Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

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

Tanping Zhou[1], Ningbo Li[1], XiaoYuan YANG[1], YiLiang HAN[1], WenChao LIU[1]

[1]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

Corresponding author: NingBo LI (e-mail:372726936@qq.com),TanPing ZHOU (e-mail:850301775@qq.com).

This work was supported by National Key R&D Program of China under Grants No. 2017YFB0802000, National Natural Science Foundation of China (Grant Nos. U1636114,61772550,61572521,61872384), State Key Laboratory of Information Security (2017-MS-18).

ABSTRACT Multi-Key Full Homomorphic Encryption (MKFHE) can perform arbitrary operations on encrypted data under different public keys (users), and the final ciphertext can be jointly decrypted by all involved users. Therefore, MKFHE has natural advantages and application value in security multi-party computation (MPC). The MKFHE scheme based on Brakerski-Gentry-Vaikuntanathan (BGV) inherits the advantages of BGV FHE scheme in aspects of encrypting a ring element, the ciphertext/plaintext ratio, and supporting the Chinese Remainder Theorem (CRT)-based ciphertexts packing technique. However some weaknesses also exist such as large ciphertexts and keys, and complicated process of generating evaluation keys. In this paper, we present an efficient BGV-type MKFHE scheme. Firstly, we construct a nested ciphertext extension for BGV and separable ciphertext extension for Gentry-Sahai-Waters (GSW), which can reduce the size of the extended ciphertexts about a half. Secondly, we apply the hybrid homomorphic multiplication between RBGV ciphertext and RGSW ciphertext to the generation process of evaluation keys, which can significantly reduce the amount of input/output ciphertexts and improve the efficiency. Finally, we construct a directed decryption protocol which allows the evaluated ciphertext to be decrypted by any target user, thereby enhancing the ability of data owner to control their own plaintext, and abolish the limitation in current MKFHE schemes that the evaluated ciphertext can only be decrypted by users involved in homomorphic evaluation.

INDEX TERMS Multi-key Full Homomorphic Encryption, ciphertext extension, evaluation key, hybrid homomorphic multiplication, directed decryption

I. 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 [1], secure multi-party computing (MPC) [2-4], cloud data analysis, etc.

Since the first ideal-based FHE scheme Gen09 was proposed in 2009 [5], many FHE schemes [6-21] was proposed following Gentry's blueprint. Multi-key FHE (MKFHE) [22-31] allows computations on ciphertexts under different secret keys, which is an extension of FHE in secure MPC. L ́opez-Alt et al. [22] first proposed a MKFHE scheme LTV12 based on the NTRU cryptosystem [32]. However, its security is based on a somewhat non-standard assumption on polynomial rings.

Clear and McGoldrick [23] proposed the first GSW-type MKFHE scheme CM15 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 [24] simplified CM15 and gave a construction of MKFHE scheme MW16 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 [25], 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 [26], which means that the bound of number of users does not need to be input during the setup procedure.

In TCC2017, Chen et al. [28] 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 process for the generation of evaluation keys. In this paper, we make the following improvements to these weaknesses:

(1) We construct a nested ciphertext extension for BGV and separable ciphertext extension for GSW, which can reduce the size of the extended ciphertexts 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 public parameters.

(3) We construct a directed decryption protocol in which the users involved in homomorphic evaluation can appoint the target user who can get the final decrypting result, thereby enhancing the ability of data owner to control their own plaintext.

These improvements can efficiently reduce the size of ciphertexts and public parameters during homomorphic evaluation, and further reduce the computational complexity of homomorphic operations.

II. PRELIMINARIES

Throughout this paper, we let λ denote the security parameter and $negl(\lambda)$ denote a negligible function of λ. We use bold lowercase symbol to denote vectors and bold uppercase symbol to denote matrixes. The *i*-th component of vector $\mathbf{a}$ is represented as $\mathbf{a}[i]$, and the element located in the *i*-th row and the *j*-th column of matrix $\mathbf{A}$ is represented as $\mathbf{A}[i,j]$. In general, vectors can be regarded as a row matrix.

Let $\Phi_m(X)$ denote the *m*-th cyclotomic polynomial with the degree $n=\phi(m)$, where $\phi(\cdot)$ is the Euler's function. We work over rings $R=\mathbb{Z}[X]/\Phi_m$ and $R_q=R/qR$ for a prime integer $q=q(\lambda)$. Addition and multiplication in these rings is done component-wise in their coefficients, and $[x]_q$ denotes that the coefficients of x are reduced in $[-q/2,q/2)$ (except for $q=2$). Let $\chi=\chi(\lambda)$ be a *B*-bound error distribution over R whose coefficients are in the range $[-B,B]$. For a probability distribution D, $x\leftarrow D$ denotes that x is sampled from D, and $x\xleftarrow{\$}D$ denotes that x is sampled uniformly from D.

For $a\in R$, we use $\|a\|_\infty=\max_{0\le i\le n-1}|a_i|$ to denote the standard l_∞-norm and use $\|a\|_1=\sum_{i=0}^{n-1}|a_j|$ to denote the standard l_1-norm.

A. THE GENERAL LEARNING WITH ERRORS (GLWE) PROBLEM

The learning with errors (LWE) problem and the ring learning with errors (RLWE) problem are syntactically identical, aside from different rings, and these two problems are summarized as GLWE problem in [BGV12].

Definition 1 (GLWE problem). Let λ be a security parameter. For the polynomial ring $R=\mathbb{Z}[X]/x^d+1$ and $R_q=R/qR$, and an error distribution $\chi=\chi(\lambda)$ over R, the GLWE problem is to distinguish the following two distributions: In the first distribution, one samples $(\mathbf{a}_i,b_i)\in R_q^{n+1}$ uniformly from R_q^{n+1}. For the second distribution, one first draws $\mathbf{a}_i\leftarrow R_q^n$ uniformly, and samples $(\mathbf{a}_i,b_i)\in R_q^{n+1}$ by choosing $\mathbf{s}\leftarrow R_q^n$ and $e_i\leftarrow\chi$ uniformly, and set $b_i=<\mathbf{a}_i,\mathbf{s}>+e_i$. 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 $n=1$.

B. LEVELED MULTI-KEY FHE

We now introduce the cryptographic definition of a leveled multi-key FHE, which is similar to the one defined in CZW17 with some modifications from LTV12.

Definition 2 (Multi-key FHE). Let $\mathcal{C}$ be a class of circuits. A leveled multi-key FHE scheme $\mathcal{E}=(\text{Setup},\text{KeyGen},\text{Enc},\text{Eval},\text{Dec})$ is described as follows:

- $\mathcal{E}.\text{Setup}(1^\lambda,1^K,1^L)$: Given the security parameter λ, the circuit depth L, and the number of distinct users K that can be tolerated in an evaluation, outputs the public parameters pp.
- $\mathcal{E}.\text{KeyGen}(pp)$: Given the public parameters pp, derives and outputs a public key pk_i, a secret key sk_i, and the evaluation keys evk_i of party i ($i=1,...,K$).
- $\mathcal{E}.\text{Enc}(pk_i,m)$: Given a public key pk_i and message μ, outputs a ciphertext ct_i.
- $\mathcal{E}.\text{Dec}((sk_{i_1},sk_{i_2},...,sk_{i_k}),ct_S)$: Given a ciphertext ct_S corresponding to a set of users $S=\{i_1,i_2,...,i_k\}\subseteq[K]$, and their secret keys $sk_S=\{sk_{i_1},sk_{i_2},...,sk_{i_k}\}$, outputs the message μ.
- $\mathcal{E}.\text{Eval}(\mathcal{C},(ct_{S_1},pk_{S_1},evk_{S_1}),...,(ct_{S_t},pk_{S_t},evk_{S_t}))$: On input a Boolean circuit $\mathcal{C}$ along with t tuples $(ct_{S_i},pk_{S_i},\ evk_{S_i})_{i=1,...,t}$, each tuple comprises of a ciphertext ct_{S_i} corresponding to a user set S_i, a set of public keys $pk_{S_i}=\{pk_j,\forall j\in S_i\}$, and the evaluation keys evk_{S_i}, outputs a ciphertext ct_S corresponding to a set of secret keys indexed by $S=\bigcup_{i=1}^t S_i\subseteq[K]$.

