An Approach to Reduce Storage for Homomorphic Computations

Jung Hee Cheon and Jinsu Kim

Seoul National University (SNU), Republic of Korea
jhcheon@snu.ac.kr, kjs2002@snu.ac.kr

Abstract. We introduce a hybrid homomorphic encryption by combining public key encryption (PKE) and somewhat homomorphic encryption (SHE) to reduce storage for most applications of somewhat or fully homomorphic encryption (FHE). In this model, one encrypts messages with a PKE and computes on encrypted data using a SHE or a FHE after homomorphic decryption. To obtain efficient homomorphic decryption, our hybrid schemes is constructed by combining IND-CPA PKE schemes without complicated message paddings with SHE schemes with large integer message space. Furthermore, we remark that if the underlying PKE is multiplicative on a domain closed under addition and multiplication, this scheme has an important advantage that one can evaluate a polynomial of arbitrary degree without recryption. We propose such a scheme by concatenating ElGamal and Goldwasser-Micali scheme over a ring $\mathbb{Z}_N$ for a composite integer N whose message space is $\mathbb{Z}_N^\times$.
To be used in practical applications, homomorphic decryption of the base PKE is too expensive. We accelerate the homomorphic evaluation of the decryption by introducing a method to reduce the degree of exponentiation circuit at the cost of additional public keys. Using same technique, we give an efficient solution to the open problem [16] partially.
As an independent interest, we obtain another generic conversion method from private key SHE to public key SHE. Differently from Rothblum [23], it is free to choose the message space of SHE.

Keywords: ElGamal, Goldwasser-Micali, Naccache-Stern, Hybrid Scheme, Multiplicative Homomorphic Encryption, Additive Homomorphic Encryption, Fully Homomorphic Encryption, Decryption Circuit, Exponentiation, Bootstrapping

1 Introduction

The concept of computation on encrypted data without decryption was firstly introduced by Rivest, Adleman and Detourzos in 1978 [22]. After thirty years, Gentry proposed a fully homomorphic encryption (FHE) based on ideal lattices [9]. This scheme is far from being practical due to its large computation costs and large ciphertexts. Since then, lots of efforts has been done to devise more efficient schemes and their successors become much more efficient [3, 11, 12]. However, most FHE schemes still have huge ciphertext size, at least millions of bits for a single ciphertext. This is a big bottleneck when it is deployed in practice.

We consider a situation: several users upload data encrypted with a FHE, a server carries out computations on encrypted data and then sends them to a client who has a decryption key of the FHE. It is common in typical applications of (public key) FHEs such as medical applications and financial applications [19]. In this situation, one approach to save storage is to use a public key encryption (PKE) to encrypt data and perform homomorphic computation on ciphertexts after converted to ciphertexts under a FHE. This approach has a great advantage in storage and communications because only small ciphertexts under a PKE are transmitted from users to a server and converted only when its homomorphic computation is required.

However, its converting complexity (homomorphic decryption of a ciphertext) is generally too large when using bit-encrypting FHEs.

In this paper, we explore efficient hybrid homomorphic encryption schemes by combining a public key encryption with small ciphertext size and a somewhat homomorphic encryption (SHE) that could evaluate the decryption circuit of the PKE. For efficient construction, we consider SHE for hybrid scheme rather than FHE.

For efficient conversion, PKE scheme needs to be IND-CPA secure without complicated message padding whose evaluation requires lots of computation. Interestingly, most candidates are additive or multiplicative homomorphic. When using these schemes as the underlying PKEs, we obtain an additional advantage in computations on encrypted data. When using additive homomorphic encryptions (AHE), it has an advantage that it can evaluate any linear function without converting to a SHE. We consider the Goldwasser-Micali scheme for encrypting a bit and the Naccache-Stern scheme for encrypting small messages. We remark that the leveled DGHV scheme gives more efficient evaluation of their decrypt circuit.

When using multiplicative homomorphic encryptions (MHE), it has a very interesting property that 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, since each monomial of $f(x_1,\ldots,x_k)$ could be computed under the PKE. One problem of this approach is that the domain of a MHE is usually not closed under an addition operation. For example, the (IND-CPA) ElGamal encryption over a ring R could take as messages only elements in a prime order subgroup, which covers only small part of R except a binary fields. However, the discrete logarithm problem in an extension field with small characteristic is not a hard problem any more [14, 2]. Thus we construct a multiplicative homomorphic encryption whose message space is $\mathbb{Z}_N^\times$ for a RSA modulus $N = p_1p_2$. The proposed scheme is obtained by combining an ElGamal over $\mathbb{Z}_N^\times$ and the Goldwasser-Micali encryption over $\mathbb{Z}_N$ and is secure under the decisional Diffie-Hellman assumption and the quadratic residuosity assumption for common $N = pq$.

One concern of this approach is the converting complexity. All the decryption circuits of PKE we considered is composed of an exponentiation by a secret exponent e. Gentry and Halevi [10] showed that its homomorphic evaluation is done by evaluating a polynomial of degree $\log e$. This degree is generally very large up to several thousands in our situation. To obtain efficient converting algorithm, we represent an integer as a binary vector of $w \log_w e$ length and reduce the degree to $\log_w e$ at the cost of $\tilde{O}(w)$ additional public keys for an arbitrary positive integer w.

Also we resolve the open problem of [16] partially that evaluate $\bmod q \bmod p$ homomorphically. We convert the double modulo reduction into depth-3 circuit and then apply the technique of [10]. Our improved technique plays important role especially in this method, since the selection of parameters heavily depends on the homomorphic capacity of FHE differently from leveled FHE.

As an independent interest, our approach gives a generic conversion from private key SHE to public key SHE. In [23], Rothblum shows the way how to transform any homomorphic private key encryption scheme whose message space is $\mathbb{Z}_2$ into a public key homomorphic encryption scheme. 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. Using our hybrid scheme, however, we obtain generic conversion from private key SHE to private key SHE whose message space is

$\mathbb{Z}_p$ for large prime p. We only add encryptions of secret key of a PKE under the private SHE to the public key instead of $\{\mathsf{SHE.Enc}_i(0)\}_i$ and $\{\mathsf{SHE.Enc}_i(1)\}_i$ as in [23].

2 Preliminaries

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

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. Denote the element in $\mathbb{Z} \cap (-\frac{n}{2}, \frac{n}{2}]$ which is equivalent to a modulo n by $a \bmod n$ or $[a]_n$. We denote 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 by $CRT_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$.

2.1 Base Problems

Definition 1 (Decisional Diffie-Hellman problem over $\mathbb{G}$). *Let $\mathbb{G}$ be a group with a generator g of order q. For given a tuple (g, g^a, g^b, g^c), the* decisional Diffie-Hellman (DDH) problem over $\mathbb{G}$ *is to decide whether $g^{ab} = g^c$ or not.*

Definition 2 (Quadratic Residuosity problem over $\mathbb{Z}_N$). *Given an odd composite integer N and $a \in \mathsf{J}_N$ where $\mathsf{J}_N := \{a \in \mathbb{Z}_N^* | \left(\frac{a}{N}\right) = 1\}$, the* quadratic residuosity problem (QR) over $\mathbb{Z}_N$ *is to decide whether a is quadratic residue modulo N or not.*

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. The QR assumption over $\mathbb{Z}_N$ is defined similarly.

