Leveled Multikey FHE with constant-size ciphertexts from RLWE

Vanesa Daza[1], Paz Morillo[2], and Sergi Rovira[1]

[1] Universitat Pompeu Fabra
{vanesa.daza,sergi.rovira}@upf.edu
[2] Universitat Politècnica de Catalunya
paz.morillo@upc.edu

Abstract. A multi-key fully homomorphic encryption (MKFHE) scheme allows a public server to evaluate arbitrary circuits over ciphertexts encrypted under different keys. One of the main drawbacks of MKFHE schemes is the need for a ciphertext expansion procedure prior to evaluation, which combines ciphertexts encrypted under different keys to a (much larger) ciphertext encrypted under a concatenated key. In this paper, we present a new (leveled) RLWE-based MKFHE scheme without ciphertext expansion.

Keywords: Multi-key fully homomorphic encryption · Ring learning with errors · Party-independent ciphertexts

1 Introduction

Fully homomorphic encryption (FHE) refers to a class of encryption schemes which are capable of performing arbitrary computations over encrypted messages without using the secret key. The interest in this type of encryption scheme skyrocketed after Gentry [Gen09] gave the first (but still inefficient [vDGHV09]) FHE scheme in 2009. This lead to a plethora of results improving Gentry's design, for example [BV11b,BGV11,Bra12,FV12,GSW13,BV13,DM15,CKKS17]. The aforementioned papers focus on the study of *single-key* FHE schemes, meaning that a public server can homomorphically evaluate ciphertexts encrypted under the same secret key. This poses a clear limitation on the applications of FHE schemes. A more general construction is *multi-key* fully homomorphic encryption (MKFHE), first defined in [LTV13]. In MKFHE, a public server can homomorphically evaluate ciphertexts encrypted under different and independent keys. Subsequent works improved upon the original construction of [LTV13] in many ways, for example, by building schemes based on standard lattice assumptions [CM14,MW16] or by removing the need for prior knowledge about the number of parties [BP16,PS16]. All these MKFHE schemes and those that followed (e.g. [CZW17,KLP18,CDKS19,LP19,AJJM20,BD21]) present the common disadvantage of running an expensive *ciphertext expansion* routine before performing any computation. Ciphertext expansion takes a collection of ciphertexts encrypted by possibly different parties and produces a new (and much larger) ciphertext encrypted under a concatenated secret key. This implies that the size of the ciphertexts depends (quadratically or linearly) on the number of parties participating in the protocol.

In this work, we present a new MKFHE scheme where the ciphertext size does not depend on the number of participants of the protocol. As we explain in the related work section of this introduction, other concurrent works have addressed the problem of ciphertext expansion in recent months. The key idea behind our scheme is to add the parties' public keys to obtain a *global* public key (and the corresponding global secret key). Our starting point is a single-key encryption scheme which is additive homomorphic with respect to its keys if we assume the CRS model. That is, given a suitable CRS, the sum of public/secret keys is a valid public/secret key. Our construction can be instantiated with any scheme with this property. The second technique that we use is called *relinearization* (or key-switching) [BV11a,BGV11]. Relinearization allows us to switch from a degree two ciphertext to a degree one ciphertext. This is the crucial technique that allows the size of the ciphertexts to stay constant during multiplication. Finally, the last building block that

we need is a way for two different parties i and j with secret keys s_i and s_j to obtain a mask of $s_i s_j$. To achieve this, we adapt the ideas of [CDKS19] to build a suitable matrix, $\mathbf{K_{i,j}}$ such that $\mathbf{K_{i,j}} \cdot (1, s_i, s_j) \approx (s_i s_j, 2 s_i s_j, \ldots, 2^h s_i s_j)$ for some $h \in \mathbb{Z}_+$.

1.1 Related work

Multi-key fully homomorphic encryption was first defined in [LTV13]. This original construction is based on the NTRU encryption scheme, and it has a ciphertext expansion of approximately $N^{1+1/\epsilon}$ where N is the number of parties and ϵ depends on the security parameter. The next step was done in [CM14] and its follow-up paper [MW16] where the authors constructed a MKFHE scheme based on the LWE assumption with ciphertext expansion of approximately N^2. One drawback of these first MKFHE designs was the fact that some prior knowledge on the number of parties, the input ciphertexts and/or the circuit to be evaluated, was required at the time of setting parameters. This problem was successfully addressed in [BP16] and [PS16] where schemes that require no information about the parties during the key generation step are presented. However, both [BP16] and [PS16] provide schemes which have linear ciphertext expansion. In [CZW17] the authors provide the first RLWE-based MKFHE scheme building upon the BGV [BGV11] scheme. In this last work the authors manage to perform a much simpler ciphertext expansion than the previously mentioned constructions, however their expansion mechanism still depends on the number of parties. Recently, a few works [Jia21,DWF22] have addressed the problem of ciphertext expansion. In [Jia21], the author constructs a LWE-based MKFHE scheme with constant-size ciphertexts in the two non-collusion server model. This construction is built upon a modification of the GSW [GSW13] scheme. In [DWF22], the authors provide two MKFHE schemes, one based on the LWE assuption and the other in the RLWE assumption. This second scheme is similar to the construction presented in this work, since it obtains a MKFHE without ciphertext expansion from the RLWE assumption. The main difference between the RLWE scheme of [DWF22] and our work lies in the relinearization process and the relinearization keys.

1.2 Our results

Our main contribution is a new leveled MKFHE scheme without ciphertext expansion based on the RLWE assumption. To build our scheme, we provide a new multi-key variant of the LPR [LPR10] where the size of the keys remain the same as in the single-key setting. Moreover, we put forward a one-round interactive protocol which allows a set of parties to compute the product of their individual secret keys.

1.3 Paper organization

In Section 2, we establish notation and provide the necessary definitions and assumptions for the rest of the paper. In Section 3 we give the details on the components used to construct our scheme. Finally, in Section 4, we present our leveled MKFHE scheme and study the error growth.