Definition 3 (Correctness of MKFHE). On input any circuit $\mathcal{C}$ of depth at most L and a set of tuples $\{(ct_{S_i},pk_{S_i})\}_{i\in\{1,...,t\}}$, let $\mu_i=\text{Dec}(sk_{S_i},ct_{S_i})$, where $sk_{S_i}=\{sk_j,\forall j\in S_i\}$, a leveled MKFHE scheme $\mathcal{E}$ is correct if it holds that

$$\Pr[\text{Dec}(sk_S,Eval(\mathcal{C},(ct_{S_i},pk_{S_i},evk_{S_i})_{i\in[t]}))\ne\mathcal{C}(\mu_1,...,\mu_t)]$$
$$=negl(\lambda)$$

Definition 4 (Compactness of MKFHE). A leveled MKFHE scheme is compact if there exists a polynomial $poly(\cdot,\cdot,\cdot)$ such that $|ct| \le poly(\lambda, K, L)$, which means that the length of ct is independent of the circuit $\mathcal{C}$, but depend on the security parameter λ, the number of users K and the circuit depth L.

C. TWO SUBROUTINES

Here we introduce two subroutines ($\mathrm{BitDecomp}(\cdot)$ and $\mathrm{Powersof}\ 2(\cdot)$) which are widely used in FHE schemes. Let $\mathbf{x} \in R_q^n$ be a polynomial of dimension n over R_q, and let $\beta = \lfloor \log q \rfloor + 1$.

$\mathrm{BitDecomp}(\mathbf{x} \in R_q^n, q)$: On input $\mathbf{x} = (x_1, ..., x_n) \in R_q^n$ and the modulus q, outputs $(x_{1,0}, ..., x_{1,\beta-1}, ..., x_{n,0}, ..., x_{n,\beta-1}) \in \{0,1\}^{n \cdot \beta}$ where $x_{i,j}$ is the j-th bit in x_i 's binary representation (ordered from least significant to most significant), namely $\mathbf{x} = (\sum_{j=0}^{\beta-1} 2^j x_{1,j}, ..., \sum_{j=0}^{\beta-1} 2^j x_{n,j})$.

$\mathrm{Powersof}\ 2(\mathbf{y} \in R_q^n, q)$: On input $\mathbf{y} = (y_1, ..., y_n) \in R_q^n$ and the modulus q, outputs $(y_1, 2y_1, ..., 2^{\beta-1} y_1, ..., y_n, 2y_n, ..., 2^{\beta-1} y_n) \in R_q^{n \cdot \beta}$.

It's straightforward to verify that for arbitrary $\mathbf{x}, \mathbf{y} \in R_q^n$, it holds that

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

D. TWO TECHNIQUES

In this section, we introduced two powerful techniques (key-switching and modulus-switching) from [BGV12], which are applied to control the dimension and noise of ciphertext during homomorphic evaluation.

Key-switching : The key switching technique can be used to reduce the dimension of an expanded ciphertext to a normal level, but more generally can be used to transform a ciphertext $\mathbf{c}_1 \in R_q^{n_1}$ (under the secret key $\mathbf{s}_1$) to another ciphertext $\mathbf{c}_2 \in R_q^{n_2}$ (under the secret key $\mathbf{s}_2$) with the corresponding message unchanged. For a FHE scheme $\mathcal{E}$, let $\beta = \lfloor \log q \rfloor + 1$, the key switching process mainly consists of two procedures:

- $\mathcal{E}.\mathrm{SwitchKeyGen}(\mathbf{s}_1 \in R_q^{n_1}, \mathbf{s}_2 \in R_q^{n_2})$: Compute $\overline{\mathbf{s}} = \mathrm{Powersof}\ 2(\mathbf{s}_1) \in R_q^{n_1 \cdot \beta}$, and output
$$\tau_{\mathbf{s}_1 \to \mathbf{s}_2} := \{\mathcal{K}_i = \mathrm{Enc}_{\mathbf{s}_2}(\overline{\mathbf{s}}[i]) \in R_q^{n_2}\}_{i=1,...,n_1\beta}$$
- $\mathcal{E}.\mathrm{SwitchKey}(\tau_{\mathbf{s}_1 \to \mathbf{s}_2}, \mathbf{c}_1, q)$: Compute $\overline{\mathbf{c}}_1 = \mathrm{BitDecomp}(\mathbf{c}_1) \in R_q^{n_1 \cdot \beta}$, and output
$$\mathbf{c}_2 = \sum_{i=1}^{n_1\beta} \mathcal{K}_i \cdot \overline{\mathbf{c}}_1[i] \in R_q^{n_2}$$

Lemma 1 (BGV12). Let $\mathbf{c}_1$ be a ciphertext under the key $\mathbf{s}_1$ for modulus q such that $e_1 \leftarrow [<\mathbf{c}_1, \mathbf{s}_1>]_q$ has length at most B and $m = [e_1]_2$ ($p = 2$). Let $c_2 \leftarrow \mathcal{E}.\mathrm{SwitchKey}(\tau_{\mathbf{s}_1 \to \mathbf{s}_2}, \mathbf{c}_1, q)$, and let $e_2 \leftarrow [<\mathbf{c}_2, \mathbf{s}_2>]_q$. Then e_2 (the new noise) has length at most $B + 2 \cdot \gamma_R \cdot B_\chi \cdot \lceil \log q \rceil \cdot \sqrt{n}$, and (assuming this noise length is less than $q/2$) we have $m = [e_2]_2$.

Modulus-switching : Since the noise involved in the ciphertext grows with homomorphic operations, modulus switching which can change the inner modulus q_{l+1} of ciphertext $\mathbf{c}_1$ to a smaller number q_l is used to reduce the noise term roughly by the ratio q_{l+1}/q_l, while preserving the correctness of decryption under the same secret key.

- $\mathcal{E}.\mathrm{ModulusSwitch}(\mathbf{c}_1, q_{l+1}, q_l)$: On input $\mathbf{c}_1 \in R_{q_{l+1}}^{n_1}$ and another smaller modulus q_l, output $\mathbf{c}_2 \in R_{q_l}^{n_1}$ which is the closest element to $(q_l / q_{l+1}) \cdot \mathbf{c}_1$.

Lemma 2 (BGV12). Let $\mathbf{c}_1$ be a ciphertext under the key $\mathbf{s}_1$ for modulus q_{l+1} such that $e_{l+1} \leftarrow [<\mathbf{c}_1, \mathbf{s}_1>]_{q_{l+1}}$ has length at most B and $m = [e_1]_2$. Let $c_2 \leftarrow \mathcal{E}.\mathrm{ModulusSwitch}(\mathbf{c}_1, q_{l+1}, q_l)$, and let $e_l \leftarrow [<\mathbf{c}_2, \mathbf{s}_1>]_{q_l}$. Then e_l (the new noise) has length at most $(q_l / q_{l+1}) \cdot B + \sqrt{n} \cdot \gamma_R \cdot B_\chi$, and (assuming this noise length is less than $q_l / 2$) we have $m = [e_l]_2$.

We just give a brief introduction of the two techniques above, and more details can be seen in BGV12.

III. EFFICIENT COMPONENTS IN OUR MKFHE

In this section, we present the details of some efficient techniques for homomorphic operations in our scheme, including: two optimized algorithms for ciphertext extension (nested ciphertext extension for BGV and separable ciphertext extension for GSW), generation of evaluation keys, and directed decryption process.