2.2 Homomorphic Encryption Schemes

Definition 3 (ElGamal Encryption over a Ring). *Let R be a ring and $\mathbb{G}_q$ a multiplicative cyclic subgroup of prime order q in R. The* ElGamal encryption scheme $\mathsf{EIG} = (\mathsf{EIG.KeyGen}, \mathsf{EIG.Enc}, \mathsf{EIG.Dec})$ *consists of the following algorithms:*

$\underline{\mathsf{EIG.KG}(\lambda)}$: *Take as input a 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 4 (Goldwasser-Micali Encryption). *The* Goldwasser-Micali encryption scheme $\mathsf{GM} = (\mathsf{GM.KeyGen}, \mathsf{GM.Enc}, \mathsf{GM.Dec})$ *consists of the following algorithms:*

$\underline{\mathsf{GM.KG}(\lambda)}$: *Choose random primes p, q and compute $N = pq$. Compute quadratic non residue x modulo N satisfying $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$. Output a public key $pk_{\mathsf{GM}} = (N, x)$ and a secret key $sk_{\mathsf{GM}} = (p, q)$.*

$\underline{\mathsf{GM.Enc}(pk_{\mathsf{GM}}, m)}$: *For a plaintext* $m \in \{0,1\}$*, choose* $y \in [0, N)$ *such that* $\gcd(y, N) = 1$*. Output a ciphertext* $c = y^2 x^m \bmod N$*.*

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

Definition 5 (Naccache-Stern Encryption). *The* Naccache-Stern encryption scheme $\mathsf{NS} = (\mathsf{NS.KeyGen}, \mathsf{NS.Enc}, \mathsf{NS.Dec})$ *consists of the following algorithms:*

$\underline{\mathsf{NS.KG}(\lambda)}$: *Choose random small primes* $p_1, \ldots, p_k$ *and 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* $N = pq$*. 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* $sk_{\mathsf{NS}} = (p, q)$

$\underline{\mathsf{NS.Enc}(pk_{\mathsf{NS}}, m)}$: *For a plaintext* $m \in \mathbb{Z}/\sigma\mathbb{Z}$*, 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* $c \in \mathbb{Z}_N$*, compute* $c^{\phi(N)/p_i}$ *and obtain* m_i *by comparing it with* $g^{j\phi(N)/p_i}$ *for all* $j = 1, \ldots, p_i - 1$*. Output* $m = CRT_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$*.*

3 Encrypt with PKE and Compute with SHE

In this section, we give a concept of hybrid scheme by combining a PKE and a SHE. A message is encrypted under the PKE and it is converted to a ciphertext under the SHE when it needs homomorphic computations on the message. The ciphertext is decrypted under SHE.

3.1 A Hybrid Scheme of a PKE and a 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 which has large computing power. Until now, fully homomorphic encryptions are good solutions for the delegation of computations [5] ignoring bandwidth of the client and storage of the server. However these are most significant measures in construction of cloud environment since they are directly connected the cost in real.

One can reduce bandwidth and storage by combining a PKE with small ciphertext size and a SHE that could evaluate the decryption circuit of the PKE not its own decryption circuit. We propose a hybrid scheme to improve an efficiency of SHE when one use it in cloud computing environment. Let $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public key encryption with depth-d decryption circuit and $\mathsf{SHE} = (\mathsf{SHE.KG}, \mathsf{SHE.Enc}, \mathsf{SHE.Dec}, \mathsf{SHE.Eval})$ be a somewhat homomorphic encryption. Suppose that there exists a multivariate polynomial f_{Dec} such that $f(sk_1, \ldots, sk_n, c) = m$ for all $c = \mathsf{PKE.Enc}(m)$ where $(sk_1, \ldots, sk_n)$ is the expanded secret key of sk_{PKE}. 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

$$pk_{\mathsf{Hyb}} = (pk_{\mathsf{PKE}}, pk_{\mathsf{SHE}}, \overline{S}_{sk_{\mathsf{PKE}}}, f_{\mathsf{Dec}})$$
$$sk_{\mathsf{Hyb}} = (sk_{\mathsf{SHE}})$$

where $S_{sk_{\mathsf{PKE}}}$ is an expanded secret key set and $\overline{S}_{sk_{\mathsf{PKE}}} = \left\{\mathsf{SHE.Enc}(a) : a \in S_{sk_{\mathsf{PKE}}}\right\}$.

$\underline{\mathsf{Hyb.Enc}(pk_{\mathsf{Hyb}}, m)}$: For a plaintext $m \in \mathbf{M}_{pk_{\mathsf{PKE}}}$, 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 $\overline{S}_{sk_{\mathsf{PKE}}}$. That is, compute and output a ciphertext C

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

$\underline{\mathsf{Hyb.Eval}(pk_{\mathsf{Hyb}}, f, c_1, \ldots, c_t)}$: For given circuit f and t ciphertexts $c_1, \ldots, c_t$ under SHE, output $\mathsf{SHE.Eval}(pk_{\mathsf{SHE}}, f, c_1, \ldots, c_t)$.

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

Remark 1. In $\mathsf{Hyb.Conv}$ algorithm, we remark that

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

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

Remark 2. In a hybrid scheme, the homomorphic capacity of the SHE at least exceeds the degree of the decryption circuit f_{Dec} of the PKE.

Theorem 1 (Semantic Security of Hybrid Scheme). *If both the public key encryption and the somewhat homomorphic encryption are semantically secure, then so is the hybrid scheme.*

The security follows by the standard hybrid argument similar to Theorem 4.2.3 in [8]. We omit the details.

Let $\mathbf{M}, \mathbf{C}$ be message space and ciphertext space of the PKE, respectively, and $\mathcal{M}$ be a message space of the SHE. If $\mathbf{M}, \mathbf{C}$ and $\mathcal{M}$ are in the same category and there exist efficient computable homomorphisms $\phi : \mathbf{M} \to \mathcal{M}^k$ and $\psi : \mathbf{C} \to \mathcal{M}^k$ for some positive integer k, one could construct more efficient hybrid scheme. In this paper, therefore, we only focus on the case that M, C and $\mathcal{M}$ satisfy the above conditions.

Remark 3. Although the elliptic curve ElGamal cryptosystem [18] has small ciphertext, we do not consider in this paper since it is hard to evaluate the inverse map of message encoding and the addition of points on the elliptic curve.

3.2 Additive Homomorphic Encryptions for the PKE in Hybrid Scheme

Goldwasser-Micali [13], Paillier [21], Okamoto-Uchiyama [20] and Joye-Libert [15] encryptions are candidates for an additive homomorphic IND-CPA PKE in constructing a hybrid scheme. The decryption circuit of each system requires an additional circuit besides an exponentiation, the Chinese remaindering algorithm for Goldwasser-Micali and Naccache-Stern encryptions and integer division for Paillier, Okamoto-Uchiyama encryptions. Since it seems hard to evaluate integer division part efficiently, the latter encryptions could not be used for the PKE. Thus we only consider Goldwasser-Micali and Naccache-Stern encryptions for a PKE in hybrid scheme.

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

$$m = CRT_{(p,q)}(c^{(p-1)/2}, c^{(q-1)/2}).$$

Using homomorphic evaluation of secret exponentiation and the Chinese remaindering algorithm, one could evaluate the decryption circuit of Goldwasser-Micali encryption. Since p and q are secret, encryptions of $q(q^{-1} \bmod p)$ and $p(p^{-1} \bmod q)$ must be added to the public key of the hybrid scheme. In this case, the message space of GM encryption is $\mathbb{Z}_2$ and the message space of the SHE is $\mathbb{Z}_{2N}$. The degree of the decryption circuit is about $2 \log p$ ($\approx \lambda^2$) using Gentry and Halevi's technique [10].

Naccache-Stern Encryption The decryption of Naccache-Stern encryption is to compute $c^{\phi(N)/p_i}$ and get m_i by comparing it with $g^{j\phi(N)/p_i}$ for all $j = 1, \ldots, p_i - 1$. The message m is recovered by computing $m = CRT_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$. To compute m_i homomorphically, at first one needs to compute a polynomial $f_i(x)$ of degree $p_i - 1$ such that $f_i(g^{m\phi(n)/p_i}) = m$ for all $m \in \mathbb{Z}_{p_i}$ using the Lagrange interpolation. And then compute $CRT_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k)$ homomorphically. In Naccache-Stern encryption, the message space is $\mathbb{Z}_M$ where $M = \prod_{i=1}^{k} p_i$ and the message space of the SHE is $\mathbb{Z}_Q$ where $Q = MN$. The degree of the decryption circuit is about $\log \phi(n) + \max_i\{p_i - 1\}$.

3.3 Multiplicative Homomorphic Encryptions for the PKE in Hybrid Scheme

We may consider ElGamal encryption [7] 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 encrypted under ElGamal encryption securely. Although one can 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, one could not use the ElGamal encryption over $\mathbb{F}_{2^n}$ for a PKE, since a DLP in an extension field with small characteristic is not a hard problem any more [14, 2]. Unlike the extension field case, it is impossible to construct an ElGamal encryption whose message space is a full domain over a 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 $\mathbb{Z}_N$ and the Goldwasser-Micali encryption scheme over $\mathbb{Z}_N$ [13] for common $N = p_1p_2$, and the ciphertext consists of three elements in $\mathbb{Z}_N^\times$. We call this scheme by *EGM* encryption.