2 Preliminaries

2.1 Basic notation

We will work over the polynomial rings $R := \mathbb{Z}[x]/\langle x^n+1\rangle$ and $R_q := R/qR$, where n is a power of 2 and q an odd integer. Single ring elements are written in lowercase ($r \in R$), vectors of ring elements are written in bold lowercase ($\mathbf{r} \in R^d$) and matrices are written in bold uppercase ($\mathbf{B} \in R^{d_1 \times d_2}$). Given a vector of ring elements $\mathbf{r} \in R^d$ we will denote the i-th element of the vector by $\mathbf{r}[i]$. Given $r \in R$, we will write $[r]_q$ to denote the reduction modulo q of each of the coefficients of the polynomial r. This reduction is done to the integers in the interval $(-\frac{q}{2}, \frac{q}{2}]$. The same notation will be used for vectors of polynomials in R^d to mean the reduction of each coefficient of each polynomial

in the vector. The infinity norm of a ring element $r = \sum_{i=0}^{n-1} r_i x^i \in R$ is defined as $\|r\| = \max_i |r_i|$ and the infinity norm of a vector of ring elements $\mathbf{r} \in R^d$ is defined as $\|\mathbf{r}\| = \max_i \|r[i]\|$. The expansion factor γ_R of R is defined as $\gamma_R = \max\{\|a \cdot b\|/\|a\|\|b\| : a, b \in R\}$. We will make use of the following polynomial functions over R_q^n:

$$\text{Powersof2} : R_q^n \longrightarrow R_q^{n \cdot \lceil \log_2 q \rceil} \qquad \text{BitDecomp} : R_q^n \longrightarrow R_q^{n \cdot \lceil \log_2 q \rceil}$$

$$\mathbf{p} \mapsto (\mathbf{p}, 2\mathbf{p}, \ldots, 2^{\lfloor \log_2 q \rfloor}\mathbf{p}) \qquad \sum_{i=0}^{\lceil \log_2 q \rceil} 2^i \cdot \mathbf{u}_i \mapsto (\mathbf{u}_0, \ldots, \mathbf{u}_{\lfloor \log_2 q \rfloor})$$

For the rest of the paper, λ denotes the security parameter, χ denotes the error distribution over R_q, B will be the upper bound on the errors sampled from χ, $d = \lceil \log_2 q \rceil$, and N will correspond to the number of parties participating in the protocol.

2.2 Definitions

Definition 1 (Decisional RLWE problem). *For security parameter λ, let $f(x) = x^n + 1$ where $n = n(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be an integer. Let $R = \mathbb{Z}[x]/\langle x^n+1 \rangle$ and $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be a distribution over R. The decisional $RLWE_{n,q,\chi}$ problem is to distinguish the following two distributions: In the first distribution, one samples (a_i, b_i) uniformly at random from R_q^2. In the second distribution, one samples $s \leftarrow R_q$ uniformly and then samples $(a_i, b_i = a_i \cdot s + e_i) \in R_q^2$ by sampling $a_i \leftarrow R_q$ uniformly and $e_i \leftarrow \chi$.*
The decisional $RLWE_{n,q,\chi}$ assumption is that the $RLWE_{n,q,\chi}$ problem is infeasible.

Definition 2 (Multi-key FHE encryption scheme).

- ***Setup:*** *$pp \leftarrow \mathtt{Setup}(1^\lambda)$. Takes the security parameter and outputs the public parameters for the rest of the algorithms.*
- ***Key generation:*** *$(sk, pk) \leftarrow \mathtt{KeyGen}(\mathtt{pp})$. Outputs a pair of secret and public keys.*
- ***Encryption:*** *$ct \leftarrow \mathtt{Enc}(\mathtt{pk}, \mathtt{m})$. Encrypts a plaintext $m \in \mathcal{M}$ and outputs a ciphertexts $c \in \mathcal{C}$.*
- ***Decryption:*** *$m \leftarrow \mathtt{Dec}(\{\mathtt{sk_i}\}_{\mathtt{i} \in [\mathtt{N}]}, \mathtt{ct})$. Given a ciphertext encrypted under the secret keys $\{sk_i\}_{i \in [N]}$ outputs a plaintext $m \in \mathcal{M}$. This procedure is normally done using one round threshold distributed decryption:*
  - *$p_i \leftarrow \mathtt{PartDec}(\mathtt{sk_i}, \mathtt{ct})$. On input a ciphertext ct encrypted under possibly many different secret keys it outputs a partial decryption p_i.*
  - *$m \leftarrow \mathtt{FinDec}(\mathtt{p_1}, \ldots, \mathtt{p_N})$. On input N partial decryptions, it outputs a plaintext $m \in \mathcal{M}$.*
- ***Evaluation:*** *$ct \leftarrow \mathtt{Eval}(\mathtt{C}, (\mathtt{ct_1}, \ldots, \mathtt{ct_N}), \{\mathtt{pk_i}\}_{\mathtt{i} \in [\mathtt{N}]})$. Given a circuit C, a tuple of ciphertexts $(ct_1, \ldots, ct_N)$ and the corresponding public keys $\{pk_i\}_{i \in [N]}$ outputs a ciphertext ct.*

Moreover, a MKFHE scheme should satisfy the following properties:

- **Correctness:** Let $(ct_1, \ldots, ct_N)$ be a collection of ciphertexts such that $\mathtt{Dec}(\{\mathtt{sk_i}\}_{\mathtt{i} \in [\mathtt{N}]}, \mathtt{ct_j}) = m_j$ for $1 \leq j \leq N$. Let C be a circuit and $ct \leftarrow \mathtt{Eval}(\mathtt{C}, (\mathtt{ct_1}, \ldots, \mathtt{ct_N}), \{\mathtt{pk_i}\}_{\mathtt{i} \in [\mathtt{N}]})$. Then, we say that the corresponding MKFHE scheme is correct if

$$\mathtt{Dec}(\{\mathtt{sk_i}\}_{\mathtt{i} \in [\mathtt{N}]}, \mathtt{ct}) = C(m_1, \ldots, m_N)$$

  with overwhelming probability.
- **Compactness:** Let $ct \leftarrow \mathtt{Eval}(\mathtt{C}, (\mathtt{ct_1}, \ldots, \mathtt{ct_N}), \{\mathtt{pk_i}\}_{\mathtt{i} \in [\mathtt{N}]})$. We say that the corresponding MKFHE scheme is compact if there exists a polynomial P such that $|ct| \leq P(\lambda, N)$. That is, the size of an evaluated ciphertext is independent of the size of the circuit being evaluated.

