Polynomial Functional Encryption Scheme with Linear Ciphertext Size

Jung Hee Cheon, Seungwan Hong, Changmin Lee, Yongha Son

Seoul National University (SNU), Republic of Korea

Abstract. In this paper, we suggest a new selective secure functional encryption scheme for degree d polynomial. The number of ciphertexts for a message with length ℓ in our scheme is $O(\ell)$ regardless of d, while it is at least $\ell^{d/2}$ in the previous works.
Our main idea is to generically combine two abstract encryption schemes that satisfies some special properties. We also gives an instantiation of our scheme by combining ElGamal scheme and Ring-LWE based homomorphic encryption scheme, whose ciphertext length is exactly $2\ell+1$, for any degree d.

1 Introduction

Functional encryption is an encryption scheme that, given the function key S_f for the function f and the encrypted message m, the decryption algorithm with the key S_f only outputs the evaluation value $f(m)$. Unlike to the traditional encryptions which decrypt all or nothing, functional encryption can provide only limited decryption capacity with a function key and so has a lot of interesting applications. As a specific subclass of such a functional encryption, AIBE, IBE, ABE were realized [9,22,25,31,32]. Because these are important applications in the field of cryptography, it shows the usefulness of functional encryption.

Unfortunately, it has not yet been fully clarified to construct the practical functional encryption that support the general circuit. A recent study show that indistinguishable obfuscation or cryptographic multilinear map can be a candidate to construct the functional encryption for all circuits. Since this indistinguishable obfuscation is implemented via multilinear maps, the results lead to the research of designing multilinear maps. However, until now, due to the absence of exact multilinear map of degree > 2, the functional encryption for general circuit has not been realized. On the other hand, instead of the exact multilinear map, a method using an approximate multilinear map has been suggested [18–20, 23], but the security of these maps is suspected due to several attacks in many cases [12,13,16,17,27,30].

Instead of supporting general functionality, a functional encryption that provide only certain functionality have been presented. Previously known results such as equality testing [7,9,24], keyword search [1,8,28], boolean formulate [26], inner product predicates [28], can be examples of the functional encryption. Besides, a functional encryption for inner product functionality (i.e. linear function) [2,3,6] and quadratic functions [5,21] are known to date. It also enables numerous applications. In particular, since all multivariate polynomial functions are represented by the inner product of the monomials, the functional encryption for general polynomial can be constructed using the functional encryption for inner product. In this case, however, ciphertexts of ℓ^d monomials are needed to compute the degree d polynomial for ℓ messages, which is far from practical. In summary, it remains an open problem to design a secure functional encryption with $O(\ell)$ ciphertexts, for degree $d > 2$ polynomials.

1.1 Our Contribution

In this paper, we provide a new generic construction of functional encryption with $O(\ell)$ ciphertexts for message vector length ℓ and degree-d polynomial functionality in secret key setting.

More specifically, we illustrate a generic method to construct a functional encryption by combining two encryption schemes satisfying certain properties. Briefly, the first scheme should have a decryption algorithm of linear arithmetic circuit. The second scheme supports module operations among ciphertexts. Refer to Section 3.1 for more details. We also present the our scheme achieves both function privacy and selective security, which is a security model that declares messages M_0 and M_1 in the semantic security game before generating the master key.

As a concrete example, we suggest a functional encryption scheme for degree-3 polynomial. It is designed by combining a ring LWE based homomorphic encryption scheme and an Elgamal public key encryption scheme. The scheme provides only $2\ell + 1$ ciphertexts, which is independent to degree. As another option, we propose a way to use the LTV homomorphic encryption scheme instead of ring LWE based scheme. In this case, the number of ciphertexts decreased to by $\ell + 1$.

