in [0, w-1]$ and $n = \lfloor \log_w e \rfloor$. We define an embedding map π as:

$$\begin{aligned} \pi : W &\longrightarrow \mathbb{Z}^w \\ a &\longmapsto \mathbf{f}_{a+1} \end{aligned}$$

where $W := \{0, 1, \ldots, w-1\}$ and $F := \{\mathbf{f}_1, \ldots, \mathbf{f}_w\}$ is the standard basis in $\mathbb{Z}^w$. We denote $\pi(e_i) = (e_{i0}, \ldots, e_{i(w-1)}) \in \mathbb{Z}^w$, where $e_{ik} \in \{0, 1\}$ for all $i \in [0, n]$ and $k \in [0, w-1]$. We also define a vector $\mathbf{v}_i := (1, v^{w^i}, v^{2w^i}, \cdots, v^{(w-1)w^i}) \in \mathbb{G}_q^w$. Then it can easily be verified that $v^{e_i w^i} = \langle \mathbf{v}_i, \pi(e_i) \rangle$, where $\langle \cdot, \cdot \rangle$ is the ordinary inner product in $\mathbb{Z}^w$. To operate this procedure publicly, we add encryptions of $e_{\ell k}$ for all $\ell \in [0, n]$, $k \in [0, w-1]$ under SHE to the public key. In fact, we can omit an encryption of e_{i0}, since e_{i0} is equal to $1 - \sum_{k=1}^{w-1} e_{ik}$ which can be computed homomorphically.

$\underline{\mathsf{Eval.Exp.Setup}(pk_{\mathsf{SHE}}, e, w)}$: Take as input a public key pk_{SHE} of SHE, secret exponent e, and expansion parameter w.

1. Expand $e = \sum_{i=0}^{n} e_i w^i$ and compute $\pi(e_i) = (e_{i0}, \ldots, e_{i(w-1)})$ for all $i \in [0, n]$, where $n = \lfloor \log_w e \rfloor$.

2. Output
$$\bar{E}_e := \Big\{ \mathsf{SHE.Enc}(e_{ik}) : i \in [0, n], k \in [0, w-1] \Big\}.$$

$\underline{\mathsf{Eval.Exp}(pk_{\mathsf{SHE}}, \bar{E}_e, w, v)}$: Take as input the public key pk_{FHE} of FHE, set $\bar{E}_e$ output by $\mathsf{Eval.Exp.Setup}$ and v.

1. Encrypt v^{kw^i} for all $i \in [0, n]$, $k \in [0, w-1]$ under SHE.

2. Output
$$c := \prod_{i=0}^{n} \Big(\sum_{k=0}^{w-1} \mathsf{SHE.Enc}(e_{ik}) \cdot \mathsf{SHE.Enc}(v^{kw^i}) \Big).$$

Remark 3.3.1. In Step 1, we assume v, w, n are public so that v^{kw^i} can be evaluated publicly.

Remark 3.3.2. In Step 2, we use the trivial encryption of v^{kw^i} for all i and k, since they contain no secret information.

Theorem 3.3.1 (Correctness)**.** *If*
$$c \leftarrow \mathsf{Eval.Exp}(pk_{\mathit{SHE}}, \bar{E}_e, w, v),$$
then $v^e \leftarrow \mathit{SHE}.\mathsf{Dec}_{sk_{\mathit{SHE}}}(c)$.

The proof of Theorem 3.3.1 is straightforward. It has been verified that the degree of exponentiation is approximately $2 \log_w e$, which is $\log w$ times smaller than the original method. Using our method, the exponentiation of a large secret exponent can be homomorphically evaluated with a small degree polynomial at a cost of $\tilde{O}(w)$ additional public keys for an arbitrary integer w.

3.3.2 Improve the Bootstrapping without Squashing

Gentry and Halevi [GH11a] proposed a new method to construct FHE without squashing, called *chimeric* FHE. The chimeric FHE uses a multiplicative homomorphic encryption (MHE) to bootstrap a SHE without squashing, thereby removing the assumption on the hardness of the sparse subset sum problem. Gentry and Halevi's technique in [GH11a] expresses the decryption of the SHE scheme as a depth-3 ($\sum\prod\sum$) arithmetic circuit. They temporarily switch to a ciphertext under MHE, such as ElGamal, to compute $\prod$ part. Then they homomorphically evaluate the decryption circuit of MHE to obtain a ciphertext under SHE. Using their method, SHE only needs to evaluate the MHE decryption circuit of fixed degree $2\log e$, rather than its own decryption circuit. Using our efficient evaluation of exponentiation given in Section 3.3.1, we can reduce the degree from $2\log e$ to $2\log_w e$. Moreover, our method can handle a more general class of SHE whose decryption circuit is a composition of restricted depth-3 circuit $\sum\prod\sum$ and several low depth circuits. By applying our technique, we show that any SHE with a decryption circuit of type $[\cdot]_q \bmod p$ can be bootstrapped if it can evaluate degree $2\log_w e$ circuits.

Evaluate Double Modulo Reduction

The bottleneck of the bootstrapping process is the homomorphic computation of $[\cdot]_q \bmod p$. We use the terminology "double modulo reduction" to denote $[\cdot]_q \bmod p$. In general, when the plaintext is $\mathbb{Z}_2$, the bootstrapping proceeds via bit operations on binary representations of integers. However, it is not easy to bootstrap a ciphertext when the message space is $\mathbb{Z}_p$, where $p > 2$. We propose a method to evaluate $[\cdot]_q \bmod p$ homomorphically for large p using Gentry and Halevi's idea [GH11a]. As an application, we present a bootstrapping method of [CCK+13] with sufficiently large message. Since the modulus switching technique cannot be used to handle errors of ciphertexts in batch FHE [CCK+13], the selection of

parameters for FHE is heavily dependent on the homomorphic capacity of the scheme. Thus our improved technique plays an important role in bootstrapping ciphertexts.

We assume that q is equivalent to 1 modulo p. Then $[c]_q \bmod p$ can be written as in the DGHV scheme [DGHV10]:

$$[c]_q \bmod p = c - \lfloor c/q \rceil \cdot q \bmod p = c - \lfloor c/q \rceil \bmod p.$$

In comparison with the DGHV scheme, it is hard to express division by p as a low degree polynomial over $\mathbb{Z}_p$ when p is greater than two. To apply the technique in [GH11a], we first modify the division part using Gentry's squashing technique [Gen09]. Let us consider parameters κ, Θ, θ such that $\kappa = \gamma\eta/\rho', \Theta = \omega(\kappa \cdot \log \lambda)$, and $\theta = \lambda$, where γ is a bit length of c, η is a bit length of q which is larger than $\rho' = 2\lambda$.

Set $x_q = \lfloor 2^\kappa/q \rceil$ and choose a Θ-bit random vector $\mathbf{s} = (s_1, \ldots, s_\Theta)$ with Hamming weight θ. Choose a random integer $u_i \in \mathbb{Z} \cap [0, p \cdot 2^\kappa)$ for $i = 1, \ldots, \Theta$ such that $\sum_i s_i u_i = x_q \pmod{p \cdot 2^\kappa}$. Set $y_i = u_i/2^\kappa$, which is smaller than p with κ precision after the binary point. In addition, $[\sum_i s_i y_i]_p = (1/q) - \Delta_q$ for some $|\Delta_q| < 2^{-\kappa}$. To bootstrap a ciphertext c output by a permuted circuit, we first compute $z_i \leftarrow [c \cdot y_i]_p$, keeping only $n = \lceil \log_2 \theta \rceil + 3$ precision after the binary point for $i = 1, \ldots, \Theta$. That is, $[c \cdot y_i] = z_i - \Delta_i$ for some Δ_i with $|\Delta_i| \leq 1/16\theta$. We have

$$\begin{aligned}
\left[(c/q) - \sum_{i=1}^{\Theta} s_i z_i\right]_p &= \left[(c/q) - \sum_{i=1}^{\Theta} s_i [c \cdot y_i]_p - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_p \\
&= \left[(c/q) - c\left[\sum_{i=1}^{\Theta} s_i \cdot y_i\right]_p - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_p \\
&= \left[(c/q) - c\,[(1/q) - \Delta_q]_p - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_p \\
&= \left[c \cdot \Delta_q - \sum_{i=1}^{\Theta} s_i \Delta_i\right]_p .
\end{aligned}$$

Since the bit length of c is at most $2^{\gamma(\eta-4)/(\rho'+2)} < 2^{\kappa-4}$, we have $c \cdot \Delta_q \leq 1/16$. Additionally, it can be observed that $|\sum s_i \Delta_i| \leq \theta \cdot 1/16\theta = 1/16$. Thus, we have

$$[c]_q \bmod p = c - \lfloor c/q \rceil \bmod p = c - \left\lfloor \sum s_i z_i \right\rceil \bmod p. \tag{3.3.1}$$

To apply the chimeric technique [GH11a], we convert the above subset sum into a $\sum\prod\sum$ form.Equation (3.3.1) can be converted as follows:

$$\begin{aligned}
[c]_q \bmod p &\equiv c - \lfloor c/q \rceil \bmod p \\
&\equiv c - \left\lfloor c \cdot \sum_{i=1}^{\Theta} s_i z_i \right\rceil \bmod p \\
&\equiv \underbrace{c - \sum_{i=1}^{\Theta} s_i z_i'}_{\text{simple part}} - \underbrace{\left\lfloor 2^{-n} \sum_{i=1}^{\Theta} s_i z_i'' \right\rceil}_{\text{complicated part}} \bmod p,
\end{aligned}$$

where $z_i = z_i' + z_i'' \cdot 2^{-n}$ for integers $z_i' \in [0, p)$ and $z_i'' \leq 2^n \leq 8\Theta$. As well as the simple part, the "complicated part" can be expressed as a $\mathcal{L}_A$-restricted depth-3 circuit C, provided we choose p such that $p > 8\Theta^2$. Thus we obtain the following theorem:

Theorem 3.3.2. *Let q, p be primes such that $q > p > 8\Theta^2$. For any $A \subset \mathbb{Z}_p$ of cardinality at least $8\Theta^2+1$, the double modulo reduction $[\cdot]_q \bmod p$ can be expressed as $\mathcal{L}_A$-restricted depth-3 circuit C of $\mathcal{L}_A$-degree at most $8\Theta^2$ having at most $8\Theta^2 + \Theta + 1$ product gates. Thus, the double modulo reduction circuit can be evaluated using the chimeric technique.*

Bootstrapping in [CCK$^+$13]

Cheon *et al.* [CCK$^+$13] proposed a FHE based on the Chinese remainder theorem. The decryption of the scheme consists of $[\cdot]_{p_i} \bmod Q_i$ with $i \in [1, k]$. Their scheme only achieves "bootstrapping" when all Q_i's are two. They raised the problem of evaluating the double modulo reduction $[\cdot]_{p_i} \bmod Q_i$ homomorphically when

some of Q_i are greater than 2. We can partially solve this problem with the above technique, and so complete the bootstrapping stage for sufficiently large Q_i. This gives a FHE that deals with large integers. Note that if the Q_i's are relatively prime, we can map a plaintext on $\mathbb{Z}_M$ with $M = \prod_i Q_i$ into $\prod_i \mathbb{Z}_{Q_i}$, thus enabling large integer arithmetic on $\mathbb{Z}_M$. In this bootstrapping procedure, our improved technique for the evaluation of exponentiation plays an important role, since the parameters of the FHE are heavily dependent on the homomorphic capacity of the scheme. Using our method, the homomorphic capacity can be reduced from $2\log e$ to $2\log_w e$ at the cost of public key size $\tilde{\mathcal{O}}(w)$.

3.4 Discussions

We now discuss some typical applications of FHE in database and cloud computing environments and analyze the advantages of our hybrid scheme under various scenarios.

3.4.1 Application Model

Database Encryption

Let us consider the situation in which a government agency collects medical records of patients from a hospital, and extracts some statistical information from the records. When lots of data are stored, it becomes more important to protect the data from misuse by insiders or hacking by outsiders. To reduce the risk, the data may be encrypted prior to storage. Under this scenario, we give an efficient storage solution using a hybrid scheme.