The aim of the system is to perform homomorphic operations on ciphertexts of different users in the cloud. In the initialization phase, the users upload the RBGV ciphertexts corresponding to their messages to the cloud, along with some materials used in the generation process of evaluation keys.

Step 1 (Ciphertext extension): The users respectively extend their RBGV ciphertexts to ones corresponding to the user set S before homomorphic evaluation.

Step 2 (Homomorphic evaluation): Do homomorphic computations on the user's **extended** ciphertexts and get a high-dimensional RBGV ciphertext.

Step 3 (Evaluation keys): Do ciphertext extension on materials in the initialization phase and **perform** hybrid homomorphic multiplication on RBGV and RGSW ciphertexts to obtain the evaluation keys.

Step 4 (Key switching): Perform key-switching operation on the high-dimensional RBGV **ciphertext** in step2 using the evaluation keys.

Step 5 (Modulus switching): Perform modulus-**switching** operation on the result ciphertext of step 4 and output the final ciphertext.

Note that the step 1, 2, 3 in our system can be performed simultaneously, and the flowchart of homomorphic operation in MKFHE scheme is shown in Figure 1.

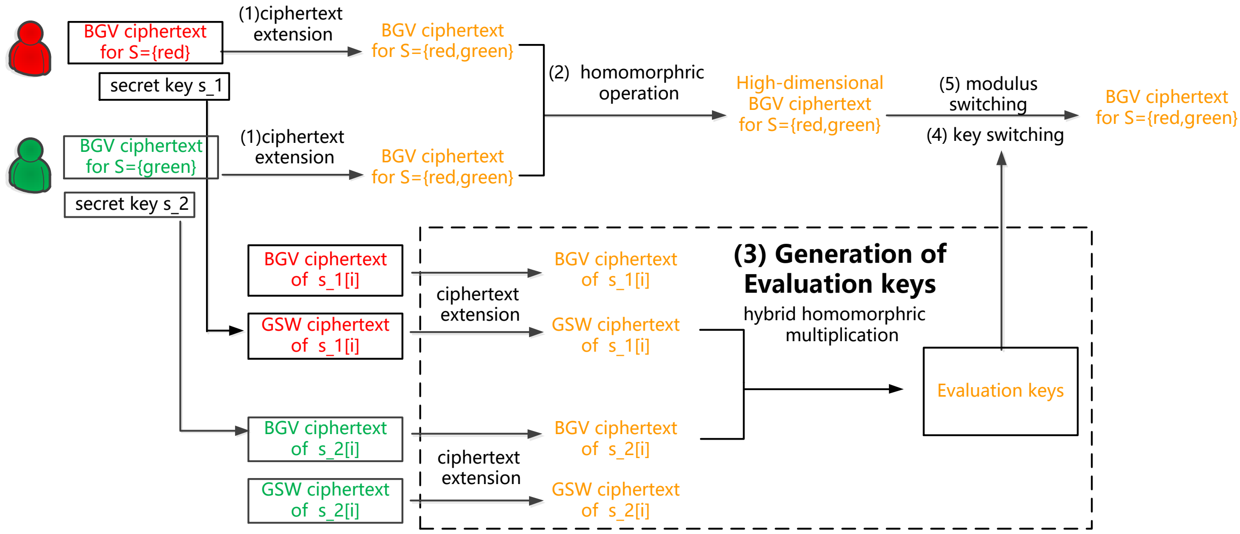

FIGURE 1. The process of homomorphic operation in MKFHE scheme

A. CIPHERTEXT EXTENSION

1) NESTED CIPHERTEXT EXTENSION FOR BGV

Here we present the basic ring-LWE based BGV scheme with some modifications to the original scheme in [BGV12].

- RBGV.Setup$(1^\lambda, 1^L)$: For the security parameter λ, 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_l, let $\beta_l = \lfloor \log q_l \rfloor + 1$. We work over rings $R = \mathbb{Z}[X]/\Phi_m$ and $R_{q_l} = R/q_l R$ defined above. Let $\chi = \chi(\lambda)$ be a B-bound error distribution over R whose coefficients are in the range $[-B, B]$.
- RBGV.KeyGen$(1^n, 1^L)$: Generate keys of circuit depth l for the j-th party ($l = 0, \dots, L$).
  1. Sample $z_{l,j} \leftarrow R_3$ and set secret key
  $$sk_{l,j} = \mathbf{s}_{l,j} := (1, -z_{l,j}) \in R_3^2$$
  2. Generate $a_{l,j} \xleftarrow{\$} R_q$ and $e_{l,j} \xleftarrow{\$} \chi$ randomly, and

compute the public key for the j-th user
$$pk_{l,j} = \mathbf{p}_{l,j} := (a_{l,j} z_{l,j} + pe_{l,j} \bmod q_l, a_{l,j}) = (b_{l,j}, a_{l,j}) \in R_{q_l}^2$$

- RBGV.Enc$(pk_{l,j}, \mu)$: On input a message $\mu \in R_p$ and the public key $pk_{L,j}$, sample random elements $r, e, e' \leftarrow \chi$, compute level-L ciphertext
$$\mathbf{c} = (c^{(0)}, c^{(1)}) = (rb_{L,j} + pe + \mu, ra_{L,j} + pe') \in R_{q_L}^2$$

Let S be an ordered set containing all indexes of users that the ciphertext corresponding to, and we assume that the indexes are arranged from small to large and S has no duplicate elements, thus we can describe a ciphertext as a tuple $ct = \{\mathbf{c}, S, l\}$. Here we set $S = \{j\}$, $l = L$, and output $ct = \{\mathbf{c}, \{j\}, L\}$.

- RBGV.Dec$(\mathbf{sk}_S, ct = (\mathbf{c}, S, l))$: On input a level-l ciphertext $ct = (\mathbf{c}, S, l)$ where $S = \{j_1, \dots, j_k\}$, and its corresponding secret keys $\{\mathbf{s}_{j_1,l}, \dots, \mathbf{s}_{j_k,l}\} \in R_3^{2k}$. Let $\overline{\mathbf{s}}_{S,l} = (1, -z_{j_1,l}, \dots, -z_{j_k,l}) \in R_3^{k+1}$, output the message
$$\mu \leftarrow \langle \mathbf{c}, \overline{\mathbf{s}}_{S,l} \rangle \bmod q_l \bmod p$$
- RBGV.CTExt$(\mathbf{c}_l, S')$: On input a ciphertext tuple $ct = \{\mathbf{c} \in R_{q_l}^{k+1}, S = \{i_1, \dots, i_k\}, l\}$ corresponding to k parties and another user set $S' = \{j_1, \dots, j_{k'}\}$ for $S \in S'$, output an extended tuple $ct' = \{\overline{\mathbf{c}} \in R_{q_l}^{k'+1}, S' = \{j_1, \dots, j_{k'}\}, l\}$. The extending algorithm is as follows:

(a) Divide the ciphertext $\mathbf{c}$ into $k+1$ sequential sub-vectors indexed by $S = \{i_1, \dots, i_k\}$ (except for the first sub-vector), i.e.,
$$\mathbf{c} = (c_S^{(0)} \mid c_{i_1}^{(1)} \mid \cdots \mid c_{i_k}^{(1)}) \in R_{q_l}^{k+1}$$
where the corresponding secret key is $\mathbf{s}_{S,l} = (1, -z_{i_1,l}, \dots, -z_{i_k,l})$.

(b) The extended ciphertext $\overline{\mathbf{c}}$ consists of $k'+1$ sequential sub-vectors, which can be indexed by $S' = \{j_1, \dots, j_{k'}\}$, i.e.,
$$\overline{\mathbf{c}} = (c_{S'}^{\prime(0)} \mid c_{j_1}^{\prime(1)} \mid \cdots \mid c_{j_{k'}}^{\prime(1)}) \in R_{q_l}^{k'+1}$$

Set $c_{S'}^{\prime(0)} = c_S^{(0)}$. If index j in S' is also included in S, we set $c_j^{\prime(1)} = c_j^{(1)}$,otherwise we set $c_j^{\prime(1)} = 0$. The corresponding secret key for decryption is $\overline{\mathbf{s}}_{S',l} = (1, -z_{l,j_1}, \dots, -z_{l,j_{k'}}) \in R_3^{k'+1}$.