2.3 Assumptions

- **Circular Security.** In order to perform relinearization in our leveled MKFHE scheme and avoid circular security, each party would need to generate a secret key for each level of the circuit and produce masks of those keys (see Section 3.3) using a public key generated from the secret key of the previous level. For the sake of clarity and simpler notation, we have decided against generating multiple secret keys for each party, and we will allow each party to use a single secret key for all levels of the circuit, therefore assuming circular security. Moreover, since our relinearization keys consists of a mask of secret keys, we are assuming that publicly sharing these keys does not compromise security. As mentioned in [FV12], this is a weak form of circular security.
- **CRS model.** We work over the CRS model. That is, the parties have a common vector of polynomials in R_q which allows them to compute the global public key and the global relinearization key.
- **Semi-honest model.** We work in the semi-honest model, that is, we assume that all parties and the public server follow the specifications of the protocol.

3 Building blocks

In this section we detail the building blocks that we will use to construct our MKFHE scheme. Our starting point is a single-key encryption scheme which is additive homomorphic on its keys if we assume the CRS model. That is, given a suitable CRS, the sum of public keys is a valid public key and the same happens with the secret keys. Our construction can be instantiated with any scheme with this property. The second technique that we will use is called relinearisation (or key-switching) which was first introduced in [BV13]. Relinearisation allows us to switch from a ciphertext of size 3 to a ciphertext of size 2 both decrypting to the same plaintext. This is the key technique that allows the size of the ciphertexts to stay the same during multiplication. Finally, the last building block that we need is a way for two different parties i and j with secret keys s_i and s_j to obtain approximately Powersof2 $(s_i s_j)$.

3.1 Single-key encryption scheme

We will use the LPR encryption scheme [LPR10] as our single-key encryption scheme. We present it as it is done in [FV12]. Recall that we work over the polynomial ring $R_q = R/qR$, where $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$, n is a power of 2 and q an odd integer. Also, recall that χ denotes the error distribution over R_q.

- **Secret Key Generation:** $sk \leftarrow \texttt{LPR.SecretKeyGen}(1^\lambda)$. Sample the secret key $sk = s \leftarrow \chi$.
- **Public Key Generation:** $\mathbf{pk} \leftarrow \texttt{LPR.PublicKeyGen}(\texttt{sk})$. Sample a polynomial $a \leftarrow R_q$ uniformly at random, an error $e \leftarrow \chi$ from the error distribution, and set the public key to be

$$\mathbf{pk} = \left([-(a \cdot s + e)]_q, a\right) = (b, a) \in R_q^2.$$

- **Encryption:** $\mathbf{ct} \leftarrow \texttt{LPR.Enc}(\mathbf{pk}, \texttt{m})$. Sample $u \leftarrow \chi$, errors $e_1, e_2 \leftarrow \chi$ from the error distribution and set the encryption of $m \in R_2$ to be

$$\mathbf{ct} = \left(\left[b \cdot u + e_1 + \left\lfloor \frac{q}{2} \right\rfloor \cdot m\right]_q, [a \cdot u + e_2]_q\right) \in R_q^2$$

- **Decryption:** $m \leftarrow \texttt{LPR.Dec}(\texttt{sk}, \mathbf{ct})$. Given a ciphertext $\mathbf{ct} = (c_0, c_1)$ and the corresponding secret key s we can recover the plaintext by

$$m = \left[\left\lfloor \frac{2 \cdot [c_0 + c_1 \cdot s]_q}{q} \right\rceil\right]_2.$$

Observe that this is equivalent to first computing

$$\hat{m} = [c_0 + c_1 \cdot s]_q$$

and then for each coefficient i of $\hat{m}$ check if it is closer to 0 or to $\lfloor \frac{q}{2} \rfloor$. In the former set $m[i] = 0$ and in the latter $m[i] = 1$.

3.2 Multi-key encryption scheme

In this section we transform the scheme presented in 3.1 to a multi-key encryption scheme. The key property used is the fact that the LPR scheme is additive homomorphic on its keys (if we assume the CRS model).

- **Setup:** $\texttt{LMFV.Setup}(1^\lambda)$. A third trusted party samples $a \leftarrow R_q$ uniformly at random and shares it with the N parties participating in the protocol.
- **Secret key generation:** $\texttt{LMFV.SecretKeyGen}(1^\lambda)$. Each party i samples its secret key $s_i \leftarrow \chi$ from the error distribution.
- **Public key generation:** $\texttt{LMFV.PublicKeyGen}(\texttt{sk})$. Each party samples an error $e_i \leftarrow \chi$ and sets its public key as $\mathbf{pk_i} = ([-(a \cdot s_i + e_i)]_q, a)$. Then, using all the individual public keys each party can compute the global public key

$$\mathbf{pk} = \left(\sum_{i=1}^{N} \mathbf{pk_i}[0], a \right) = ([-(a \cdot s_+ + e_+)]_q, a) \in R_q^2,$$

  where $s_+ = \sum_{i=1}^{N} s_i$ and $e_+ = \sum_{i=1}^{N} e_i$.
- **Encryption:** $\mathbf{ct} \leftarrow \texttt{LMFV.Enc}(\mathbf{pk}, \texttt{m}) = \texttt{LPR.Enc}(\mathbf{pk}, \texttt{m})$. Encryption is the same as the single-key setting but using the global public key $\mathbf{pk}$ instead of the individual public keys $\mathbf{pk_i}$.
- **Decryption:** $m \leftarrow \texttt{LMFV.Dec}(\texttt{sk}, \mathbf{ct})$. Since no party should have the global secret key s_+, decryption is done by a standard secret sharing protocol. That is, given a ciphertext $\mathbf{ct}$ each party i computes the partial decryption $m_i = c_0 + c_1 s_i$ and secret-shares it with the rest of the parties. Once this is done, the parties can easily recover $c_0 + c_1 s_+$ and obtain the corresponding plaintext.

