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

Tanping Zhou¹, Ningbo Li¹, XiaoYuan YANG¹, YiLiang HAN¹, WenChao LIU¹

¹Key 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 RGV 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 \( \lambda \) denote the security parameter and \( \text{negl}(\lambda) \) denote a negligible function of \( \lambda \).

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 \( x \overset{\text{unif}}{\leftarrow} D \) denotes that \( x \) is sampled uniformly from \( D \).

For \( a \in R \), we use \( \|a\|_\infty = \max_{\text{coeffs of } a} |a| \) to denote the standard \( l_\infty \)-norm and use \( \|a\|_1 = \sum_{i=0}^{n-1} |a_i| \) 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 \( \lambda \) be a security parameter. For the polynomial ring \( R = \mathbb{Z}[X]/X^n + 1 \) 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 \( (\mathbf{a}, b) \in R^{n+1}_q \) uniformly from \( R^{n+1}_q \). For the second distribution, one first draws \( \mathbf{a} \leftarrow R^n_q \) uniformly, and samples \( (\mathbf{a}, b) \in R^{n+1}_q \) by choosing \( \mathbf{s} \leftarrow R^n_q \) and \( \epsilon \leftarrow \chi \) uniformly, and set \( b = \langle \mathbf{a}, \mathbf{s} \rangle + \epsilon \). 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:

- \( \mathcal{E}.\text{Setup}(1^{|\lambda|^6}, l^X, l^Y) \): Given the security parameter \( \lambda \), 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 \( i (i = \ldots, K) \).
- \( \mathcal{E}.\text{Enc}(pk, m) \): Given a public key \( pk \) and message \( m \), outputs a ciphertext \( ct \).
- \( \mathcal{E}.\text{Dec}(sk, ct) \): Given a ciphertext \( ct \) corresponding to a set of users \( S = \{i_1, i_2, \ldots, i_t\} \subseteq [K] \), and their secret keys \( sk_j = \{sk_{i,j}, sk_{i,j}, \ldots, sk_{i,j}\} \), outputs the message \( m \).
- \( \mathcal{E}.\text{Eval}(C, (ct_{i_1}, pk_{i_1}, evk_{i_1}), \ldots, (ct_{i_t}, pk_{i_t}, evk_{i_t})) \): On input a Boolean circuit \( C \) along with \( t \) tuples \((ct_{i_j}, pk_{i_j}, evk_{i_j})_{i_j=1}^{t} \), each tuple comprises of a ciphertext \( ct_{i_j} \) corresponding to a user set \( S_{i_j} \), a set of public keys \( pk_{i_j} = \{pk_{j}, \forall j \in S_{i_j}\} \), and the evaluation keys \( evk_{i_j} \), outputs a ciphertext \( ct_{i_j} \) corresponding to a set of secret keys indexed by \( S = \bigcup_{i_j=1}^{t} S_{i_j} \subseteq [K] \).

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

\[
\Pr[\text{Dec}(sk_{i_j}, \text{Eval}(C, (ct_{i_j}, pk_{i_j}, evk_{i_j}), \mu_{i_1}), \ldots, \mu_{i_t})) \neq C(\mu_{i_1}, \ldots, \mu_{i_t})] = \text{negl}(\lambda)
\]
Definition 4 (Compactness of MKFHE). A leveled MKFHE scheme is compact if there exists a polynomial poly(\(\cdot, \cdot, \cdot\)) such that \(|\varepsilon| \leq \text{poly}(\lambda, K, L)\), which means that the length of \(\varepsilon\) is independent of the circuit \(\mathcal{C}\), but depend on the security parameter \(\lambda\), the number of users \(K\) and the circuit depth \(L\).