Technical Overview. Here, we describe some intuitions with respect to construction idea and its security proof. First we recall the Abdella *et al.*'s functional encryption scheme for inner product functionality (for short, IPE scheme). In order to evaluate $\langle \vec{m}, \vec{f} \rangle$ for a message vector $\vec{m} = (m_1, \cdots, m_\ell) \in \mathbb{Z}^\ell$ and coefficients $\vec{f} = (f_1, \cdots, f_\ell) \in \mathbb{Z}^\ell$, it is the main idea of the IPE scheme to encrypt the each message entry m_i using a module encryption scheme, which is an encryption scheme supporting scalar multiplication and addition between ciphertexts. Then, by its properties, it multiplies the f_i coefficient and adds all ciphertexts to get the ciphertext of $\langle \vec{m}, \vec{f} \rangle$. Additionally the IPE scheme would have a special public information $S_{\vec{f}}$ that helps one to only decrypt the ciphertext. It implies that we obtain the inner product value of $\vec{m}$ and $\vec{f}$.

Now we consider the functional encryption for a polynomial function of order greater than 1. For the sake of simplicity, we only consider degree $d = 2$ polynomial F here. For an ℓ-length message vector $\vec{m} = (m_i)$, degree 2 polynomial function $F(\vec{m})$ can be represented by

$$F(\vec{m}) = \sum_{0\le i,j\le \ell} F_{i,j} \cdot m_i \cdot m_j$$

for some coefficient $F_{i,j}$. The representation implies that ciphertexts for all monomials of $m_i \cdot m_j$ is enough to evaluate function F like the existing IPE schemes. However, since existing IPE schemes do not support multiplication between ciphertexts, it looks nontrivial to devise a scheme of linear ciphertext size.

Our main idea is to use a hybrid technique, proposed by Cheon *et al.* [15]. In other words, we propose a functional encryption scheme by combining a scheme LDE, that provide a decryption circuit using only a linear arithmetic and an $\mathcal{R}$ module encryption scheme RMod, where the ring $\mathcal{R}$ is a underlying space of LDE's ciphertext.

For the sake of simplicity, we assume that the encryption of $\mathsf{LDE.Enc}\,(\mathsf{sk}_i, m_i)$ with secret key sk_i is decrypted with just multiplication, i.e. $m_i = \mathsf{LDE.Enc}\,(\mathsf{sk}_i, m_i) \cdot \mathsf{sk}_i$ Therefore, from each ciphertext of m_i $1 \le i \le \ell$ encrypted by LDE, one can generate the encryption of $m_i \cdot m_j$, say $\mathsf{LDE.Enc}\,(\mathsf{sk}_{i,j}, m_i \cdot m_j)$ by multiplying each ciphertexts, where $\mathsf{sk}_{i,j} := \mathsf{sk}_i \cdot \mathsf{sk}_j$ is the its secret key.

Now, to convert this ciphertext to that of module scheme, so called RMod, we provide evaluation keys. They are all encrypted with LDE's secret key with module scheme and of the form $\mathsf{RMod.Enc}(\mathsf{sk}_{i,j})$ (to simplify, we omit the secret key of RMod as an input of algorithm $\mathsf{RMod.Enc}$). We put here a coefficient factor $F_{i,j}$ so that we can naturally obtain the ciphertext of $F_{i,j} \cdot m_i \cdot m_j$. In other words, encryptions $\mathsf{RMod.Enc}(F_{i,j} \cdot \mathsf{sk}_{i,j})$ are published. Then, by multiplying it to $\mathsf{LDE.Enc}\,(\mathsf{sk}_{i,j}, m_i \cdot m_j)$, we obtain intermediate term, which can be expressed as follows.

$$\begin{aligned}
&\mathsf{LDE.Enc}\,(\mathsf{sk}_{i,j}, m_i \cdot m_j) \cdot \mathsf{RMod.Enc}(F_{i,j} \cdot \mathsf{sk}_{i,j}) \\
&= \mathsf{RMod.Enc}(F_{i,j} \cdot \mathsf{sk}_{i,j} \cdot \mathsf{LDE.Enc}\,(\mathsf{sk}_{i,j}, m_i \cdot m_j)) \\
&= \mathsf{RMod.Enc}(F_{i,j} \cdot m_i \cdot m_j)
\end{aligned}$$

Since RMod encryption scheme supports addition, by adding the all intermediate term, we then get the encryption of $F(\vec{m})$ for the RMod scheme. Finally, by decrypting the ciphertext using a special form, denoted c_0, which plays a similar role to the function key S_F in the Abdella *et al.*'s IPE scheme, we can recover the evaluation value $F(\vec{m})$.