Let $N = p_1p_2$ with $p_1 = 2q_1 + 1$ and $p_2 = 4q_2 + 1$ for distinct primes p_1, p_2, q_1, q_2. Then the order of $\mathbb{Z}_N^\times$ is $\phi(N) = 8q_1q_2$ and 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 use a bijective map ι,

$$\iota : \mathbb{Z}_N^\times \to \mathsf{J}_N \times \{0,1\}, \quad m \mapsto (\hat{m}, \check{m}) = \left(m\left(\frac{m}{N}\right), \left(1 - \left(\frac{m}{N}\right)\right)/2\right),$$

where $\mathsf{J}_N := \{a \in \mathbb{Z}_N^* | \left(\frac{a}{N}\right) = 1\}$. 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 $sk_{\mathsf{EGM}} = (e, p_2)$.

$\underline{\mathsf{EGM.Enc}(pk_{\mathsf{EGM}}, m)}$: For a plaintext $m \in \mathbb{Z}_N^\times$, compute $\iota(m) = (\hat{m}, \check{m})$ and choose a random $r \in [0, N^2)$[1] 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 $m \equiv c_1^e c_2 \cdot CRT_{(p_1,p_2)}(c_3^{q_1}, c_3^{2q_2}) \mod N$.

The EGM encryption is multiplicative homomorphic over $\mathbb{Z}_N^\times$, which covers almost all nonzero element of $\mathbb{Z}_N$. We remark that $CRT_{(p_1,p_2)}(c_3^{q_1}, c_3^{2q_2}) = 1$ if $\check{m}$ is even, and $CRT_{(p_1,p_2)}(c_3^{q_1}, c_3^{2q_2}) = -1$ otherwise. From this, the EGM encryption is correct for unlimited number of multiplications on encrypted data. Also the EGM encryption is semantically secure under the DDH assumption over J_N and the QR assumption over $\mathbb{Z}_N$ for common $N = p_1p_2$.

Theorem 2 (Multiplicative Homomorphic). *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 a multiplication of two ciphertexts is defined by componentwise. That is, EGM encryption is multiplicative 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

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

We remark that if $\sum_{i=1}^k \check{m}_i$ is even, $T(C_3) = 1$ and $\prod_{i=1}^k (\frac{m_i}{N}) = 1$. Unless, $T(C_3) = -1$ and $\prod_{i=1}^k (\frac{m_i}{N}) = -1$. Thus we obtain

$$C_1^e C_2 T(C_3) = \Big(\prod_{i=1}^{k} \hat{m}_i\Big) \cdot T(C_3) = \Big(\prod_{i=1}^{k} m_i \Big(\frac{m_i}{N}\Big)\Big) \cdot T(C_3) = \prod_{i=1}^{k} m_i \pmod{p_2}.$$

□

Theorem 3 (Semantic Security). *The* EGM *scheme is semantically secure under the DDH assumption over J_N and the QR assumption over $\mathbb{Z}_N$.*

Sketch of Proof. We use a hybrid argument to prove semantic security. We define a sequence of games.

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

[1] 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, one can choose $N^{1+\epsilon}$ for some $\epsilon > 0$ instead of N^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 $\sigma' = r^2$ for a randomly chosen r from $\mathbb{Z}_N$. That is, $pk_1 = (N, g, y, \sigma')$.

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.

Since the challenge ciphertext c in Game$_2$ is independent from message m_b, $\Pr[S_2]$ is $1/2$, where S_i is the event that the attacker succeeds in Game$_i$. Thus the advantage of the attacker in Game$_0$ is negligible under the DDH assumption over J_N and the QR assumption over $\mathbb{Z}_N$. □

Encryption of Zero Since the EGM scheme has the message space $\mathbb{Z}_N^\times$, one can encrypt neither the zero nor a multiple of p_1 or p_2 in $\mathbb{Z}_N$. Since the probability that an encryptor chooses the multiples of p_1 or p_2 is negligible, we only care for the message zero.

By borrowing the idea in [24], we could modify the scheme by appending λ GM encryptions of encoding defined by $0 \mapsto \boldsymbol{r} \in_R \{0,1\}^\lambda$ 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 appended ciphertext is preserved under multiplications with $1 - 2^{-\lambda}$ probability. The decryption algorithm is similar to the original EGM by 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 GM encryption, one could 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 ciphertext space is $(\mathbb{Z}_N^\times)^3$. One needs to choose the message space of SHE as $\mathbb{Z}_N$ in construction of hybrid scheme.

4 Homomorphic Evaluation of Exponentiation

To enhance a performance of the hybrid scheme, one needs to evaluate a modular exponentiation by a secret exponent efficiently that is related to the decryption circuit of GM, NS and EGM encryptions. Actually, this problem is dealt with by Gentry and Halevi [10] when 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 decryption circuit of the ElGamal is represented by 4λ degree polynomial which is large to evaluate efficiently where λ is the security parameter. We give an improved algorithm to evaluate an exponentiation with a small degree multivariate polynomial.

4.1 Improved Exponentiation using Vector Decomposition

Gentry and Halevi [10] firstly proposed a method to evaluate an exponentiation by a secret exponent homomorphically. They expand 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 in computing $v^{e_i 2^i}$, that is $v^{e_i 2^i} = e_i v^{2^i} + (1 - e_i)v^0$. The degree of their exponentiation circuit is about $2 \log e$ ($\approx 4\lambda$) which is large where λ is the security parameter. Reducing the degree of exponentiation by secret exponent is a meaningful approach since the degree is directly related to selection in parameters of a FHE. One may consider a w-ary representation of the secret e with $w > 2$ to reduce a degree of the exponentiation circuits. When one uses w-ary representation, there are $\log_w e$ terms of individual $v^{e_i w^i}$ which is smaller than $\log_2 e$. However it requires degree $(w-1)$ polynomial to express each $v^{e_i w^i}$ for $e_i \in [0, w-1]$ when using the Lagrange interpolation. Indeed this increases the degree of exponentiation by e from $2 \log e$ to $w \log_w e$.

We use a vector representation of e_i to reduce the degree of $v^{e_i w^i}$ instead of the Lagrange interpolation. At 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 a map π

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

where $W = [0, w-1]$ and $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_w\}$ 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 $\boldsymbol{v}_i := (1, v^{w^i}, v^{2w^i}, \cdots, v^{(w-1)w^i}) \in \mathbb{G}_q^w$. Then one can easily verify that $v^{e_i w^i} = \langle \boldsymbol{v}_i, \pi(e_i) \rangle$, 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 the 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, a 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, the 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 the SHE.
2. Output $c := \prod_{i=0}^n \Big(\sum_{k=0}^{w-1} \mathsf{SHE.Enc}(e_{ik}) \cdot \mathsf{SHE.Enc}\big(v^{kw^i}\big) \Big)$.

Theorem 4 (Correctness). *Suppose that* $c \leftarrow \mathsf{Eval.Exp}(pk_{\mathsf{SHE}}, \bar{E}_e, w, v)$. *Then* $v^e \leftarrow \mathsf{SHE.Dec}_{sk_{\mathsf{SHE}}}(c)$.

The proof of Theorem 4 is straightforward. It is verified that the degree of exponentiation is about $2\log_w e$ which is $\log w$ times smaller than the original method. Using our method, one can evaluate the exponentiation by a large secret exponent homomorphically with a small degree polynomial at the cost of $\tilde{O}(w)$ additional public keys for an arbitrary integer w.