C. TWO SUBROUTINES

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

\(\text{BitDecomp}(x \in R_q^n, q)\): On input \(x = (x_1, \ldots, x_n) \in R_q^n\) and the modulus \(q\), outputs \((x_{1,0}, \ldots, x_{1,q-1}, x_{2,0}, \ldots, x_{2,q-1}, \ldots, x_{n,0}, \ldots, x_{n,q-1}) \in \{0,1\}^{\beta n}\) where \(x_{j,0}\) is the \(j\)-th bit in \(x_j\)’s binary representation (ordered from least significant to most significant), namely \(x = \sum_{j=0}^{\beta-1} 2^j x_{j,0} + \sum_{j=0}^{\beta-1} 2^j x_{j,1} + \cdots \sum_{j=0}^{\beta-1} 2^j x_{j,n} \). The correct result ciphertext of step 4 and output the ciphertext corresponding to the user’s message.

\(\text{Powersof 2}(y \in R_q^n, q)\): On input \(y = (y_1, \ldots, y_q) \in R_q^n\) and the modulus \(q\), outputs \(\left(y_1, 2y_1, \ldots, 2^{q-1}y_1, y_2, 2y_2, \ldots, 2^{q-1}y_q\right) \in R_q^{nq}\).

It’s straightforward to verify that for arbitrary \(x, y \in R_q^n\), it holds that

\[\text{BitDecomp}(x, q), \text{Powersof 2}(y, q) = (x, y) \mod q\]

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 \(c_i \in R_q^n\) (under the secret key \(s_i\)) to another ciphertext \(c_i \in R_q^n\) (under the secret key \(s_2\)) with the corresponding message unchanged. For a FHE scheme \(\mathcal{E}\), let \(\beta = \lceil \log q \rceil + 1\), the key switching process mainly consists of two procedures:

- \(\mathcal{E}'.\text{SwitchKeyGen}(s_i \in R_q^n, s_2 \in R_q^n)\): Compute \(\overline{s} = \text{Powersof 2}(s_i \in R_q^n, q)\), and output \(\tau_{s_i \rightarrow s_2} = \{\mathcal{A}_i = \text{Enc}_{s_2}(\overline{s}[i]) \in R_q^n\}_{i=1\ldots,n_{\beta}}\).

- \(\mathcal{E}'.\text{SwitchKey}(\tau_{s_i \rightarrow s_2}, c_i, q)\): Compute \(\overline{c}_i = \text{BitDecomp}(c_i) \in R_q^{nq}\), and output \(c_i = \sum_{i=1}^{n_{\beta}} \mathcal{A}_i \cdot \overline{c}_i[i] \in R_q^n\).

Lemma 1 (BGV12). Let \(c_i\) be a ciphertext under the key \(s_i\) for modulus \(q\) such that \(e_i \leftarrow [c_i, s_i > 1]_q\) has length at most \(B\) and \(m = [e_i]_2\). Let \(e_2 \leftarrow \mathcal{E}'.\text{SwitchKey}(\tau_{s_i \rightarrow s_2}, \overline{c}_i, q)\), and let \(e_2 \leftarrow [c_i, s_i > 1]_q\). Then \(e_2\) (the new noise) has length at most \(B + 2\cdot \gamma_R \cdot B \cdot \lceil \log q \rceil \cdot \sqrt{n_{\beta}}\), and (assuming this noise length is less than \(q/2\)) we have \(m = [e_i]_2\).

Modulus-switching: Since the noise involved in the ciphertext grows with homomorphic operations, modulus switching which can change the inner modulus \(q\) of ciphertext \(c_i\) to a smaller number \(q_i\) is used to reduce the noise term roughly by the ratio \(q_i / q\), while preserving the correctness of decryption under the same secret key.

- \(\mathcal{E}'.\text{ModulusSwitch}(c_i, q_i, q_i)\): On input \(c_i \in R_q^n\) and another smaller modulus \(q_i\), output \(c_i \in R_q^n\) which is the closest element to \([q_i / q_i] \cdot c_i\).