3.3 Relinearisation

Relinearization (or key-switching) is a very powerful technique which allows us to change the key under which a plaintext is encrypted without the need of decryption. This tool was introduced in [BV11a] to obtain the first homomorphic encryption scheme based only on the learning with errors (LWE) assumption. We will explain relinearization directly within the context of our construction.

Suppose that $\mathbf{ct_1}, \mathbf{ct_2} \in R_q^2$ are two ciphertexts produced by the scheme LMFV. In section 4.2 we will see that when we compute the multiplication of $\mathbf{ct_1}$ and $\mathbf{ct_2}$ we obtain a ciphertext $\mathbf{ct}_{\text{mult}} \in R_q^3$ which can be decrypted by computing $\langle \mathbf{ct}_{\text{mult}}, (1, s_+, s_+^2) \rangle$. This is a problem since the size of the ciphertext has increased from 2 ring elements to 3 ring elements and it will keep increasing each time that we do a multiplication. Fortunately, using relinearization we are able to construct another ciphertext $\mathbf{ct}'_{\text{mult}}$ which encrypts the same plaintext as $\mathbf{ct}_{\text{mult}}$ but which is of size 2 and can be decrypted by computing $\langle \mathbf{ct}'_{\text{mult}}, (1, s_+) \rangle$. In order be able to compute $\mathbf{ct}'_{\text{mult}}$ the parties will need to generate auxiliary information, the relinearization keys. Informally, the relinearization key of party i consists of a mask of Powersof2 $\left(s_i^2\right)$. Using the individual keys of each party we are able to obtain the global relinearization key which consists of maks of Powersof2 $\left(s_+^2\right)$. The careful reader will have realized that in order to obtain the global relinearization key we need something more than the individual relinearization keys of the parties. Indeed, what we need is a mask of Powersof2 $(s_i s_j)$ for all parties $i > j$.

3.4 The CDKS scheme

In order for two different parties to obtain $\text{Powersof2}(s_i s_j)$ we will use the (basic) CDKS scheme which consists of the following three algorithms.

1. **Setup:** $\texttt{CDKS.Setup}(1^\lambda)$. Samples a uniformly random vector $\mathbf{a} \leftarrow R_q^d$
2. **Key Generation:** $\texttt{CDKS.KeyGen}(\mathbf{a})$. Samples the secret key $s \leftarrow \chi$. Sample an error vector $\mathbf{e} \leftarrow \chi^d$ and set the public key as $\mathbf{b} = [-\mathbf{a} \cdot s + \mathbf{e}]_q$ in R_q^d.
3. **Encryption:** $\texttt{CDKS.UniEnc}(\mu; s)$. For an input plaintext $\mu \in R$, generate a ciphertext $\mathbf{D} = (\mathbf{d_0}|\mathbf{d_1}|\mathbf{d_2}) \in R_q^{d\times 3}$ as follows:
   - Sample $r \leftarrow \chi$
   - Sample $\mathbf{d_1} \leftarrow R_q^d$, and set $\mathbf{d_0} = [-\mathbf{d_1} \cdot s + \mathbf{e_1} + \text{Powersof2}(r)]_q$
   - Sample $\mathbf{e_2} \leftarrow \chi^d$ and set $\mathbf{d_2} = [\mathbf{a} \cdot r + \mathbf{e_2} + \text{Powersof2}(\mu)]_q$.

Using CDKS, we can obtain public information that allows two parties i and j to obtain $\text{Powersof2}(s_i s_j)$. To do so, party i needs to encrypt its secret key s_i and publish

$$\mathbf{D_i} = \texttt{CDKS.UniEnc}(s_i, s_i) = (\mathbf{d}_{i,0} \mid \mathbf{d}_{i,1} \mid \mathbf{d}_{i,2}).$$

Once this is done, party i can take the public key $\mathbf{b_j}$ of party j and compute

$$\mathbf{k_{i,j,0}}[z] = \langle \text{BitDecomp}(\mathbf{b}_j[z]), d_{i,0} \rangle,\ \mathbf{k_{i,j,1}}[z] = \langle \text{BitDecomp}(\mathbf{b}_j[z]), d_{i,1} \rangle,\ \mathbf{k_{i,j,2}} = \mathbf{d_{i,2}}$$

to obtain the matrix

$$\mathbf{K_{i,j}} = (\mathbf{k_{i,j,0}} \mid \mathbf{k_{i,j,1}} \mid \mathbf{k_{i,j,2}}) \in R_q^{d\times 3}.$$

Now notice that $\mathbf{K_{i,j}} \cdot (1, s_i, s_j) \approx \text{Powersof2}(s_i s_j)$. In section 4.1 we will explain how two parties can use $\mathbf{K_{i,j}}$ to add $\text{Powersof2}(s_i s_j)$ to the evaluation key.

4 Our leveled MKFHE encryption scheme

4.1 Adding homomorphic capabilities

To provide homomorphic capabilities to our scheme LMFV we will use relinearization (see 3.3).

