Efficient Multi-key FHE with short extended ciphertexts and less public parameters

Tanping Zhou[1,2], Ningbo Li[1], Xiaoyuan Yang[1,2], Yiliang Han[1], and Wenchao Liu[1]

[1] College of Cryptography Engineering, Engineering University of People's Armed Police, Xi'an, 710086, China
`372726936@qq.com,850301775@qq.com`

[2] Key Laboratory of Network & Information Security under the People's Armed Police, College of Cryptography Engineering, Engineering University of People's Armed Police, Xi'an, 710086, China
`yxyangyxyang@163.com`

Abstract. Multi-Key Full Homomorphic Encryption scheme (MKFHE) can perform arbitrary operation on encrypted data under different public keys (users), and the final ciphertext can be jointly decrypted. Therefore, MKFHE has natural advantages and application value in security multi-party computation (MPC). For BGV-type MKFHE scheme, the amount of ciphertexts and keys are relatively large, and the process of generating evaluation keys is complicated. In this paper, we presented an efficient BGV-type MKFHE scheme with short extended ciphertexts and less public parameters. Firstly, we construct a nested ciphertext extension for BGV and separable ciphertext extension for GSW, which can reduce the amount of the extended ciphertext. Secondly, we construct a hybrid homomorphic multiplication between RBGV ciphertext and RGSW ciphertext, which can reduce the size of input ciphertext and improve the computational efficiency. Finally, the coefficient of user's secret key is limited to $\{-1, 0, 1\}$, which can reduce the ciphertext size in key switching process. Comparing to CZW17 proposed in TCC17, analysis shows that the our scheme reduces the amount of ciphertext from $2k$ to $(k+1)$, and the evaluation key generation materials are reduced from $\sum_{l=0}^{L} 24\beta_l^2$ to $\sum_{l=0}^{L} 4\beta_B + 4\beta_l$, and the amount of evaluation keys are reduced from $4k^2\beta_l$ to $(k+1)^2\beta_B$, where k is the number of users participating in the homomorphic evaluations, L is a bound on the circuit depth, β_l and β_B relatively denotes the bit length of modulus q_l and the noise bound B. The reduction in the amount of data may lead to improvement in computational efficiency. Further more, the separable ciphertext extension for GSW can also be used in GSW-type MKFHE scheme such as CM15 to reduce the amount of ciphertext and improve the efficiency of homomorphic operations.

Keywords: Multi-key FHE, BGV scheme, ciphertext extension, public parameter, evaluation key, hybrid homomorphic multiplication.

1 Introduction

Full-homomorphic encryption (FHE), which can perform arbitrary operations on encrypted data without knowing the secret key, has the exchangeable property for encryption and computation. It has high research value in the current cloud computing environment, and can be widely used in ciphertext retrieval, secure multi-party computing (MPC), cloud data analysis, etc.

Since the first ideal-based FHE scheme Gen09 was proposed in 2009, many FHE schemes [DGHV10, BV11a, BV11b, BGV12, GSW13, AP14] was proposed following Gentry's blueprint.

Multi-key FHE (MKFHE) allows computations on ciphertexts under different secret keys, which is an extension of FHE in secure MPC. The concept of MKFHE was first introduced in [LATV12], and proposed a MKFHE scheme based on NTRU cryptosystem. However, the security of the NTRU-based cryptosystem can't be reduced to hard problems in lattice strictly. Its security has not been fully proved and needs to be further verified.

Clear and McGoldrick [CM15] proposed the first GSW-type MKFHE scheme based on the learning with error (LWE) problem whose security can be reduced to the worst-case hardness of problems on ideal lattices. Mukherjee and Wichs [MW16] simplified [CM15] and gave a construction of MKFHE scheme based on LWE. [MW16] can be used to construct a simple 1-round threshold decryption protocol and a two-round MPC protocol.

Both [CM15] and [MW16] need to determine the parties involved in homomorphic computation in advance and any new party cannot be allowed to join in during the homomorphic computation. This type of MKFHE is called single-hop in [PS16], comparing to multi-hop MKFHE whose result ciphertext can be employed to further evaluation with new parties, i.e. any new party can dynamically join the homomorphic evaluation at any time. Another similar concept named fully dynamic MKFHE was proposed in [BP16], which means that the bound of number of users does not need to be input during the setup procedure.

In TCC2017, Chen et al. proposed a BGV-type multi-hop MKFHE scheme [CZW17], which supports the Chinese Remainder Theorem (CRT)-based ciphertexts packing technique, and simplifies the ciphertext extension process in MKFHE. What's more, [CZW17] admits a threshold decryption protocol and two-round MPC protocol.

Our Contributions. At present, the BGV-type MKFHE scheme supporting batched multi-hop operations is represented by CZW17. This type of MKFHE scheme has the weaknesses of large ciphertexts and public parameters, and complicated computation for the generation of evaluation keys. In this paper, we improve these weaknesses as follows:

(1) We optimize the ciphertext extension process to transform BGV and GSW ciphertexts under different secrets keys to ciphertexts under the combination of all involving secrets keys, thus reduce the size of extended ciphertext by about a half.

(2) We optimize the generation process of evaluation keys. The hybrid homomorphic multiplication between RBGV ciphertexts and RGSW ciphertexts

are adopted in our scheme instead of homomorphic multiplication between two RBGV ciphertexts, thus reduce the size of the public parameters.

(3) We limit the coefficient of user's secret key to $\{-1, 0, 1\}$, thus reduce the amount and size of ciphertexts in the generation process of evaluation keys. These improvements can efficiently reduce the amount of data during the homomorphic operations, which may further reduce the homomorphic operations' computational complexity.

2 Preliminaries

In this paper, the bold upper case letters denote matrices, and the bold lower case letters denote vectors, and all the vectors are represented as columns. For a vector $\mathbf{a}$,we use $\mathbf{a}[i]$ to denote the i-th element in $\mathbf{a}$. For a matrice,we use $\mathbf{A}[i, j]$ to denote the element of the i-th row and j-th column in $\mathbf{A}$, and use $\mathbf{A}[i, :](\mathbf{A}[:, j])$ to denote the element of the i-th row (j-th column) in $\mathbf{A}$.

For security parameter I, let $\Phi_m(X)$ be the m-th cyclotomic polynomial which the degree $n = \phi_m$, where $\phi(\cdot)$ is the Euler's function. We define the ring $R = [X]/\Phi_m$, and $R_q = R/qR$ denotes the residue ring of R modulo an integer $q = plot(n)$, which means that the coefficients in R_q are in $[-q/2, q/2)$(except for $q = 2$). For $a \in R$, we use $\|a\|_\infty = \max_{0 \leqslant i \leqslant n-1} |a_i|$ to denote the standard l_∞-norm and use $\|a\|_1 = \sum_{i=0}^{n-1} |a_j|$ to denote the standard l_1-norm.

2.1 GLWE problem

GLWE problem. For security parameter , the GLWE problem is to distinguish the following two distributions: First distribution is the uniform samples $(\mathbf{a}_i, b_i) \in R_q^{n+1}$. In the second distribution, sampled $\mathbf{a}_i \leftarrow R_q^n$ and $\mathbf{s} \leftarrow R_q^n$ uniformly, $e_i \leftarrow c$, and the second distribution is the samples $(\mathbf{a}_i, b_i) \in R_q^{n+1}$ where $b_i =< \mathbf{a}_i, \mathbf{s} > +e_i$. The GLWE assumption is that the GLWE problem is infeasible.
LWE problem. The LWE problem is simply GLWE problem instantiated with $d = 1$.
RLWE problem. The RLWE problem is GLWE problem instantiated with $n = 1$.