First, the agency generates $(pk_{\mathsf{Hyb}}, sk_{\mathsf{Hyb}})$ using the hybrid scheme. The secret key sk_{Hyb} is stored in a secure area, the public key pk_{PKE} is made public to hospitals, and pk_{Hyb} is made public to a database. Each hospital uploads its medical records to the database after encryption under pk_{PKE}. The agency requests the

database to perform some computations on the patient data to extract information. The database performs homomorphic computations on the encrypted data using pk_{Hyb}. After evaluating the requested computations, the resulting ciphertexts are sent to the agency. The decryption is carried out in the agency's secure area.

Outsourcing of Computations in Cloud Environment

Suppose that a client who has limited computing power wants to perform heavy computation on private data. Our hybrid scheme can be used to protect his privacy, i.e., the computations of PKE-encrypted data are outsourced along with pk_{Hyb} to a cloud that has huge computing power and storage. The cloud performs the outsourced computations, and returns the resulting ciphertext encrypted under SHE. Although the fact that the client must send the large-size pk_{Hyb} appears to be a weakness of our hybrid scheme, this is only a minor concern, since pk_{Hyb} is sent to the cloud only once in the outsourcing procedure.

Remark 3.4.1. Since the client may not trust the cloud, it is desirable that the cloud can prove that the computations on the encrypted data were done correctly. Proof of the correctness when using only FHE was given in [CKV10]. Further study on this problem is needed when the hybrid scheme is used to delegate the computations.

3.4.2 Advantages

The advantages of using our scheme in the above scenarios include small bandwidth, reduced storage requirements, and computational efficiency. In this section, we consider the scale invariant fully homomorphic encryption based on RLWE proposed in [FV12].

Schemes	SHE only	AES with SHE	EGM with SHE
Transmitted Ciphertext	$\lceil \frac{\mu}{2 \cdot 10^3} \rceil \cdot$600KB	128$\cdot$ 16MB+ $\lceil \frac{\mu}{128} \rceil \cdot$128bit	$\lceil \frac{\mu}{1024} \rceil \cdot$3072bit
Ciphertext Expansion	2400	1	3
SHE Ciphertext	600KB	16MB	3GB
Public-Key	1.2MB	32MB	3TB
Multiplicative Depth of SHE	10	50	20

Table 3.1: Comparison between SHE schemes, the hybrid scheme of AES with SHE, and the hybrid scheme of EGM with SHE in regard to the transmitted ciphertext size, ciphertext expansion ratio, SHE ciphertext size, public key size, and multiplicative depth of SHE.

Small Bandwidth and Storage Saving

In a cloud environment, each client encrypts their messages using limited computing power and storage and a server manages the encrypted data with its large computing power and storage. However, the current FHE's may not be suited to this environment, because their large ciphertext size entails a large communication cost. For example, the ciphertext size of scale-invariant FHE based on RLWE allowing multiplicative depth ten in [FV12] is about 600KB for 80-bit security: the dimension n is 10067 and the modulus q is a 230-bit integer.

In the above scenario, encryptors (each hospital or client) can encrypt data using an efficient AES or PKE scheme to reduce the bandwidth, instead of an inefficient SHE only. Ciphertexts of only a few thousand bits are then sent to

the database. This could reduce the encryptors' bandwidth dramatically. When transmitting μ-bit data using the hybrid scheme of AES and SHE, encryptors need to send encryptions of their own AES secret key to convert AES ciphertexts into SHE ciphertexts as well as data encryptions. We call it by *conversion key*. Since the size of SHE ciphertext which supports multiplicative depth fifty (forty for decryption of AES and ten for homomorphic evaluation) is 16MB, the encryptors send additional 128×16MB conversion key.[†]

On the other hand, when using the hybrid scheme of EGM and SHE, encryptors only send EGM encryption of size $\lceil \frac{\mu}{1024} \rceil \cdot 3072$bit to send μ-bit without any additional conversion key. The conversion key of EGM scheme are sent to the server by decryptor (the government agency).

After receiving the ciphertext from each encryptors, the server stores and computes on the ciphertexts which contain secret information of encryptors. The server can reduce the storage requirement by storing only small AES or PKE ciphertexts, rather than large SHE ciphertexts. The server converts these to SHE ciphertexts and computes the necessary operations only when required. In the case of hybrid scheme of AES and SHE, the server have to store the conversion key of each clients as well as the public key (containing evaluation key) to evaluate functions on encrypted data. In the hybrid scheme of AES and SHE, the public key size is $4n \log q = 32$MB. [‡]

Let us consider the hybrid scheme of EGM and SHE. The message space of the EGM scheme is $\mathbb{Z}_N^\times$ for a 1024 -bit integer $N = pq$, and the ciphertext is of the form $(C_1, C_2, C_3) \in (\mathbb{Z}_N^\times)^3$. To evaluate the decryption of EGM, we choose the message space of SHE to be $\mathbb{Z}_N$. In scale invariant SHE based on RLWE [FV12],

[†] We refer parameter analysis of [GHS12b]. In state-wise bit-slicing variant of AES evaluation, they encrypt each bit of AES secret key separately. When the SHE allows homomorphic multiplications unto depth fifty, the degree n of the base ring is 51234, and the modulus q is a 1250-bit integer. We can obtain $2n \log q (= 16$MB) size SHE ciphertext.

[‡]Here, if we use a new key switching technique proposed in [GHS12b], the evaluation key size is $2n \log q$.

we choose the dimension and modulus as follows:

$$\begin{aligned} n &= \frac{(L(\log t + \log n + 23) - 8.5)(\kappa + 110)}{7.2} \\ \log q &= L(\log t + \log n + \log\log q), \end{aligned}$$

for multiplicative depth L, message space $\mathbb{Z}_t$ and security parameter κ. Thus, we obtain the ciphertext size of 3GB by substituting $L = 20$ and $\log t = 1024$. In this case, we have to add encryptions of EGM secret key in the public key to convert EGM ciphertexts into SHE ciphertexts. Since q_1 and q_2 are 512 bit integers, we add 1024 additional ciphertexts to the public key when using binary expansion of q_1 and q_2. Therefore, the size of public key is 3TB. If we can evaluate the modulo N arithmetic under SHE with a smaller message space, we expect to reduce the size of ciphertext and public key of SHE. In the hybrid scheme of EGM and SHE, the public key size is much more huge. However, differently from the hybrid scheme of AES and SHE, the advantage is that the server do not need to store each encryptor's conversion key.

Our hybrid scheme of EGM and SHE has more efficient bandwidth and storage than the hybrid scheme of AES and SHE scheme when more than 1500 encryptors participate in the above scenario and each encryptor sends data whose size is smaller than 1GB. [§]

We summarize the theoretical comparison of the size of the transmitted ciphertext, SHE-ciphertext and public key in Table 3.1 when each cryptosystem allows homomorphic evaluation of depth ten multivariate functions.

Efficient Computing

In our hybrid scheme, we can choose an additive or multiplicative homomorphic PKE depending on the property of the circuit we are to evaluate. Suppose that the

[§]Suppose that m encryptors participate in and each encryptor sends k-bit data to the server. Then our approach is more advantageous when 3TB $<$ $(2m)$GB and $3k$-bit $<$ (2GB+ k-bit).

server is to evaluate multivariate polynomials f and g, where f has polynomially many monomials on inputs and g has polynomially many linear factors. We will use a multiplicative homomorphic PKE in the first case, and an additive homomorphic PKE in the second case.

First, let us consider

$$f(x_1, \ldots, x_n) = \sum_{i \in I} M_i(x_1, \ldots, x_n),$$

which is an n-variable polynomial over a ring R. Each M_i is a monomial of f. If the degree of f is large, several decryption or modulus switching procedures are required when using ordinary FHE. These are slow, or increase the ciphertext size. However, using our hybrid scheme, we can evaluate f without bootstrapping, regardless of its degree. Suppose the ciphertexts are encrypted under a multiplicative encryption $\mathcal{E}$ with key (pk, sk) and SHE can evaluate the decryption circuit $\mathcal{D}(sk, \cdot)$ of $\mathcal{E}_{pk}$. Given $c_1 = \mathcal{E}_{pk}(m_1), \ldots, c_n = \mathcal{E}_{pk}(m_n)$, we can compute $\mathsf{SHE.Enc}(f(m_1, \ldots, m_n))$ with $\mathsf{SHE.Enc}(sk)$. Below, we denote $\mathsf{SHE.Enc}$ by SHE and $\mathsf{SHE.Dec}(a) = \mathsf{SHE.Dec}(b)$ by $a \equiv b$.

$$\begin{aligned}
& \mathsf{SHE}(f(m_1, \ldots, m_n)) \\
\equiv \quad & \sum_i \mathsf{SHE}(M_i(m_1, \ldots, m_n)) \\
& (\because \mathsf{SHE} \text{ is additive homomorphic.}) \\
= \quad & \sum_i \mathsf{SHE}\{\mathcal{D}(sk, \mathcal{E}_{pk}(M_i(m_1, \ldots, m_n)))\} \\
& (\because \mathcal{D} \circ \mathcal{E} \text{ is an identity}) \\
= \quad & \sum_i \mathsf{SHE}\{\mathcal{D}(sk, M_i(\mathcal{E}_{pk}(m_1), \ldots, \mathcal{E}_{pk}(m_n)))\} \\
& (\because \mathcal{E} \text{ is multiplicative homomorphic.}) \\
\equiv \quad & \sum_i \bar{\mathcal{D}}(\mathsf{SHE}(sk), \mathsf{SHE}(M_i(\mathcal{E}_{pk}(m_1), \ldots, \mathcal{E}_{pk}(m_n)))),
\end{aligned}$$

where $\bar{\mathcal{D}}$ is the decryption circuit encrypted with SHE and the last equality holds because SHE can evaluate $\mathcal{D}$ with $\mathsf{SHE}(sk)$. More specifically, we follow the steps:

1. Given $c_1 = \mathcal{E}_{pk}(m_1), \cdots, c_n = \mathcal{E}_{pk}(m_n)$, compute $M_i(c_1, \cdots, c_n) = \mathcal{E}_{pk}(M_i(m_1, \cdots, m_n))$ for each i.

2. Encrypt $\mathcal{E}_{pk}(M_i(m_1, \cdots, m_n))$ with SHE and then evaluate the decryption circuit $\bar{\mathcal{D}}$ with the encrypted secret key $\mathsf{SHE}(sk)$ of $\mathcal{E}_{pk}$. (Observe: one may use trivial encryption on $\mathcal{E}_{pk}(M_i)$.)

3. Add them to obtain $\mathsf{SHE}(f(m_1, \ldots, m_n))$.

Now let us consider that the server is to compute a polynomial $g(x_1, \ldots, x_n) = \prod_{i \in I} L_i(x_1, \ldots, x_n)$ where each L_i is a linear multivariate factor of g. Suppose that the ciphertexts are encrypted under an additive homomorphic encryption $\mathcal{E}$. In this case, we follow the steps:

1. Given $c_1 = \mathcal{E}_{pk}(m_1), \cdots, c_n = \mathcal{E}_{pk}(m_n)$, compute $L_i(c_1, \cdots, c_n) = \mathcal{E}_{pk}(L_i(m_1, \cdots, m_n))$ for each i.

2. Encrypt $\mathcal{E}_{pk}(L_i(m_1, \cdots, m_n))$ with SHE and then evaluate the decryption circuit $\bar{\mathcal{D}}$ with the encrypted secret key $\mathsf{SHE}(sk)$ of $\mathcal{E}_{pk}$. (Observe: one may use trivial encryption on $\mathcal{E}_{pk}(L_i)$.)

3. Multiply them to obtain $\mathsf{SHE}(f(m_1, \ldots, m_n))$.

Remark 3.4.2. After converting the ciphertextsm we could compute on them under SHE rather than PKE. Therefore, better use is made of the hybrid scheme when evaluating fixed multivariate polynomials.

3.5 Generic Conversion of SHE from Private-Key to Public-Key

Rothblum [Rot11] showed the way how to transform any additively homomorphic private-key encryption scheme into a public-key homomorphic encryption scheme

when the message is $\mathbb{Z}_2$. To apply this method, the private-key SHE needs to be compact which means that the length of a homomorphically generated encryption is independent of the number of ciphertexts from which it was created. An additive homomorphic encryption is converted from private-key to public key by adding a number of encryptions of zero and one to the public key.