4.2 Improve the Bootstrapping without Squashing

Gentry and Halevi [10] 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 they remove the assumption on the hardness of the sparse subset sum problem. The Gentry and Halevi's technique in [10] is to express the decryption of a SHE scheme as a depth-3 ($\sum\prod\sum$) arithmetic circuit. They temporarily switch to a ciphertext under a MHE, such as ElGamal, to compute $\prod$ part. And then they homomorphically evaluate the decryption circuit of MHE to get a ciphertext under SHE. Using their method, SHE only needs to evaluate MHE's decryption circuit of fixed degree $2\log e$, not its own decryption circuit. Using our efficient evaluation of exponentiation given in Section 4.1, we can reduce the degree from $2\log e$ to $2\log_w e$. Moreover, we point out that one can handle more general class of SHE whose decryption circuit is a composition of restricted depth-3 circuit $\sum\prod\sum$ and several low depth circuits. This will be given in Appendix C. By applying our technique, we show that any SHE with decryption circuit of $[\cdot]_q \bmod p$ type is bootstrappable if it could evaluate degree $2\log_w e$ circuits.

Evaluate Double Modulo Reduction The bottleneck of bootstrapping is to compute $[\cdot]_q \bmod p$ homomorphically. We call $[\cdot]_q \bmod p$ as a *double modulo reduction circuit*. In general, when the plaintext is $\mathbb{Z}_2$, the bootstrapping is done by using bit operations on binary representations of integers. However it is not easy to bootstrap a ciphertext when the message space is $\mathbb{Z}_p$ with $p > 2$. We propose a method to evaluate $[\cdot]_q \bmod p$ homomorphically for large p using Gentry and Halevi's idea [10]. As a application, we give a bootstrapping method of [4] when a message is large enough. Since one could not use modulus switching technique to handle errors of ciphertexts in batch fully homomorphic encryption [4], the selection of parameters of the FHE heavily depends on the homomorphic capacity of the scheme. Thus our improved technique plays important role in bootstrapping ciphertexts.

We assume that q is equivalent to 1 modulo p. Then $[c]_q \bmod p$ can be written in the form as DGHV [27],

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

In comparison with DGHV scheme, it is hard to express the division by p by a low degree polynomial over $\mathbb{Z}_p$ when p is larger than two. To apply the technique in [10], we first modify the division part using the Gentry's squashing technique [9]. Let us consider parameters κ, Θ, θ such that $\kappa = \gamma\eta/\rho' \log p, \Theta = \omega(\kappa \cdot \log \lambda)$ 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 $x_q = \lfloor p^\kappa/q \rceil$ and choose Θ-bit random vector $\boldsymbol{s} = (s_1, \ldots, s_\Theta)$ with Hamming weight θ. Choose random integer $u_i \in \mathbb{Z} \cap [0, p^{\kappa+1})$ for $i = 1, \ldots, \Theta$ such that $\sum_i s_i u_i = x_q \pmod{p^{\kappa+1}}$. Set $y_i = u_i/p^\kappa$ which is smaller than p with κ precision after p-ary point. Also $[\sum_i s_i y_i]_p = (1/q) - \Delta_q$ for some $|\Delta_q| < p^{-\kappa}$. We firstly compute $z_i \leftarrow [c \cdot y_i]$, keeping only $n = \lceil \log_p \theta \rceil +$

$\lceil \log_p 8 \rceil$ precision after p-ary 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

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

Since the bit length of c is at most $2^\gamma < p^{\kappa-4}$, thus $c \cdot \Delta_q \leq 1/16$. Also we observe that $|\sum s_i \Delta_i| \leq \theta \cdot 1/16\theta = 1/16$. Thus we have

$$[c]_q \bmod p = c - \lfloor c/q \rceil \bmod p = c - \left\lfloor \sum s_i z_i \right\rceil \bmod p. \tag{1}$$

To apply chimeric technique [10], we convert the above subset sum into a $\sum\prod\sum$ form, defined in Appendix D. The equation (1) can be converted as follows:

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

where $z_i = z'_i + z''_i \cdot Q^{-\kappa}$ with positive integers z'_i, z''_i. 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 p such that $p > 2\Theta^2$ by the Lemmas in Appendix D. Thus we obtain the following Theorem:

Theorem 5. *Let q, p be primes such that $q > p > 2\Theta^2$ and $q \equiv 1 \bmod p$. For any $A \subset \mathbb{Z}_p$ of cardinality at least $2\Theta^2 + 1$, 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 $2\Theta^2$ having at most $2\Theta^2 + \Theta + 1$ product gates. Thus one can evaluate the double modulo reduction circuit using chimeric technique.*

Bootstrapping in [4] Cheon *et al.* [4] proposed a fully homomorphic encryption based on Chinese remainder theorem. The decryption of the scheme consists of $[\cdot]_{p_i} \bmod Q_i$ with $i \in [1, k]$. They only achieve "bootstrapping" when all Q_i's are two. They raised a problem to evaluate the double modulo reduction $[\cdot]_{p_i} \bmod Q_i$ homomorphically when some of Q_i is

greater than 2. We can solve this problem partially with the above technique and so complete the bootstrapping stage for large enough Q_i's. It gives a fully homomorphic encryption dealing with large integers. Note that if Q_i's are relatively prime, we can map a plaintext on $\mathbb{Z}_M$ with $M = \prod_i Q_i$ into $\prod_i \mathbb{Z}_{Q_i}$ and so it enables large integer arithmetics on $\mathbb{Z}_M$. In this bootstrapping procedure, our improved technique in evaluation of exponentiation plays important role, since the parameters of the FHE heavily depends 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 $\tilde{\mathcal{O}}(w)$.

5 Discussions

We give typical applications of fully homomorphic encryption in database and cloud computing environment and analyze advantages of our hybrid scheme under some scenarios.

5.1 Application Model

Database Encryption Let us consider the situation that a government agency collects medical records of patients from the hospital and extracts some statistical information from the records. When lots of data are stored in a storage, it becomes more serious problem to protect data from insider's misuse or outsider's hacking. To reduce the risk, one may store data after encryption. We are to give a storage efficient solution using a hybrid scheme under this scenario.

At first, the agency generates $(pk_{\mathsf{Hyb}}, sk_{\mathsf{Hyb}})$ of hybrid scheme and stores sk_{Hyb} in secure area and makes public only pk_{PKE} to hospitals and pk_{Hyb} to a database. Each hospital uploads its medical records after encrypted under the pk_{PKE} in the database. The agency requests for some computations on patients data to extract information to the databased. Then the database performs homomorphic computation on encrypted data using pk_{Hyb}. After evaluation of requested computations, the resulting ciphertexts are sent to the agency. The decryption is carried out in secure area of the agency.

Outsourcing of Computations in Cloud Environment Suppose that a client who has small computing power wants to compute heavy computation on private data. He can use a hybrid scheme to protect his privacy, that is, he outsources computations of PKE-encrypted data along with pk_{Hyb} to a cloud that has huge computing power and storage. The cloud only performs the outsourced computations and returns the results ciphertext encrypted under SHE. Although the fact that the client must send pk_{Hyb} with large size seems as a weakness of our hybrid scheme, it doesn't matter since pk_{Hyb} is sent to the cloud only once in the procedure of outsourcing.

Remark 4. Since the client may not trust the cloud, it is desirable to have the cloud prove that the computation on encrypted data was done correctly. In [5], they give a solution to prove the correctness when using FHE only. It needs more study on this problem when one uses the hybrid scheme in delegation of computations.

5.2 Advantages

The advantages of using our scheme in the above scenarios includes small bandwidth, storage save and efficient computations.

Small Bandwidth In cloud environment, each client encrypts their messages with small computing power and storage and a server manages encrypted data with large computing power and storage. However, the current FHE's may not be suitable to this environment, because it has a large ciphertext size which causes large communication cost. In the above scenario, each hospital and the client can encrypt data using efficient PKE scheme instead of using an inefficient FHE. And then they send ciphertexts with only few thousands of bits to the database and the cloud. It could reduce the bandwidth of clients dramatically.

Storage Save After receiving the ciphertext from each clients, the server stores ciphertexts which contain secret information of clients. The server can save the storage by storing only small PKE ciphertexts rather than large FHE ciphertexts. The server convert them to SHE ciphertexts and compute some operations only when it is required.

Efficient Computing In our hybrid scheme, one could choose a PKE among additive homomorphic and multiplicative homomorphic depending on the property of a circuit we are to evaluate. Suppose that the server is to evaluate a multivariate polynomial f and g where f has polynomially many monomials on inputs and g has polynomially many linear factors. We will use multiplicative homomorphic encryption as PKE in the first case, and additive homomorphic encryption as PKE for the second case.