2.2 Modulus switching

Modulus switching technique, which is proposed in [BGV12], is used to decrease the noise involved in the ciphertext by changing the original modulus q_l to another smaller q_{l-1} without change the corresponding plaintext. A randomized rounding function $[\cdot]_{q_l:q_{l-1}} : \mathbb{Z}_{q_l} \to \mathbb{Z}_{q_{l-1}}$ is used in modulus switching process. Given an integer x, the rounding process is defined as

$$[x]_{q_l:q_{l-1}} = \lfloor (q_{l-1}/q_l)x \rceil$$

Where $\lfloor\cdot\rceil$ is the rounding function. Actually the rounding error $[\mathbf{c}]_{q_l:q_{l-1}} - (q_{l-1}/q_l)\mathbf{c}$ is a sub-Gussian distribution with parameter $\sqrt{2\pi}$. Once given a RLWE ciphertext $\mathbf{c} = (b, \mathbf{a}) \in R_{q_l}^{n+1}$ with a modulus q_l and another modulus q_{l-1}, the plaintext modulus p, compute

$$\text{ModulusSwitch}(\mathbf{c}, q_l, q_{l-1}, p) = [b, \mathbf{a}]_{q_l:q_{l-1}} = ([b]_{q_l:q_{l-1}}, [a_1]_{q_l:q_{l-1}}, ..., [a_n]_{q_l:q_{l-1}}) \in R_{q_l}^{n+1}$$

Lemma 1. *On input the secret key $\mathbf{s} \in R_q^n$, a ciphertext $\mathbf{c} \in R_q^n$ in which the noise is a sub-Gussian distribution with the parameter σ, the output of ModulusSwitch$(\mathbf{c}, q_l, q_{l-1}, p)$ contains the noise which is a sub-Gussian distribution with parameter $\sqrt{(q_{l-1}/q_l\sigma)^2 + 2\pi(||\mathbf{s}||^2 + 1)}$.*

2.3 Key Switching

The key switching technique can be used to not only reduce the dimension of the ciphertext, but more generally can be used to transform a ciphertext $\mathbf{c_1}$ under one secret key vector $\mathbf{s_1}$ to a different ciphertext $\mathbf{c_2}$ that encrypts the same message, but is now decryptable under a second secret key vector $\mathbf{s_2}$. Given a ciphertext $\mathbf{c}_1 \in R_q^{n_1}$ under the secret key $\mathbf{s}_1 = (1, -\mathbf{z}_1) \in R_2^{n_1}$ and another secret key $\mathbf{s}_2 = (1, -\mathbf{z}_2) \in R_2^2$, let $\beta = \lfloor \log q \rfloor + 1$, the key switching process consists of two procedures:

- SwitchKeyGen$(\mathbf{s}_1 \in R_2^{n_1}, \mathbf{s}_2 \in R_2^{n_2})$: Sample $n_1 \cdot \beta$ RLWE instances $(\mathbf{a}_i\mathbf{z}_2 + pe_i, \mathbf{a}_i) \in R_q^{n_2}$, $i = 1, ..., n_1\beta$, compute $\bar{\mathbf{s}} = \text{Powersof}\,2(\mathbf{s}_1) \in R_q^{n_1 \cdot \beta}$, and output the switching keys:

$$\tau_{\mathbf{s}_1 \to \mathbf{s}_2} := \{\mathcal{K}_i = (\mathbf{a}_i\mathbf{z}_2 + pe_i + \bar{\mathbf{s}}[i], \mathbf{a}_i) \in R_q^{n_2}\}_{i=1,...,n_1\beta}$$

  Notice that $\langle \mathcal{K}_i, \mathbf{s}_2 \rangle = pe + \text{Powersof2}(\mathbf{s}_1)[i] \bmod q$.
- SwitchKey$(\tau_{\mathbf{s}_1 \to \mathbf{s}_2}, \mathbf{c}_1)$: Compute $\bar{\mathbf{c}}_1 = \text{BitDecomp}(\mathbf{c}_1) \in R_q^{n_1 \cdot \beta}$, and output the new ciphertext $\mathbf{c}_2$ under the secret key $\mathbf{s}_2$:

$$\mathbf{c}_2 = \sum\nolimits_{i=1}^{n_1\beta} \mathcal{K}_i \cdot \bar{\mathbf{c}}_1[i] \in R_q^{n_2}$$

Lemma 2. *On input $\mathbf{z} \in \mathbb{Z}_q^n$, a message $m \in \mathbb{Z}_t$, a ciphertext $\mathbf{c} \in LWE_{\mathbf{z}}^{t/q}(m)$ in which the noise is a sub-Gussian distribution with parameter α, an evaluation key $\mathbf{k}_{i,j,v} \in LWE_{\mathbf{s}}^{q/q}(v \cdot z_i B_{ks}^j)$ in which the noise is a sub-Gussian distribution with parameter σ, the output of key switching KeySwitch$(\mathbf{c}, \{\mathbf{k}_{i,j,v}\})$ contains the noise which is a sub-Gussian distribution with parameter $\sqrt{\alpha^2 + Nd_{ks}\sigma^2}$.*

2.4 Cryptographic Definitions for Leveled MKFHE Scheme

Definition 1. *A leveled multi-key FHE scheme consists of a set of algorithms described as follows:*

- Setup$(1^\lambda, 1^K, 1^L)$*: Given the security parameter λ, a bound K on the number of keys, a bound L on the circuit depth,output the public parameter pp.*

- Gen(pp)*: Given the public parameter pp,output the public key and secret key of party $i(i=1,...,K)$,and output the materials which are required for the generation of evaluation keys evk.*
- Enc(pp,pk_i,m)*: Given the public key pk_i of party i and a message μ, output the ciphertext ct_i which contains the index of the corresponding secret key and the level tag.*
- Dec(pp, $(sk_{i_1}, sk_{i_2}, ..., sk_{i_k}), ct_S$)*: Given a ciphertext ct_S corresponding to a set of parties $S = \{i_1, i_2, ..., i_k\} \subseteq [K]$,and their secret keys $sk_S = \{sk_{i_1}, sk_{i_2}, ..., sk_{i_k}\}$, output the message μ.*
- Eval($pp, evk, C, (ct_{S_1}, pk_{S_1}), ..., (ct_{S_t}, pk_{S_t})$)*: Given t tuples $\{(ct_{S_i}, pk_{S_i})\}_{i=1,...,t}$ and a boolean circuit $\mathcal{C}$ which is needed to be evaluated,each tuple contains a ciphertext ct_{S_i} corresponding to a set of secret keys indexed by $S_i = i_1, ..., i_{k_i} \subseteq [K]$ and a set of public keys $pk_{S_i} = \{pk_j, \forall j \in S_i\}$. Output a ciphertext ct corresponding to a set of secret keys indexed by $S = \cup_{i=1}^t S_i \subseteq [K]$.*