To proceed to security proof, we observe that the public information that an adversary can obtain consists of

1) Ciphertexts c_i encrypted by LDE,
2) Encrypted secret information $\mathsf{sk}_{i,j}$ related to c_i encrypted by RMod,
3) An additional information c_0 that helps to decrypt $\mathsf{RMod.Enc}(F(\vec{m}))$.

In the case of 1) and 2), for $T = O(\ell^d)$, we prove that the semantic security of LDE(with the T number of message query) or RMod(with polynomial number of message query) is not provided if our functional encryption does not satisfy the semantic security due to 1) or 2) information leakage with T number of message query. In this argument we use a similar idea to the security proof of Cheon *et al.*'s hybrid scheme [15]. In other words, if the two base schemes satisfy the semantic security, 1) and 2) reveals no information. Finally, we show that c_0 can be computed by already revealed information from LDE and RMod, so we can conclude that c_0 do not reveal any other information in hybrid scheme. For more details, please refer the section 3.3.

2 Preliminary

In this section, we introduce notations used in this paper and recall the basic notions including the functional encryption and the somewhat homomorphic encryption needed to construct our schemes.

Notation For any set S, we write $|S|$ by the size of S. We denote $[n] = \{1, 2, \cdots, n\}$ for an positive integer n. For a positive integer p, we fix the represents of the group $\mathbb{Z}_p$ by $(-p/2, p/2] \cap \mathbb{Z}$.

Without other special mention, $\mathcal{R}$ denotes an abstract ring. We use bold letter to represent a polynomial. A vector having elements in $\mathcal{R}$(or polynomial) is denoted with upward arrow as $\vec{a}$(or $\vec{\boldsymbol{a}}$), and for a vector $\vec{a}$, we denote the i-th component of any vector $\vec{a}$ (or $\vec{\boldsymbol{a}}$) as a_i (or $\boldsymbol{a}_i$). We also denote a vector by $(a_i)_{i\in[\ell]}$ when we want to emphasize each entry and the length; if the length is obvious in the context, we simply write it by (a_i).

We use $\|\cdot\|_\infty$ to denote the infinity norm of vector. Moreover, for a polynomial $\boldsymbol{a}$, $\|\boldsymbol{a}\|$ denotes the infinity norm of the coefficient vector.

For a positive integers d and ℓ, we define the set $\mathcal{I}_{d,\ell}$ by the collection of every multi-subset[1] of $[\ell]$ having size d. We call an element of $I \in \mathcal{I}_{d,\ell}$ by an index, and if there is no confusion, we omit the subscripts d, ℓ. For any vector $\vec{\boldsymbol{x}} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_\ell)$ of ring elements and an index $I \in \mathcal{I}$, we define

$$\boldsymbol{x}^{(I)} = \prod_{i\in I} \boldsymbol{x}_i.$$

Given two vectors $\vec{a} = (a_1, \cdots, a_m)$ and $\vec{b} = (b_1, \cdots, b_n)$, the tensor product of $\vec{a}, \vec{b}$ is defined by $\vec{a} \otimes \vec{b} = (a_i \cdot b_j)_{(i,j)\in[m]\times[n]}$. Note that the tensor product of vectors of length m and n has the length mn.

For an algorithm $\mathcal{A}$, $a \leftarrow \mathcal{A}$ means that the algorithm A outputs a. For the distribution $\mathcal{D}$ and the set S, the algorithms $\mathcal{A}_\mathcal{D}$ and $\mathcal{A}_S$ refer to the algorithm for sampling according to the distribution $\mathcal{D}$ and the algorithm for sampling uniformly in the set S, respectively. If there is no confusion, we abolish the notation of $d \leftarrow D$ and $s \leftarrow S$ instead of $d \leftarrow \mathcal{A}_\mathcal{D}$ and $s \leftarrow \mathcal{A}_S$, respectively. When we computing between the outputs of algorithms, we use the operation between algorithms. For instance, $b = \mathcal{A}_1 + \mathcal{A}_2$ means that we first sample $a_1 \leftarrow \mathcal{A}_1$ and $a_2 \leftarrow \mathcal{A}_2$, and then compute $b = a_1 + a_2$.