We can consider our hybrid scheme to be a generic conversion of SHE from private-key to public key whose message space is $\mathbb{Z}_p$ for some large prime p. We need only add encryptions of the secret key of a PKE under the private SHE to the public key instead of $\{\mathsf{SHE.Enc}_i(0)\}_i$ and $\{\mathsf{SHE.Enc}_i(1)\}_i$. The encryption algorithm of "public-key" SHE is made up of $\mathsf{Hyb.Enc}$ and $\mathsf{Hyb.Conv}$. Given a hybrid scheme $\mathsf{Hyb} = (\mathsf{Hyb.KG}, \mathsf{Hyb.Enc}, \mathsf{Hyb.Conv}, \mathsf{Hyb.Dec}, \mathsf{Hyb.Eval})$ of PKE and a private key SHE scheme $\mathsf{PrivSHE}$, we could construct a public key SHE PubSHE as follows:

$\underline{\mathsf{PubSHE.KG}(\lambda)}$: Run $\mathsf{PKE.KG}$ to obtain pk_{PKE} and sk_{PKE}. Output a public key $pk_{\mathsf{PubSHE}} = (pk_{\mathsf{Hyb}})$ and a secret key $sk_{\mathsf{PubSHE}} = (sk_{\mathsf{Hyb}})$.

$\underline{\mathsf{PubSHE.Enc}(pk_{\mathsf{PubSHE}}, m)}$: For a plaintext $m \in \mathbb{Z}_p$, encrypt m under $\mathsf{Hyb.Enc}$ and then output a ciphertext $C \leftarrow \mathsf{Hyb.Conv}(pk_{\mathsf{PubSHE}}, c)$ where $c \leftarrow \mathsf{Hyb.Enc}(m)$.

$\underline{\mathsf{PubSHE.Dec}(sk_{\mathsf{PubSHE}}, C)}$: For a ciphertext C, output a message $m \leftarrow \mathsf{Hyb.Dec}(sk_{\mathsf{PubSHE}}, C)$.

The semantic security of this conversion follows that of the hybrid scheme.

Chapter 4

A Hybrid Asymmetric Homomorphic Encryption

In [GHS12b], Gentry et. al suggested a hybrid scheme to encrypt a message by the block cipher AES with a session key S and send it with an encryption of the session key S under a FHE. By homomorphically evaluating (decrypting with the key S while keeping all the informations encrypted under the FHE) them, the ciphertexts $\mathrm{AES}_S(m)$ under AES are converted into ciphertexts $\mathsf{FHE}(m)$ under the FHE of the same message. This scheme can reduce the ciphertext size significantly, but requires a large computation cost for conversion. Also the underlying FHE should be able to homomorphically evaluate circuits of more than 40 depth, and so can not be used with a somewhat homomorphic encryption scheme of homomorphic capacity 40 or less.

We start with an asymmetric leveled SHE having a *switch key* $\mathrm{SWK}_{S:S_L}$, with which a conversion algorithm transforms a ciphertext $\mathrm{SHE}_S(m)$ of a message m with the private key S into $\mathrm{SHE}_{S_L}(m)$ of the same message with the private key S_L of lower level. We have several candidate schemes with such a property [BV11, BGV12, Bra12, CNT12, CLT14].

We consider a public key compression technique in [CNT12] to reduce the SHE-

ciphertext size. In the DGHV scheme [DGHV10] and the LWE-based schemes [BV11, Bra12], the public key is a set of encryptions of the zero and so the public key compression techniques is essentially the ciphertext compression in its symmetric version. More precisely, in the DGHV shceme, the $\mathrm{SSHE}_S(m)$ is compressed into a seed se and its correction value $\delta(m)$ such that $\mathrm{PRNG}(\mathsf{se}) + \delta(m) = \mathrm{SSHE}_S(m)$. In the LWE-based schemes, the ciphertext is of the form $(\mathbf{b}, \mathbf{A})$ where a matrix $\mathbf{A}$ is generated from $\mathrm{PRNG}(\mathsf{se})$ and can be compressed into a small seed se and its correction value $\delta(m) = \mathbf{b}$. However, this technique can not be applied to its *asymmetric* versions where an encryption of a message m is made from a sparse subset sum of the ciphertexts of the zero instead of choosing a random parts of ciphertext.

Then a hybrid encryption of a message m is composed of the compressed ciphertext $(\mathsf{se}, \delta(m))$ of $\mathrm{SSHE}_S(m)$ along with the switch key $\mathrm{SWK}_{(S:S_L)}$. On receiving a ciphertext $(\mathrm{SWK}_{(S:S_L)}, \mathsf{se}, \delta(m))$, recover $\mathrm{SSHE}_S(m)$ from $(\mathsf{se}, \delta(m))$ and convert it to $\mathrm{SSHE}_{S_L}(m)$ with $\mathrm{SWK}_{(S:S_L)}$. This procedure is possible even when the SSHE has low homomorphic capacity. A conversion is done by a matrix multiplications for LWE-base SHE and inner products for the DGHV scheme and so very fast. In the leveled homomorphic encryption schemes, the switch key $\mathrm{SWK}_{(S:S_L)}$ is made by one who knows both of the private key S and S_L, but in this scenario the secret key S_L is not available to an encryptor. We provide an algorithm to make the switch key $\mathrm{SWK}_{(S:S_L)}$ without knowing the secret key S_L.

We apply this technique to the scale-invariant SHE by Brakerski [Bra12]. In that case, the switch key size is $n^2 \log^2 q$. By adopting the technique in [GHS12b], we reduce it to $n^2 \log q$. It can be further reduced into $n \log q$ if an encryptor sends his session key S to the data owner having S_L in order to make a compressed switch key for $\mathrm{SWK}_{(S:S_L)}$. We also apply this technique to the scale-invariant DGHV scheme [CLT14], in which the switch key size is 600 GB too large. But it has fairly good advantages to large integer arithmetics when the compressed switch key is available through communications between an encryptor and the

data owner.

Our hybrid scheme can be applied to a situation where a server needs to perform homomorphic computation on ciphertexts with different decryption keys: each user encrypts their data using symmetric FHE and send them to the server along with conversion key. The server converts ciphertexts when it needs homomorphic computations.

4.1 Preliminaries

Notations. For an integer q, $[x]_q$ denotes an integer in $(-q/2, q/2] \cap \mathbb{Z}$ that is equvalent to $x \pmod q$. We use $x \leftarrow \mathcal{D}$ to denote that x is sampled from a distribution $\mathcal{D}$. Similarly, $x \leftarrow S$ denotes that x is sampled from the uniform distribution over a set S. We use $\lfloor x \rceil$ to indicate rounding x to the nearest integer, and $\lfloor x \rfloor$, $\lceil x \rceil$ (for $x \geq 0$) to indicate rounding down or up. The $\log x$ denotes the logarithm of x to base 2.

Vectors, Matrices and Tensors. We denote scalars in plain as a and (column) vectors in bold lowercase as $\mathbf{a}$, and matrices in bold uppercase as $\mathbf{A}$. We use $(\mathbf{x}, \mathbf{y})$ to refer to the vector $[\mathbf{x}^T||\mathbf{y}^T]^T$ where $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. The ℓ_i norm of a vector of $\mathbf{v}$ is denoted by $||\mathbf{v}||_i$. The inner product of $\mathbf{v}$ and $\mathbf{u}$ is denoted by $\langle \mathbf{v}, \mathbf{u} \rangle$, i.e. $\langle \mathbf{v}, \mathbf{u} \rangle = v^T u$. For a n dimensional vector $\mathbf{v}$, the i-th element of $\mathbf{v}$ is denoted by $v[i]$ for $i = 1, ..., n$. The operations $[\cdot]_q, \lfloor \cdot \rceil, \lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are applied element-wise when they applied to vectors. The tensor product of two vectors $\mathbf{v}, \mathbf{w}$ of dimension n, denoted $\mathbf{v} \otimes \mathbf{w}$, is the n^2 dimensional vector $(\mathbf{v}[1]\mathbf{w}, \ldots, \mathbf{v}[n]\mathbf{w})$. Note that $\langle \mathbf{v} \otimes \mathbf{w}, \mathbf{x} \otimes \mathbf{y} \rangle = \langle \mathbf{v}, \mathbf{x} \rangle \langle \mathbf{w}, \mathbf{y} \rangle$.

4.2 A Hybrid Approach to Asymmetric FHE with Compressed Ciphertext

In this section, we present building blocks from previous works are used in our construction.

4.2.1 Main Tools

We adapt the concept of (fully) homomorphic encryption introduced in [Gen09].

Public-key Homomorphic Encryption.

A public-key homomorphic encryption scheme PHE has four algorithms $\mathsf{KG}_{\mathsf{PHE}}$, $\mathsf{Enc}_{\mathsf{PHE}}$, $\mathsf{Dec}_{\mathsf{PHE}}$ and an additional algorithms $\mathsf{Eval}_{\mathsf{PHE}}$ that takes as input the evaluation key ek, a circuit C and a tuple of ciphertexts $(c_1, \ldots, c_n)$; it outputs a ciphertext c. The computational complexity of all of these algorithms must be polynomial in security parameter λ and the size of C. PHE is correct for circuits in $\mathcal{C}_{\mathsf{PHE}}$, if for any key pair (sk, pk) output by $\mathsf{KG}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathsf{PHE}}$, any plaintexts $(m_1, \ldots, m_n)$ and any ciphertexts $(c_1, \ldots, c_n)$ with $c_i \leftarrow \mathsf{Enc}_{\mathsf{PHE}}(pk, m_i)$, it is the case that

$$c \leftarrow \mathsf{Eval}_{\mathsf{PHE}}(\mathsf{ek}, C, (c_1, \ldots, c_n)) \Rightarrow C(m_1, \ldots, m_n) = \mathsf{Dec}_{\mathsf{PHE}}(sk, c).$$

Definition 4.2.1 (Homomorphic Encryption)**.** PHE is homomorphic for circuits in $\mathcal{C}_{\mathsf{PHE}}$ if PHE is correct for $\mathcal{C}_{\mathsf{PHE}}$ and $\mathsf{Dec}_{\mathsf{PHE}}$ can be expressed as a circuit of size $\mathrm{poly}(\lambda)$.

Definition 4.2.2 (Fully Homomorphic Encryption)**.** PHE is fully homomorphic if it is homomorphic for all circuits.

Compressible Secret-key Homomorphic Encryption.

We introduce a symmetric-key somewhat homomorphic encryption SSHE with a key sk has *compressible ciphertexts* using Pseudo-Random Number Generator (PRNG). We consider a public key compression technique in [CNT12]. In the DGHV scheme [DGHV10] and some LWE-based schemes [BV11, Bra12], the public key is a set of encryptions of the zero and so the public key compression techniques is essentially the ciphertext compression in its symmetric version. More precisely, in the LWE-based schemes, the ciphertext is of the form $(\mathbf{b}, \mathbf{A})$ where a matrix $\mathbf{A}$ is generated from PRNG(se) and can be compressed into a small seed se. In DGHV, the $\mathsf{SSHE}(sk, m)$ is compressed into a seed $\mathbf{se}$ and its correction value $\delta_{sk}(m)$ such that PRNG(se) + $\delta_{sk}(m)$ = $\mathsf{SSHE}(sk, m)$. We give the formal definition of *compressible* symmetric-key homomorphic encryption.

Definition 4.2.3 (α-Compressible Symmetric-key Homomorphic Encryption)**.** Let SSHE be a symmetric-key somewhat homomorphic encryption. SSHE is α-*compressible* if it satisfies the following:

- For a random number r and a message m, there is a correction function $\delta_{sk} : \mathbb{Z} \times \mathsf{M} \to \{0,1\}^{|\mathbf{C}|}$ such that $\mathsf{SSHE.Dec}_{sk}(f(r, \delta_{sk}(r, m))) = m$ for some fixed (publicly evaluated) function f.

- The following distribution

  $$\mathcal{C}_{sk}(m) = \Big\{ r \leftarrow \{0,1\}^{|\mathbf{C}|} : \text{ Output } f\big(r, \delta_{sk}(r, m)\big) \text{ such that } \mathsf{SSHE.Dec}_{sk}\big(f(r, \delta_{sk}(r, m))\big) = m \Big\}$$

  is computationally close to the distribution $\mathsf{SSHE.Enc}_{sk}(m)$ under some hard problem.

- A ciphertext $\mathsf{SSHE.Enc}(m)$ is compressed into $(\mathsf{se}, \delta_{sk}(r, m))$ when a random r is generated by pseudorandom number generator with seed $\mathsf{se} \in \{0,1\}^{\lambda}$ for the security parameter λ.