It's easy to verify that $\langle \mathbf{c}, \mathbf{s}_{S,l} \rangle = \langle \overline{\mathbf{c}}, \overline{\mathbf{s}}_{S',l} \rangle \bmod q_l$.

2) NESTED CIPHERTEXT EXTENSION FOR GSW

In this section, we describe a variant of Ring-LWE based GSW scheme.

- RGSW.Setup(1^λ): For the security parameter λ, 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_l, let $\beta_l = \lfloor \log q_l \rfloor + 1$. We work over rings $R = \mathbb{Z}[X]/\Phi_m$ and $R_{q_l} = R/q_l R$ defined above. Let $\chi = \chi(\lambda)$ be a B-bound error distribution over R whose coefficients are in the range $[-B, B]$.
- RGSW.KeyGen(1^n) : Sample $z \leftarrow R_3$, choose a random vector $\mathbf{a} \in R_{q_l}^{2\beta_l}$ and $\mathbf{e} \leftarrow \chi^{2\beta_l}$ uniformly, output the secret key $\mathbf{s} = (1, -z)^T \in R_3^2$ and public key $\mathbf{P} = [\mathbf{a}z + p\mathbf{e}, \mathbf{a}] = [\mathbf{b}, \mathbf{a}] \in R_{q_l}^{2\beta_l \times 2}$.
- RGSW.EncRand$(r, \mathbf{P})$: This procedure is to generate the encryption of randomness which is used in the ciphertext extension. On input $r \leftarrow R_{q_l}$, sample

$r_i \leftarrow \chi \quad (i=1,...,\beta_l)$ and two vectors $\mathbf{e}_1', \mathbf{e}_2' \leftarrow \chi^{\beta_l}$ randomly, output the encryption of the randomness:

$$\mathrm{RGSW.EncRand}_{\mathbf{s}}(r) = \mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2] \in R_{q_l}^{\beta_l \times 2}$$

where $\mathbf{f}_1[i] = \mathbf{b}[i]r_i + p\mathbf{e}_1'[i] + \mathrm{Powersof}\ 2(r)[i] \in R_{q_l}$

$\mathbf{f}_2[i] = \mathbf{a}[i]r_i + p\mathbf{e}_2'[i] \in R_{q_l}$.

Notice that $\mathbf{Fs} = [p\tilde{\mathbf{e}} + \mathrm{Powersof}\ 2(r)] \in R_{q_l}^{\beta_l}$ for some small $\tilde{\mathbf{e}} = \mathbf{e}[i]r_i + \mathbf{e}_1'[i] - \mathbf{e}_2'[i]z$.

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

$$\begin{aligned}\mathrm{RGSW.Enc}(\mu, \mathbf{P}) = \mathbf{C} &= r\mathbf{P} + p\mathbf{E} + \mu\mathbf{G} \\ &= r[\mathbf{b}, \mathbf{a}] + p\mathbf{E} + \mu\mathbf{G} \\ &= r[\mathbf{a}z + p\mathbf{e}, \mathbf{a}] + p\mathbf{E} + \mu\mathbf{G} \\ &= [r\mathbf{a}z + p(r\mathbf{e} + \mathbf{e}_1), r\mathbf{a} + p\mathbf{e}_2] + \mu\mathbf{G}\end{aligned}$$

where $\mathbf{G} = (\mathbf{I}_2, 2\mathbf{I}_2, ..., 2^{\beta_l - 1}\mathbf{I}_2)^T \in R_q^{2\beta_l \times 2}$ and $\mathbf{I}_2$ is a 2×2 identity matrix. Notice that $\mathbf{C} \cdot \mathbf{s} = p\tilde{\mathbf{e}} + \mu\mathbf{G} \cdot \mathbf{s} \in R_q^{2\beta_l}$.

- $\mathrm{RGSW.CTExt}(\mathbf{C}_i, \mathbf{F}_i, \{\mathbf{P}_j, j=1,...,k\})$: On input the i-th user's ciphertext $\mathbf{C}_i = [\mathbf{C}_{i,0}, \mathbf{C}_{i,1}] \in R_q^{2\beta_l \times 2}$, an encryption $\mathbf{F}_i$ of randomness $r_i \in R_{q_l}$, and the public keys of all involved users $\mathbf{P}_j = [\mathbf{b}_j, \mathbf{a}_j]$, $j = 1,...,i-1,i+1,...k$. Output the extended ciphertext:

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

where $\mathbf{X}_j = [\mathbf{X}_{j,0}, \mathbf{X}_{j,1}] = [\mathrm{BitDecomp}(\tilde{\mathbf{b}}_j[u])\mathbf{F}_i] \in R_q^{2\beta_l \times 2}$, $\tilde{\mathbf{b}}_j[u] = \mathbf{b}_j[u] - \mathbf{b}_i[u]$, $u = 1,...,2\beta_l$, and the corresponding secret key $\bar{\mathbf{s}} = (1, -z_1, \ldots -z_k) \in R_3^{k+1}$.

Correctness of Ciphertext Extension: In order to ensure the correctness of the extending algorithm of GSW ciphertext, it is necessary to verify that the j-th row in $\bar{\mathbf{C}}_i$ satisfies:

$$(\mathbf{X}_{j,0} + \mathbf{C}_{i,0}) - \mathbf{C}_{i,1}z_j - \mathbf{X}_{j,1}z_i = \mathbf{C}_i\mathbf{s}_j + \mathbf{X}_j\mathbf{s}_i = p\tilde{\mathbf{e}}' + \mu_i\mathbf{G}\mathbf{s}_j \in R_q^{2\beta_l}$$

where $\tilde{\mathbf{e}}' \in R^{2\beta_l}$ is a small noise vector. The analysis process is as follows:

$$\begin{aligned}\mathbf{C}_i\mathbf{s}_j &= r_i[\mathbf{a}z_i + p\mathbf{e}_i - \mathbf{a}z_j] + p\mathbf{E}\mathbf{s}_j + \mu_i\mathbf{G}\mathbf{s}_j \\ &= r_i[\mathbf{a}z_i + p\mathbf{e}_i - \mathbf{a}z_j - p\mathbf{e}_j] + p\mathbf{E}\mathbf{s}_j + \mu_i\mathbf{G}\mathbf{s}_j - r_i p\mathbf{e}_j \\ &= -r_i\tilde{\mathbf{b}}_j + \mu_i\mathbf{G}\mathbf{s}_j + p\mathbf{E}\mathbf{s}_j - r_i p\mathbf{e}_j \\ &= (p\mathbf{E}\mathbf{s}_j - r_i p\mathbf{e}_j) + \mu_i\mathbf{G}\mathbf{s}_j - r_i\tilde{\mathbf{b}}_j\end{aligned}$$

$$\begin{aligned}\mathbf{X}_j\mathbf{s}_i &= \mathrm{BitDecomp}(\tilde{\mathbf{b}}_j)\mathbf{F}_i \cdot \mathbf{s}_i \\ &= \mathrm{BitDecomp}(\tilde{\mathbf{b}}_j)[p\tilde{\mathbf{e}} + \mathrm{Powersof}\ 2(r_i)] \\ &= \mathrm{BitDecomp}(\tilde{\mathbf{b}}_j)p\tilde{\mathbf{e}} + r_i\tilde{\mathbf{b}}_j\end{aligned}$$

Then we have:

$$\begin{aligned}\mathbf{C}_i\mathbf{s}_j + \mathbf{X}_j\mathbf{s}_i &= \mu_i\mathbf{G}\mathbf{s}_j + \mathrm{BitDecomp}(\tilde{\mathbf{b}}_j)p\tilde{\mathbf{e}} + (p\mathbf{E}\mathbf{s}_j - r_i p\mathbf{e}_j) \\ &= \mu_i\mathbf{G}\mathbf{s}_j + p\tilde{\mathbf{e}}' \in R_q^{2\beta}\end{aligned}$$

Finally we can get

$$\mathbf{C}_i\mathbf{s}_j + \mathbf{X}_j\mathbf{s}_i = \bar{\mathbf{C}}_j\bar{\mathbf{s}} = p\tilde{\mathbf{e}} + \mu_i\bar{\mathbf{G}}\bar{\mathbf{s}}$$

where $\bar{\mathbf{G}} = (\mathbf{I}_{2k}, 2\mathbf{I}_{2k}, ..., 2^{\beta-1}\mathbf{I}_{2k})^T \in R_q^{2k\beta_l \times 2k}$.