Lemma 2 (BGV12). Let \(c_i\) be a ciphertext under the key \(s_i\) for modulus \(q_i\) such that \(e_i \leftarrow [c_i, s_i > 1]_q\) has length at most \(B\) and \(m = [e_i]_2\). Let \(e_2 \leftarrow \mathcal{E}'.\text{ModulusSwitch}(c_i, q_i, q_i)\), and let \(e_i \leftarrow [c_i, s_i > 1]_q\). Then \(e_i\) (the new noise) has length at most \((q_i / q_i) \cdot B + \sqrt{n_{\beta}} \cdot \gamma_r \cdot B\), and (assuming this noise length is less than \(q_i / 2\)) we have \(m = [e_i]_2\).

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 step 2 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.
\section{Ciphertext Extension}

1) \textit{Nested Ciphertext Extension for BGV} 

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

\textbf{RBGV.Setup}(1, 1) : For the security parameter \(\lambda\), given a bound \(K\) on the number of keys, a bound \(L\) on the circuit depth with \(L\) decreasing modulus \(q_0 \gg q_{L-1} \gg \cdots \gg q_1\), for each level and a small integer \(p\) coprime with all \(q_i\), let \(\beta = \log q_0 + 1\). We work over rings \(R = \mathbb{Z}[X]/\phi_0\) and \(R_i = R / q_i R\) defined above. Let \(\chi = \chi(\lambda)\) be a B-bound error distribution over \(R\) whose coefficients are in the range \([-B, B]\).

\textbf{RBGV.KeyGen}(1, 1) : Generate keys of circuit depth \(l\) for the \(j\)-th party (\(l = 0, \ldots, L\)).

1. Sample \(z_{i,j} \sim R\) and set secret key \(sk_{i,j} = (1, -z_{i,j}) \in R_i^2\).
2. Generate \(a_{i,j} \sim R\) and \(e_{i,j} \sim \chi\) randomly, and compute the public key for the \(j\)-th user
\[
pk_{i,j} = \pi_{i,j} := (a_{i,j}, z_{i,j} + pe_{i,j} \mod q_i, a_{i,j}) \in R_i^{2}\]

\textbf{RBGV.Enc}(pk_{i,j}, \mu) : On input a message \(\mu \in R\) and the public key \(pk_{i,j}\), sample random elements \(r, e, e' \sim \chi\), compute level-\(l\) ciphertext
\[
c = (c^{(0)}, c^{(1)}) = (rb_{i,j} + pe + \mu, ra_{i,j} + pe') \in R_i^{2}\]

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 \(ct = \{e, S, l\}\). Here we set \(S = \{j\}\), \(l = L\), and output
\[
ct = \{e, \{j\}, L\}.
\]

\textbf{RBGV.Dec}(sk, ct) : On input a level-1 ciphertext \(ct = \{e, S, l\}\) where \(S = \{j_1, \ldots, j_l\}\), and its corresponding secret keys \((s_{j_1}, \ldots, s_{j_l}) \in R_i^{2}\). Let \(\bar{s}_{j_1} = (1, -z_{j_1}, \ldots, -z_{j_L}) \in R_i^{2\times l}\), output the message
\[
\mu \leftarrow \langle \bar{e}, \bar{s}_{j_1}\rangle \mod q_i \mod p
\]