At first, let us consider $f(x_1, \ldots, x_n) = \sum_{i \in I} M_i(x_1, \ldots, x_n)$ be an n-variable polynomial over a ring R, where 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 FHEs, which are slow or increase the ciphertext size. However, using our hybrid scheme, one may evaluate f without a bootstrapping regardless of the degree f. Suppose the ciphertexts are encrypted under a multiplicative encryption E with the key (pk, sk) and the SHE can evaluate the decryption circuit $\mathsf{D}(sk, \cdot)$ of E_{pk}. Given $c_1 = \mathsf{E}_{pk}(m_1), \ldots, c_n = \mathsf{E}_{pk}(m_n)$, 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 $a \equiv b$.

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

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

1. Given $c_1 = \mathsf{E}_{pk}(m_1), \cdots, c_n = \mathsf{E}_{pk}(m_n)$, compute $M_i(c_1, \cdots, c_n) = \mathsf{E}_{pk}(M_i(m_1, \cdots, m_n))$ for each i.
2. Encrypt $\mathsf{E}_{pk}(M_i(m_1, \cdots, m_n))$ with SHE and then evaluate the decryption circuit $\bar{\mathsf{D}}$ with the encrypted secret key $\mathsf{SHE}(sk)$ of E_{pk}. (Observe: one may use trivial encryption on $\mathsf{E}_{pk}(M_i)$)

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 E. In this case, we follow the steps:

1. Given $c_1 = \mathsf{E}_{pk}(m_1), \cdots, c_n = \mathsf{E}_{pk}(m_n)$, compute $L_i(c_1, \cdots, c_n) = \mathsf{E}_{pk}(L_i(m_1, \cdots, m_n))$ for each i.
2. Encrypt $\mathsf{E}_{pk}(L_i(m_1, \cdots, m_n))$ with SHE and then evaluate the decryption circuit $\bar{\mathsf{D}}$ with the encrypted secret key $\mathsf{SHE}(sk)$ of E_{pk}. (Observe: one may use trivial encryption on $\mathsf{E}_{pk}(L_i)$)
3. Multiply them to obtain $\mathsf{SHE}(f(m_1, \ldots, m_n))$

Remark 5. As you see, one could compute on ciphertexts under a SHE not a PKE after conversion of ciphertexts. Therefore one makes more good use of the hybrid scheme when evaluating fixed multivariate polynomials.

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

In [23] , Rothblum shows 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.

One could consider our hybrid scheme as a generic conversion of SHE from private-key to public key whose message space is $\mathbb{Z}_p$ for large prime p. We only add encryptions of secret key of a PKE under the private SHE to the public key instead of $\{\mathsf{SHE.Enc}_i(0)\}_i$ and $\{\mathsf{SHE.Enc}_i(1)\}_i$. And 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 a PKE and 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 $pk_{\mathsf{PubSHE}} = \overline{(pk_{\mathsf{Hyb}})}$ 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 hybrid scheme.

6 Implementation of Hybrid Scheme

In this section, we describe an implementation of the hybrid scheme $\mathsf{Hyb} = (\mathsf{Hyb.KG}, \mathsf{Hyb.Enc}, \mathsf{Hyb.Conv}, \mathsf{Hyb.Dec})$ using the EGM encryption scheme over $\mathbb{Z}_N$ and a symmetric version of the leveled DGHV scheme proposed by Coron *et al.* [6].

6.1 Implementation Model

We consider a scenario that a client delegates of computations on encrypted data. The client encrypts a message under EGM given in Section 3.3 and sends the ciphertext with FHE evaluation key. We use symmetric version of [6], since there is no difference between symmetric version and public version in this scenario.

Hybrid Scheme of EGM and leveled-DGHV The hybrid scheme of the EGM over $\mathbb{Z}_N$ and the leveled-DGHV [6] with is as follows:

$\underline{\mathsf{Hyb.KG}(\lambda, \mathsf{EGM.KG}, \mathsf{FHE.KG})}$: Take as input security parameter λ, $\mathsf{EGM.KG}$ and $\mathsf{FHE.KG}$ algorithms. Run $\mathsf{EGM.KG}$ and $\mathsf{FHE.KG}$ to get $(N, g, y, \sigma, \kappa, e, p_1 = 2q_1 + 1, p_2 = 4q_2 + 1, pk_{\mathsf{FHE}}, sk_{\mathsf{FHE}})$ where $\mathsf{J}_N = \langle g \rangle$ and $y = g^e$ with $e \in [0, 2^{2\lambda})$. Let $k_0 = \lfloor \log_w e \rfloor$ and $k_1 = \lfloor \log_w q_1 \rfloor$. Define a vector $\boldsymbol{v}_i := (1, v^{w^i}, v^{2w^i}, \cdots, v^{(w-1)w^i})$. Define a multivariate polynomial f_{Dec} as

$$f_{\mathsf{Dec}}(x_0, \ldots, x_{k_0}, y_0, \ldots, y_{k_1}, z_0, \ldots, z_{k_1}, u, v, t) := \prod_{i=0}^{k_0} \langle \boldsymbol{v}_i, \pi(x_i) \rangle \cdot v \cdot \left(\alpha_1 \prod_{i=0}^{k_1} \langle \boldsymbol{t}_i, \pi(y_i) \rangle + \alpha_2 \prod_{i=0}^{k_1} \langle \boldsymbol{t}_i, \pi(z_i) \rangle \right)$$

where $\alpha_1 = p_2 \cdot (p_2^{-1} \bmod p_1)$ and $\alpha_2 = p_1 \cdot (p_1^{-1} \bmod p_2)$. Output $pk_{\mathsf{Hyb}} = (N, g, y, \bar{S}_1, \bar{S}_2, f_{\mathsf{Dec}}, \mathsf{FHE.Enc}(\alpha_1), \mathsf{FHE.Enc}(\alpha_2))$ and $sk_{\mathsf{Hyb}} = (sk_{\mathsf{FHE}})$, where

$$\bar{S}_0 = \left\{ \mathsf{FHE.Enc}(e_{ij}) \middle| e = \sum_{i=0}^{k_0} e_i w^i,\ \pi(e_i) = (e_{ij})_{j=0}^{w-1} \right\},$$

$$\bar{S}_1 = \left\{ \mathsf{FHE.Enc}(q_{1,ij}) \middle| q_1 = \sum_{i=0}^{k_1} q_{1,i} w^i,\ \pi(q_{1,i}) = (q_{1,ij})_{j=0}^{w-1} \right\},$$

$$\bar{S}_2 = \left\{ \mathsf{FHE.Enc}(q_{2,ij}) \middle| 2q_2 = \sum_{i=0}^{k_1} q_{2,i} w^i,\ \pi(q_{2,i}) = (q_{2,ij})_{j=0}^{w-1} \right\},$$

$\underline{\mathsf{Hyb.Enc}(pk_{\mathsf{Hyb}}, m)}$: Take as input the public key pk_{Hyb} and a plaintext $m \in \mathbb{Z}_N^\times$. Output $\mathsf{EGM.Enc}(m)$.

$\underline{\mathsf{Hyb.Conv}(pk_{\mathsf{Hyb}}, c)}$: Take as input the public key pk_{Hyb} and a ciphertexts $c = (c_1, c_2, c_3)$ under EGM encryption. Compute and output

$$C = f_{\mathsf{Dec}}(\mathsf{FHE.Enc}(e_{1,00}), \ldots, \mathsf{FHE.Enc}(q_{1,00}), \ldots, \mathsf{FHE.Enc}(q_{2,00}), \ldots, \mathsf{FHE.Enc}(q_{2,k_1 w-1}), c_1, c_2, c_3).$$

$\underline{\mathsf{Hyb.Dec}(sk_{\mathsf{Hyb}}, C)}$: Take as input the secret key sk_{Hyb} and a ciphertext C. Compute and output a message $M = \mathsf{FHE.Dec}(sk_{\mathsf{FHE}}, C)$.

Remark 6. The degree of decryption circuit f_{Dec} is $2(k_0 + k_1) \approx O(\lambda^2)$ for the security parameter λ.