B. GENERATION OF EVALUATION KEY

In this paper, we optimize the generation of evaluation keys during the key-switching process in [CZW17]. We apply the hybrid homomorphic multiplication in [19] between RBGV ciphertexts and RGSW ciphertexts instead of homomorphic multiplication between two RBGV ciphertexts, thus decrease the noise involved in the evaluation keys. What's more, we limit the coefficient of user's secret key to $\{-1, 0, 1\}$ so that $\mathrm{BitDecomp}(\cdot)$ and $\mathrm{Powersof}\ 2(\cdot)$ techniques are no longer required in key-switching, thus reduce the number of ciphertexts during key-switching process. For convenience, we use $\mathrm{RGSW.Enc}_{\mathbf{s}}(\mu)$ (or $\mathrm{RBGV.Enc}_{\mathbf{s}}(\mu)$) to denote a GSW/BGV ciphertext that can be decrypted to μ with the secret key $\mathbf{s}$.

- $\mathrm{MKFHE.EVKGen}(em_S, pk_S)$: Given a level-$l$ extended secret key $\hat{\mathbf{s}}_l = \bar{\mathbf{s}}_l \otimes \bar{\mathbf{s}}_l \in R_3^{(k+1)^2}$ for $\bar{\mathbf{s}}_l = (1, -z_{l,j_1}, ..., -z_{l,j_k})$, and corresponding public keys $[\mathbf{b}_{l-1,j}, \mathbf{a}_{l-1,j}]_{j \in \{j_1,...,j_k\}}$ for the user set $S = \{j_1, ..., j_k\}$. For $j \in \{1,...,k\}$, $m \in \{0,...,\beta_l - 1\}$, compute

$$\Psi_{l,j} \triangleq \mathrm{RGSW.Enc}_{\mathbf{s}_{l-1,j}}(z_{l,j})$$

$$\mathbf{F}_{l,j} \triangleq \mathrm{RGSW.EncRand}(r_{l,j}, pk_{l-1,j})$$

$$\Phi_{l,j,m} \triangleq \mathrm{RBGV.Enc}_{\mathbf{s}_{l-1,j}}(2^m \cdot z_{l,j})$$

Output the evaluation keys $evk = \{\mathcal{K}_{m,\xi} \in R_{q_l}^2\}$, $\xi \in \{1, ..., (k+1)^2\}$, and the process is shown in Algorithm1.

Algorithm1: the generation of $evk = \{\mathcal{K}_{m,\xi}\}$
Input: $\Psi_{l,j}$, $\mathbf{F}_{l,j}$, $\Phi_{l,j,m}$
1. for $\zeta' \in \{0,...,k\}$ $\bar{\Psi}_l[\zeta'] \triangleq \begin{cases} \mathrm{RGSW.Enc}_{\bar{\mathbf{s}}_{l-1}}(1) & \zeta' = 0 \\ \mathrm{RGSW.CTExt}_{\bar{\mathbf{s}}_{l-1}}(\Psi_{l,\zeta'}, \mathbf{F}_{l,j}, \mathbf{P}_{l,j}) & \text{else} \end{cases}$ 2. for $\zeta \in \{0,...,k\}$ for $m \in \{0,...,\beta_l - 1\}$ $\bar{\Phi}_{l,m}[\zeta] \triangleq \begin{cases} \mathrm{RBGV.Enc}_{\bar{\mathbf{s}}_{l-1}}(2^m) & \zeta = 0 \\ \mathrm{RBGV.CTExt}_{\bar{\mathbf{s}}_{l-1}}(\Phi_{l,\zeta,m}, S) & \text{else} \end{cases}$ 3. for $\zeta' = [0,...,k]$ for $\zeta = [0,...,k]$ for $m = [0,...,\beta_l - 1]$ $\mathcal{K}_{m,(k+1)\zeta'+\zeta} = \bar{\Psi}_{l,j}[\zeta'] \boxdot \bar{\Phi}_{l,j',m}[\zeta]$ 4. output $evk = \{\mathcal{K}_{m,\xi}\}_{m \in \{0,...,\beta_l - 1\}; \xi \in \{1,...,(k+1)^2\}}$

where "$\boxdot$" denotes the hybrid homomorphic multiplication between RBGV ciphertexts and RGSW ciphertexts.

Definition 5 (Hybrid homomorphic multiplication). We define the product $\boxdot$ as

$$\boxdot: \text{RGSW}\times\text{RBGV}\to\text{RBGV}$$
$$(\mathbf{C}_2,\mathbf{c}_1)\to\mathbf{C}_2\boxdot\mathbf{c}_1=\text{BD}(\mathbf{c}_1)\bullet\mathbf{C}_2$$

The definition of hybrid homomorphic multiplication based on RLWE is a variant of the external product based on TLWE in [19], and it can be used in evaluating process to reduce the amount of ciphertexts and noise, thereby improving the efficiency of homomorphic evaluation.

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 $\mu_2\bullet\mu_1$ and $\|Err(\mathbf{C}_2\boxdot\mathbf{c}_1)\|_\infty\le(2\beta)n\bullet2\sigma\|Err(\mathbf{C}_2)\|_\infty+\|\mu_2\|_\infty\|Err(\mathbf{c}_1)\|_\infty$, $Var(Err(\mathbf{C}_2\boxdot\mathbf{c}_1))\le2p\beta(2n+1)Var(\mathbf{e})+pnVar(\mathbf{e}_1)$, where n is the degree of the cyclotomic polynomial, p is an integer, β is the bound of the noise coefficients, σ is the standard deviation of the error distribution χ , $p\mathbf{e}_1$ is the noise of $\mathbf{c}_1$, $\mathbf{e}\leftarrow\chi$ is the noise involved in $\mathbf{C}_2$.

C. DIRECTED DECRYPTION PROTOCOL

MKFHE can be applied to realize secure computation among multi-parties, and the evaluated ciphertext can be jointly decrypted by all involved users. However, sometimes we do not prefer the final decrypting result to be known by all involved users, and only want the designated and recognized legitimate user(s) to get the decrypting result, even the user(s) does not participate in the computing process. For this scenario in Figure 2, a directed decryption protocol is essential to enhance the ability of data owner to control their own plaintext.