\section{TENSION FOR BGV}

This process of homomorphic operation in \textit{MKFHE scheme}

\begin{itemize}
\item \textbf{RBGV.CTExt}(\(c_i, S\)) : On input a ciphertext tuple \(ct = \{e \in R_i^{2+l}, S = \{i_1, \ldots, i_{l}\}\) corresponding to \(k\) parties and another user set \(S' = \{j_1, \ldots, j_{l}\}\) for \(S \neq S'\), output an extended ciphertext \(ct' = \{\bar{e} \in R_i^{2+l}, S' = \{j_1, \ldots, j_{l}\}\}\). The extending algorithm is as follows:
\begin{enumerate}
\item (a) Divide the ciphertext \(e\) into \(k+1\) sequential sub-vectors indexed by \(S = \{i_1, \ldots, i_{l}\}\) (except for the first sub-vector), i.e.,
\[
\mathbf{e} = \{(c_{i_0}^{(0)}, c_{i_0}^{(1)}, \ldots, c_{i_0}^{(l)}) \in R_i^{2+l}\}
\]
where the corresponding secret key is \(s_{j_0} = (1, -z_{j_0}, \ldots, -z_{j_l})\).
\item (b) The extended ciphertext \(\bar{e}\) consists of \(k' + 1\) sequential sub-vectors, which can be indexed by \(S' = \{j_1, \ldots, j_{l}\}\), i.e.,
\[
\mathbf{\bar{e}} = \{(c_{j_0}^{(0)}, c_{j_0}^{(1)}, \ldots, c_{j_0}^{(l)}) \in R_i^{2+l}\}
\]
Set \(c_{j_0}^{(0)} = c_{j_0}^{(0)}\). If index \(j\) in \(S'\) is also included in \(S\), we set \(c_{j_0}^{(1)} = c_{j_0}^{(1)}\); otherwise we set \(c_{j_0}^{(0)} = 0\). The corresponding secret key for decryption is \(s_{j_0} = (1, -z_{j_0}, \ldots, -z_{j_0}) \in R_i^{2\times l}\).
It’s easy to verify that \(\langle \mathbf{e}, s_{j_0} \rangle = \langle \bar{e}, s_{j_0} \rangle \mod q_i\).
\end{enumerate}
\end{itemize}

2) \textit{Nested Ciphertext Extension for GSW}

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

\textbf{RGSW.Setup}(1) : For the security parameter \(\lambda\), given a bound \(K\) on the number of keys, a bound \(L\) on the circuit depth with \(L\) decreasing modulus \(q_0 \gg q_{L-1} \gg \cdots \gg q_0\) for each level and a small integer \(p\) coprime with all \(q_i\), let \(\beta = \log q_0 + 1\). We work over rings \(R = \mathbb{Z}[X]/\phi_0\) and \(R_i = R / q_i R\) defined above. Let \(\chi = \chi(\lambda)\) be a B-bound error distribution over \(R\) whose coefficients are in the range \([-B, B]\).

\textbf{RGSW.KeyGen}(1) : Sample \(z \sim R\), choose a random vector \(a \in R_i^{2n}\) and \(e \sim \chi\) uniformly, output the secret key \(s = (1, -z) \in R_i^{2}\) and public key \(\mathbf{P} = [az + pe, a] \in R_i^{2n+2}\).

\textbf{RGSW.EncRand}(r, P) : This procedure is to generate the encryption of randomness which is used in the ciphertext extension. On input \(r \sim R_i\), sample...
Finally we can get

\[ C_\beta s_j + X_s s_j = C_j s = p\bar{e} + \mu G s \]

where \( G = (1_{2^\beta}, 2_{2^\beta}, \ldots, 2^{2\beta-1}_{2^\beta})^T \in R^{2^{2\beta+2}} \).

**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 RGV ciphertexts and RSW ciphertexts instead of homomorphic multiplication between two RGV 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 BitDecomp() and Powersof 2() techniques are no longer required in key-switching, thus reduce the number of ciphertexts during key-switching process. For convenience, we use RSW.Enc(\(\mu\)) (or RGV.Enc(\(\mu\)) to denote a GSW/SGV ciphertext that can be decrypted to 0 with the secret key \(s\).

**MKFHE.EVKGen(\(em_s,pk_s\))**: Given a level-\(l\) extended secret key \(\vec{s} = \vec{s} \otimes \vec{z} = \{1,0,1\}^{l+1}\) for \(\vec{z} = (1, z_{i,1}, \ldots, z_{i,j})\), and corresponding public keys \(\{b_{i,j}, a_{i,j}\}_{z_{i,j}=0}^{z_{i,j}=1}\) for the user set \(S = \{j_1, \ldots, j_k\}\). For \(j \in \{1, \ldots, k\}\), \(m \in \{0, \ldots, \beta_i - 1\}\), compute

\[ \Psi_{j,i} = \text{RGSW.Enc}_{b_{i,j}}(z_{i,j}) \]
\[ F_{j,i} = \text{RGSW.EncRand}(r, \beta_{i,j}) \]
\[ \Phi_{j,i,m} = \text{RGBV.Enc}(2^m, z_{i,j}) \]

Output the evaluation keys \(evk = \{\mathcal{A}_{m,z} \in R_2^l\}\), \(z \in \{1, (k+1)\}^l\), and the process is shown in Algorithm 1.