6.2 Results

Asymptotic Key Size Let $N = p_1 p_2$ be the modulus of EGM scheme, ρ and κ be the bit length of errors and p_i, respectively. We refer Lenstra's analysis [17] to choose κ and

lattice attacks on approximate gcd problem [26, 27] to choose η and γ. In our construction, the message space of the EGM and leveled-DGHV is $\mathbb{Z}_N$ and the set of secret key for leveled-DGHV is $\{P_1, \ldots, P_L\}$ where P_i is i-th modulus with size $(i+1)\mu$ bits for each $i = 1, \ldots, L$. To remain the ciphertext noise as the same, one should choose μ such that $\mu \geq 2\rho+\kappa+40$. We give concrete analysis in Appendix B. We choose the parameters asymptotically as follows: $\rho = O(\lambda), \kappa = O(\lambda^2), \mu = O(\lambda^2), \eta_L = L \cdot O(\lambda^2)$ and $\gamma = \omega(\mu^2 \log \lambda) = \omega(\lambda^4 \log \lambda)$.

Optimization Let d be a degree of the decryption circuit of EGM. If one use Gentry and Halevi's technique [10], d becomes $2\kappa + 2\log e$. We could reduce this to $(2\kappa + 2\log e)/\log w$ when using w-ary expansion of secret exponent. Thus the depth L of leveled DGHV scheme become smaller as the parameter w increase. Also, we use W-bit decomposition and powers instead of BitDecomp and Powersof2 as in [6] to reduce running time of SwitchKey by a factor of W at the cost of increasing the noise by W bits. We take $w = 128$ and $W = \mu/2$ in our implementations.

Storage Save As in Table 1, we can use EGM ciphertexts instead of FHE ciphertexts, which results in about *four thousands* times storage save.

Instances	Security	ρ	κ	$\log e$	μ	W	w	L	$\eta_L \times 10^{-4}$	$\gamma \times 10^{-7}$	FHE size	EGM size
Toy	42	80	80	84	480	256	2	8	0.43	0.19	237 KB	60 Byte
							128	5	0.28	0.07	87 KB	60Byte
Small	52	106	220	104	1040	512	2	9	1.04	0.96	1.2 MB	165 Byte
							128	6	0.72	0.59	0.7 MB	165 Byte
Medium	62	126	343	124	1600	768	2	9	1.6	1.16	1.5 MB	258 Byte
							128	7	1.28	0.83	1.1 MB	258 Byte
Large	72	145	506	144	2400	1024	2	10	2.64	6.96	8.7 MB	380 Byte
							128	7	1.92	3.69	4.6 MB	380 Byte

Table 1. The parameter setting of the hybrid scheme.

Timing Results We test the timing of the hybrid scheme on a PC with Intel core i7-2600 3.4GHz and 192GB memory and use the NTL library [25] with the GMP library [1]. Table 2 shows that it takes about 52 minutes 11 seconds in FHE.KG and about 1 minute 25 seconds in Hyb.Conv algorithm under "Medium" parameter setting with $w = 128$. We verify that our efficient homomorphic evaluation of exponentiation technique gives about eight times faster than that of the original method [10].

7 Conclusion

We propose a hybrid scheme of a public key encryption and a somewhat homomorphic encryption. The proposed scheme is suitable for cloud computing environment since it has small band width, storage and supports efficient computing on encrypted data. At the cost of conversion from PKE ciphertext to FHE ciphertext, we can reduce the storage size about four thousands times smaller. If we can batch fully homomorphic encryption over the integers with modulus switching technique, we may expect to further enhance the performance of the hybrid scheme.

Instances	w	FHE.KG	EGM.KG	Hyb.KG	Hyb.Conv
Toy	2	1 min 40 s	0.09 s	2.02 s	16.61 s
	128	15.2 s	0.06 s	14.3 s	1.38 s
Small	2	28 min 13 s	1.28 s	32.2 s	4 min 38 s
	128	7 min 3 s	0.03 s	5 min 15 s	31.7 s
Medium	2	1 h 36 min	2.43 s	1 min 4 s	**8 min 15 s**
	128	52 min 11 s	2.1 s	13 min 16 s	**1 min 25 s**

Table 2. Timing results of the hybrid scheme.

References

1. GMP: The gnu multiple precision arithmetic library. `http://gmplib.org`, 2013.
2. R. Barbulescu, P. Gaudry, A. Joux, and E. 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`.
3. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) Fully homomorphic encryption without bootstrapping. In S. Goldwasser, editor, *Innovations in Theoretical Computer Science 2012*, pages 309–325. ACM, 2012.
4. J. Cheon, J.-S. Coron, J. Kim, M. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In T. Johansson and P. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer Berlin Heidelberg, 2013.
5. K.-M. Chung, Y. Kalai, and S. Vadhan. Improved delegation of computation using fully homomorphic encryption. In T. Rabin, editor, *Advances in Cryptology CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 483–501. Springer Berlin Heidelberg, 2010.
6. J.-S. Coron, D. Naccache, and M. Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer Berlin Heidelberg, 2012.
7. T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology - CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.
8. C. Gentry. A fully homomorphic encryption scheme. In *PhD thesis*. Stanford University, 2009. crypto.stanford.edu/craig.
9. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing - STOC 2009*, pages 169–178. ACM, 2009.
10. C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In R. Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science - FOCS 2011*, pages 107–109. IEEE, 2011.
11. C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.
12. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.
13. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
14. A. Joux. A new index calculus algorithm with complexity L(1/4+o(1)) in very small characteristic. *IACR Cryptology ePrint Archive*, 2013.
15. M. Joye and B. Libert. Efficient cryptosystems from 2^k-th power residue symbols. In *Advances in Cryptology, EUROCRYPT 2013*.
16. J. Kim, M. S. Lee, A. Yun, and J. H. Cheon. CRT-based fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/057, 2013. *The merged paper appears in Eurocrypt 2013 [4].*
17. *A. K. Lenstra. Key length, 2004.*
18. *A. Menezes and S. A. Vanstone. Elliptic curve cryptosystems and their implementations.* J. Cryptology*, 6:209–224, 1993.*

19. *M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In C. Cachin and T. Ristenpart, editors,* Proceedings of the 3rd ACM Cloud Computing Security Workshop - CCSW 2011*, pages 113–124. ACM, 2011.*
20. *T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor,* Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of* Lecture Notes in Computer Science*, pages 308–318. Springer, 1998.*
21. *P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor,* Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of* Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.*
22. *R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphism.* Foundations of Secure Computation*, pages 168–177, 1978.*
23. *R. Rothblum. Homomorphic encryption: From private-key to public-key. In Y. Ishai, editor,* TCC*, volume 6597 of* Lecture Notes in Computer Science*, pages 219–234. Springer, 2011.*
24. *T. Sander, A. L. Young, and M. 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.*
25. *V. Shoup. NTL: A library for doing number theory ver. 6.0.0.* `http://www.shoup.net/ntl/`*, 2013.*
26. *J. H. Silverman, editor.* Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of* Lecture Notes in Computer Science*. Springer, 2001.*
27. *M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In H. Gilbert, editor,* Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of* Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.*

A Proofs

Proof of Theorem 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 $\mathsf{EGM.KeyGen}$ to obtain a public key $pk_0 = (N, g, y, \sigma)$ and a secret key $sk_0 = (e, \phi(N), \sigma_1, \sigma_2)$.

Game$_1$: this game is the same as Game_0, except for the following modification to the key generation. Instead of choosing $\sigma \in \mathbb{Z}_N^*$ with $(\frac{\sigma}{p_1}) = (\frac{\sigma}{p_2}) = -1$, we choose it as $\sigma' = r^2$ for a randomly chosen r 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 $\mathbb{Z}_N$. □

B Leveled DGHV Scheme with Large Message Space

In [6], they firstly propose the leveled fully homomorphic encryption over the integers. One could not use the modulus switching technique of BGV scheme [3] directly to DGHV scheme since p and p' must remain secret. For given ciphertext $c = pq + r$, they use virtual ciphertext of the form $c' = 2^k \cdot q' + r'$ with $[q']_2 = [q]_2$ to switch secret modulus.