- We define $\alpha = \frac{\lambda+|\{\delta_{sk}(r,m)\}|}{|\mathsf{SSHE.Enc}_{sk}(m)|} < 1$, where $|A|$ denotes a bound for bit-length of all elements in the set A.

Public Generation of Switching Key.

The public key of (leveled) fully homomorphic encryption includes additional components to enable converting a valid ciphertext with respect to one key into a valid ciphertext encrypting the same plaintext with respect to another key [CLT14, BGV12, Bra12]. This can be used to convert the product ciphertext which is valid with respect to a high-dimension key (e.g. tensored secret key) back to a ciphertext with respect to the original low-dimension key. The key consists of encryptions of secret key under the original key in some sense.

Almost all the previous FHE schemes use the key switching technique to enhance the efficiency of the homomorphic evaluation on ciphertexts. Therefore, a party who runs SwitchKG algorithm to obtain the switching key knows both secret key. Differently from them, in our scenario the data owner who encrypts data generates the key switching gadget that converts ciphertexts under the data owner's key into ciphertexts under the data analyst's key. It is reasonable that the data owner cannot access the secret-key of the data analyst. To enable public generation of switching key, the data analyst executes supplementary algorithm that provides additional key, we call this algorithm by *auxiliary key generation* AKG and this key by *auxiliary key* ak. The auxiliary key contains secret information on the original secret key in encrypted form. For given auxiliary key ak, encryptor publicly generates switching key that can convert the underlying secret key from data owner's into data analyst, we call it by *conversion key* ck. The data owner runs *public switching key generation algorithm* PubSwitchKG using auxiliary key to obtain conversion key.

4.2.2 Hybrid Encryption with Compressed Ciphertexts

In this section, we construct a *hybrid encryption with compressed ciphertexts* using a public-key somewhat homomorphic encryption with switching key algorithm and a compressible secret-key somewhat homomorphic encryption. We consider a situation that there are independent three parties who participate in this scheme, a key generator(decryptor), an encryptor and an evaluator.

- The key generator makes key pair $(pk, sk, \mathsf{evk}, \mathsf{ak})$ and send (pk, ak) to the encryptor and (pk, evk) to the evaluator.

- The encryptor uses his own secret to encrypt data and send them with conversion key to the evaluator.

- The evaluator converts key of a bundle of ciphertext to be evaluated, and then evaluate the function to be requested.

We give the formal construction on hybrid encryption scheme with compressed ciphertexts. Let $\mathsf{PSHE} = (\mathsf{PSHE.KG}, \mathsf{PSHE.Enc}, \mathsf{PSHE.Eval}, \mathsf{PSHE.Dec})$ be a public-key somewhat homomorphic encryption and $\mathsf{SSHE} = (\mathsf{SSHE.KG}, \mathsf{SSHE.Enc}, \mathsf{SSHE.Eval}, \mathsf{SSHE.Dec})$ be a α-compressible secret-key somewhat homomorphic encryption. We construct a *hybrid public-key encryption scheme* Hyb by combining PSHE and SSHE.

- $\mathsf{Hyb.KG}(\lambda)$: Run $\mathsf{PSHE.KG}(\lambda)$ to obtain (pk, sk, evk) and auxiliary key generation algorithm AKG to obtain auxiliary key ak. Output a public key pk, an evaluation key evk, a secret key sk, and an auxiliary key ak.

- $\mathsf{Hyb.ConvKG}(pk, \mathsf{ak})$: Run $\mathsf{SSHE.KG}(\lambda)$ and $\mathsf{PubSwitchKG}(pk, \mathsf{ak}, sk')$ to obtain (sk', evk') and conversion key ck', respectively. (Observation: ck' resembles an encryption of sk', but does not decrypt to it)

- $\mathsf{Hyb.Enc}(sk', m)$: Compute $r = \mathsf{PRNG}(\mathsf{se})$ for a random seed se and find a correction $\delta_{sk}(r, m)$ such that $\mathsf{SSHE.Dec}_{sk'}(f(r, \delta_{sk}(r, m)) = m$. Output a ciphertext $(\mathsf{se}, \delta_{sk}(r, m))$.

- $\mathsf{Hyb.Conv}(\mathsf{ck}', (\mathsf{se}, \delta(m)))$: Compute $r = \mathsf{PRNG}(\mathsf{se})$ to obtain a ciphertext $c' = f(r, \delta_{sk}(r, m))$. Run $\mathsf{SwitchKey}_{\mathsf{ck}'}(c')$ to obtain $c = \mathsf{PSHE.Enc}_{pk}(m)$. (Observation: $\mathsf{SwitchKey}_{\mathsf{ck}'}$ converts the underlying secret key of c' from sk' into sk.)

- $\mathsf{Hyb.Eval}(\mathsf{evk}, \mathsf{ck}', C, (\mathsf{se}_1, \delta_{sk}(r_1, m_1)), \ldots, (\mathsf{se}_n, \delta_{sk}(r_n, m_n)))$: Compute $\mathsf{PSHE.Eval}(\mathsf{evk}, C, c_1, \ldots, c_n)$ for $c_i \leftarrow \mathsf{Hyb.Conv}(\mathsf{ck}', ((\mathsf{se}_i, \delta_{sk}(r_i, m_i)))$.

- $\mathsf{Hyb.Dec}(sk, c)$: Output $m' \leftarrow \mathsf{PSHE.Dec}_{sk}(c)$.

4.3 Concrete Hybrid Constructions

4.3.1 Hybrid Encryptions based on DGHV and Its Variants

In this section, we consider a variant of DGHV scheme that has key switching algorithm and its compressible symmetric version. We adapt the key switching procedure and public key compression technique introduced in [CNT12].

First, let us recall the scale-invariant DGHV scheme proposed by [CLT14]. Let p be a η bit odd integer for secret key and $\mathbb{Z}_t$ be a message space for a integer $t \geq 2$. A ciphertext is of the form $c = q \cdot p^2 + r + m \cdot \lfloor p/t \rceil$ for a message m where $q \leftarrow [0, 2^\gamma/p^2), r \leftarrow (-2^\rho, 2^\rho)$ for some parameters γ, ρ. The parameters ρ, η and γ are chosen by taking into consideration the multiplicative depth of the circuit to be evaluated and the security issues. Especially, a random q of bit length $\gamma - \eta$ is quite large compared to p and r. If we generate q-part in the

ciphertext c with pseudorandom number generator with seed se, we can compress the ciphertext c. The straightforward way is to use modulo reduction: compute $\delta_p(m) = -[\chi]_{p^2} + r + m \cdot \lfloor p/t \rceil$ for a γ-bit random integer χ generated by $\mathsf{PRNG}(\mathsf{se})$ and output a compressed ciphertext $(\mathsf{se}, \delta_p(m))$. Then the ciphertext size is reduced from γ-bit into η-bit.

Since the compressed ciphertext can be generated only with a secret key p, we need to provide conversion key to convert this ciphertext back to the original public key. We consider the key switching algorithm for the DGHV scheme given in [CNT12].

Key Switching.

In [CNT12], the authors described a technique for switching moduli in DGHV scheme. Given a DGHV ciphertext $c' = q' \cdot p' + r'$, they first convert c into a virtual ciphertext of the form $c'' = 2^k \cdot q'' + r''$ with $[q'']_2 = [q']_2$ using the bits s_i satisfying the followings:

$$\frac{2^k}{p'} = \sum_{i=1}^{\Theta} s_i \cdot z_i + \epsilon \bmod 2^{k+1},$$

where z_i is κ-bit precision after binary point and $|\epsilon| \leq 2^{-\kappa}$. First, the initial ciphertext c' is expanded by multiplying the $z_i's$ and then collapsed into ciphertext c'' using the secret key $\mathbf{s} = (s_1, \ldots, s_\Theta)$. However we cannot reveal $\mathbf{s}$ intactly, so instead a DGHV encryption of the secret key bit s_i under secret key p. Then the expanded ciphertext can be converted into a new ciphertext c under p instead of p' for the same plaintext. Furthermore, the noise in the ciphertext is reduced by a factor p/p' as in the RLWE scheme.

Coron et.al proposed a scale-invariant DGHV scheme by moving the plaintext bit from LSB to the MSB of $[c \bmod p]$ and working modulo p^2 [CLT14]. When multiplying two fresh ciphertexts c_1 and c_2, the resulting ciphertext contains the plaintext bit in the MSB of $[2c_1c_2 \bmod p^2]$. They give a method of converting the result of a ciphertext multiplication back to a ciphertext usable in subse-

quent homomorphic operations. Their procedure to convert uses essentially the same technique as the modulus switching technique for DGHV in [CNT12]. As the modulus switching enables to convert a ciphertext under a secret p into a new ciphertext under a secret p', their method is to convert ciphertext under p^2 back to a ciphertext under p.

In the scale-invariant DGHV scheme, they do not change the underlying secret key p. To adapt this scheme to our scenario, we slightly modify the convert algorithm in [CLT14] to convert secret-key p into p'. We consider a construction that a message is encrypted MSB of $[c \bmod p]$ and working modulo p not p^2 to reduce ciphertext size further (by factor two). That is, a fresh ciphertext c is of the form $c' = r' + m \cdot (p'-1)/2 + q' \cdot p'$ and a key switched ciphertext is $c = r + m \cdot (p-1)/2 + q \cdot p^2$. Furthermore, to reduce the size of switching key size as in [CLT14], we use words of size w bits instead of using $\mathsf{BitDecomp}$ and $\mathsf{Powersof2}$. This decreases the size of the vector by a factor w at the cost of increasing the resulting noise by roughly w bits. We define $\mathsf{BitDecomp}_w$ and $\mathsf{Powersof}_w$:

- $\mathsf{BitDecomp}_w(\mathbf{v}, \alpha)$: For a vector $\mathbf{v} \in \mathbb{Z}^n$, let $\mathbf{v}_i \in (\mathbb{Z} \cap [0, 2^w))^n$ be such that $\mathbf{v} \bmod 2^{\alpha \cdot w} = \sum_{i=0}^{\alpha-1} \mathbf{v}_i \cdot (2^w)^i$. Output the vector

$$(\mathbf{v}_0, \ldots, \mathbf{v}_{\alpha-1}) \in (\mathbb{Z} \cap [0, 2^w))^{n \cdot \alpha}$$

- $\mathsf{Powersof}_w(\mathbf{u}, \alpha)$: For a vector $\mathbf{u} \in \mathbb{Z}^n$, outputs the vector

$$(\mathbf{u}, 2^w \cdot \mathbf{u}, (2^w)^2 \cdot \mathbf{u}, \ldots, (2^w)^{\alpha-1} \cdot \mathbf{u})$$

We can easily verify

$$\langle \mathbf{v}, \mathbf{u} \rangle = \langle \mathsf{BitDecomp}_w(\mathbf{v}, \alpha), \mathsf{Powersof}_w(\mathbf{u}, \alpha) \rangle$$

for any $\alpha, w \in \mathbb{Z}$.

Now, we give a modified key-switching procedure in the scale-invariant DGHV scheme.

- $\mathsf{DGHV.SwitchKG}(pk, sk, pk', sk')$:

  1. Take as input two scale-invariant DGHV secret-keys p and p' of size η and η'. Let $\kappa = \gamma + 2$ where γ is the size of public key x_i.

  2. Generate a vector $\mathbf{z}$ of Θ random rational numbers with κ bits of precision after binary point and a random binary vector $\mathbf{s}$ of dimension Θ such that $2^\eta/p' = \langle \mathbf{s}, \mathbf{z} \rangle + \epsilon \bmod 2^\eta$ where $|\epsilon| \leq 2^{-\kappa}$. Compute the expanded secret-key $\mathbf{s}' = \mathsf{Powersof}_w(\mathbf{s}, \eta/w)$.

  3. Compute a vector of encryption σ of $\mathbf{s}'$ under p, defined as

  $$\sigma = \mathbf{q} \cdot p^2 + \mathbf{r} + \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}} \right\rceil$$

  where $\mathbf{q} \leftarrow (\mathbb{Z} \cap [0, 2^\gamma/p^2))^{(\eta/w)\cdot\Theta}$ and $\mathbf{r} \leftarrow (\mathbb{Z} \cap (-2^\rho, 2^\rho))^{(\eta/w)\cdot\Theta}$.