To do this, each party needs to compute their individual *relinearisation keys* which, in our case are:

$$\mathbf{rlk}_i = \left(\left[-(\mathbf{a} \cdot s_i + \mathbf{e}_i) + \text{Powersof2}\left(s_i^2\right) \right]_q, \mathbf{a} \right) \in R_q^{d\times 2},$$

where $\mathbf{a} \in R_q^d$ is sampled uniformly at random and it is common for all parties and $\mathbf{e_i} \in R_q^d$ is sampled from the error distribution. As we did with the public key, the parties should find a way to compute the global relinearisation key:

$$\hat{\mathbf{rlk}} = \left(\sum_{i=1}^{N} \mathbf{rlk}_i[0] + 2 \cdot \text{Powersof2}\left(\sum_{i<j} s_i s_j \right), \mathbf{a} \right) = \left(\left[-(\mathbf{a} \cdot s_+ + \mathbf{e}_{rlk}) + \text{Powersof2}\left(s_+^2\right) \right]_q, \mathbf{a} \right) \in R_q^{d\times 2},$$

where $\mathbf{e_{rlk}} = \sum_{i=1}^{N} \mathbf{e}_i$ The key $\hat{\mathbf{rlk}}$ will allow us to go from a degree 2 ciphertext (c_0, c_1, c_2) to a degree 1 ciphertext (c_0', c_1'). Indeed if we set

$$c_0' = \left[c_0 + \langle \hat{\mathbf{rlk}}[0], \text{BitDecomp}(c_2) \rangle \right]_q, \quad c_1' = \left[c_1 + \langle \hat{\mathbf{rlk}}[1], \text{BitDecomp}(c_2) \rangle \right]_q$$

we get

$$\begin{aligned} c_0' + c_1' \cdot s_+ &= c_0 + \langle -(\mathbf{a} \cdot s_+ + \mathbf{e}_{rlk}) + \text{Powersof2}\left(s_+^2\right), \text{BitDecomp}(c_2) \rangle + c_1 \cdot s_+ + \langle \mathbf{a} \cdot s_+, \text{BitDecomp}(c_2) \rangle \mod q \\ &= c_0 + c_1 \cdot s_+ + c_2 \cdot s_+^2 - \langle \mathbf{e}_{rlk}, \text{BitDecomp}(c_2) \rangle \mod q. \end{aligned}$$

Notice however, that $\hat{\mathbf{rlk}}$ cannot be directly computed from the partial evaluation keys $\mathbf{rlk}_i$ since we require the term Powersof2 $\left(\sum_{i<j} s_i s_j\right)$. For this, we will use the technique explained in 3.4. First, each party i needs to compute $\mathbf{K_{i,j}}$ using $\mathbf{D_i}$ and $\mathbf{b_j}$. Once this is done, each party i needs to publish

$$\mathbf{k_{i,j,0}} + \mathbf{k_{i,j,1}} s_i \quad \forall j > i \text{ and } \mathbf{k_{j,i,2}} s_i \quad \forall j < i.$$

Then the server can compute

$$\mathbf{k_{i,j,0}} + \mathbf{k_{i,j,1}} s_i + \mathbf{k_{i,j,2}} s_j \approx \text{Powersof2}(s_i s_j) \; \forall \; i < j.$$

Unfortunately, this idea leads to an insecure system since any adversary have both $\mathbf{k_{i,j,2}} s_j$ and $\mathbf{k_{i,j,2}}$ leaking s_j for example. To fix this issue we need to introduce suitable errors. In particular, each party i publishes

$$\mathbf{k_{i,j,0}} + \mathbf{k_{i,j,1}} s_i + 2\mathbf{e_{i,j}} \quad \forall j > i \text{ and } \mathbf{k_{j,i,2}} s_i + 2\mathbf{e'_{i,j}} \quad \forall j < i.$$

Then the server can compute

$$\mathbf{k_{i,j,0}} + \mathbf{k_{i,j,1}} s_i + \mathbf{k_{i,j,2}} s_j + 2(\mathbf{e_{i,j}} + \mathbf{e'_{i,j}}) \approx \text{Powersof2}(s_i s_j) \; \forall \; i < j,$$

In the next section we will see that there is no need to publish all $\mathbf{K_{i,j}}$, only $k_{j,i,2} s_i$ so we will not need to mask $\mathbf{k_{i,j,0}} + \mathbf{k_{i,j,1}} s_i$.

4.2 Leveled multi-key homomorphic encryption scheme

In this section, we present our leveled multi-key FHE scheme. It consists of adapting the LPR encryption scheme to the multi-key setting, together with the necessary modifications needed to introduce global keys and our relinearization methodology. More precisely, our leveled multi-key FHE encryption scheme consists of the following algorithms:

- **Setup:** $\texttt{LMFV.Setup}(1^\lambda)$. A third trusted party samples $\mathbf{crt} \leftarrow R_q^{d+1}$ and shares it with the N parties participating in the protocol. Each party sets $a = \mathbf{crt}[0]$ and $\mathbf{a} = \mathbf{crt}[1:d]$.
- **Secret key generation:** $\texttt{LMFV.SecretKeyGen}(1^\lambda)$. Each party i samples its secret key $s_i \leftarrow \chi$ from the error distribution.
- **Public key generation:** $\texttt{LMFV.PublicKeyGen}(\texttt{sk})$. Each party samples an error $e_i \leftarrow \chi$ and sets its public key as $\mathbf{pk_i} = ([-(a \cdot s_i + e_i)]_q, a)$. Then, using all the individual public keys each party can compute the global public key

$$\mathbf{pk} = \left(\sum_{i=1}^{N} \mathbf{pk_i}[0], a\right) = ([-(a \cdot s_+ + e_+)]_q, a) \in R_q^2,$$

  where $s_+ = \sum_{i=1}^N s_i$ and $e_+ = \sum_{i=1}^N e_i$.
- **Relinearisation key generation:** $\texttt{LMFV.RelinKeyGen}(1^\lambda)$. This consists of the following steps.
  1. Each party i samples $\mathbf{e}_i, \mathbf{e}'_i \leftarrow \chi^d$, $\mathbf{a'_i} \leftarrow R^d$ and uses the CRS to compute its individual relinearisation keys and the public key of the basic CDKS scheme:

$$\mathbf{rlk}_i = \left(\left[-(\mathbf{a} \cdot s_i + \mathbf{e}_i) + \text{Powersof2}\left(s_i^2\right)\right]_q, \mathbf{a}\right) \in R_q^{d \times 2}, \quad \mathbf{b_i} = [-\mathbf{a'_i} \cdot s_i + \mathbf{e'_i}]_q.$$