Algorithm1: the generation of \(evk = \{\mathcal{A}_{m,z}\}\)

Input: \(\Psi_{j,i}, \ F_{j,i}, \ \Phi_{j,i,m}\)

1. for \(\zeta' \in \{0, \ldots, k\}\)
   \[ \vec{F}_{i,j} \{\zeta'\} = \begin{cases} \text{RGSW.Enc}_{\tau_{j,i}}(1) & \zeta' = 0 \\ \text{RGSW.CTExt}_{\tau_{j,i}}(\Psi_{j,i}, F_{j,i}, P_{j,i}) & \text{else} \end{cases} \]

2. for \(\zeta \in \{0, \ldots, k\}\)
   \[ \vec{F}_{i,j} \{\zeta\} = \begin{cases} \text{RGBV.Enc}_{\tau_{j,i}}(2^m) & \zeta = 0 \\ \text{RGBV.CTExt}_{\tau_{j,i}}(\Phi_{j,i,m}, S) & \text{else} \end{cases} \]

3. for \(\zeta' = \{0, \ldots, k\}\)
   \[ for \ z \in \{0, \ldots, k\}\]
   \[ for \ m \in \{0, \ldots, \beta_i - 1\}\]
   \[ \mathcal{A}_{i,m,k} = \vec{F}_{i,j} \{\zeta'\} \circ \vec{F}_{i,j} \{\zeta\} \]

4. output \(evk = \{\mathcal{A}_{m,z}\}_{m=0, \ldots, \beta_i - 1, \zeta=0, \ldots, z_{i,j}=1}^{z_{i,j}=1, (k+1)}\)
Definition 5 (Hybrid homomorphic multiplication). We define the product $\boxdot$ as

$$\boxdot: \text{RGSW} \times \text{RBGV} \rightarrow \text{RBGV}$$

$$\langle C_1, c_1 \rangle \rightarrow C_2 \boxdot c_1 = \text{BD}(c_1 \times C_2)$$

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 $C_2$ be a valid RGSW sample of message $\mu_2$ and let $c_1$ be a valid RBGV sample of message $\mu_1$. Then $C_2 \boxdot c_1$ is a RBGV sample of message $\mu_2 \cdot \mu_1$ and $\|\text{Err}(C_2 \boxdot c_1)\|_\infty \leq (2\beta) n \cdot 2\|\text{Err}(C_2)\|_\infty + \|\mu_2\|_\infty \cdot \|\text{Err}(c_1)\|_\infty$, Var$(\text{Err}(C_2 \boxdot c_1)) \leq 2p\beta(2n+1)\text{Var}(\epsilon) + p\text{Var}(\epsilon)$, where $n$ is the degree of the cyclotomic polynomial, $p$ is an integer, $\beta$ is the bound of the noise coefficients, $\sigma$ is the standard deviation of the error distribution $\chi$, $pe_i$ is the noise of $c_1$, $e \leftarrow \chi$ is the noise involved in $C_2$.