If the input ciphertext of Eval($\cdot$) *can be fresh ciphertext or intermediate results after any homomorphic operation, the MKFHE scheme satisfies the multi-hop property.*

Definition 2 (Correctness). *A leveled multi-hop MKFHE scheme is correct if for any circuit $\mathcal{C}$ of depth at most L with t input wires and a set of tuples $\{(ct_{S_i}, pk_{S_i})\}_{i\in\{1,...,t\}}$, letting $\mu_i = \mathrm{Dec}(sk_{S_i}, ct_{S_i})$, where $sk_{S_i} = \{sk_j, \forall j \in S_i\}$, $i = 1, ..., t$, it holds that*

$$\Pr[\mathrm{Dec}(sk_S, Eval(\mathcal{C}, (ct_{S_1}, pk_{S_1}), ..., (ct_{S_t}, pk_{S_t}))) \neq \mathcal{C}(\mu_1, ..., \mu_t)] = negl(\lambda)$$

where $S = \cup_{i=1}^t S_i \subseteq [K]$, $pp \leftarrow \mathrm{Setup}(1^\lambda, 1^K, 1^L)$, $(pk_j, sk_j) \leftarrow Gen(pp)$ for $j \in [S]$.

Definition 3 (Compactness). *A leveled multi-hop MKFHE scheme is compact if there exists a polynomial $poly(\cdot,\cdot,\cdot)$ such that $|ct| \leqslant poly(\lambda, K, L)$, which means that the length of ct is independent of the circuit $\mathcal{C}$, but can depend of λ, K and L.*

2.5 The Ring-GSW Scheme

In this section, we describe a variant of ring-LWE based GSW scheme with ring element plaintext.

- RGSW.Setup(1^λ): For the security parameter λ, let $\Phi_m(X)$ be the m-th cyclotomic polynomial which the degree $n = \phi_m$, where $\phi(\cdot)$ is the Euler's function. Given a modulus $q = ploy(n)$, a small constant integer p, a B-bound discrete distribution χ in ring $R = \mathbb{Z}[X]/\Phi_m[X]$ for $B \ll q$, and an integer $N = O(nlogq)$. Let $\beta = \lfloor \log q \rfloor + 1$, we use ring $R_q = R/qR$.
- RGSW.KeyGen(1^n): Sample $z \leftarrow R_3$, choose a random vector $\mathbf{a} \in R_q^{2\beta}$ and $\mathbf{e} \leftarrow \chi^{2\beta}$ uniformly,output the secret key $\mathbf{s} = (1, -z)^T \in R_3^2$ and public key $\mathbf{P} = [\mathbf{a}z + p\mathbf{e}, \mathbf{a}] = [\mathbf{b}, \mathbf{a}] \in R_q^{2\beta\times 2}$.

– RGSW.EncRand$(r, \mathbf{P})$: This procedure is to generate the encryption of randomness which is used in the real encryption. On inputs the message $r \leftarrow R_q$, sample $r_i \leftarrow \chi(i = 1, ..., \beta)$ and two vectors $\mathbf{e'}_1, \mathbf{e'}_2 \leftarrow \chi^\beta$, output the encryption of the randomness:

$$\text{RGSW}\,.EncRand_{\mathbf{s}}(r) = \mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2] \in R_q^{\beta\times 2}$$

where for $i = 1, ..., \beta$, $\mathbf{f}_1[i] = \mathbf{b}[i]r_i + p\mathbf{e'}_1[i] + \text{Powersof}\,2(r)[i] \in R_q$, $\mathbf{f}_2[i] = \mathbf{a}[i]r_i + p\mathbf{e'}_2[i] \in R_q$. Notice that $\mathbf{Fs} = [p\tilde{\mathbf{e}} + \text{Powersof}\,2(r)] \in R_q^\beta$ for some small $\tilde{\mathbf{e}} = \mathbf{e}[i]r_i + \mathbf{e'}_1[i] - \mathbf{e'}_2[i]z$.

RGSW . Enc$(\mu, \mathbf{P})$: On inputs $\mu \in R_q$ and the public key $\mathbf{P} = [\mathbf{b}, \mathbf{a}] \in R_q^{2\beta\times 2}$,sample a random element $r \leftarrow \chi$ and an error matrix $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2] \leftarrow \chi^{2\beta\times 2}$, output the ciphertext

$$\begin{aligned}\text{RGSW.Enc}_{\mathbf{s}}(\mu) &= \mathbf{C} = r\mathbf{P} + p\mathbf{E} + \mu\mathbf{G} \\ &= r[\mathbf{b}, \mathbf{a}] + p\mathbf{E} + \mu\mathbf{G} \\ &= [r\mathbf{a}z + p(r\mathbf{e} + \mathbf{e}_1), r\mathbf{a} + p\mathbf{e}_2] + \mu\mathbf{G} \in R_q^{2\beta\times 2}\end{aligned}$$

where $\mathbf{G} = (\mathbf{I}, 2\mathbf{I}, ..., 2^{\beta-1}\mathbf{I})^T \in R_q^{2\beta\times 2}$. Notice that $\mathbf{C}\cdot\mathbf{s} = p\tilde{\mathbf{e}} + \mu\mathbf{G}\cdot\mathbf{s} \in R_q^{2\beta}$.

3 Efficient Techniques for Homomorphic Operations

This section introduces some efficient functions and algorithms for homomorphic operations, which are our main innovations. Mainly includes: nested ciphertext extension for BGV, separable ciphertext extension for GSW, generation of evaluation keys and the hybrid homomorphic multiplication between RBGV ciphertext and RGSW ciphertext.

3.1 The Process of Homomorphic Operations

The process of homomorphic operation in MKFHE scheme is shown in Fig.1. The purpose of the system is to perform homomorphic operations on ciphertexts of different users in the cloud. In the initialization phase, the user uploads his own BGV ciphertext and the encrypted GSW and BGV ciphertext pieces of his secret key to the cloud.

Step 1(ciphertext extension): run the ciphertext extension function to the BGV ciphertext participating in the homomorphic operation, and get the extended ciphertext corresponding to the user set S.

Step 2(homomorphic operation): do homomorphic operation on the user's extended ciphertext and get a high-dimensional BGV ciphertext.

Step 3(hybrid homomorphic multiplication): respectively select the BGV/GSW ciphertext of secret key from different users and do ciphertext extension and hybrid homomorphic multiplication between these ciphertext to obtain the evaluation keys.

Step 4(key switching): perform key switching operation on the ciphertext outputted in the third step using the evaluation keys.

Step 5(modulus switching): perform modulus switching operation on the result ciphertext of step four and output the final ciphertext.

The first two steps and the third step in the system can be performed simultaneously.