  2. Each party i runs $\text{UniEnc}(s_i; s_i)$ to obtain $\mathbf{D_i} = (\mathbf{d_{0,i}} | \mathbf{d_{1,i}} | \mathbf{d_{2,i}})$.
  3. Each party i computes

$$\mathbf{K_{i,j}} = (\mathbf{k_{i,j,0}} \mid \mathbf{k_{i,j,1}} \mid \mathbf{k_{i,j,2}}) \quad \forall j > i,$$

  and publishes $\mathbf{k_{i,j,2}} \quad \forall j > i$.

4. Each party i samples $\mathbf{e_{i,j}} \leftarrow \chi^d$ for each $j < i$, and then it publishes

$$\mathbf{k_{i,j}} := \mathbf{k_{i,j,0}} + \mathbf{k_{i,j,1}} s_i \quad \forall j > i \text{ and } \mathbf{k'_{i,j}} := \mathbf{k_{j,i,2}} s_i + 2\mathbf{e_{i,j}} \quad \forall j < i.$$

5. Finally, the relinearization key can be computed as follows:

$$\mathbf{rlk} = \left(\sum_{i=1}^{N} \left(\mathbf{rlk}_i[0] + 2 \cdot \sum_{j=i+1}^{N} (\mathbf{k_{i,j}} + \mathbf{k'_{i,j}}) \right), \mathbf{a} \right) = \left(\left[-(\mathbf{a} \cdot s_+ + \mathbf{e}_{\widehat{\mathbf{rlk}}}) + \text{Powersof2}\left(s_+^2\right) + \mathbf{e_r} \right]_q, \mathbf{a} \right),$$

where $\mathbf{e}_{\widehat{\mathbf{rlk}}} = \sum_{i=1}^{N} \mathbf{e}_i$ and $\mathbf{e_r} = 4 \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathbf{e_{i,j}}$.

- **Encryption:** $\mathbf{ct} \leftarrow \texttt{LMFV.Enc}(\mathbf{pk}, \texttt{m}) = \texttt{LPR.Enc}(\mathbf{pk}, \texttt{m})$. Encryption is the same as the single-key setting but using the global public key $\mathbf{pk}$ instead of the individual public keys $\mathbf{pk_i}$.
- **Decryption:** $m \leftarrow \texttt{LMFV.Dec}(\texttt{sk}, \mathbf{ct})$. Since no party should have the global secret key s_+, decryption is done using a 1-round protocol. Each party computes a partial decryption c_i of c and then it shares it with all the other parties. Once this step of the protocol is finished, the parties can recover $m = \sum_i c_i$.
- **Addition:** $\mathbf{ct}_{\text{add}} \leftarrow \texttt{LMFV.Add}(\mathbf{ct_1}, \mathbf{ct_2})$. The addition of two ciphertext $\mathbf{ct_1}$ and $\mathbf{ct_2}$ is $\mathbf{ct}_{\text{add}} = (c_0, c_1)$, where

$$c_0 = [\mathbf{ct_1}[0] + \mathbf{ct_2}[0]]_q$$
$$c_1 = [\mathbf{ct_1}[1] + \mathbf{ct_2}[1]]_q$$

- **Multiplication:** $\mathbf{ct}_{\text{mult}} \leftarrow \texttt{LMFV.Mult}(\mathbf{rlk}, \mathbf{ct_1}, \mathbf{ct_2})$. First compute the multiplication of two ciphertext $\mathbf{ct_1}$ and $\mathbf{ct_2}$ as $\tilde{\mathbf{ct}}_{\text{mult}} = (c_0, c_1, c_2)$, where

$$c_0 = [2 \cdot \mathbf{ct_1}[0]\mathbf{ct_2}[0]]_q$$
$$c_1 = [2 \cdot (\mathbf{ct_1}[0]\mathbf{ct_2}[1] + \mathbf{ct_1}[1]\mathbf{ct_2}[0])]_q$$
$$c_2 = [2 \cdot \mathbf{ct_1}[1]\mathbf{ct_2}[1]]_q .$$

  Then call the relinearisation routine and return $\mathbf{ct}_{\text{mult}} \leftarrow \texttt{LMFV.Relin}(\mathbf{rlk}, \tilde{\mathbf{ct}}_{\text{mult}})$
- **Relinearisation:** $\mathbf{ct}_{\text{rlk}} \leftarrow \texttt{LMFV.Relin}(\mathbf{rlk}, \mathbf{ct})$. Given a degree two ciphertext $\mathbf{ct} = (c_0, c_1, c_2)$ write

$$c'_0 = [c_0 + \langle \mathbf{rlk}[0], \text{BitDecomp}(c_2) \rangle]_q , \quad c'_1 = [c_1 + \langle \mathbf{rlk}[1], \text{BitDecomp}(c_2) \rangle]_q$$

  and return the degree 1 ciphertext $\mathbf{ct}_{\text{rlk}} = (c'_0, c'_1)$.