C. DIRECTED DECRIPTION 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](image)

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 decrypting 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 $c = (b_1, a_{j_1,i_1}, a_{j_2,i_2}, \ldots, a_{j_l,i_l}) \in R^{k+1}$, the corresponding user set $S = (j_1, \ldots, j_l)$ and the plaintext $\mu = C(\mu_1, \ldots, \mu_k)$ where $C$ is the Boolean circuit. Assume that the target user who needs to get the ultimate decrypting result is $i$, and he can get the ciphertext $c$, the process of our directed decryption protocol is implemented as follows:

1. Semi-decrypting: The users in $S$ respectively do decrypting operations on ciphertext $c$ with their special extended keys. For user $j_i$ with secret key $s_{i,j_i} = (1, -z_{i,j_i})$, get the semi-decrypting result $c'_{i,j} = (c_{i,j}, 0)$ by the extended key $\overline{s}_{i,j} = (1, -z_{i,j}, 0, \ldots, 0)$:

$$c'_{i,j} = (c_{i,j}, 0) = (c, \overline{s}_{i,j}, >, 0) = (b_i - a_{i,j} \cdot z_{i,j}, 0)$$

Other users in $S$ do similar operations as user $j_i$.

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

$$c_i = \text{RBGV. Enc}(pk_{i,j}, 0) = (b_{i,j}, a_{i,j}) \in R^{k+1}$$

For user $j_i$, compute the sum of $c_i$ and the semi-decrypting result $c'_{i,j}$:

$$c^*_{i,j} = (b_i - a_{i,j} \cdot z_{i,j_i} + b_{i,j}, a_{i,j})$$

Following the same method, other users can get \{c_{i,j}^*, c_{i,j}^*, \ldots, c_{i,j}^*\} and send them to user $i$.

3. Final decrypting: When user $i$ receives \{c_{i,j}^*, c_{i,j}^*, \ldots, c_{i,j}^*\}, he compute $c_{\text{sum}} = c_{i,j}^* + c_{i,j}^* + \ldots + c_{i,j}^*$, and compute the final decrypting result as

$$\mu = (c_{\text{sum}} - (k - 1)b_i) \cdot s_{j_i} = (c_{i,j}^* + \ldots + c_{i,j}^* - (k - 1)b_i) \cdot (1, -z_{i,j_i})$$

$$= kb_i - \sum_{m=1}^k a_{i,j} \cdot z_{i,j} - (k - 1)b_i + \sum_{m=k+1}^k (b_{i,j} - a_{i,j} \cdot z_{i,j})$$

$$= b_i - \sum_{m=1}^k a_{i,j} \cdot z_{i,j} + \sum_{m=k+1}^k (b_{i,j} - a_{i,j} \cdot z_{i,j})$$

$$= C(\mu_1, \ldots, \mu_k) + e_1 + \ldots + e_{i,j} + e_i + e_{i,j} + \ldots + e_{i,j}$$

$$= C(\mu_1, \ldots, \mu_k) \mod q_i \mod p$$

VOLUME XX, 2017

9
Lemma 3: Let $B$ denotes the bound of noise in a fresh RBGV ciphertext, and $B_j$ denotes the bound of noise in a level-$l$ RBGV ciphertext, then the directed decryption process is correct if
\[ |e_{i,1} + \ldots + e_{i,j} + e_{i,j+1}| \leq |e_{i,1} + \ldots + e_{i,j}| + kB \leq kB_j + kB < q / 4.\]

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 RGSW. Enc($\mu$) (presented in Section 3.1) to denote a GSW ciphertext (or RBGV. Enc($\mu$)) that can be decrypted to $\mu$ with the secret key $s$. Also we adopt the techniques of key-switching and modulus-switching introduced in section 2.4.

A. BASIC SCHEME

MKFHE.Setup($1^k$, $1^\ell$) : For the security parameter $\lambda$, 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 $q_i$. We work over rings $R = \mathbb{Z}[X]/\Phi$, and $R_q = R / q R$ defined above. Let $\chi = \chi(\lambda)$ be a $B$-bound error distribution over $R$ whose coefficients are in the range $[-B, B]$. Let $\beta_1 = \log q_0 + 1$, $\beta_2 = \log B + 1$, and choose $L+1$ random public vectors $a_i \in R_q^{2^i}$ for $l \in \{0, \ldots, L\}$. The following algorithm implicitly take the public parameter $pp = (R, B, \chi, (q_i, a_i), p)_{i=0}^{L+1}$ 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 $ct = (c, S, L)$.