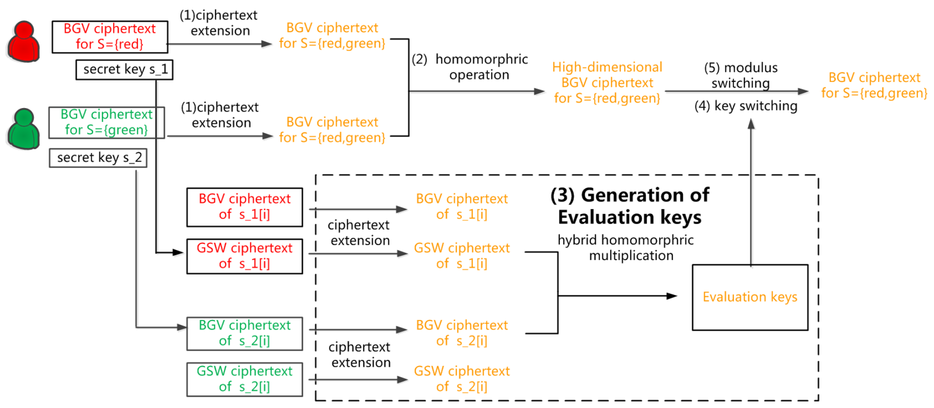

Fig. 1. The process of homomorphic operation in MKFHE scheme

3.2 Ciphertext Extension

Nested ciphertext extension for BGV $\mathrm{BGV.CTExt}(\mathbf{c}_l, S')$: Input a ciphertext tuple $ct = \{\mathbf{c} \in R_{q_l}^{k+1}, S = \{i_1, ..., i_k\}, l\}$ corresponding to k users and a set of parties $S' = \{j_1, ..., j_{k'}\}$ for $S \in S'$, output an extended tuple $ct' = \{\bar{\mathbf{c}} \in R_{q_l}^{k'+1}, S' = \{j_1, ..., j_{k'}\}, l\}$. The extending algorithm is as follows:

(a) Divide the ciphertext $\mathbf{c}$ into $k+1$ sequential sub-vectors, which can be indexed by $S = \{i_1, ..., i_k\}$ (except for the first sub-vector), i.e.,

$$\mathbf{c} = (c_{i_0} | c_{i_1} | \cdots | c_{i_k}) \in R_{q_l}^{k+1}$$

where the corresponding secret key is $\mathbf{s}_l = (1, -z_{l,i_1}, ..., -z_{l,i_k}) \in R_{q_l}^{k+1}$.

(b) The extended ciphertext $\bar{\mathbf{c}}$ consists of $k'+1$ sequential sub-vectors, which can be indexed by $S' = \{j_1, ..., j_{k'}\}$ (except for the first sub-vector), i.e.,

$$\bar{\mathbf{c}} = (c_{i_0} | c_{j_1} | \cdots | c_{j_{k'}}) \in R_{q_l}^{k+1}$$

where $c_{j_x} = \begin{cases} 0 \ , \ j_x \notin S \\ c_{i_x} \ , \ j_x \in S \end{cases}$, $x \in \{1, ..., k'\}$, and the corresponding secret key is $\bar{\mathbf{s}}_l = (1, -z_{l,j_1}, ..., -z_{l,j_{k'}}) \in R_{q_l}^{k'+1}$. It's easy to verify that $\langle \mathbf{c}, \mathbf{s}_l \rangle = \langle \bar{\mathbf{c}}, \bar{\mathbf{s}}_l \rangle$.

Separable ciphertext extension for GSW RGSW.CTExt($\mathbf{C}_i, \mathbf{F}_i, \{\mathbf{P}_j, j = 1, ..., k\}$): On input the i-th party's ciphertext $\mathbf{C}_i \in R_q^{2\beta \times 2}$, an encryption F_i of randomness r_i and the public keys of all involved parties $\mathbf{P}_j = [\mathbf{b}_j, \mathbf{a}_j]$, $j = 1, ..., i-1, i+1, ...k$. Output the extended ciphertext:

$$\bar{\mathbf{C}}_i = \begin{bmatrix} \mathbf{X}_{1,0} + \mathbf{C}_{i,0} & \mathbf{C}_{i,1} & & \mathbf{X}_{1,1} & 0 & 0 \\ \mathbf{X}_{2,0} + \mathbf{C}_{i,0} & 0 & \ddots & \vdots & & 0 \\ \vdots & \vdots & & \mathbf{C}_{i,1} & & \vdots \\ \mathbf{X}_{k-1,0} + \mathbf{C}_{i,0} & & & \vdots & \ddots & \\ \mathbf{X}_{k,0} + \mathbf{C}_{i,0} & 0 & & \mathbf{X}_{k,1} & & \mathbf{C}_{i,1} \end{bmatrix} \in R_q^{2k\beta \times (k+1)}$$

where $\mathbf{C}_i = \text{RGSW}.\text{Enc}_{\mathbf{s}_i}(\mu_i) = [\mathbf{C}_{i,0}, \mathbf{C}_{i,1}] = r_i[\mathbf{a}z_i + p\mathbf{e}_i, \mathbf{a}] + p\mathbf{E}_i + \mu_i \mathbf{G} \in R_q^{2\beta \times 2}$, $\mathbf{X}_j = [\mathbf{X}_{j,0}, \mathbf{X}_{j,1}] = [BitDecomp(\tilde{\mathbf{b}}_j[u])\mathbf{F}_i] \in R_q^{2\beta \times 2}$, $\tilde{\mathbf{b}}_j[u] = \mathbf{b}_j[u] - \mathbf{b}_i[u]$, $u = 1, ..., 2\beta$, $\beta = \lfloor \log q \rfloor + 1$ and the corresponding secret key $\bar{\mathbf{s}} = (1, -z_1, \ldots - z_k)$.

Correctness of Ciphertext Extension for GSW. In order to ensure the correctness of the extending algorithm of GSW ciphertext ($\bar{\mathbf{C}}_i \bar{\mathbf{s}} = p\tilde{\mathbf{e}} + \mu_i \bar{\mathbf{G}} \bar{\mathbf{s}}$), it is necessary to verify that the j-th row in $\bar{\mathbf{C}}_i$ satisfies:

$$(\mathbf{X}_{j,0} + \mathbf{C}_{i,0}) - \mathbf{C}_{i,1} z_j - \mathbf{X}_{j,1} z_i = \mathbf{C}_i \mathbf{s}_j + \mathbf{X}_j \mathbf{s}_i = p\tilde{\mathbf{e}} + \mu_i \mathbf{G} \mathbf{s}_j$$

where $\tilde{\mathbf{e}} \in R^{2\beta}$ is a small noise vector, and the analysis process is as follows:

$$\begin{aligned} \mathbf{C}_i \mathbf{s}_j &= r_i[\mathbf{a}z_i + p\mathbf{e}_i - \mathbf{a}z_j] + p\mathbf{E}\mathbf{s}_j + \mu_i \mathbf{G}\mathbf{s}_j \\ &= p\tilde{\mathbf{e}} + \mu_i \mathbf{G}\mathbf{s}_j - r_i \tilde{\mathbf{b}}_j \\ \mathbf{X}_j \mathbf{s}_i &= BitDecomp(\tilde{\mathbf{b}}_j)\mathbf{F}_i \cdot \mathbf{s}_i \\ &= BitDecomp(\tilde{\mathbf{b}}_j)[p\tilde{\mathbf{e}} + \text{Powersof}\,2(r_i)] \\ &= p\tilde{\mathbf{e}} + r_i \tilde{\mathbf{b}}_j \end{aligned}$$

Then we can get $\mathbf{C}_i \mathbf{s}_j + \mathbf{X}_j \mathbf{s}_i = p\tilde{\mathbf{e}} + \mu_i \mathbf{G}\mathbf{s}_j$, thus $\bar{\mathbf{C}}_i \bar{\mathbf{s}} = p\tilde{\mathbf{e}} + \mu_i \bar{\mathbf{G}} \bar{\mathbf{s}}$, where $G = (\mathbf{I}_{2k}, 2\mathbf{I}_{2k}, ..., 2^{\beta-1}\mathbf{I}_{2k})^T \in R_q^{2k\beta \times 2k}$.