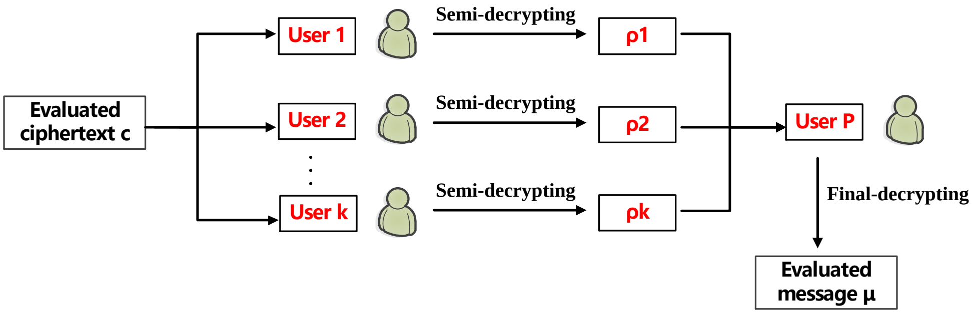

FIGURE 2. The process of directed decryption in our MKFHE scheme

In this paper, we construct a directed decryption protocol in which the users involved in homomorphic evaluation can appoint the target user who can get the final decrypting result, thereby enhancing the ability of data owner to control their own plaintext. The directed decryption protocol is realized by adding the encryption of 0 (under the public key of target user) to the intermediate decryption result of the users involved in homomorphic evaluation, and the process is as follows:

Assume that the level-l ciphertext needs to be finally decrypted is denoted by $\mathbf{c}=(b_l,a_{l,j_1},a_{l,j_2},...,a_{l,j_k})\in R_{q_l}^{k+1}$, the corresponding user set $S=(j_1,...,j_k)$ and the plaintext $\mu=\mathcal{C}(\mu_1,...,\mu_k)$ where $\mathcal{C}$ is the Boolean circuit. Assume that the target user who needs to get the ultimate decrypted result is i, and he can get the ciphertext $\mathbf{c}$, the process of our directed decryption protocol is implemented as follows:

1. **Semi-decrypting**: The users in S respectively do decrypting operations on ciphertext $\mathbf{c}$ with their special extended keys. For user j_1 with secret key $\mathbf{s}_{l,j_1}=(1,-z_{l,j_1})$, get the semi-decrypting result $\mathbf{c}'_{j_1}=(\mathbf{c}_{j_1},0)$ by the extended key $\bar{\mathbf{s}}'_{l,j_1}=(1,-z_{l,j_1},0,...,0)$:

$$\mathbf{c}'_{j_1}=(\mathbf{c}_{j_1},0)=(<\mathbf{c},\bar{\mathbf{s}}'_{l,j_1}>,0)=(b_l-a_{l,j_1}\cdot z_{l,j_1},0)$$

Other users in S do similar operations as user j_1.

2. **Adding target user's encryption of 0**: The users in S respectively compute the encryption of 0 using the public key of user i :

$$\mathbf{c}_i=\text{RBGV}.\text{Enc}(pk_{l,i},0)=(b_{l,i_1},a_{l,i_1})\in R_{q_l}^2$$

For user j_1, compute the sum of $\mathbf{c}_i$ and the semi-decrypting result $\mathbf{c}'_{j_1}$:

$$\mathbf{c}''_{j_1}=(b_l-a_{l,j_1}\cdot z_{l,j_1}+b_{l,i_1},a_{l,i_1})$$

Following the same method, other users can get $\{\mathbf{c}''_{j_1},\mathbf{c}''_{j_2},...,\mathbf{c}''_{j_k}\}$ and send them to user i.

3. **Final decryption**: When user i receives $\{\mathbf{c}''_{j_1},\mathbf{c}''_{j_2},...,\mathbf{c}''_{j_k}\}$, he compute $\mathbf{c}_{sum}=\mathbf{c}''_{j_1}+\mathbf{c}''_{j_2}+...+\mathbf{c}''_{j_k}$, and compute the final decrypting result as

$$\mu=(\mathbf{c}_{sum}-(k-1)b_l)\cdot\mathbf{s}_{l,i}=(\mathbf{c}''_{j_1}+...+\mathbf{c}''_{j_k}-(k-1)b_l)\cdot(1,-z_{l,i})$$
$$=kb_l-\sum_{m=1}^{k}a_{l,j_m}\cdot z_{l,j_m}-(k-1)b_l+\sum_{m=1}^{k}(b_{l,i_m}-a_{l,i_m}\cdot z_{l,i})$$
$$=b_l-\sum_{m=1}^{k}a_{l,j_m}\cdot z_{l,j_m}+\sum_{m=1}^{k}(b_{l,i_m}-a_{l,i_m}\cdot z_{l,i})$$
$$=\mathcal{C}(\mu_1,...,\mu_k)\ +e_{j_1}+...+e_{j_k}+e_{i_1}...+e_{i_k}$$
$$=\mathcal{C}(\mu_1,...,\mu_k)\ \bmod q_l \bmod p$$

Lemma 3: Let B denotes the bound of noise in a fresh RBGV ciphertext, and B_l denotes the bound of noise in a level-*l* RBGV ciphertext, then the directed decryption process is correct if

$$|e_{j_1}+...+e_{j_k}+e_{i_1}...+e_{i_k}|\le|e_{j_1}+...+e_{j_k}|+kB\le kB_l+kB<q/4$$

Note that in current MKFHE schemes, the result of homomorphic evaluations can only be finally decrypted by users involved in the evaluation process, and the directed decryption protocol designed in this paper allow the result ciphertext to be decrypted by any legitimate user. Moreover, as no homomorphic multiplication is involved in our protocol, there is no need of some techniques to control the noise.

IV. NEW CONSTRUCTION OF BGV-TYPE MKFHE SCHEME

In this section, we present the details of our BGV-type MKFHE scheme. For convenience, in the following we use $\mathrm{RGSW.Enc}_{\mathbf{s}}(\mu)$ (presented in Section 3.1) to denote a GSW ciphertext (or $\mathrm{RBGV.Enc}_{\mathbf{s}}(\mu)$) that can be decrypted to μ with the secret key $\mathbf{s}$. Also we adopt the techniques of key-switching and modulus-switching introduced in section 2.4.

A. BASIC SCHEME

- MKFHE.Setup$(1^\lambda,1^K,1^L)$: For the security parameter λ, 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_l. We work over rings $R=\mathbb{Z}[X]/\Phi_m$ and $R_{q_l}=R/q_lR$ defined above. Let $\chi=\chi(\lambda)$ be a *B*-bound error distribution over R whose coefficients are in the range $[-B,B]$. Let $\beta_l=\lfloor\log q_l\rfloor+1$, $\beta_B=\lfloor\log B\rfloor+1$, and choose $L+1$ random public vectors $\mathbf{a}_l\in R_{q_l}^{2\beta_l}$ for $l\in\{0,\dots,L\}$. All the following algorithms implicitly take the public parameter $pp=(R,B,\chi,\{q_l,\mathbf{a}_l\}_{l\in\{0,\dots,L\}},p)$ as input.

  Let *S* be an ordered set containing all indexes of users that the ciphertext corresponding to, and we assume that the indexes are arranged from small to large and *S* has no duplicate elements, thus we can describe a ciphertext as a tuple $ct=\{\mathbf{c},S,l\}$.
- MKFHE.KeyGen(pp): Given the public parameters pp, generate keys of circuit depth l for the *j*-th party ($l=0,...,L$).