4.3 Error growth

In this section, we present bounds on the parameters of LMFV which enables our scheme to evaluate a multiplicative circuit of bounded depth. We back this result by providing bounds on the noise introduced by encryption and multiplication (taking into account the errors introduced by relinearization). Overly generalizing, the noise of our construction can be understood as the noise of the base single-key encryption scheme changing the norm of a single key $\|s\|$ by the norm of the global secret key $\|s_+\|$ and the norm of a single error $\|e\|$ by the norm of the global error $\|e_+\|$.

Noise after encryption Consider a ciphertext $\mathbf{ct} = (c_0, c_1) \leftarrow \texttt{LMFV.Enc}(\mathbf{pk}, \texttt{m})$. Notice that after the first step of the decryption process we have

$$[c_0 + c_1 \cdot s_+]_q = t \cdot m - e_+ u + e_{c_0} + e_{c_1} s_+.$$

Therefore, the error after encryption is $e_{ct} = -e_+ u + e_{c_0} + e_{c_1} s_+$ which can be bounded as follows:

$$\|e_{ct}\| \leq \|e_{c_0}\| + \|e_{c_1} s_+\| + \|e_+ u\| \leq B + \gamma_R \|e_{c_1}\| \|s_+\| + \gamma_R \|e_+\| \|u\| \leq B + 2\gamma_R B^2 N.$$

For correct decryption we need to impose that $\|e_{ct}\| \leq \frac{q}{4}$. This proves the following lemma, which summarizes the correctness of our encryption scheme:

Lemma 1. *Let q, B, γ_R and N be as defined in Section 2.1. If*

$$B + 2\gamma_R B^2 N < \frac{q}{4},$$

then, we can decrypt correctly a ciphertext produced by `LMFV.Enc`.

Noise after addition Let $(c_0, c_1) \leftarrow \texttt{LMFV.Add}(\mathbf{ct_1}, \mathbf{ct_2})$ be the ciphertext corresponding to the homomorphic addition of the ciphertexts $\mathbf{ct_1}$ and $\mathbf{ct_2}$. Then we have that

$$[c_0 + c_1 \cdot s_+]_q = t \cdot (m_1 + m_2) - e_+(u_1 + u_2) + e^1_{c_0} + e^2_{c_0} + (e^1_{c_1} + e^2_{c_1})s_+ = t \cdot (m_1 + m_2) + e_{ct_1} + e_{ct_2},$$

and we can bound the error after addition $e_{add} = e_{ct_1} + e_{ct_2}$ by $\|e_{add}\| \leq 2B + 4\gamma_R B^2 N$. Now let us assume that we want to evaluate an arithmetic circuit of L levels consisting of fan-in 2 addition gates. At level 1 the error of the addition is $\|e_{add}\|$, at level 2 the error is $2\|e_{add}\|$, at level 3 the error is $4\|e_{add}\|$ and so on. Therefore, if we want to evaluate a circuit of L levels we need to impose that

$$\|e_{add}\| < \frac{q}{2^{L+1}}$$

or equivalently that

$$\|e_{ct}\| < \frac{q}{2^{L+2}}.$$

This proves the following lemma.

Lemma 2. *Let q, B, γ_R, d, and N be as defined in Section 2.1. If*

$$B + 2\gamma_R B^2 N < \frac{q}{2^{L+2}},$$

then, the scheme `LMFV` *can evaluate a depth L arithmetic circuit consisting of fan-in two addition gates.*

Noise after multiplication (without relinearisation) To study the noise after multiplication, we will make use of the analysis done by FV for the single-key case. Within the context of our paper, this analysis can be summarized in the following lemma.

Lemma 3. *Let $\mathbf{ct_i} \leftarrow \texttt{LPR.Enc}(\mathbf{pk}, \mathtt{m_i})$ for $i = 1, 2$ be two ciphertexts, with $[\mathbf{ct_i}(s)]_q = \lfloor \frac{q}{2} \rfloor \cdot m_i + e_{ct_i}$ and $\|e_{ct_i}\| < E < \lfloor \frac{q}{2} \rfloor$, then the error after multiplying $\mathbf{ct_1}$ and $\mathbf{ct_2}$ can be bounded by*

$$4\gamma_R E(\gamma_R \|s\| + 1) + 8\gamma_R^2(\|s\| + 1)^2.$$

Given this result, it is very easy to perform the analysis of the error for the multi-key case. We only need to take into account that we are not working with a single key s but with the collective key s_+. If we do so, we get the following lemma.

Lemma 4. *Let $\mathbf{ct_i} \leftarrow \texttt{LMFV.Enc}(\mathbf{pk}, \mathtt{m})$ for $i = 1, 2$ be two ciphertexts, with $[\mathbf{ct_i}(s_+)]_q = \lfloor \frac{q}{2} \rfloor \cdot m_i + e_{ct_i}$ and $\|e_{ct_i}\| < E < \frac{t}{2}$, then the error after multiplying $\mathbf{ct_1}$ by $\mathbf{ct_2}$ can be bounded by*

$$4\gamma_R(\gamma_R NB(E + 2NB + 4) + E + 2\gamma_R).$$

Noise introduced by relinearisation After relinearisation we have that

$$\begin{aligned} c'_0 + c'_1 \cdot s_+ &= c_0 + \langle -(\mathbf{a} \cdot s_+ + \mathbf{e_{rlk}}) + \text{Powersof2}\left(s_+^2\right) + \mathbf{e_r}, \text{BitDecomp}\left(c_2\right)\rangle + c_1 \cdot s_+ + \langle s_+ \cdot a, \text{BitDecomp}\left(c_2\right)\rangle \\ &= c_0 + c_1 s_+ + c_2 s_+^2 + \langle \mathbf{e_r} - \mathbf{e_{rlk}}, \text{BitDecomp}\left(c_2\right)\rangle. \end{aligned}$$

Recall that the error in the relinearisation key is given by $\mathbf{e_r} = 4\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathbf{e_{i,j}}$ which can be upper bounded as follows:

$$\|\mathbf{e_r}\| \leq 4\sum_{i=1}^{N}\sum_{j=i+1}^{N}\|\mathbf{e_{i,j}}\| \leq 4\sum_{i=1}^{N}\sum_{j=i+1}^{N}\cdot B = 2\cdot B\cdot(N-1) \tag{4.1}$$