3.3 Generation of Evaluation Key

In this paper, we optimize the generation of evaluation keys during the key-switching process in [CZW17]. We admit the hybrid homomorphic multiplication between RBGV ciphertexts and RGSW ciphertexts instead of homomorphic multiplication between two RBGV ciphertexts, thus the noise involved in the evaluation keys is decreased. What's more, we limit the coefficient of user's secret key to $\{-1, 0, 1\}$, thus the BitDecomp($\cdot$) and Powersof 2($\cdot$) techniques are no longer required in the key-switching process, thus reduce the number of ciphertexts during key-switching process.

MKFHE.EVKGen(em_S,pk_S): Given a level-l extended secret key $\hat{\mathbf{s}}_l = \bar{\mathbf{s}}_l \otimes \bar{\mathbf{s}}_l \in R_3^{(k+1)^2}$,where $\bar{\mathbf{s}}_l = (1, -z_{l,j_1}, ..., -z_{l,j_k}) \in R_3^{k+1}$, and all the level-$(l-1)$

public keys $[\mathbf{b}_{l-1,j}, \mathbf{a}_{l-1,j}]_{j\in\{j_1,...,j_k\}}$ involved in $S = \{j_1, ..., j_k\}$. For the user $j \in \{1, ..., k\}$, define the ciphertexts $\Psi_{l,j} \triangleq RGSW.Enc_{\mathbf{s}_{l-1,j}}(z_{l,j})$ and $\Phi_{l,j,m} \triangleq RBGV.Enc_{\mathbf{s}_{l-1,j}}(2^m \cdot z_{l,j})$, where $m \in \{0, ..., \beta_l - 1\}$. Output the evaluation keys $evk = \{\mathcal{K}_{m,\xi} \in R^2_{q_l}\}_{m\in\{0,...,\beta_l-1\};\xi\in\{1,...,(k+1)^2\}}$

The generation process of the evaluation keys is presented in Algorithm1.

Algorithm 1 the generation of $evk = \{\mathcal{K}_{m,\xi}\}$

```
Input:$\Psi_{l,j}$,$\mathbf{F'}_{l,j}$,$\Phi_{l,j,m}$,$m \in \{0, ..., \beta_l - 1\}$,$j \in \{1, ..., k\}$,$\zeta \in \{0, ..., k\}$
  for $\zeta' \in \{0, ..., k\}$ do
    $\bar{\Psi}_l[\zeta'] \triangleq \begin{cases} RGSW.Enc_{\bar{\mathbf{s}}_{l-1}}(1) \; \zeta' = 0 \\ \text{RGSW} .CTExt_{\bar{\mathbf{s}}_{l-1}}(\Psi_{l,\zeta'}) \text{ else} \end{cases}$
  for $\zeta \in \{0, ..., k\}$ do
    for $m \in \{0, ..., \beta_l - 1\}$ do
      $\bar{\Phi}_{l,m}[\zeta] \triangleq \begin{cases} RBGV.Enc_{\bar{\mathbf{s}}_{l-1}}(2^m) \; \zeta = 0 \\ \text{RBGV} .CTExt_{\bar{\mathbf{s}}_{l-1}}(\Phi_{l,\zeta,m}) \text{ else} \end{cases}$
  for $\zeta' = [0, ..., k]$ do
    for $\zeta = [0, ..., k]$ do
      for $m = [1, ..., \beta_B]$ do
        $\mathcal{K}_{m,(k+1)\zeta'+\zeta} = \bar{\Psi}_{l,j}[\zeta'] \boxdot \bar{\Phi}_{l,j',m}[\zeta]$
  Output $evk = \{\mathcal{K}_{m,\xi}\}_{m\in\{0,...,\beta_l-1\};\xi\in\{1,...,(k+1)^2\}}$
```

3.4 Hybrid Homomorphic Multiplication

Definition 4 (Hybrid homomorphic multiplication). *We define the product $\boxdot$ as*

$$\boxdot : RGSW \times RBGV \to RBGV$$

$$(\mathbf{C}_2, \mathbf{c}_1) \to \mathbf{C}_2 \boxdot \mathbf{c}_1 = BD(c_1) \cdot \mathbf{C}_2$$

Corollary 1. Let $\mathbf{C}_2$ be a valid RGSW sample of message μ_2 and let $\mathbf{c}_1$ be a valid RBGV sample of message μ_1. Then $\mathbf{C}_2 \boxdot \mathbf{c}_1$ is a RBGV sample of message $\mu_2 \cdot \mu_1$ and $\|Err(\mathbf{C}_2 \boxdot \mathbf{c}_1)\|_\infty \leqslant (2\beta)N \cdot 2\sigma||Err(\mathbf{C}_2)||_\infty + \|\mu_2\|_\infty||Err(\mathbf{c}_1)||_\infty$, $Var(Err(\mathbf{C}_2 \boxdot \mathbf{c}_1)) \leqslant 2p\beta(2N + 1)Var(\mathbf{e}) + pNVar(\mathbf{e}_1)$, where $p\mathbf{e}_1$ is the noise of c_1, $\mathbf{e} \leftarrow \chi$ is the noise involved in $\mathbf{C}_2$.

4 New Construction of BGV-type MKFHE Scheme

In this section, we present the details of our MKFHE scheme based on [BGV12]. For convenience, in the following we use RGSW .Enc$_\mathbf{s}(\mu)$ to denote a GSW ciphertext (which may not be fresh) that can be decrypted to μ with the secret key $\mathbf{s}$. Also we adopt the same subroutines such as ModulusSwitch and SwitchKey in [BGV12] scheme. For details of the original BGV scheme, see Appendix 1.