  4. Output $(\mathbf{z}, \sigma)$.

- $\mathsf{DGHV.SwitchKey}(\mathbf{z}, \sigma, c')$

  1. Compute the expanded ciphertext $\mathbf{c} = (\lfloor c' \cdot z_i \rceil \bmod 2^\eta)_{1 \leq i \leq \Theta}$ and let $\mathbf{c}' = \mathsf{BitDecomp}_w(\mathbf{c}, \eta/w)$.

  2. Output $c = 2\langle \sigma, \mathbf{c}' \rangle$.

Lemma 4.3.1 (Correctness)**.** *Let $c' = r' + m \cdot (p'-1)/2 + q' \cdot p'$ with $|r'| \leq 2^{\rho'}$. Then the procedure* SwitchKey *converts the ciphertext c' into a ciphertext $c = r + (2r^* + m) \cdot (p-1)/2 + q \cdot p^2$ with noise $|r^*| \leq 2^w \cdot \Theta$ and $|r| \leq 2^{\rho'+\eta-\eta'+4} + \Theta \cdot \eta/w \cdot 2^{\rho+w+2}$.*

Proof. We have

$$c = 2\langle \sigma, \mathbf{c}' \rangle = 2p^2 \cdot \langle \mathbf{q}, \mathbf{c}' \rangle + 2\langle \mathbf{r}, \mathbf{c}' \rangle + 2 \left\langle \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}}, \right\rceil, \mathbf{c}' \right\rangle.$$

Since the components of $\mathbf{c}'$ are w bits integers, we have

$$2\left\langle \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}}, \right\rceil, \mathbf{c}' \right\rangle = \frac{p}{2^{\eta}} \cdot \langle \mathbf{s}', \mathbf{c}' \rangle + \nu_1, \tag{4.3.1}$$

where $\nu_1 \leq 2^w \cdot \Theta \cdot \eta / w$. We observe that $\langle \mathbf{s}', \mathbf{c}' \rangle = \langle \mathbf{s}, \mathbf{c} \rangle + q_1 \cdot 2^{\eta}$ by the definition of $\mathsf{Powersof}_w$ and $\mathsf{BitDecomp}_w$. We verify

$$\langle \mathbf{s}, \mathbf{c} \rangle = \sum_{i=1}^{\Theta} s_i \lfloor c \cdot z_i \rceil + q_2 2^{\eta} = \sum_{i=1}^{\Theta} s_i \cdot c \cdot z_i + \delta_1 + q_2 \cdot 2^{\eta} = c \cdot \langle \mathbf{s}, \mathbf{z} \rangle + \delta_1 + q_2 \cdot 2^{\eta},$$

for some $q_2 \in \mathbb{Z}$ and $|\delta_1| \leq \Theta/2$. Using $\langle \mathbf{s}, \mathbf{z} \rangle = 2^{\eta}/p' - \epsilon - \mu \cdot 2^{\eta}$ and $c = r' + m \cdot (p'-1)/2 + q' \cdot p'$, we obtain

$$\begin{aligned} \langle \mathbf{s}, \mathbf{c} \rangle &= c \cdot \left(\frac{2^{\eta}}{p'} - \epsilon - \mu \cdot 2^{\eta} \right) + \delta_1 + q_2 \cdot 2^{\eta} \\ &= q' \cdot 2^{\eta} + m \cdot 2^{\eta-1} - m \cdot \frac{2^{\eta-1}}{p'} + r \cdot \frac{2^{\eta}}{p'} - c \cdot \epsilon + \delta_1 + (q_2 - c \cdot \mu) \cdot 2^{\eta}, \end{aligned}$$

and we can write $\langle \mathbf{s}, \mathbf{c} \rangle = q_3 \cdot 2^{\eta} + m \cdot 2^{\eta-1} + r_1^*$ with $|r_1^*| \leq 2^{\rho'+\eta-\eta'+3}$. From the Equation (4.3.1), we have

$$2\left\langle \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}}, \right\rceil, \mathbf{c}' \right\rangle = \frac{p}{2^{\eta}}\big((q_1+q_3)\cdot 2^{\eta} + m \cdot 2^{\eta-1} + r^*\big) + \nu_1 = q_4 \cdot p + m \cdot \frac{p}{2} + \frac{p}{2^{\eta}} \cdot r_1^* + \nu_1,$$

with $|q_4| \leq 2^w \cdot \Theta$; namely the components of $p/2^{\eta+1} \cdot \mathbf{s}'$ are smaller than p and $\mathbf{c}'$ is a w-bits integer vector. This gives

$$2\left\langle \left\lfloor \mathbf{s}' \cdot \frac{p}{2^{\eta+1}}, \right\rceil, \mathbf{c}' \right\rangle = (2q_4 + m) \cdot \frac{p-1}{2} + r_2^*,$$

with $|r_2^*| \leq 2^{\rho'+\eta-\eta'+4} + 2^w \cdot \Theta \cdot \eta / w$. Finally, we obtain

$$\begin{aligned} c =& 2p^2 \cdot \langle \mathbf{q}, \mathbf{c}' \rangle + 2\langle \mathbf{r}, \mathbf{c}' \rangle + (2q_4 + m) \cdot \frac{p-1}{2} + r_2^* \\ =& 2q \cdot p^2 + (2q_4 + m) \cdot \frac{p-1}{2} + r, \end{aligned}$$

where $|r| \leq |r_2^*| + \Theta \cdot \eta / w \cdot 2^{\rho+w+1} \leq 2^{\rho'+\eta-\eta'+4} + \Theta \cdot \eta / w \cdot 2^{\rho+w+2}$, which proves the Lemma. □ □

Remark 4.3.1. In the key switching procedure described above depends on subset sum problem when sharing the secret value $2^{\eta}/p'$. We can avoid the subset problem as follows:

1. For two secret key p and p', define $\sigma = (\sigma_1, \ldots, \sigma_{\gamma/w})$ by

$$\sigma_i = q_i \cdot p^2 + r_i + \left\lfloor \frac{p}{2} \cdot \left[\frac{(2^w)^i \cdot 2}{p'} \right]_2 \right\rceil .$$

2. To switch key of ciphertext $c' = q' \cdot p' + r + m \cdot (p'-1)/2$ from p' into p, compute

$$c = \langle \mathsf{BitDecomp}_w(c', \gamma/w), \sigma \rangle$$

Let us denote $\mathsf{BitDecomp}_w(c', \gamma/w)$ by $\mathbf{c}$. We write $\lfloor (p/2) \cdot \left[(2^w)^i \cdot 2/p'\right]_2 \rceil$ as $(p/2) \cdot \left[(2^w)^i \cdot 2/p'\right]_2 + \delta_i$ for $|\delta_i| \leq 1/2$. We have, modulo 2,

$$\left\langle \mathbf{c}, \left\lfloor \frac{p}{2} \cdot \left[\frac{(2^w)^i \cdot 2}{p'} \right]_2 \right\rceil \right\rangle = \delta + \frac{p}{2} \cdot (2a + m)$$

with $|a| \leq 2^{w-1} \cdot \gamma/w$ and $|\delta| \leq 2^{w-1} \cdot \gamma/w + |r'| \cdot |p|/|p|'$ by the following equations:

$$\begin{aligned}
\left\langle \mathbf{c}, \left\lfloor \frac{p}{2} \cdot \left[\frac{(2^w)^i \cdot 2}{p'} \right]_2 \right\rceil \right\rangle &= \langle \mathbf{c}, \delta \rangle + \left\langle \mathbf{c}, \frac{p}{2} \cdot \left[\frac{(2^w)^i \cdot 2}{p'} \right]_2 \right\rangle \\
&= \delta' + \frac{p}{2} \cdot \left(c' \cdot \frac{2}{p'} \right) \pmod 2 \\
&= \delta' + \frac{p}{2} \cdot (2q' + (2r' - m)/p' + m) \pmod 2 \\
&= \delta + \frac{p}{2} \cdot (2q' + m) \pmod 2,
\end{aligned}$$

where $|\delta| \leq 2^{w-1} \cdot \gamma/w + |r'| \cdot |p|/|p|'$. Here, we have

$$\left| \langle \mathbf{c}, (p/2) \cdot \left[(2^w)^i \cdot 2/p'\right]_2 \rangle \right| \leq (p/2) \cdot 2^w \cdot \gamma/w,$$

that implies $|a| \leq 2^{w-1} \cdot \gamma/w$. Therefore, we obtain

$$\begin{aligned}
c &= \langle \mathbf{c}, \mathbf{q} \rangle \cdot p^2 + \langle \mathbf{c}, \mathbf{r} \rangle + \left\langle \mathbf{c}, \left\lfloor \frac{p}{2} \cdot \left[\frac{(2^w)^i \cdot 2}{p'} \right]_2 \right\rceil \right\rangle \\
&= q \cdot p^2 + r + \frac{p}{2} \cdot (2a + m),
\end{aligned}$$

where $|r| \leq 2^{\rho+w+1} \cdot \gamma/w + 2^{\rho+1}|p|/|p'|$ and $|a| \leq 2^{w-1} \cdot \gamma/w$. In this case, the number of switching key is γ/w and so the size of switching key is γ^2/w.

Public Generation of Switching-key.

When generating the switch key $(\mathbf{z}, \sigma)$, the key generator needs to know both secret key p and p'. On the other hand, two separate party possess their own secret key p and p' in our model. Therefore, we need the public generation algorithm for switching key in our scenario: an encryptor who does not know secret p' can generate the switching key with auxiliary key. We observe that s'_i is in $\{(2^w)^i : 0 \leq i \leq \eta/w - 1\}$ for $\mathbf{s}' = (s'_1, \ldots, s'_{\Theta \cdot \eta'/w})$. Therefore one can generate σ without knowing secret p', when we have

$$A_1 := \left\{ p^2 \cdot q_i + r_i : q_i \leftarrow [0, 2^\gamma/p^2), r_i \leftarrow (-2^\rho, 2^\rho) \right\},$$
$$A_2 := \left\{ p^2 \cdot q_j + r_j + \left\lfloor (2^w)^j \cdot \frac{p}{2^{\eta+1}} \right\rceil : q_j \leftarrow [0, 2^\gamma/p^2), r_j \leftarrow (-2^\rho, 2^\rho) \right\}.$$

A_1 which is necessary for the re-randomize the switching key is actually a set of encryptions of zero contained in public key pk'. A_2 is auxiliary key for the public generation of switching key. When generating conversion key from $\mathcal{A}_1$ and $\mathcal{A}_2$, we use left-over hash lemma to prove the security, and therefore the noise in conversion key is slightly increased compared to the original switching key.

α-Compressible Homomorphic Encryption

We define a function $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ by $f(a, b) = a + b$ and $\delta_{sk} : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ by $\delta_{sk}(r, m) = -[r]_p + e_1 \cdot p + e_2 + m \cdot (p-1)/2$ for a secret key $sk = p$ and random integers e_1, e_2. Let us consider the following distribution:

$$\begin{aligned}\mathcal{C}_{sk}(m) &= \left\{ r \leftarrow \{0,1\}^\gamma : \text{ Output } f\big(r, \delta_{sk}(r, m)\big) \right\} \\ &= \Big\{ r \leftarrow \{0,1\}^\gamma, e_1 \leftarrow [0, 2^{\lambda+\eta}/p), e_2 \leftarrow (-2^\lambda, 2^\lambda) : \\ &\qquad\qquad \text{Output } r + (-[r]_p + e_1 \cdot p + e_2 + m \cdot (p-1)/2) \Big\}\end{aligned}$$

For $a \leftarrow \mathcal{C}_{sk}(m)$, we verify that $\mathsf{DGHV.Dec}_{sk}(a) = m$ under the DGHV parameter setting. Furthermore, the distribution $\mathcal{C}_{sk}(m)$ is computationally close to the encryption distribution $\mathsf{DGHV.Enc}_{sk}(m)$ by Lemma 1 in [CNT12]. In short, we obtain $\alpha = (\lambda + (\lambda + \eta))/\gamma$-compressible symmetric homomorphic encryption.

Construction.