In hybrid scheme with the EGM scheme and the leveled DGHV scheme, the message space of the FHE is $\mathbb{Z}_N$ for RSA modulus N. In this section, we modify the Lemmas in [6] to deal with the leveled DGHV scheme whose message space is $\mathbb{Z}_Q$.

Lemma 1 (Lemma 4 in [6]). *Let p be an odd integer. Let $c = pq + r$ be a ciphertext of DGHV. Let k, κ be an integer such that $|c| < 2^\kappa$. Let $\boldsymbol{y}$ be a vector of Θ numbers with κ bits of precision after the binary point, and $\boldsymbol{s}$ be a vector of Θ bits such that $Q^k/p = \langle \boldsymbol{s}, \boldsymbol{y}\rangle \bmod Q^{k+1}$ with $|\epsilon| < 2^{-\kappa}$. Let $\boldsymbol{c} = (\lfloor c \cdot y_i \rceil \bmod Q^{k+1})_{1\leq i\leq \Theta}$ and $c' = \langle \boldsymbol{s}, \boldsymbol{c}\rangle$. Then $c' = Q^k \cdot q' + r'$ with $[q']_Q = [q]_Q$ and $r' = \lfloor r \cdot Q^k/p \rfloor + \delta$ where $\delta \leq 2 + \Theta/2$.*

Lemma 2 (Lemma 5 in [6]). *Let p and p' be odd integers such that $p \equiv p' \bmod Q$. Let k be integer such that $p' < Q^k$ and $c = pq + r$ be a ciphertext. Let $\boldsymbol{y}$ be a vector of Θ numbers with κ bits precision after the binary point, and let $\boldsymbol{s}$ be a vector of Θ bits such that $Q^k/p = \langle \boldsymbol{s}, \boldsymbol{y}\rangle + \epsilon \bmod Q^{k+1}$ where $|\epsilon| < 2^{-\kappa}$. Let $\boldsymbol{\sigma} = p'\boldsymbol{q} + \boldsymbol{r} + \lfloor s' \cdot pQ^{k+1} \rceil$. Let $\boldsymbol{c} = (\lfloor c \cdot y_i \rceil)_{1\leq i\leq \Theta}$ and $\boldsymbol{c}' = \mathsf{BitDecomp}(\boldsymbol{c}, \mathsf{k})$ be the expanded ciphertext. Let $c'' = Q\langle \boldsymbol{\sigma}, \boldsymbol{c}'\rangle + [c]_Q$. Then $c'' = p'q'' + r''$ where $r'' = \lfloor r \cdot p'/p \rfloor + \delta'$ for some δ with $|\delta| \leq Q \cdot 2^{\rho+1} \cdot \Theta \cdot (k+1)$*

Now we give the modulus switching algorithm for DGHV with message space $\mathbb{Z}_Q$.

$\underline{\mathsf{SwitchKeyGen}(pk, sk, pk'sk')}$:

1. Take as input two DGHV secret keys p and p' of size η and η' respectively. Let $\kappa = 2\gamma + \eta$ where γ is the size of the public key integers x_i under p. Let $\eta'' = \lfloor \eta'/\log Q \rfloor$.
2. Generate a vector $\boldsymbol{y}$ of Θ random modulo $Q^{\eta''+1}$ with κ bits of precision after the binary point, and a random vector $\boldsymbol{s}$ of Θ bits such that $Q^{\eta''}/p = \langle \boldsymbol{s}, \boldsymbol{y}\rangle + \epsilon \bmod Q^{\eta''+1}$ where $|\epsilon| < 2^{-\kappa}$. Generate the expanded secret key $\boldsymbol{s}' = \mathsf{Powersof2}(\boldsymbol{s}, \eta'')$
3. Compute a vector encryption $\boldsymbol{\sigma}$ of $\boldsymbol{s}'$ under sk', defined as follows:

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

where $\boldsymbol{q} \leftarrow (\mathbb{Z} \cap [0, q_0))^{(\eta''+1)\Theta}$ and $\boldsymbol{r} \leftarrow (\mathbb{Z} \cap (-2^{\rho'}, 2^{\rho'}))^{(\eta''+1)\Theta}$, where q'_0 is from $x'_0 = p'q'_0 + r'$ in pk'.
4. Output $\tau_{pk \to pk'} = (\boldsymbol{y}, \boldsymbol{\sigma})$

$\underline{\mathsf{SwitchKey}(\tau_{pk \to pk'}, c)}$:

1. Let $(\boldsymbol{y}, \boldsymbol{\sigma}) \leftarrow \tau_{pk \to pk'}$
2. Compute the expanded ciphertext $\boldsymbol{c} = (\lfloor c \cdot y_i \rceil \bmod Q^{\eta''+1})_{1 \le i \le \Theta}$ and let $\boldsymbol{c}' = \mathsf{BitDecomp}(\boldsymbol{c}, \eta'')$.
3. Output $c'' = Q\langle \boldsymbol{\sigma}, \boldsymbol{c}' \rangle + [c]_Q$.

C Improved Bootstrapping without Squashing

Suppose that $\boldsymbol{s} := (s_1, \ldots, s_n) \in \{0,1\}^n$ is a secret key of the SHE and $\boldsymbol{c} := (c_1, \ldots, c_n) \in \{0,1\}^n$ is a ciphertext of plaintext $m \in \mathbb{Z}_p$ under the SHE. Suppose that the decryption of SHE consists of one or several following functions:

$$f(\boldsymbol{c}) = T\left(\sum_{j=1}^{n+1} \lambda_j \prod_{i=1}^{n} (a_j + s_i \cdot c_i)\right), \tag{3}$$

where the a_j's and λ_j's are the publicly known constants in $\mathbb{Z}_p$ and T is a circuit of degree d. Actually, a_j's are chosen such that a_j and $a_j + 1$ are quadratic residues modulo p and λ_j is chosen depending on a_j's. We allow that T is a composition of several $\sum$ and $\prod$, and so may have exponentially many terms when represented as a sum of monomials. Now we describe recrypting a ciphertext $\boldsymbol{c} = \mathsf{SHE.Enc}(m)$ using our new technique. The public key of FHE contains encryptions of the secret key of SHE under $\mathsf{ElGamal}$ and encryptions of the secret key of $\mathsf{ElGamal}$ under SHE additionally. $\mathsf{FHE.Recrypt_{low}.Setup}$ and $\mathsf{FHE.Recrypt_{low}}$ are as follows:

$\underline{\mathsf{FHE.Recrypt_{low}.Setup}(pk_{\mathsf{SHE}}, sk_{\mathsf{SHE}}, pk_{\mathsf{EIG}}, sk_{\mathsf{EIG}}, \{\lambda_j\}, \{a_j\}, w)}$: Take as input the public key and secret key pair of SHE and EIG with publicly known constant $\{\lambda_j\}$ and $\{a_j\}$.

1. Encrypt $a_j + s_i$ and a_j for all $i \in [1, n]$ and $j \in [1, n+1]$ where $\boldsymbol{s} := (s_1, \ldots, s_n)$ is the secret key of the SHE.
2. Expand e as a w-ary representation, $e = \sum_{\ell=0}^{\lfloor \log_w e \rfloor} e_\ell w^\ell$ with $e_\ell \in [0, w-1]$.
3. Encrypt $e_{\ell k}$ for $\ell \in [0, \lfloor \log_w e \rfloor]$ and $k \in [0, w-1]$, where $\pi(e_\ell) = (e_{\ell 0}, \ldots, e_{\ell(w-1)})$.
4. Output

$$E_1 := \left\{\mathsf{EIG.Enc}(a_j + s_i) \middle| i \in [1,n] \text{ and } j \in [1, n+1]\right\} \cup \left\{\mathsf{EIG.Enc}(a_j) \middle| j \in [1, n+1]\right\}$$
$$E_2 := \left\{\mathsf{SHE.Enc}(e_{\ell k}) \middle| \ell \in [0, \lfloor \log_w e \rfloor] \text{ and } k \in [0, w-1]\right\}.$$

$\underline{\mathsf{FHE.Recrypt_{low}}(pk_{\mathsf{SHE}}, E_1, E_2, w, c)}$: Take as input the public key of SHE, the set E_1, E_2 output by $\mathsf{FHE.Recrypt_{low}.Setup}$ and a ciphertext $c = (c_1, \ldots, c_n)$ to recrypt.