4.1 Basic Scheme

- MKFHE.Setup($1^\lambda, 1^K, 1^L$): For the security parameter λ, let $\varPhi_m(X)$ be the m-th cyclotomic polynomial which the degree $n = \phi_m$, where $\phi(\cdot)$ is the Euler's function. Given a bound K on the number of keys, a bound L on the circuit depth with L decreasing modulus for each level and a small integer p coprime with all q_l, a B-bound discrete distribution χ in ring $R = \mathbb{Z}[X]/\varPhi_m(X)$ for $B \ll q_l$. Let $\beta = \lfloor \log q \rfloor + 1$, $\beta_l = \lfloor \log q_l \rfloor + 1$, $\beta_B = \lfloor \log B \rfloor + 1$ and choose $L+1$ random public vectors $a_l \in R_{q_l}^{2\beta_l}$ for $l \in \{0, \ldots, L\}$. All the following algorithms implicitly take the public parameter $pp = (R, B, \chi, \{q_l, \mathbf{a}_l\}_{l \in \{0,\ldots,L\}}, p)$ as input.
  Let S be an ordered set containing all indexes of the parities that the ciphertext corresponding to. Without loss of generality, we assume that the indexes in S are always arranged from small to large and S has no duplicate elements. Usually, the ciphertext tuple $ct = \{\mathbf{c}, S, l\}$ contains the real ciphertext $\mathbf{c}$, the user set S and a level tag l.
- MKFHE.KeyGen($j \in K$): Generate keys for the j-th party. For l from L down to 0, do the following:
  (a) Sample $z_{l,j} \leftarrow \chi$, and set $\mathbf{s}_{l,j} = (1, -z_{l,j})^T \in R_{q_l}^2$. The secret key for the j-th party is $sk_j = \{\mathbf{s}_{l,j}\}$, $l \in \{L, ..., 0\}$.
  (b) Choose a random vector $\mathbf{e}_{l,j} \leftarrow \chi^{2\beta_l}$, and generate $2\beta_l$ ring-LWE instances:
  $$\mathbf{p}_{l,j} := [\mathbf{a}_{l,j} z_{l,j} + p\mathbf{e}_{l,j}, \mathbf{a}_{l,j}] = [\mathbf{b}_{l,j}, \mathbf{a}_{l,j}] \in R_q^{2\beta_l \times 2}$$
  The public key for the j-th party is $pk_{l,j} = \{\mathbf{p}_{l,j}\}$, $l \in \{L, ..., 0\}$.
  (c) For $j \in \{1, ..., k\}$, compute the materials used in the generation of evaluation keys.
  $$em_j = \{(\varPhi_{l,j} \in R_{q_l}^{2\beta_B}), (\varPsi_{l,j} \in R_{q_l}^{2\beta_l \times 2}, \mathbf{F'}_{l,j} \in R_{q_l}^{\beta_l \times 2})\}_{l=\{L,\ldots,0\}}$$
  (i) For $m \in \{0, ..., \beta_l - 1\}$, $j \in \{1, ..., k\}$, $\zeta \in \{0, ..., k\}$, compute
  $$\begin{aligned}\varPhi_{l,j,m} &\triangleq \text{RBGV.Enc}_{\mathbf{s}_{l-1,j}}(2^m \cdot z_{l,j}) \\ &= \{r_{l,j,m}\mathbf{b}_{l-1,j} + 2e_{l,j,m} + 2^m \cdot z_{l,j}, r_{l,j,m}\mathbf{a}_{l-1} + 2e'_{l,j,m}\} \in R_{q_l}^2\end{aligned}$$
  $$\bar{\varPhi}_{l,m}[\zeta] \triangleq \begin{cases} \text{RBGV.Enc}_{\bar{\mathbf{s}}_{l-1}}(2^m) & \zeta = 0 \\ \text{RBGV.CTExt}_{\bar{\mathbf{s}}_{l-1}}(\varPhi_{l,\zeta,m}) & \text{else} \end{cases}$$
  (ii) For $i \in \{0, ..., \beta_l - 1\}$, $j \in \{1, ..., k\}$, $\zeta \in \{0, ..., k\}$ compute
  $$\begin{aligned}\varPsi_{l,j} &\triangleq \text{RGSW.Enc}_{\bar{\mathbf{s}}_{l-1}}(z_{l,j}) \\ &= \{r'_{l,j}[\mathbf{b}_{l-1}, \mathbf{a}_{l-1}] + p\mathbf{E'}_{l,j} + z_{l,j}\mathbf{G}\} \in R_{q_l}^{2\beta_l \times 2}\end{aligned}$$
  $$\bar{\varPsi}_l[\zeta'] \triangleq \begin{cases} \text{RGSW.Enc}_{\bar{\mathbf{s}}_{l-1}}(1) & \zeta' = 0 \\ \text{RGSW.CTExt}_{\bar{\mathbf{s}}_{l-1}}(\varPsi_{l,\zeta'}) & \text{else} \end{cases}$$
  $$\mathbf{F'}_{l,j} = \text{RGSW.EncRand}(r'_{l,j}, pk_{l-1,j}) \in R_{q_l}^{\beta_l \times 2}$$

– MKFHE.Enc(pk_j, μ): Set $S = \{j\}$, and input the public key of the j-th party $\mathbf{p}_{L,j}$ and a message $\mu \in R_p$, choose a random and an error matrix , generate the level-L ciphertext of the message $\mu \in R_p$:

$$\mathbf{c} = (c_{j,0}, c_{j,1}) = (r\mathbf{b}_{L,j}[1] + pe + \mu_j, r\mathbf{a}_L[1] + pe') \in R_{q_L}^2$$

And output the tuple $ct = \{\mathbf{c}, \{j\}, L\}$.

– MKFHE. Dec($sk_S, ct = \{\mathbf{c}, S, l\}$): Suppose that $S = \{j_1, ..., j_k\}$ and sk_S consist of all the parties' secret keys whose indexes are contained in S, i.e. $sk_S = \{sk_{j_1}, sk_{j_2}, ..., sk_{j_k}\}$. Let

$$\bar{\mathbf{s}}_l = (1, -z_{l,j_1}, \ldots - z_{l,j_k})$$

Once given a level-l ciphertext c, we can get the message

$$\mu =< \mathbf{c}, \bar{\mathbf{s}}_l > \mathrm{mod} q_l \bmod p$$

– MKFHE.Eval($(pk_{l,j_1}, \ldots, pk_{l,j_k}), em_S, \mathcal{C}, (ct_1, \ldots ct_t)$): Assume that the sequence of ciphertexts $ct_i = \{\mathbf{c}_i, S_i, l\}_{i\in\{1,\ldots,t\}}$ are at the same level-l (If needed, use SwitchKey and ModulusSwitch to make it so). Let $S = \cup_{i=1}^t S_i = (j_1, \ldots, j_k)$. Then the outline of evaluation of the Boolean circuit C is as follows.
(a) For $i \in \{1, \ldots, t\}$, compute BGV.CTExt($\mathbf{c}_i, S$) to get an extended ciphertext $\bar{\mathbf{c}}_i$ which encrypt the same message under $\bar{\mathbf{s}}_l$. Here $\bar{\mathbf{s}}_l := (1, -z_{l,j_1}, \ldots - z_{l,j_k})$ is indexed by S.
(b) Compute $evk_S = $ MKFHE.EvkGen(em_S) to generate the evaluation key for the extended ciphertext.
(c) Evaluate each gate of the circuit $\mathcal{C}$ by using the two basic homomorphic operations MKFHE.EvalAdd($evk_S, \bar{\mathbf{c}}_{i_1}, \bar{\mathbf{c}}_{i_2}$) and MKFHE.EvalMult($evk_S, \bar{\mathbf{c}}_{i_1}, \bar{\mathbf{c}}_{i_2}$).

In the following subsections, we will detail how to perform the two basic homomorphic operations MKFHE.EvalAdd($\cdot$) and MKFHE.EvalMult($\cdot$) on two (extended) ciphertext $\bar{\mathbf{c}}_1$, $\bar{\mathbf{c}}_2 \in R_{q_l}^{k+1}$ corresponding to the user set $S = \{j_1, ..., j_k\}$. The evaluation key is defined as :

$$evk_l = \tau_{\hat{\mathbf{s}}_l \to \bar{\mathbf{s}}_{l-1}} = \{\mathcal{K}_{m,\xi}\}_{m=1,\ldots,\beta_l, \xi=1,\ldots,(k+1)^2}$$

where $\bar{\mathbf{s}}_l = (1, -z_{l,j_1}, ..., -z_{l,j_k}) \in R_3^{k+1}$, $\hat{\mathbf{s}}_l = \bar{\mathbf{s}}_l \otimes \bar{\mathbf{s}}_l$, $\bar{\mathbf{s}}_{l-1} = (1, -z_{l-1,j_1}, ..., -z_{l-1,j_k}) \in R_3^{k+1}$ and $\mathcal{K}_{m,\xi} \in R_{q_l}^{k+1}$ such that $\langle \mathcal{K}_{m,\xi}, \bar{\mathbf{s}}_{l-1} \rangle = pe_{m,\xi} + 2^{m-1}\hat{\mathbf{s}}_l[\xi] \in R_{q_l}$, and the canonical form of $e_{m,\xi}$ is small.