We describe the construction of scale invariant fully homomorphic encryption over the integer with compressed ciphertext. We denote λ by the security parameter, η by the bit length of secret key p, ρ the bit length of the noise in a fresh ciphertext, τ the number of elements in public key, γ their bit length. Let us consider the following distribution:

$$\mathcal{D}^{\rho}_{p,q_0} := \{q \cdot p^2 + r : q \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/p),\ r \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})\}.$$

In our description of the scheme, we will use the same key switching algorithm and the evaluation algorithm in [CLT14] and denote them by $\mathsf{DGHV.SwitchKey}$ and $\mathsf{DGHV.Eval}$, respectively.

- $\underline{\mathsf{HybDGHV.KG}(1^{\lambda})}$: Generate an odd η-bit integer p and γ-bit integer $x_0 = q_0 \cdot p^2 + r_0$ with $q_0 \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/p^2)$ and $r_0 \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})$. Let $x_i \leftarrow \mathcal{D}^{\rho}_{p,q_0}$ for $1 \leq i \leq \tau$. Let also $y = y' + (p-1)/2$ for $y' \leftarrow \mathcal{D}^{\rho}_{p,q_0}$. Let $\mathbf{z}$ be a vector of Θ numbers with $\kappa = 2\gamma + 2$ bits of precision after the binary point, and let $\mathbf{s}$ be a vector of Θ bits such that

  $$\frac{2^{\eta}}{p^2} = \langle \mathbf{s}, \mathbf{z} \rangle + \epsilon \bmod 2^{\eta}$$

  with $|\epsilon| \leq 2^{-\kappa}$. Define a vector

  $$\sigma = \mathbf{q} \cdot p^2 + \mathbf{r} + \left\lfloor \mathsf{Powersof}_w(\mathbf{s}, \eta/w) \cdot \frac{p}{2^{\eta+1}} \right\rceil,$$

  where $\mathbf{q} \leftarrow (\mathbb{Z} \cap [0, q_0))^{\Theta \cdot \eta/w}$ and $\mathbf{r} \leftarrow (\mathbb{Z} \cap (-2^{\rho}, 2^{\rho}))^{\Theta \cdot \eta/w}$. The secret key is $sk = \{p\}$, the public parameter $pk = (x_0, x_1, \ldots, x_{\tau}, y)$ and the evaluation key $\mathsf{ek} = (\sigma, \mathbf{z})$

  Define a vector $\omega = (\omega_0, \ldots, \omega_{\eta/w-1})$ by

  $$\omega_j = q_j \cdot p^2 + r_j + \left\lfloor (2^w)^j \cdot \frac{p}{2^{\eta+1}} \right\rceil$$

  where $q_j \leftarrow \mathbb{Z} \cap [0, q_0)$ and $r_j \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})$ for $j = 0, \ldots, \eta/w - 1$. Output the secret key $sk = p$, the public key $pk = \Big(\{x_i\}_{0 \leq i \leq \tau}, y\Big)$, the evaluation key $\mathsf{ek} = (\mathbf{z}, \sigma)$ and the auxiliary key $\mathsf{ak} = \omega$.

- $\underline{\mathsf{HybDGHV.ConvKG}(pk, \mathsf{ak}, p')}$: Given η'-bit odd integer p', let $\mathbf{z}'$ be a vector of Θ numbers with $\kappa' = \gamma + 2$ bits of precision after the binary point, and let $\mathbf{t}$ be a vector of Θ bits such that

$$\frac{2^\eta}{p'} = \langle \mathbf{t}, \mathbf{z}' \rangle + \epsilon \bmod 2^\eta$$

with $|\epsilon| \leq 2^{-\kappa'}$. Let $\mathbf{t}' = (t'_1, \ldots, t'_{\Theta \cdot \eta/w}) = \mathsf{Powersof}_w(\mathbf{t}, \eta/w)$ and define a vector $\sigma' = (\sigma'_1, \ldots, \sigma'_{\Theta \cdot \eta/w})$ by

$$\sigma'_i = \Big[\sum_{j \in S} x_j + \omega_{t'_i} \Big]_{x_0}$$

for a random subset $S \subset \{1, \ldots, \tau\}$. Output the conversion key $\mathsf{ck} = (\mathbf{z}', \sigma')$.

- $\underline{\mathsf{HybDGHV.Enc}(pk, \mathsf{ak}, (m_1, \ldots, m_k) \in \{0,1\}^k)}$: Choose an odd η'-bit integer p' and compute $\mathsf{ck} = (\mathbf{z}', \sigma') \leftarrow \mathsf{HybDGHV.ConvKG}(pk, \mathsf{ak}, p')$. Initialize a pseudo-random number generator with a random seed se. Run $\mathsf{PRNG}(\mathsf{se} + i)$ to obtain $\chi_i \in [0, 2^\gamma)$ and compute

$$c_i = -[\chi_i]_{p'} + \xi_i \cdot p' + r + \frac{p' - 1}{2} \cdot m_i,$$

where $\xi_i \leftarrow \mathbb{Z} \cap [0, 2^{\eta+\lambda}/p')$ and $r_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$. Output the compressed ciphertexts

$$(\mathsf{se}, c_1, \ldots, c_k)$$

along with conversion key $\mathsf{ck} = (\mathbf{z}', \sigma')$. These ciphertexts are parsed into $(\mathsf{ck}, \mathsf{se}, (i, c_i))$ for $i = 1, \ldots, k$.

- $\underline{\mathsf{HybDGHV.Convert}(\mathsf{ck}, \mathsf{se}, (i, c))}$: Output

$$\mathbf{C} \leftarrow \mathsf{DGHV.SwitchKey}(\mathbf{z}', \sigma', C')$$

where $\mathbf{C}' = \chi + c$ for $\chi \leftarrow \mathsf{PRNG}(\mathsf{se} + i)$.

- $\underline{\mathsf{HybDGHV.Eval}(\mathsf{ek}, (\mathsf{ck}_1, \mathsf{se}_1, i_1, c_1,), \cdots, (\mathsf{ck}_n, \mathsf{se}_n, i_n, c_n))}$: Output

$$C \leftarrow \mathsf{DGHV.Eval}(\mathsf{ek}, C_1, \ldots, C_2),$$

where $C_j \leftarrow \mathsf{HybDGHV.Convert}(\mathsf{ck}_j, \mathsf{se}_j, (i_j, c_j)))$ for $j = 1, \ldots, n$.

- $\underline{\mathsf{HybDGHV.Dec}(sk, (\mathsf{ck}, \mathsf{se}, i, c))}$: Compute $C \leftarrow \mathsf{HybDGHV.Convert}(\mathsf{ck}, \mathsf{se}, (i, c))$ and output

$$m \leftarrow ((2 \cdot C) \bmod p) \bmod 2.$$

We set $\rho = 2\lambda, \eta = \tilde{\mathcal{O}}(L + \lambda), \gamma = \tilde{\mathcal{O}}(L^2\lambda + \lambda^2)$ and $\Theta = \tilde{O}(L\lambda)$, where L is the multiplicative depth of the circuit to be evaluated. In this hybrid scheme of DGHV encryption, the size of conversion key is about $\mathcal{O}(\eta \cdot \Theta \cdot \gamma / w)$ which is about $\tilde{\mathcal{O}}(\lambda^5)$.

To state the security of the hybrid scheme, we consider the following problem given in [CNT12].

Definition 4.3.1 (Decisional Approximate GCD)**.** The (ρ, η, γ)-Decisional Approximate GCD Problem is: For a random η-bit odd integer p, given polynomially many samples from $\mathcal{D}^{\rho}_{p,q_0}$, and given an integers $z = x + b \cdot \lfloor 2^j \cdot p / 2^{\eta+1} \rceil$ for a given random integer $j \in [0, \eta)$, where $x \leftarrow \mathcal{D}^{\rho}_{p,q_0}$ and $b \leftarrow \{0, 1\}$, find b.

The Decisional Approximate GCD assumption is defined in the usual way.

Theorem 4.3.1 (Security)**.** *The proposed hybrid scheme of asymmetric homomorphic encryption is semantically secure under the Decisional Approximate GCD assumption and under the hardness of subset sum assumption.*

Return Small Ciphertexts.

After homomorphic evaluating on ciphertexts, the resulting ciphertext has huge size –$\mathcal{O}(\lambda^3)$ bits under several security analysis [DGHV10, CN12a]. We can reduce the size of resulting ciphertext as Dijk et. al compressed the DGHV ciphertext

size [DGHV10]. In the modified scheme, we provide an additional public key with the description of a cyclic group G with generator g whose order is a multiple of secret p. The compressed ciphertext is simply $c' = g^c$. To decrypt the compressed ciphertext, one computes $m \leftarrow (\mathbf{DL}_g(c') \bmod p) \bmod 2$. We refer to [DGHV10] for the details.

4.3.2 Hybrid Encryptions based on LWE

In this section, we describe a hybrid encryption based on LWE, combining LWE-based symmetric and assymmetric encryption schemes. As in the DGHV case, we use a pseudo-random number generator to reduce the size of the ciphertexts [CNT12]. Let us first recall the LWE encryption. Let $q = q(n)$ be an integer and $\chi = \chi(n)$ be an error distribution over $\mathbb{Z}_q$. Let a secret key $\mathbf{s}$ be chosen on a distribution χ^n. Let $\mathbb{Z}_t$ be a message space for a small integer $t \geq 2$. The ciphertext has the form of $\mathbf{c} = (b, -\mathbf{a}) = (\mathbf{a} \cdot \mathbf{s} + e + m\lfloor q/t \rfloor, -\mathbf{a})$ where $e \leftarrow \chi$ for a message $m \in \mathbb{Z}_t$. Given a ciphertext, decryption can be done by computing $m = \lfloor t(b - \mathbf{a} \cdot \mathbf{s})/q \rceil$.

Since the vector $\mathbf{a}$ in the ciphertext is random in $\mathbb{Z}_q^n$, generating compressed ciphertext is easy: Generate $\mathbf{a}$ using pseudo-random number generator with a randomly chosen seed se. Then (se, b) is a compressed ciphertext. Decompression is trivial. Note that the security is same as before as long as the length of se is at least λ for the security parameter λ, assuming that pseudo-random number generator is secure. Since the compressed ciphertext can be generated only with a secret key, we need to provide conversion key to convert this ciphertext back to the original public key such that it can be evaluated and finally decrypted by the data analyzer. And the size of this conversion key should not be too large. In this reason, we will use a variant of key switching method described in [GHS12b].

Key Switching in LWE-based encryption.

The usual key switching process is done by a multiplication of a $(n+1) \times (n+1)\lceil \log q \rceil$ matrix and the bit-decomposed ciphertext to control the noise growth. In [GHS12b], instead of bit-decomposition, temporarily increased modulus is used, which can be scaled back to the original modulus by modulus switching technique. Let a ciphertext $\mathbf{c}$ encrypt a message m. Then we have an equality $\langle \mathbf{c}, (1, \mathbf{s}) \rangle = e + m\lfloor q/t \rfloor$ over $\mathbb{Z}_q$. This implies $\langle \mathbf{c}, (p, p\mathbf{s}) \rangle = e' + m\lfloor pq/t \rfloor$ over $\mathbb{Z}_{pq}$ for every odd integer p, where $e' = pe - m(p-1)/t$. Since $e/q \approx e'/pq$, $\mathbf{c}$ is still a valid ciphertext that encrypts the same message m with respect to a secret key $p(1, \mathbf{s})$ and a modulus pq. By taking p large enough, we can ensure that the norm of $\mathbf{c}$ is sufficiently small relative to the modulus pq so that decomposition of the ciphertext is not strictly needed. Then a switching key matrix $\mathbf{P}_{p\mathbf{s}:\mathbf{s}'}$ modulo pq is included in the evaluation key. After the key-switching, modulus switching algorithm with inputs pq and q scales down the ciphertext by a factor p.

Detailed description with word decomposition is in the following. We use the method to decompose vectors in a way that preserves the inner product. Let $\ell = \lceil \log_w q \rceil$. Our notation is generally adopted from [BGV12].