MKFHE.KeyGen($pp$) : Given the public parameters $pp$, generate keys of circuit depth $l$ for the $j$-th party ($j = 0, \ldots, L$).

1. Sample $z_{i,j} \leftarrow \chi$ and set secret key $sk_{i,j} = sk_{i,j} = (1, -z_{i,j}) \in R_q^2$.

2. Choose $e_{i,j} \leftarrow \chi^{2^i}$ randomly, and compute the public key for the $j$-th user $pk_{i,j} = p_{i,j} := [a_i, z_{i,j} + pe_{i,j}, a_{i,j}]\in R_q^{2^{i+2}}$.

3. Compute the materials used in the generation of evaluation keys:
\[ em_j = \{ (\Phi_{i,j,m} \in R_q^{2^{i+2}}, (\Psi_{i,j} \in R_q^{2^{i+2}}), F_{i,j} \in R_q^{2^{i+2}}) \}_{i=0}^{L+1}. \]

where
\[ \Phi_{i,j,m} = \text{RBGV. Enc}_{b_{i,j}}(2^m \cdot z_{i,j}) = [t_{i,j,m} \in R_q^{2^{i+2}}, \Psi_{i,j} \in R_q^{2^{i+2}}, F_{i,j} \in R_q^{2^{i+2}}]_1, \ldots, L \]

\[ \Psi_{i,j} = \text{RGSW. Enc}_{b_{i,j}}(z_{i,j}) = [t_{i,j} \in R_q^{2^{i+2}}, p e_{i,j} + z_{i,j}, G] \in R_q^{2^{i+2}} \]

\[ F_{i,j} = \text{RGSW. EncRand}(t_{i,j}, pk_{i-1,j}) \in R_q^{2^{i+2}} \]

MKFHE.Enc($pk_{i-1,j}, \mu_j$) : On input a message $\mu_j \in R_q'$ and the public key $pk_{i,j}$, sample random elements $r$, $e' \leftarrow \chi$, compute level-$l$ ciphertext $c = (t_{i,j}, e_{i,j}) = (rb_{i,j}, \{ pe + \mu_j, ra_{i,j} + pe' \}) \in R_q'$ and output the tuple $ct = (c, \{j, L\})$.

MKFHE.Dec($sk_{i,j}, ct$) : On input a level-$l$ ciphertext $ct = (c, S, L)$ where $S = \{j_1, \ldots, j_k\}$, and its corresponding secret keys $\{s_{j_1}, \ldots, s_{j_k}\} \in R_q^{2 k}$. Let $\tilde{s}_{i,j} = (1, -z_{i,j}, \ldots, -z_{i,j}) \in R_q^L$. Output the message $\hat{\mu} = \{(c, \tilde{s}_{i,j}) \mod q_i\}$.

MKFHE.Eval($\{pk_{i-1,j}, \ldots, pk_{i,j}\}, em_{j}, c_{i}, ct_{i,\ldots, ct_j}$) : Assume that the sequence of ciphertexts $ct_i = (c, S, L)_{i=1}^k$ are at the same level-$l$ (If needed, use key-switching and modulus-switching to make it so). Let $S = \bigcup_{i=1}^k S_i = \{j_1, \ldots, j_k\}$. Then the outline of evaluation on Boolean circuit $C$ is as follows.

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

2. Generate the evaluation keys by compute $evk_{s_i} = \tau_{s_i} \cdot \tilde{s}_i = \text{MKFHE.EVKeyGen}(em_{j}, pk_{i,j})$.

3. Evaluate the circuit $C$ by using the two basic homomorphic operations MKFHE.EvalAdd($evk_{s_i}, \tilde{c}_i, \tilde{c}_{i'})$ and MKFHE.EvalMult($evk_{s_i}, \tilde{c}_i, \tilde{c}_{i'})$.