– MKFHE.EvalAdd($evk_S, \bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2$): On input two (extended) ciphertext $\bar{\mathbf{c}}_1$, $\bar{\mathbf{c}}_2 \in R_{q_l}^{k+1}$ at the same level-l under the same secret key $\bar{\mathbf{s}}_l \in R_3^{k+1}$ (If needed, use SwitchKey and ModulusSwitch to make it so).
(a) Compute $\bar{\mathbf{c}}_3' \triangleq \bar{\mathbf{c}}_1 + \bar{\mathbf{c}}_2 \bmod q_l$ under the secret key $\bar{\mathbf{s}}_{l-1} \in R_3^{k+1}$.
(b) Compute $\bar{\mathbf{c}}_3'' \triangleq$ SwitchKey($\bar{\mathbf{c}}_3', \tau_{\hat{\mathbf{s}}_l \to \bar{\mathbf{s}}_{l-1}}, q_l$) under the secret key $\bar{\mathbf{s}}_{l-1} \in R_3^{k+1}$.
(c) Compute $\bar{\mathbf{c}}_3 \triangleq$ ModulusSwitch($\bar{\mathbf{c}}_3'', q_{l-1}$).

– MKFHE.EvalMult$(evk_S, \bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2)$: On input two (extended) ciphertext $\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2 \in R_3^{k+1}$ at the same level-l under the same secret key $\bar{\mathbf{s}}_l \in R_3^{k+1}$ (If needed, use SwitchKey and ModulusSwitch to make it so).
  (a) Compute $\bar{\mathbf{c}}_3' \triangleq \bar{\mathbf{c}}_1 \otimes \bar{\mathbf{c}}_2 \bmod q_l$ under the secret key $\hat{\mathbf{s}}_l = \bar{\mathbf{s}}_l \otimes \bar{\mathbf{s}}_l \in R_3^{(k+1)^2}$.
  (b) Compute $\bar{\mathbf{c}}_3'' \triangleq \text{SwitchKey}(\bar{\mathbf{c}}_3', \tau_{\hat{\mathbf{s}}_l \to \bar{\mathbf{s}}_{l-1}}, q_l)$ under the secret key $\bar{\mathbf{s}}_{l-1} \in R_3^{k+1}$.
  (c) Compute $\bar{\mathbf{c}}_3 \triangleq \text{ModulusSwitch}(\bar{\mathbf{c}}_3'', q_{l-1})$.

4.2 Analysis

Security analysis. The basic encryption scheme in this paper and CZW17 adopt the same BGV encryption scheme and GSW encryption scheme. There are two main differences between our scheme and CZW17: on the one hand, we propose the nested ciphertext extension for BGV and separable ciphertext extension for GSW, and construct a hybrid homomorphic multiplication between RBGV ciphertext and RGSW ciphertext. The input and output of these three functions are ciphertext, and the homomorphic operations are all performed in ciphertext, so the security of our scheme will not be reduced comparing to CZW17. On the other hand, we limit the coefficient of user's secret key to {-1,0,1} (comparing to a B-bound discrete distribution χ in ring R), which can greatly reduce the ciphertext size generated by key switching process. So the dimension of polynomial in ring R is needed to increase to some extent to ensure the security of the scheme.

Tab.1. Comparison of storage overhead between our scheme and CZW17

	Ciphertext Size (k users)	Evaluation Key Generation Materials Size	Evaluation Key Size (level-l)
CZW17	$2k\beta_l N$	$\sum_{l=0}^{L} 24\beta_l^3 N$	$4k^2\beta_l^2 N$
Our Scheme	$(k+1)\beta_l N$	$\sum_{l=0}^{L}(4\beta_B + 4\beta_l)\beta_l N$	$(k+1)^2\beta_B\beta_l N$

Efficiency analysis. Table 1 shows that our scheme has obvious advantages in terms of the three main factors which will affecting the scheme's efficiency: ciphertext size (for k users), size of evaluation key generation materials and evaluation keys. For homomorphic operations on large-scale users, our scheme improve the efficiency obviously. From the process of homomorphic operation in the whole MKFHE scheme we can see that, as we limit the coefficient of user's secret key to {-1,0,1} , thus the ciphertext size of the secret key is reduced to β_B and the efficiency of our scheme is improved, which can make up for the increase of computational complexity caused by the increase of polynomial dimension N.

5 Conclusion

In this paper, we proposed an efficient multi-key FHE by constructing some efficient techniques such as nested ciphertext extension for BGV and separable ciphertext extension for GSW, and the hybrid homomorphic multiplication between RBGV ciphertext and RGSW ciphertext, which reduce the size of public parameters, and evaluation keys, and thus improve the efficiency of BGV-type MKFHE scheme. Furthermore, the separable ciphertext extension for GSW can also be used in GSW-type MKFHE scheme such as CM15 to improve the efficiency of homomorphic operations.

Appendix

1 BGV Scheme

- BGV.Setup$(1^\lambda, 1^L)$: For the security parameter λ, let $\Phi_m(X)$ be the m-th cyclotomic polynomial which the degree $n = \phi_m$, where $\phi(\cdot)$ is the Euler's function. Given a bound K on the number of keys, a bound L on the circuit depth with L decreasing modulus $q_L \gg q_{L-1} \gg \cdots \gg q_0$ for each level and a small integer p coprime with all q_1, a B-bound discrete distribution χ in ring $R = \mathbb{Z}[X]/\Phi_m[X]$ for $B \ll q_l$, $N = n \cdot \text{polylog}(q)$. We use $R_q = R/qR$.
- BGV.KeyGen$(1^n, , 1^L)$: For l from L down to 0, do the following:
  (a) Sample $\mathbf{z}_l \leftarrow \chi^n$, the secret key is defined as $sk = \mathbf{s}_l \leftarrow (1, -z_l[1], ..., -z_l[n]) \in R_{q_l}^{n+1}$.
  (b) Sample a random matrix $\mathbf{A'}_l \leftarrow R_{q_l}^{N\times n}$ and an error vector $\mathbf{e}_l \leftarrow \chi^N$, compute $\mathbf{b}_l \leftarrow \mathbf{A'}_l\mathbf{z}_l + 2\mathbf{e}_l \in R_{q_l}^N$, output the public key $pk := \mathbf{A}_l = [\mathbf{b}_l\,|\mathbf{A'}_l] \in R_{q_l}^{N\times(n+1)}$.
  (c) Let $\mathbf{s'}_l = \mathbf{s}_l \otimes \mathbf{s}_l \in R_{q_l}^{(n+1)^2}$, the evaluation keys is defined as
  $evk = \tau_{\mathbf{s'}_l \to \mathbf{s}_{l-1}} \leftarrow \text{SwitchKeyGen}(\mathbf{s'}_l, \mathbf{s}_{l-1})$(omit this step when $l = 0$)
- BGV.Enc(pk, μ): On input a message $m \in R_p$, set $\mathbf{m} \leftarrow (m, 0, ..., 0) \in R_p^{n+1}$, choose a random vector $\mathbf{r}_l \in R_2^N$, output the ciphertext