- $D_{w,q}(\mathbf{x})$: For a vector $\mathbf{x} \in \mathbb{Z}^n$, let $\mathbf{x}_i \in \{-2^w/2 + 1, \ldots, 2^w/2\}^n$ such that $\mathbf{x} = \sum_{i=0}^{\ell-1} (2^w)^i \cdot \mathbf{x}_i$. Output the vector

$$(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}) \in \{-w/2 + 1, \ldots, w/2\}^{n\ell}$$

- $P_{w,q}(\mathbf{y})$: For a vector $\mathbf{y} \in \mathbb{Z}^n$, output

$$\left[\left(\mathbf{y}, 2^w\mathbf{y}, (2^w)^2\mathbf{y}, \ldots, (2^w)^{\ell-1}\mathbf{y} \right) \right]_q \in \mathbb{Z}_q^{n\ell}$$

We can easily verify that for any $q \in \mathbb{Z}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, it holds

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle D_{w,q}(\mathbf{x}), P_{w,q}(\mathbf{y}) \rangle \bmod q.$$

Let $\mathbf{c_s}$ be a ciphertext encrypted by a secret key $(1, \mathbf{s})$. Let w be a word size of the decomposition. Let $\ell = \lfloor \log_w q \rfloor + 1$. Let $Q = pq$ where p is a large integer of size $\mathcal{O}(\log q)$ bits that boost up the modulus q. To switch a ciphertext under secret-key $(1, \mathbf{s}) \in \mathbb{Z}_q^{n+1}$ into a ciphertext under secret key $(1, \mathbf{t}) \in \mathbb{Z}_q^{n+1}$, we need a key-switching matrix:

$$\mathbf{P_{s:t}} = [\mathbf{b_{s:t}} \| - \mathbf{A_{s:t}}] \in \mathbb{Z}_Q^{(n+1)\ell \times (n+1)},$$

where

$$\mathbf{b_{s:t}} := [A_{\mathbf{s:t}}\mathbf{t} + \mathbf{e_{s:t}} + pP_{w,q}(1, \mathbf{s})]_Q \in \mathbb{Z}_Q^{(n+1)\ell}. \tag{4.3.2}$$

And the convert procedure becomes

$$\mathbf{c'_t} = \left[\mathbf{P}_{\mathbf{s:t}}^T D_{w,q}(\mathbf{c_s})\right]_Q, \mathbf{c_t} \leftarrow \mathsf{Scale}(\mathbf{c'_t}, Q, q, t).$$

Now we analyze the noise growth in this procedure. Let $\|\mathbf{e_{s:t}}\|_\infty < B$ and $\langle \mathbf{c_s}, (1, \mathbf{s}) \rangle \equiv \lfloor q/t \rfloor m + e \pmod q$ where $|e| < B'$. Consider the inner product of $\mathbf{c'_t}$ and $(1, \mathbf{t})$. Since

$$\begin{aligned} \mathbf{b}_{\mathbf{s:t}}^T D_{w,q}(\mathbf{c_s}) &= \mathbf{t}^T {\mathbf{A}_{\mathbf{s:t}}}^T D_{w,q}(\mathbf{c_s}) + {\mathbf{e}_{\mathbf{s:t}}}^T D_{w,q}(\mathbf{c_s}) + pP_{w,q}(1, \mathbf{s})^T D_{w,q}(\mathbf{c_s}) \\ &\equiv \mathbf{t}^T {\mathbf{A}_{\mathbf{s:t}}}^T D_{w,q}(\mathbf{c_s}) + E_1 + \lfloor Q/t \rfloor m + E_2 \pmod Q \end{aligned}$$

where $|E_1| < Bw(n_s + 1)\ell$ and $|E_2| < pB'$. Now we have

$$\begin{aligned} \langle (1, \mathbf{t}), \mathbf{c'_t} \rangle &\equiv (1, \mathbf{t})^T \mathbf{P}_{\mathbf{s:t}}^T D_{w,q}(\mathbf{c_s}) \equiv \mathbf{b}_{\mathbf{s:t}}^T D_{w,q}(\mathbf{c_s}) - \mathbf{t}^T \mathbf{A}_{\mathbf{s:t}}^T D_{w,q}(\mathbf{c_s}) \\ &\equiv \lfloor Q/t \rfloor m + E_1 + E_2 \pmod Q. \end{aligned}$$

By scaling, the error $E_1 + E_2$ becomes E where $|E| < |E_1 + E_2|/p + \|(1, \mathbf{t})\|_1$. Thus, the overall noise is less than $Bw(n+1)\ell/p + B' + \|(1, \mathbf{t})\|_1$. Using appropriate p, the noise grows only moderately.

Non-interactive Generation of the Key-switching Matrix.

In our model, two seperate party knows the respective secret keys $\mathbf{s}$ and $\mathbf{t}$. Thus, the key-switching matrix needs to be generated differently. One can easily see from

equation (4.3.2) that these can be done seperately. First party generates a matrix over $\mathbb{Z}_Q$ whose rows are the encryptions of zeroes which is similar to the previous key-switching matrix with the omission of $pP_{w,q}(1, \mathbf{s})$. Using the LWE assumption, these encryptions are indistinguishable to the random vectors. Then the second party generates key-switching matrix using $\mathbf{P}_Q$ by linear combination of these encryptions and add the entries of $pP_{w,q}(1, \mathbf{s})$ with the secret key $\mathbf{s}$. Using the left-over hash lemma, the security can be proved. And the resulting key-switching matrix has only slightly larger noise compared to before. And homomorphic computations are carried out after the key converting process for each fresh ciphertexts.

In the following, the overall construction is given.

Construction.

Let $\ell = \lfloor \log_w q \rfloor + 1$, $N = (n+1)\lceil \log q \rceil$, $N_1 = (n+1)(\lceil \log q \rceil + O(1))$, and $N_2 = (n+1)(\lceil \log Q \rceil + O(1))$.

- $\underline{\mathsf{C.SI\text{-}LWE.KG}(1^n)}$: Generate a secret vector $\mathbf{s} \leftarrow \chi^n$. Generate a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{N_1 \times n}$, $\mathbf{e} \leftarrow \chi^{N_1}$, and compute $\mathbf{b} := [\mathbf{As} + \mathbf{e}]_q$, and define a matrix

  $$\mathbf{P} := [\mathbf{b} \| -\mathbf{A}] \in \mathbb{Z}_q^{N_1 \times (n+1)}.$$

  Generate a uniform random matrix $\mathbf{A}' \leftarrow \mathbb{Z}_q^{N^2 \lceil \log q \rceil \times n}$ and a noise $\mathbf{e}_i \leftarrow \chi^{N^2 \lceil \log q \rceil}$. Define

  $$\tilde{\mathbf{s}} := D_{2,q}(1, \mathbf{s}) \otimes D_{2,q}(1, \mathbf{s}) \in \{0,1\}^{N^2},$$

  and compute

  $$\mathbf{P}' = [\mathbf{b}' \| -\mathbf{A}'] \in \mathbb{Z}_q^{N^2 \lceil \log q \rceil \times (n+1)},$$

  where

  $$\mathbf{b}' := \left[\mathbf{A}'\mathbf{s} + \mathbf{e}' + P_{2,q}(\tilde{s})\right]_q \in \mathbb{Z}_q^{N^2 \lceil \log q \rceil}.$$

  Generate a uniform random matrix $\mathbf{A}_Q \leftarrow \mathbb{Z}_Q^{N_2 \times n}$ and a noise $\mathbf{e}_Q \leftarrow \chi^{N_2}$. Compute

  $$\mathbf{P}_Q = [\mathbf{b}_Q \| -\mathbf{A}_Q]$$

where

$$\mathbf{b}_Q = \mathbf{A}_Q\mathbf{s} + \mathbf{e}_Q.$$

Output the secret-key $sk = \mathbf{s}$, the public-key $pk = \mathbf{P}$, the auxiliary key $\mathsf{ak} = \mathbf{P}_Q$, and the evaluation key $\mathsf{ek} = \mathbf{P}'$.

- $\underline{\mathsf{C.SI\text{-}LWE.ConvKG}(\mathsf{ak}, \mathbf{s}')}$: Let $\tilde{\mathbf{s}}' = P_{w,q}(\mathbf{s}')$. Sample $\mathbf{R} \in \{0,1\}^{n\lceil \log_w q\rceil \times N_2}$ and output the conversion key

$$\mathsf{ck} = \mathbf{P}_{\mathbf{s}':\mathbf{s}} := \mathbf{R}\mathbf{P}_Q + \left[\tilde{\mathbf{s}'}||\mathbf{0}\right] \in \mathbb{Z}_q^{n\lceil \log_w q\rceil \times (n+1)}$$

where $\mathbf{0}$ is a zero matrix.

- $\underline{\mathsf{C.SI\text{-}LWE.Enc}(pk, \mathsf{ak}, \mathbf{m} = (m_1, \ldots, m_k) \in \mathbb{Z}_t^k)}$: Sample a secret vector $\mathbf{s}' \leftarrow \chi^n$ and compute

$$\mathbf{P}_{\mathbf{s}':\mathbf{s}} \leftarrow \mathsf{C.SI\text{-}LWE.ConvKG}(\mathsf{ak}, \mathbf{s}').$$

Initialize a pseudo-random number generator with a random seed se. For $i = 1, \ldots, k$, run $f(\mathsf{se}+i)$ to obtain $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ and compute $c_i = \langle \mathbf{a}_i, \mathbf{s}'\rangle + e_i + \left\lfloor \frac{q}{t} \right\rfloor \cdot m_i \pmod q$, where $e_i \leftarrow \chi$. Output the compressed ciphertexts

$$(\mathsf{ck}, \mathsf{se}, \mathbf{c})$$

where $\mathbf{c} = (c_1, \ldots, c_k)$ and $\mathsf{ck} = (\mathbf{P}_{\mathbf{s}':\mathbf{s}})$.

- $\underline{\mathsf{C.SI\text{-}LWE.Convert}(\mathsf{ck}, \mathsf{se}, c_1, \ldots, c_k)}$: Run f with seed se to generate $\mathbf{a}_i \leftarrow f(\mathsf{se}+i)$ and output the decompressed ciphertext $\mathbf{C}' = (\mathbf{c}, -\mathbf{A})$. Output

$$\mathbf{C} \leftarrow \left[\mathbf{P}_{\mathbf{s}':\mathbf{s}} D_{w,q}(\mathbf{C}')\right]_q$$

Once conversion is done, the rest is the same as the underlying encryption scheme. Thus we omit Add, Mul, and Dec.

In this LWE-based hybrid scheme, the size of conversion key is $O(n^2\ell \log Q)$. We remark that the public key and evaluation key can be reduced also, using

pseudo-random number generator and seed with increased evaluation time.

To state the security of the scheme, we define the following distribution:

$$A_{q,\mathbf{s},\chi} := \left\{ \left(\mathbf{a}, \left[\langle \mathbf{a}, \mathbf{s} \rangle + e \right]_q \right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q : \mathbf{a} \leftarrow \mathbb{Z}_q^n, e \leftarrow \chi \right\}.$$

Definition 4.3.2 (Decisional LWE: DLWE [Bra12])**.** For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the (average-case) decision learning with errors problem, denoted $\mathsf{DLWE}_{n,q,\chi}$, is to distinguish (with non-negligible advantage) polynomially many samples chosen according to $A_{q,\mathbf{s},\chi}$ (for random $\mathbf{s} \leftarrow \chi^n$), from random samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Lemma 4.3.2 (Security)**.** *Let n, q, Q, χ be some parameters such that $\mathsf{DLWE}_{n,q,\chi}$ and $\mathsf{DLWE}_{n,Q,\chi}$ holds with same random $\mathbf{s} \leftarrow \chi^n$. Then for any $m \in \mathbb{Z}_t$, if $(sk, \mathbf{P}, \mathbf{P}_Q, \mathbf{P}') \leftarrow \mathsf{C.SI\text{-}LWE.KG}(1^n)$ and $(\mathsf{ck}, \mathsf{se}, c) \leftarrow \mathsf{C.SI\text{-}LWE.Enc}(\mathbf{P}, \mathbf{P}_Q, m)$, it holds that the joint distribution $(\mathbf{P}, \mathbf{P}_Q, \mathsf{ck}, c)$ is computationally indistinguishable from the uniform distribution. And $\mathbf{P}'$ is computationally indistinguishable from the uniform distribution assuming the circular security.*

Sketch of the proof. We can prove this lemma by applying leftover hash lemma.

Return Small Ciphertexts.

After homomorphic evaluation, the resulting ciphertext can be reduced to be transmitted. Since the main part of the decryption consists of the inner product with the secret key, using Paillier encryption scheme [Pai99] with encrypted secret key, one can nearly decrypt the ciphertext, but still encrypted under Paillier scheme, as is described in [Yin13]. In this way, the resulting ciphertext can be compressed to the size of the Paillier encryption scheme. We refer to [Yin13] for the details.