  1. Sample $z_{l,j}\leftarrow\chi$ and set secret key
  $$sk_{l,j}=\mathbf{s}_{l,j}:=(1,-z_{l,j})\in R_3^2.$$
  2. Choose $\mathbf{e}_{l,j}\xleftarrow{\$}\chi^{2\beta_l}$ randomly, and compute the public key for the *j*-th user
  $$pk_{l,j}=\mathbf{p}_{l,j}:=[\mathbf{a}_{l,j}z_{l,j}+p\mathbf{e}_{l,j},\mathbf{a}_{l,j}]=[\mathbf{b}_{l,j},\mathbf{a}_{l,j}]\in R_q^{2\beta_l\times 2}$$
  3. Compute the materials used in the generation of evaluation keys:
  $$em_j=\{(\Phi_{l,j,m}\in R_{q_l}^{2\beta_B}),(\Psi_{l,j}\in R_{q_l}^{2\beta_l\times 2},\mathbf{F}_{l,j}\in R_{q_l}^{\beta_l\times 2})\}_{l=\{L,\dots,0\}}$$

  where
  $$\Phi_{l,j,m}\triangleq\mathrm{RBGV.Enc}_{\mathbf{s}_{l-1,j}}(2^m\cdot z_{l,j})$$
  $$=\{r_{l,j,m}\mathbf{b}_{l-1,j}[1]+2e_{l,j,m}+2^m\cdot z_{l,j},r_{l,j,m}\mathbf{a}_{l,j}[1]+2e'_{l,j,m}\}\in R_{q_l}^2$$
  $$\Psi_{l,j}\triangleq\mathrm{RGSW.Enc}_{\mathbf{s}_{l-1}}(z_{l,j})$$
  $$=\{r'_{l,j}[\mathbf{b}_{l-1},\mathbf{a}_{l-1}]+p\mathbf{E}'_{l,j}+z_{l,j}\mathbf{G}\}\in R_{q_l}^{2\beta_l\times 2}$$
  $$\mathbf{F}_{l,j}\triangleq\mathrm{RGSW.EncRand}(r_{l,j},pk_{l-1,j})\in R_{q_l}^{\beta_l\times 2}$$
- MKFHE.Enc$(pk_{L,j},\mu_j)$: On input a message $\mu_j\in R_p$ and the public key $pk_{L,j}$, sample random elements r, e, $e'\leftarrow\chi$, compute level-*L* ciphertext
  $$\mathbf{c}=(c_{j,0},c_{j,1})=(r\mathbf{b}_{L,j}[1]+pe+\mu_j,r\mathbf{a}_L[1]+pe')\in R_{q_L}^2$$
  and output the tuple $ct=\{\mathbf{c},\{j\},L\}$.
- MKFHE.Dec$(\mathbf{sk}_S,ct=(\mathbf{c},S,l))$: On input a level-*l* ciphertext $ct=(\mathbf{c},S,l)$ where $S=\{j_1,...,j_k\}$, and its corresponding secret keys $\{\mathbf{s}_{j_1,l},...,\mathbf{s}_{j_k,l}\}\in R_3^{2k}$. Let $\overline{\mathbf{s}}_{S,l}=(1,-z_{j_1,l},...,-z_{j_k,l})\in R_3^{k+1}$, output the message
  $$\mu\leftarrow\langle\mathbf{c},\overline{\mathbf{s}}_{S,l}\rangle\bmod q_l\bmod p$$
- MKFHE.*Eval*$((pk_{l,j_1},\dots,pk_{l,j_k}),em_S,\mathcal{C},(ct_1,\dots ct_t))$: Assume that the sequence of ciphertexts $ct_i=\{\mathbf{c}_i,S_i,l\}_{i\in\{1,\dots,t\}}$ are at the same level-*l* (If needed, use key-switching and modulus-switching to make it so). Let $S=\bigcup_{i=1}^t S_i=(j_1,\dots,j_k)$. Then the outline of evaluation on Boolean circuit $\mathcal{C}$ is as follows.

  1. For $i\in\{1,\dots,t\}$, compute RBGV.CTExt$(\mathbf{c}_i,S)$ to get an extended ciphertext $\overline{\mathbf{c}}_i$ under extended secret key $\overline{\mathbf{s}}_l:=(1,-z_{l,j_1},\dots-z_{l,j_k})$.

  2.Generate the evaluation keys by compute $evk_S=\tau_{\hat{\mathbf{s}}_l\to\overline{\mathbf{s}}_{l-1}}$=MKFHE.EVKGen$(em_S,pk_{l,S})$

  3. Evaluate the circuit $\mathcal{C}$ by using the two basic homomorphic operations MKFHE.EvalAdd$(evk_S,\overline{\mathbf{c}}_{i_1},\overline{\mathbf{c}}_{i_2})$ and MKFHE.EvalMult$(evk_S,\overline{\mathbf{c}}_{i_1},\overline{\mathbf{c}}_{i_2})$.

B. HOMOMORPHIC OPERATIONS

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 $\overline{\mathbf{c}}_1$, $\overline{\mathbf{c}}_2\in R_{q_l}^{k+1}$ corresponding to the user set $S=\{j_1,...,j_k\}$. The evaluation key is defined as :

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

where $\hat{\mathbf{s}}_l=\overline{\mathbf{s}}_l\otimes\overline{\mathbf{s}}_l$, $\overline{\mathbf{s}}_l=(1,-z_{l,j_1},...,-z_{l,j_k})\in R_3^{k+1}$, $\overline{\mathbf{s}}_{l-1}=(1,-z_{l-1,j_1},...,-z_{l-1,j_k})\in R_3^{k+1}$, and it holds that

$$<\mathcal{K}_{m,\xi},\overline{\mathbf{s}}_{l-1}>=pe_{m,\xi}+2^{m-1}\hat{\mathbf{s}}_l[\xi]-z_{l-1,j_1},...,-z_{l-1,j_k})\in R_3^{k+1}$$

where the canonical form of $e_{m,\xi}$ is small.

- MKFHE.EvalAdd$(evk_S,\overline{\mathbf{c}}_1,\overline{\mathbf{c}}_2)$: On input two (extended) ciphertext $\overline{\mathbf{c}}_1$, $\overline{\mathbf{c}}_2\in R_{q_l}^{k+1}$ at the same level-*l* under the same secret key $\overline{\mathbf{s}}_l\in R_3^{k+1}$ (If needed, use key-switching and modulus-switching to make it so) .

  1. Compute $\overline{\mathbf{c}}_3\triangleq\overline{\mathbf{c}}_1+\overline{\mathbf{c}}_2\bmod q_l$ under the secret key $\overline{\mathbf{s}}_l\in R_3^{k+1}$.

2. Compute $\bar{\mathbf{c}}_3' \triangleq \mathrm{SwitchKey}(\tau_{\hat{\mathbf{s}}_l \to \bar{\mathbf{s}}_{l-1}}, \bar{\mathbf{c}}_3)$ under the secret key $\bar{\mathbf{s}}_{l-1} \in R_3^{k+1}$.

3. Compute $\bar{\mathbf{c}}_3'' \triangleq \mathrm{ModulusSwitch}(\bar{\mathbf{c}}_3', q_{l-1})$.

- MKFHE.EvalMult$(evk_S, \bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2)$: On input two (extended) ciphertext $\bar{\mathbf{c}}_1$, $\bar{\mathbf{c}}_2 \in R_{q_l}^{k+1}$ at the same level-l under the same secret key $\bar{\mathbf{s}}_l \in R_3^{k+1}$ (If needed, use key-switching and modulus-switching to make it so).

1. Compute $\bar{\mathbf{c}}_3 \triangleq \bar{\mathbf{c}}_1 \otimes \bar{\mathbf{c}}_2 \bmod q_l$ under the secret key $\hat{\mathbf{s}}_l = \bar{\mathbf{s}}_l \otimes \bar{\mathbf{s}}_l \in R_3^{(k+1)^2}$.

2. Compute $\bar{\mathbf{c}}_3' \triangleq \mathrm{SwitchKey}(\tau_{\hat{\mathbf{s}}_l \to \bar{\mathbf{s}}_{l-1}}, \bar{\mathbf{c}}_3)$ under the secret key $\bar{\mathbf{s}}_{l-1} \in R_3^{k+1}$.

3. Compute $\bar{\mathbf{c}}_3'' \triangleq \mathrm{ModulusSwitch}(\bar{\mathbf{c}}_3', q_{l-1})$.

C. ANALYSIS

1) SECURITY ANALYSIS

The basic BGV and GSW encryption scheme in our scheme are same as CZW17, and the main differences between us lies in: (1) we construct the nested ciphertext extension for BGV and separable ciphertext extension for GSW (2) we apply the 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 on ciphertext, so the security of our scheme is same as CZW17.