Therefore, the error after relinearisation is bounded as follows:

$$\|\langle \mathbf{e_r}-\mathbf{e_{rlk}}, \text{BitDecomp}\,(c_2)\rangle\| \leq \sum_{i=1}^{d}\|\mathbf{e_r}[i]\|+\|\mathbf{e_{rlk}}[i]\| \leq \sum_{i=1}^{d} 2\cdot B\cdot(N-1)\cdot N+N\cdot B = d\cdot N\cdot B\cdot(2\cdot N-1), \tag{4.2}$$

where we have used the bound in 4.1 and $\|\mathbf{e_{rlk}}[i]\| \leq \sum_{j=1}^{N}\|\mathbf{e}_j[i]\| \leq N\cdot B$.

Noise after multiplication (with relinearisation) Using the bound on the multiplication noise given in 4 and the bound on the relinearisation noise obtained in (4.2) we can find a bound for the error after multiplication followed by relinearisation. To simplify the analysis we will use approximations on the obtained bounds. Recall that the error after encryption is $\approx 2\gamma_R B^2 N$. We will assume that when using 4 we have $\gamma_R E > NB$. This allows us to approximate the error after multiplication by $4\gamma_R^2 NBE$. With these approximations we see that after a single multiplication ($l=1$) we have an error of $8\gamma_R^3 N^2 B^3$. If we multiply two ciphertexts at level 1 we get an error of $32\gamma_R^5 N^3 B^4$ and so on. This means that at level l of multiplication we have a multiplication error of $\approx (2\delta_R)^{2l+1} N^{l+1} B^{l+2}$. Doing the same analysis for the relinearisation error we see that it is $\approx 2^{2l-1}\gamma_R^{2l-2} N^{l+1} B^l d$. Therefore the error at level l after multiplying and performing relinearisation is

$$\approx (2\delta_R)^{2l+1} N^{l+1} B^{l+2} + 2^{2l}\gamma_R^{2l-2} N^{l+1} B^l d.$$

To decrypt we need this error to be smaller than $\frac{q}{4}$, imposing this condition we obtain the following bound

$$N^{l+1}\gamma_R^{2l-2} B^l (2\gamma_R^3 B^2 + d) < \frac{q}{2^{2l+2}}.$$

This proves the following theorem.

Theorem 1. *Let q, B, γ_R, d, and N be as defined in Section 2.1. If*

$$N^{L+1}\gamma_R^{2L-2} B^L (2\gamma_R^3 B^2 + d) < \frac{q}{2^{2L+2}}.$$

then, the scheme `LMFV` *can evaluate a depth L arithmetic circuit consisting of fan-in two multiplication gates.*

References

AJJM20. Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. 2020. 1

BD21. Chinmoy Biswas and Ratna Dutta. Secure and efficient multi-key fhe scheme supporting multi-bit messages from LWE preserving non-interactive decryption. Cryptology ePrint Archive, Report 2021/1431, 2021. https://ia.cr/2021/1431. 1

BGV11. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS '12 Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325, 2011. 1, 2

BP16. Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *CRYPTO*, pages 190–213. Springer, 2016. 1, 2

Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in Cryptology - Crypto 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012. 1

BV11a. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Cryptology ePrint Archive, Report 2011/344, 2011. https://ia.cr/2011/344. 1, 5

BV11b. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, volume 6841 of *Lecture Notes in Computer Science*, page 501. Springer, 2011. 1

BV13. Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. Cryptology ePrint Archive, Report 2013/541, 2013. https://ia.cr/2013/541. 1, 4

CDKS19. Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. Cryptology ePrint Archive, Report 2019/524, 2019. https://ia.cr/2019/524. 1, 2

CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT (1)*, pages 409–437. Springer, 2017. 1

CM14. Michael Clear and Ciarán McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. Cryptology ePrint Archive, Report 2014/798, 2014. https://ia.cr/2014/798. 1, 2

CZW17. Long Chen, Zhenfeng Zhang, and Xueqing Wang. Batched multi-hop multi-key fhe from ring-lwe with compact ciphertext extension. In *TCC (2)*, pages 597–627. Springer, 2017. 1, 2

DM15. Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT (1)*, pages 617–640. Springer, 2015. 1

DWF22. Xiaokang Dai, Wenyuan Wu, and Yong Feng. Key lifting : Multi-key fully homomorphic encryption in plain model. Cryptology ePrint Archive, Report 2022/055, 2022. https://ia.cr/2022/055. 2

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144. 1, 4

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery. 1

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92. Springer, 2013. 1, 2

Jia21. Bingbing Jiang. Multi-key FHE without ciphertext-expansion in two-server model. *Frontiers of Computer Science*, 16(1), 2021. 2

KLP18. Eunkyung Kim, Hyang-Sook Lee, and Jeongeun Park. Towards round-optimal secure multiparty computations: Multikey fhe without a crs. Cryptology ePrint Archive, Report 2018/1156, 2018. https://ia.cr/2018/1156. 1

LP19. Hyang-Sook Lee and Jeongeun Park. On the security of multikey homomorphic encryption. Cryptology ePrint Archive, Report 2019/1082, 2019. https://ia.cr/2019/1082. 1

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. 2, 4

LTV13. Adriana Lopez, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. Cryptology ePrint Archive, Report 2013/094, 2013. https://ia.cr/2013/094. 1, 2

MW16. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT (2)*, pages 735–763. Springer, 2016. 1, 2

PS16. Chris Peikert and Sina Shiehian. Multi-key fhe from lwe, revisited. In *TCC (B2)*, pages 217–238. Springer, 2016. 1, 2

vDGHV09. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2009/616, 2009. https://ia.cr/2009/616. 1