$$\mathbf{c}_l = \mathbf{m} + \mathbf{A}_l{}^T\mathbf{r}_l \in R_{q_l}^{n+1}$$

- BGV. Dec$(\mathbf{s}_l, \mathbf{c}_l)$: On input the ciphertext $\mathbf{c}_l \in R_{q_l}^{n+1}$ and its corresponding secret key $\mathbf{s}_l \in R_{q_l}^{n+1}$, output the message:

$$\mu \leftarrow [[< \mathbf{c}_l, \mathbf{s}_l >]_{q_l}]_p$$

- BGV. Hom Add$(evk, \mathbf{c}_1, \mathbf{c}_2)$: On input two (extended) ciphertext $\mathbf{c}_1, \mathbf{c}_2 \in R_{q_l}^{n+1}$ at the same level-l under the same secret key $\mathbf{s}_l \in R_{q_l}^{n+1}$ (If needed, use SwitchKey and ModulusSwitch to make it so).
  (a) Compute $\bar{\mathbf{c}}_3 = \mathbf{c}_1 + \mathbf{c}_2 \in R_{q_l}^{n+1}$ under the secret key $\mathbf{s'}_l = \mathbf{s}_l \otimes \mathbf{s}_l \in R_{q_l}^{(n+1)^2}$

(b) Pad zeros to $\bar{\mathbf{c}}_3$ and get $\mathbf{c}'_3 \in R_{q_l}^{(n+1)^2}$, and compute $\mathbf{c}''_3 = \mathrm{SwitchKey}(\mathbf{c}'_3, \tau_{\mathbf{s}'_l \to \mathbf{s}_{l-1}}) \in R_{q_l}^{n+1}$ under the secret key $\mathbf{s}_{l-1} \in R_{q_l}^{n+1}$.
(c) Compute $\mathbf{c}_3 = \mathrm{Modulus\,Switch}(\mathbf{c}'_3, q_{l-1}) \in R_{q_{l-1}}^{n+1}$.

– BGV. Hom Mult$(evk, \mathbf{c}_1, \mathbf{c}_2)$: On input two (extended) ciphertext $\mathbf{c}_1, \mathbf{c}_2 \in R_{q_l}^{n+1}$ at the same level-l under the same secret key $\mathbf{s}_l \in R_{q_l}^{n+1}$ (If needed, use SwitchKey and ModulusSwitch to make it so).
(a) Compute $\bar{\mathbf{c}}_3 = \mathbf{c}_1 \otimes \mathbf{c}_2 \in R_{q_l}^{(n+1)^2}$ under the secret key $\mathbf{s}'_l = \mathbf{s}_l \otimes \mathbf{s}_l \in R_{q_l}^{(n+1)^2}$;
(b) Compute $\mathbf{c}'_3 = \mathrm{SwitchKey}(\bar{\mathbf{c}}_3, \tau_{\mathbf{s}'_l \to \mathbf{s}_{l-1}}) \in R_{q_l}^{n+1}$ under the secret key $\mathbf{s}_{l-1} \in R_{q_l}^{n+1}$.
(c) Compute $\mathbf{c}_3 = \mathrm{Modulus\,Switch}(\mathbf{c}'_3, q_{l-1}) \in R_{q_{l-1}}^{n+1}$.

2 Two Subroutines

Here introduce two subroutines BitDecomp$(\cdot)$ and Powersof 2$(\cdot)$ which are widely used in FHE schemes. Let $\beta = \lfloor \log q \rfloor + 1$. We describe these two subroutines as follows.

BitDecomp$(\mathbf{x} \in R_q^n, q)$: Decomposes $\mathbf{x} \in R_q^n$ into its bit representation. Namely write $\mathbf{x} = \sum_{j=0}^{\beta-1} 2^j \mathbf{u}_j$ with all $\mathbf{u}_j \in \{0,1\}^{n \times d}$, and output $[\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{\beta-1}] \in \{0,1\}^{n \cdot \beta}$. Powersof 2$(\mathbf{x} \in R_q^n, q)$: Let $\mathbf{v}_j = 2^j \mathbf{x} \mod q \in R_q^n$, $j \in \{0, 1, ..., \beta - 1\}$, and output $[\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_{\beta-1}] \in R_q^{n \cdot \beta}$. It's obviously to verify that for $\mathbf{U}, \mathbf{V} \in R_q^{n \times d}$, we have

$$\langle \mathrm{BitDecomp}(\mathbf{x}, q), \mathrm{Powersof\ 2}(\mathbf{y}, q) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \bmod q$$

References

[AP13] ALPERIN-SHERIFF J, PEIKERT C. Practical bootstrapping in quasilinear time. In: Advances in Cryptology—CRYPTO 2013. Springer Berlin Heidelberg, 2013: 1–20.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Advances in Cryptology-CRYPTO 2014, pages 297-314. Springer, 2014.

[Bra12] BRAKERSKI Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In: Advances in Cryptology—CRYPTO 2012. Springer Berlin Heidelberg, 2012: 868–886.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pages 309-325.ACM, 2012.

[BP16] BRAKERSKI Z, PERLMAN R. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Advances in Cryptology—CRYPTO 2016. Springer Berlin Heidelberg, 2016: 190–213.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Foundations of Computer Science Annual Symposium on, 2011(2):97-106, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Advances in Cryptology-CRYPTO 2011, pages 505-524. Springer, 2011.

[CGG16] Chillotti I, Gama N, Georgieva M, et al. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. International Conference on the Theory and Application of Cryptology and Information Security—ASIACRYPT 2016. Springer, Berlin, Heidelberg, 2016:3-33.

[CGGI17] Chillotti I, Gama N, Georgieva M, et al. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. International Conference on the Theory and Application of Cryptology and Information Security—ASIACRYPT 2016. Springer, Cham, 2017:377-408.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Advances in Cryptology - CRYPTO 2015, Proceedings, Part II, pages 630-656, 2015.

[CZW17] Chen L, Zhang Z, Wang X. Batched Multi-hop Multi-key FHE from Ring-LWE with Compact Ciphertext Extension, Theory of Cryptography Conference. Springer, Cham, 2017:597-627.

[DGHV10] Dijk M V, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers, International Conference on Theory and Applications of Cryptographic Techniques. Springer-Verlag, 2010:24-43.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In STOC, volume 9, pages 169-178, 2009.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attributebased. In Advances in Cryptology-CRYPTO 2013, pages 75-92. Springer, 2013.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ringbased public key cryptosystem. In International Symposium on Algorithmic Number Theory, pages 267-288, 1998.

[LATV12] Adriana L´opez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pages 1219-1234. ACM, 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Advances in Cryptology - EUROCRYPT 2016 , Proceedings, Part II, pages 735-763, 2016.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from lwe, revisited. In Theory of Cryptography - 14th International Conference, TCC 2016-B, Proceedings, Part II, pages 217-238, 2016.