2) EFFICIENCY ANALYSIS

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

	CZW17	Our scheme
Ciphertext size (k users)	$2k\beta_l n$	$(k+1)\beta_l n$
Materials size	$\sum_{l=0}^{L} 24\beta_l^3 n$	$\sum_{l=0}^{L} (4\beta_B + 4\beta_l)\beta_l n$
Evaluation key size	$4k^2\beta_l^2 n$	$(k+1)^2 \beta_B \beta_l n$

As the range of secret key is limited to {-1,0,1} , 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*.

V. CONCLUSION

In this paper, we propose an efficient multi-key FHE scheme by constructing some efficient techniques such as nested ciphertext extension for BGV and separable ciphertext extension for GSW, and we apply the hybrid homomorphic multiplication between RBGV ciphertext and RGSW ciphertext, which can reduce the size of public parameters and evaluation keys, thus improve the efficiency of BGV-type MKFHE scheme. We also construct a directed decryption protocol which allows the evaluated ciphertext to be decrypted by any target user, thereby enhancing the ability of data owner to control their own plaintext.

REFERENCES

[1] Hamlin A., Shelat A., Weiss M., Wichs D, "Multi-Key Searchable Encryption, Revisited," in *IACR International Workshop on Public Key Cryptography – PKC 2018*, Berlin, Germany: Springer, vol. 10769, pp. 95-124, Mar.2018.

[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," in *STOC*, pp. 218-229, 1987.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)," in *STOC*, pp. 1-10, 1988.

[4] Chaum D., Crépeau C., Damgård I., "Multiparty Unconditionally Secure Protocols (Abstract)," in *Advances in Cryptology—CRYPTO '87*, Berlin, Germany: Springer, vol. 293, pp. 462-462, 1987.

[5] Craig Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, vol. 9, pp. 169-178, 2009.

[6] Dijk M V, Gentry C, Halevi S, *et al*., "Fully homomorphic encryption over the integers," in *International Conference on Theory and Applications of Cryptographic Techniques*. Berlin, Germany: Springer, pp. 24-43, 2010.

[7] Zvika Brakerski and Vinod Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *IEEE 52nd Annual Symposium on Foundations of Computer Science*, vol.2, pp. 97-106, 2011.

[8] Zvika Brakerski and Vinod Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Advances in Cryptology-CRYPTO 2011*, Berlin, Germany: Springer, pp. 505-524, 2011.

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

[10] Ivan Damg˚ard, Valerio Pastro, Nigel Smart, and Sarah Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology-CRYPTO 2012*, Springer, pp. 643-662, 2012.

[11] Craig Gentry, Shai Halevi, and Nigel P Smart, "Fully homomorphic encryption with polylog overhead," in *Advances in Cryptology-EUROCRYPT 2012*, Springer, pp. 465-482, 2012.

[12] Craig Gentry, Shai Halevi, and Nigel P Smart, "Homomorphic evaluation ofthe AES circuit," In *Advances in Cryptology-CRYPTO 2012*, Springer, pp. 850-867, 2012.

[13] Craig Gentry, Amit Sahai, and Brent Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology-CRYPTO 2013*, Heidelberg, Germany: Springer, pp. 75-92, 2013.

[14] BRAKERSKI Z, VAIKUNTANATHAN V, "Lattice-

based FHE as secure as PKE," in: *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ACM, pp. 1-12, 2014.
[15] Jacob Alperin-Sheriff and Chris Peikert, "Faster bootstrapping with polynomial error," in *Advances in Cryptology-CRYPTO 2014*, Berlin, Germany: Springer, pp, 297-314. 2014.
[16] Shai Halevi and Victor Shoup, "Algorithms in HElib," in *Advances in Cryptology-CRYPTO 2014, Proceedings, Part I*, pp. 554-571, 2014.
[17] L ́eo Ducas and Daniele Micciancio, "Fhew: Bootstrapping homomorphic encryption in less than a second, in *Advances in Cryptology-EUROCRYPT 2015*, Springer, pp. 617-640, 2015.
[18] Shai Halevi and Victor Shoup, "Bootstrapping for HElib," in *Advances in Cryptology-EUROCRYPT 2015*, Springer, pp. 641-670, 2015.
[19] Chillotti I, Gama N, Georgieva M, *et al*., "Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds," in *International Conference on the Theory and Application of Cryptology and Information Security—ASIACRYPT 2016*, Berlin, Heidelberg, Springer, pp.3-33, 2016.
[20] Chillotti I, Gama N, Georgieva M, *et al*., "Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE," in International Conference on the Theory and Application of Cryptology and Information Security—ASIACRYPT 2017. Cham, Springer, pp. 377-408, 2017.
[21] T. Zhou, X. Yang, L. Liu, W. Zhang and N. Li, "Faster Bootstrapping With Multiple Addends," in *IEEE Access*, vol. 6, pp. 49868-49876, 2018.
[22] 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*, ACM, pp. 1219-1234, 2012.
[23] 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*, pp. 630-656, 2015.
[24] Pratyay Mukherjee and Daniel Wichs, "Two round multiparty computation via multi-key FHE," in *Advances in Cryptology - EUROCRYPT 2016 , Proceedings, Part II*, pp. 735-763, 2016.
[25] Chris Peikert and Sina Shiehian, "Multi-key FHE from lwe, revisited," in *Theory of Cryptography - 14th International Conference, TCC*, pp. 217-238, 2016.
[26] BRAKERSKI Z, PERLMAN R，"Lattice-based fully dynamic multi-key FHE with short ciphertexts，" in: *Advances in Cryptology*—*CRYPTO 2016*. Heidelberg, Germany: Springer, pp. 190-213, 2016.
[27] Doröz, Y., Hu, Y, and Sunar, B, "Homomorphic AES evaluation using the modified LTV scheme," Designs, Codes and Cryptography, vol.80, pp. 333-358, 2016.
[28] Chen L, Zhang Z, Wang X, "Batched Multi-hop Multi-key FHE from Ring-LWE with Compact Ciphertext Extension," in *Theory of Cryptography Conference, TCC*. Springer, pp. 597-627, 2017.
[29] Chongchitmate W., Ostrovsky R, "Circuit-Private Multi-key FHE," in Public-Key Cryptography–PKC 2017, Heidelberg, Berlin, Springer, vol. 10175, pp.241-270, 2017.
[30] Rishab Goyal, "Quantum Multi-Key Homomorphic Encryption for Polynomial-Sized Circuits," https://eprint.iacr.org/2018/443.pdf.
[31] Hamlin A., Shelat A., *et al*., "Multi-Key Searchable Encryption, Revisited," in *Public-Key Cryptography – PKC 2018*, Springer, vol. 10769, pp. 95-124, 2018.
[32] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, "NTRU: A ring-based public key cryptosystem," in *International Symposium on Algorithmic Number Theory*, pp. 267-288, 1998.

NINGBO LI was born in Sanmenxia, China in 1992. He received the B.S. and M.S. degrees in Engineering University of People's Armed Police. Now he is a Ph.D. candidate in Engineering University of People's Armed Police. His main research interests include fully homomorphic encryption, encryption scheme based on lattice.
(372726936@qq.com)

TANPING ZHOU was born in Yingtan, China in 1989. He received Ph.D degrees in Engineering University of People's Armed Police. His main research interests include fully homomorphic encryption, encryption scheme based on lattice.
(850301775@qq.com).

XIAOYUAN YANG was born in Xi’an, China in 1959. He is a PhD supervisor in Engineerin g University of People's Armed Police. His ma in research interests include information securi ty and cryptology.

YILIANG HAN was born in 1978. PhD and PhD supervisor. His main research interests include information security and cryptology.

WENCHAO LIU was born in Wuwei, China i n 1994. He received the B.S. degrees in Sun ya t-sen university. Now he is a master in Enginee ring University of People's Armed Police. His main research interests include fully homomor phic encryption, encryption scheme based on l attice.