1. Given ciphertext $\boldsymbol{c} := (c_1, \ldots, c_n)$, if $c_i = 1$, choose $b_{ji} := \mathsf{EIG.Enc}(a_j + s_i)$, otherwise choose $b_{ji} = \mathsf{EIG.Enc}(a_j)$ from the public key and compute $b_j := \prod_{i=1}^{n} b_{ji}$ for all $j \in [1, n+1]$.

2. Let $b_j = (y_j, h_j)$. Compute $\mathsf{SHE.Enc}((y_j^{w^\ell})^k)$ and $\mathsf{SHE.Enc}(h_j)$ for all $j \in [1, n+1]$, $\ell \in [0, \lfloor \log_w e \rfloor]$ and $k \in [0, w-1]$.
3. Compute $A_j := \mathsf{SHE.Enc}(h_j) \cdot \prod_{\ell=0}^{\lfloor \log_w e \rfloor} \Big(\sum_{k=0}^{w-1} \mathsf{SHE.Enc}(e_{\ell k}) \cdot \mathsf{SHE.Enc}((y_j^{w^\ell})^k) \Big)$ for all $j \in [1, n+1]$.
4. Encrypt the publicly known constant λ_j under SHE, and define $d_j := \mathsf{SHE.Enc}(\lambda_j)$ for all $j \in [1, n+1]$.
5. Output $B := T(\sum_{j=1}^{n+1} d_j A_j)$.

In the step 4 and 5,

$$A_j = \mathsf{SHE.Enc}\Big(h_j \cdot \prod_{\ell=0}^{\lfloor \log_w e \rfloor} (\sum_{k=0}^{w-1} e_{\ell k} \cdot y_j^{e_\ell 2^\ell}) \Big) = \mathsf{SHE.Enc}\Big(h_j \cdot y_j^e\Big) = \mathsf{SHE.Enc}\Big(\prod_{i=1}^{n} (a_j + s_i \cdot c_i)\Big),$$

since $y^{e_\ell w^\ell} = \sum_{k=0}^{w-1} e_{\ell k} \cdot (y^{w^\ell})^k$ for $e_\ell \in [0, w-1]$ and

$$B = T\Big(\sum_{j=1}^{n+1} \mathsf{SHE.Enc}\Big(\lambda_j \prod_{i=1}^{n} (a_j + s_i \cdot c_i)\Big)\Big) = \mathsf{SHE.Enc}\Big(T\Big(\sum_{j=1}^{n+1} \lambda_j \prod_{i=1}^{n} (a_j + s_i \cdot c_i)\Big)\Big).$$

Thus $B = \mathsf{SHE.Enc}(m)$ which is a refreshed ciphertext of m by the equation (3). We observe that B is not a completely refreshed ciphertext, since computing B involves evaluating $(2\lfloor \log_w e \rfloor + 3) + d$ degree polynomials in ciphertexts.

In the original $\mathsf{FHE.Recrypt}$ algorithm in [10], the SHE needs to support evaluating polynomials of degree $2\lfloor \log e \rfloor + 2$ ($\approx 4\lambda$). To ensure this, one must increase the parameters of the SHE scheme until it can handle them. We give the alternative approach to solve this problem. The w-ary representation instead of binary representation reduces the parameter required to enable the bootstrapping. The SHE only needs to support degree $2\lfloor \log_w e \rfloor + 2$ polynomial in our variant which is smaller than $2\lfloor \log e \rfloor + 2$. Our contribution is the reduction in the degree of polynomial required to bootstrap. It prevents the parameter rise which causes inefficiency overall scheme.

D Definitions and Lemmas

Definition 6. *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 $\boldsymbol{x} := (x_1, \ldots, x_n)$ if there exists multi sets $S_1, \ldots, S_t \subset \mathcal{L}$ and constants $\lambda_0, \lambda_1, \ldots, \lambda_t$ such that*

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

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

Lemma 3 ([10]). *Let Q be a prime with $Q > 2\Theta^2$. Regarding the "complicated part" of the decryption circuit, there is an univariate polynomial $f(x)$ of degree $\leq 2\Theta^2$ such that $f(\sum_{i=1}^{\Theta} s_i z_i'') = \lfloor Q^{-n} \sum_{i=1}^{\Theta} s_i z_i'' \rceil \bmod Q$.*

Lemma 4 ([10]). *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*

$$f(\sum_{i=1}^{\Theta} s_i z_i'') = M_f(\underbrace{s_1, \ldots, s_1}_{z_1''}, \underbrace{0, \ldots, 0}_{T-z_1''}, \ldots, \underbrace{s_\Theta, \ldots, s_\tau}_{z_\Theta''}, \ldots, \underbrace{0, \ldots, 0}_{T-z_\Theta''}), \tag{4}$$

for all $\boldsymbol{s} = (s_1, \ldots, s_\Theta) \in \{0,1\}^\Theta$ and $z_1'', \ldots, z_\Theta'' \in [0, T]$, .

Lemma 5 ([10]). *Let $Q \geq \Theta + 1$ be a prime, let $A \subset \mathbb{Z}_Q$ have cardinality $\Theta + 1$, let $\boldsymbol{x} = (x_1, \ldots, x_\Theta)$ be variables, denote $\mathcal{L}_A = \{(a + x_i : a \in A, 1 \leq i \leq \Theta\}$. For every multilinear symmetric polynomial $M(\boldsymbol{x})$ over $\mathbb{Z}_Q$, there is a circuit $C(\boldsymbol{x})$ such that:*

- *C is a $\mathcal{L}_A$-restricted depth-3 circuit over $\mathbb{Z}_Q$ such that $C(\boldsymbol{x}) \equiv M(\boldsymbol{x}) \mod Q$.*
- *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 $M(\boldsymbol{x})$ at all $\boldsymbol{x} = 1^i 0^{\Theta - i}$*

Sketch of Proof. Since $M(\boldsymbol{x})$ is multilinear (degree 1 with respect to each x_i) and symmetric, $M(\boldsymbol{x}) = \sum_{i=0}^{\tau} \ell_i e_i(\boldsymbol{x})$, where e_i is i-th elementary symmetric polynomial. Consider $f(x) = \prod_{i=1}^{\tau}(x + x_i) = e_\tau(\boldsymbol{x}) + e_{\tau-1}(\boldsymbol{x})x + \cdots + e_o(\boldsymbol{x})x^\tau$. Thus

$$\begin{bmatrix} f(\alpha_0) \\ \vdots \\ f(\alpha_\tau) \end{bmatrix} = \begin{bmatrix} 1 & \alpha_0 & \cdots & \alpha_0^\tau \\ \vdots & & \ddots & \vdots \\ 1 & \alpha_\tau & \cdots & \alpha_\tau^\tau \end{bmatrix} \begin{bmatrix} e_\tau(\boldsymbol{x}) \\ \vdots \\ e_0(\boldsymbol{x}) \end{bmatrix}$$

Thus $M(\boldsymbol{x})$ can be written in the form,

$$M(\boldsymbol{x}) = \begin{bmatrix} \ell_\tau & \cdots & \ell_0 \end{bmatrix} \begin{bmatrix} e_\tau(\boldsymbol{x}) \\ \vdots \\ e_0(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \ell_\tau & \cdots & \ell_0 \end{bmatrix} \begin{bmatrix} 1 & \alpha_0 & \cdots & \alpha_0^\tau \\ \vdots & & \ddots & \vdots \\ 1 & \alpha_\tau & \cdots & \alpha_\tau^\tau \end{bmatrix}^{-1} \begin{bmatrix} f(\alpha_0) \\ \vdots \\ f(\alpha_\tau) \end{bmatrix} = \begin{bmatrix} \lambda_0 & \cdots & \lambda_\tau \end{bmatrix} \begin{bmatrix} f(\alpha_0) \\ \vdots \\ f(\alpha_\tau) \end{bmatrix}.$$

We can write $M(\boldsymbol{x}) = \sum_{j=0}^{\tau} \lambda_j f(\alpha_j) = \sum_{j=0}^{\tau} \lambda_j \prod_{i=1}^{\tau}(\alpha_j + x_i)$. □