4.4 Discussion

4.4.1 Comparison to Other Approaches

The large ciphertext size of existing FHE schemes is a main obstacle when it is employed for practical applications. It will cause large bandwidth and storage requirement if the applications require transmitting ciphertexts through the network.

As is mentioned in [NLV11, GHS12b], the hybrid scheme of AES encryption and FHE scheme can be used to optimize the communication cost in cloud computing applications. Since the ciphertext expansion of most FHE schemes is huge, data can be encrypted under AES with ciphertext expansion equals to one, and sent along with the additional public key that encrypts the secret key of AES. After receiving the data and the public key, the cloud homomorphically evaluate the decryption circuit of AES firstly, and then performs homomorphic operations on the data under the FHE scheme.

In a recent report by Gentry, Halevi and Smart in a updated implementation [GHS12c], the homomorphic evaluation the AES circuit takes about 4 minutes. When SIMD techniques are used, amortized rate is about 2 seconds per block. We note that the FHE scheme needs to allow depth-40 homomorphic computations on ciphertexts to evaluate the AES circuit.

If the amount of transferred data is large, packing a bundle of messages into one ciphertext can be an another solution to reduce the bandwidth. Most FHEs [SV14, BGV12, GHS12a, CCK$^+$13, CLT14] that support the message packing use Chinese remaindering theorem on specific rings, except LWE scheme [BGH13]. The ciphertext expansion ratio gets better by packing more messages in one encryption, but it increases public key size and slows down encryption/decryption speed per bit.

4.4.2 Other Fully Homomorphic Encryptions

In this section, we discuss the GSW [GSW13] and LTV [LATV12] cryptosystems. It turns out that those FHEs are not suitable to apply our approach. The reason is as follows. For the GSW cryptosystem, the ciphertext is a matrix whose rows resembles the LWE ciphertexts. Since compressing LWE ciphertext is easy, we can similarly compress GSW ciphertexts.

Although GSW ciphertexts are compressible, it is not easy to see how to apply key switching technique to GSW cryptosystem, since GSW cryptosystem does not have switching key. One can try to switch key using the key-switching technique in LWE-based cryptosystem. However, it is not straightforward since the rows in the ciphertext is different from the LWE ciphertext.

For the LTV cryptosystem, the ciphertext is only a single ring element. And it does not seem to be compressible. Thus, our approach does not give a benefit when LTV cryptosystem is needed.

Chapter 5

Conclusion

Gentry proposed the first fully homomorphic encryption scheme based on ideal lattices which supports arbitrarily many additions and multiplications on encrypted bits [Gen09]. His breakthrough paper drew an explosive interest and lead numerous researches in this area [DGHV10, CMNT11, CNT12, GH11b, SV10, SS10, SS11, GHS12a, BV11, BGV12, Bra12, GSW13]. Even though FHE schemes can support both additions and multiplications on encrypted data infinitely, FHE schemes are still far from being practical because of its large computational cost and large ciphertexts.

In this dissertation we proposed several methods to improve the efficiency of fully homomorphic encryption over the integers and move them to practice.

We extend the fully homomorphic encryption scheme over the integers of van Dijk et al. (DGHV) into a batch fully homomorphic encryption scheme, i.e. to a scheme that supports encrypting and homomorphically processing a vector of plaintexts as a single ciphertext. Our scheme has an advantage over [GHS12a] in applications requiring larger message space. We reduce the security of our Somewhat Homomorphic Encryption scheme to a decisional version of Approximate GCD problem (DACD).

We introduce a hybrid homomorphic encryption that combines public-key en-

cryption (PKE) and somewhat homomorphic encryption (SHE) to reduce the storage requirements of most somewhat or fully homomorphic encryption (FHE) applications. In this model, messages are encrypted with a PKE and computations on encrypted data are carried out using SHE or FHE after homomorphic decryption.

We propose an approach to compress ciphertext of somewhat homomorphic encryption with low capacity by using the public key compression technique and the key switching method. The proposed scheme is suitable for cloud computing environment since it has small bandwidth, storage and supports efficient conversion on ciphertexts. Our scheme also supports homomorphic computations on ciphertexts that might be encrypted under different key, which is useful in real applications.

Bibliography

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in lwe-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2013.

[BGJT13] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. *IACR Cryptology ePrint Archive*, 2013. `http://eprint.iacr.org/2013/400`.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012*, pages 309–325. ACM, 2012.

[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 *Lecture Notes in Computer Science*, pages 868–886. Springer Berlin / Heidelberg, 2012.

BIBLIOGRAPHY

[BV11] 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 *Lecture Notes in Computer Science*, pages 505–524. Springer Berlin / Heidelberg, 2011.

[BY88] Ernest F. Brickell and Yacov Yacobi. On privacy homomorphisms (extended abstract). In David Chaum and WynL. Price, editors, *Advances in Cryptology - EUROCRYPT 1987*, volume 304 of *Lecture Notes in Computer Science*, pages 117–125. Springer Berlin Heidelberg, 1988.

[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 *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 315–335, 2013.

[CH11] Henry Cohn and Nadia Heninger. Approximate common divisors via lattices. *IACR Cryptology ePrint Archive*, 2011:437, 2011.

[CK13] Jung Hee Cheon and Jinsu Kim. An approach to reduce storage for homomorphic computations. Cryptology ePrint Archive, Report 2013/710, 2013. `http://eprint.iacr.org/`.

[CKN06] Jung Hee Cheon, Woo-Hwan Kim, and Hyun Soo Nam. Known-plaintext cryptanalysis of the domingo-ferrer algebraic privacy homomorphism scheme. *Inf. Process. Lett.*, 97(3):118–123, 2006.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In Tal Rabin, editor, *Advances in Cryptology CRYPTO 2010*, volume 6223 of *Lecture*

Notes in Computer Science, pages 483–501. Springer Berlin Heidelberg, 2010.

[CLT13] Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. Batch fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/036, 2013.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 311–328, 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 *Lecture Notes in Computer Science*, pages 487–504. Springer Berlin / Heidelberg, 2011.

[CN12a] Yuanmi Chen and Phong 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 *Lecture Notes in Computer Science*, pages 502–519. Springer Berlin / Heidelberg, 2012.

[CN12b] Yuanmi Chen and PhongQ. 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 â EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 502–519. Springer Berlin Heidelberg, 2012.

BIBLIOGRAPHY

[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 *Lecture Notes in Computer Science*, pages 446–464. Springer Berlin / Heidelberg, 2012.

[DGHV10] 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 *Lecture Notes in Computer Science*, pages 24–43. Springer Berlin / Heidelberg, 2010.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive, 2012:144*, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 116–137. Springer Berlin / Heidelberg, 2010.

[GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Rafail Ostrovsky,

editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science - FOCS 2011*, pages 107–109. IEEE, 2011.

[GH11b] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In Kenneth Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer Berlin / Heidelberg, 2011.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer Berlin / Heidelberg, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer Berlin / Heidelberg, 2012.

[GHS12c] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. Cryptology ePrint Archive, Report 2012/099, 2012. `http://eprint.iacr.org/`.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology*

Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[HG01] Nick Howgrave-Graham. Approximate integer common divisors. In *CaLC*, pages 51–66, 2001.

[JL] Marc Joye and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. In *Advances in Cryptology, EUROCRYPT 2013.*

[Jou13] Antoine Joux. A new index calculus algorithm with complexity L(1/4+o(1)) in very small characteristic. *IACR Cryptology ePrint Archive*, 2013.

[KLYC13] Jinsu Kim, Moon Sung Lee, Aaram Yun, and Jung Hee Cheon. CRT-based fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/057, 2013. *The merged paper appears in Eurocrypt 2013 [CCK+13].*

[Lag85] J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. SIAM J. Comput.*, 14(1):196–209, 1985.*

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1219–1234. ACM, 2012.*

[Len87] Jr. Lenstra, H. W. Factoring integers with elliptic curves. The Annals of Mathematics*, 126(3):pp. 649–673, 1987.*

BIBLIOGRAPHY

[MV93] *Alfred Menezes and Scott A. Vanstone. Elliptic curve cryptosystems and their implementations.* J. Cryptology, *6:209–224, 1993.*

[NLV11] *Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In* CCSW, *pages 113–124, 2011.*

[NS98] *David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In Li Gong and Michael K. Reiter, editors,* Proceedings of the ACM Conference on Computer and Communications Security - 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* Lecture Notes in Computer Science, *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* Lecture Notes in Computer Science, *pages 223–238. Springer, 1999.*

[RAD78] *R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphism.* Foundations of Secure Computation, *pages 168–177, 1978.*

[Rot11] *Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Yuval Ishai, editor,* TCC, *volume 6597 of* Lecture Notes in Computer Science, *pages 219–234. Springer, 2011.*

[SS10] *Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor,* Advances in Cryptology - ASIACRYPT 2010, *volume 6477 of* Lecture Notes in Computer Science, *pages 377–394. Springer Berlin / Heidelberg, 2010.*

BIBLIOGRAPHY

[SS11] *Peter Scholl and Nigel Smart. Improved key generation for gentry's fully homomorphic encryption scheme. In Liqun Chen, editor,* Cryptography and Coding*, volume 7089 of* Lecture Notes in Computer Science*, pages 10–22. Springer Berlin / Heidelberg, 2011.*

[SV10] *N. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Nguyen and David Pointcheval, editors,* Public Key Cryptography – PKC 2010*, volume 6056 of* Lecture Notes in Computer Science*, pages 420–443. Springer Berlin / Heidelberg, 2010.*

[SV14] *Nigel P. Smart and Frederik Vercauteren. Fully homomorphic simd operations.* Des. Codes Cryptography*, 71(1):57–81, 2014.*

[SYY99] *Tomas Sander, Adam L. Young, and Moti Yung. Non-interactive cryptocomputing for* NC^1*. In* 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 554–567, 1999.*

[Wag03] *David Wagner. Cryptanalysis of an algebraic privacy homomorphism. In* ISC*, pages 234–239, 2003.*

[Yin13] *Hu Yin. Ph.D. thesis in WORCESTER POLYTECHNIC INSTITUTE, 2013.*

국문초록

동형 암호는 비밀키를 모르는 주체가 별도의 복호화 과정없이 암호문 간의 덧셈 및 곱셈의 연산을 허용하는 암호 알고리즘이다. 동형 암호는 Gentry에 의해 2009년 최초로 개발되었으며, 그 이후 안전성 및 효율성 향상과 응용에 대한 수많은 후속 연구가 이루어졌다.

현재까지 제안된 대부분의 동형 암호 알고리즘은 암호문 크기가 매우 크다는 단점 갖고 있으며 이는 암복호화 과정 및 암호문간의 연산 속도에 현저하게 영향을 주어 현실에서 사용하 데에 큰 걸림돌이 되고 있다. 본 학위 논문에서 이를 해결하기 위해 다음과 같이 두 가지 큰 방향으로 연구를 진행한다.

우선, Dijk 등이 제안한 정수기반 동형 암호의 메시지 공간을 확장함으로써, 같은 크기의 암호문으로 많은 양의 메시지에 대한 연산을 수행하는 방법에 대해서 연구한다. 중국인 나머지 정리를 활용하여 새로운 동형 암호를 설계하였으며, 이 스킴은 Howgrave-Graham이 제안한 근사 최대 공약수 문제에 기반한다.

또한, 기존의 공개키 암호와 동형 암호의 결합을 통해 암호문의 크기를 줄이는 기법을 제안함으로써 클라우드 환경에 적합한 하이브리드 암호 모델에 대한 연구를 진행한다. 이 모델에서 암호화하고자 하는 주체는 암호문의 크기가 현저히 작은 공개키 암호를 이용하고, 암호문 간의 연산은 동형 암호로 바꾼 뒤에 수행하게 된다.

동형 암호에서 연산 효율성을 위해 제안된 키 변환 기법과 공개키 축소 기법을 결합하여 동형 암호의 암호문을 줄이는 방법에 대한 연구를 진행한다. 이 경우, 연산을 위해 필요한 암호문 복구 기법이 상대적으로 간단하다는 장점을 갖는다.

주요어휘: 동형암호, 준동형 암호, 하이브리드 스킴, 근사 최대공약수 문제, 암호문 압축

학번: 2009-20264