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 $\tilde{c}_i, \tilde{c}_{i'} \in R_q^{k+1}$ corresponding to the user set $S = \{j_1, \ldots, j_k\}$.

The evaluation key is defined as : $evk_{s_i} = \tau_{s_i} \cdot \tilde{s}_i = (\tau_{s_1} \cdot \tilde{s}_1) \cdot \ldots \cdot \tau_{s_k} \cdot \tilde{s}_k$ where $\tilde{s}_i = \tilde{s}_i \otimes \tilde{s}_j$ , $\tilde{s}_i = (1, -z_{i,j_1}, \ldots, -z_{i,j_k}) \in R_q^{k+1}$, $\tilde{s}_i = (1, -z_{i,j_1}, \ldots, -z_{i,j_k}) \in R_q^{k+1}$, and it holds that $\tilde{s}_i \cdot \tilde{s}_j = pe_{m_{s_i,j}} + 2^m \tilde{s}_j[\tilde{s}_i] - z_{i,j_1}, \ldots, -z_{i,j_k}] \in R_q^{k+1}$ where the canonical form of $m_{s_i,j}$ is small.

MKFHE.EvalAdd($evk_{s_i}, \tilde{c}_i, \tilde{c}_{i'}$) : On input two (extended) ciphertext $\tilde{c}_i, \tilde{c}_{i'} \in R_q^{k+1}$ at the same level-$l$ under the same secret key $\tilde{s} \in R_q^{k+1}$ (If needed, use key-switching and modulus-switching to make it so).

1. Compute $\tilde{c}_i \doteq \tilde{c}_i + \tilde{c}_{i'}$ mod $q_i$ under the secret key $\tilde{s} \in R_q^{k+1}$.
2. Compute $\overline{c}_3 \triangleq \text{SwitchKey}(\tau_3, \overline{c}_3)$ under the secret key $\overline{s}_3 \in R_{t+1}^l$.
3. Compute $\overline{c}_4 \triangleq \text{ModulusSwitch}(\overline{c}_3, q_{l+1})$.

- MKFHE.EvalMult(evk, $\overline{c}_3, \overline{c}_4$): On input two (extended) ciphertext $\overline{c}_3, \overline{c}_4 \in R_{t+1}^l$ at the same level-$l$ under the same secret key $\overline{s} \in R_{t+1}^l$, if needed, use key-switching and modulus-switching to make it so.
  1. Compute $\overline{c}_3 \triangleq \overline{c}_3 \otimes \overline{c}_4 \mod q_l$ under the secret key $\overline{s} = \overline{s} \otimes \overline{s} \in R_{t+1}^l$.
  2. Compute $\overline{c}_4 \triangleq \text{SwitchKey}(\tau_4, \overline{c}_4)$ under the secret key $\overline{s}_4 \in R_{t+1}^l$.
  3. Compute $\overline{c}_5 \triangleq \text{ModulusSwitch}(\overline{c}_4, q_{l+1})$.

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 RGV 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

<table>
<thead>
<tr>
<th>Material</th>
<th>CZW17</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphertext size ($k$ users)</td>
<td>$2k\beta n$</td>
<td>$(k+1)\beta n$</td>
</tr>
<tr>
<td>Materials size</td>
<td>$\sum_{i=0}^L 24\beta_i n$</td>
<td>$\sum_{i=0}^L (4\beta_i + 4\beta_i)\beta_i n$</td>
</tr>
<tr>
<td>Evaluation key size</td>
<td>$4k^2\beta^2 n$</td>
<td>$(k+1)^2\beta^2 n$</td>
</tr>
</tbody>
</table>

As the range of secret key is limited to $\{-1,0,1\}$, the ciphertext size of the secret key is reduced to $\beta_y$ 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 RGV 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


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 Engineering University of People's Armed Police. His main research interests include information security and cryptology.

YILJANG 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 in 1994. He received the B.S. degrees in Sun yangt-sen university. Now he is a master in Engineering University of People's Armed Police. His main research interests include fully homomorphic encryption, encryption scheme based on lattice.