[1] Multiset is modification of the concept of a set that, unlike a set, allows for multiple instances for each of its elements.

2.1 Functional Encryption

We first provide a definition of functionality and a secret key version of functional encryption based on Boneh *et al.*'s paper [10]. Next, we present security notions of the functional encryption.

Definition 1 (Functionality). *The functionality $\mathcal{F}$ defined over $(\mathcal{K}, \mathcal{M})$ is a function $\mathcal{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, where $\mathcal{K}, \mathcal{M}$, and $\mathcal{Y}$ are the key space, the message space, and the output space, respectively.*

Definition 2 (Secret key Functional Encryption (sFE)). *The secret key functional encryption for a functionality $\mathcal{F}$ over $(\mathcal{K}, \mathcal{M})$ consists of the algorithms* $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$. *Each algorithm is defined by*

- $\mathsf{FE.Setup}(1^\lambda)$ *: This algorithms runs following two subalgorithms :*
  - $\mathsf{FE.Param}(1^\lambda)$ *: Given security parameter λ, output public parameter* pp.
  - $\mathsf{FE.SkGen}(\mathsf{pp})$ *: Given public parameter* pp*, output master secret key* msk.

  Then output $(\mathsf{pp}, \mathsf{msk})$.
- $\mathsf{FE.KeyGen}(\mathsf{msk}, k)$ *: For a key $k \in \mathcal{K}$ and a master secret key* msk*,* KeyGen *outputs a secret key* sk_k.
- $\mathsf{FE.Enc}(\mathsf{msk}, m)$ *: For a message $m \in \mathcal{M}$ and a master secret key* msk*,* Enc *outputs a ciphertext* Ct.
- $\mathsf{FE.Dec}(\mathsf{sk}_k, \mathsf{Ct})$ *Given a secret key* sk_k *and a ciphertext* Ct, Dec *outputs $y \in \mathcal{Y}$.*

and satisfy the correctness condition: $y = \mathcal{F}(k, m)$ with overwhelming probability.

We remark that every algorithms defined after Setup need the public parameter pp as its input. However, we omit it for the sake of simplicity.

Security Intuitively, indistinguishability security model of sFE is to distinguish between the functional encryption of one input $m_0 \in \mathcal{M}$ and that of another $m_1 \in \mathcal{M}$, if one can only obtain secret keys for a functions which output the same values on m_0 and m_1. Depending on whether the two challenge inputs (m_0, m_1) are chosen adaptively or fixed, adaptive indistinguishability functional security and selective one are defined.

Since, in this work, we are mainly interested in the later security model, we only provide the definition of the selective security against T number of chosen-plaintext attacks via the security game below.

Definition 3 (T-Selective-Message IND **security with function Privacy for sFE(T-**SEL-IND-FP**)).** *There are two participants in T-*SEL-IND-FP *game. An adversary $\mathcal{A}$ wants to show its ability to break the* IND *security of chosen sFE scheme with T number of selected messages and a challenger $\mathcal{C}$ wants to make sure the $\mathcal{A}$'s argument is correct. For each purpose, they proceed the following game.*

1. *$\mathcal{C}$ generates* $\mathsf{pp} \leftarrow \mathsf{FE.Param}$ *and gives* pp *to $\mathcal{A}$.*

2. *$\mathcal{A}$ chooses $\vec{m}^0 = (m_1^0, m_2^0, \cdots, m_T^0), \vec{m}^1 = (m_1^1, m_2^1, \cdots, m_T^1) \in \mathcal{M}^T$ and gives $(\vec{m}^0, \vec{m}^1)$ to $\mathcal{C}$.*
3. *$\mathcal{C}$ generates $\mathsf{msk} \leftarrow \mathsf{FE.SkGen}$. Then $\mathcal{C}$ chooses random bit $b \leftarrow \{0,1\}$ and gives key generating oracle $\mathsf{FE.KeyGenO}_{\mathsf{msk},b}(\cdot, \cdot, \vec{m}^0, \vec{m}^1)$ that can be used polynomially many times to $\mathcal{A}$. This oracle is defined as follows.*
   - *$\mathsf{FE.KeyGenO}_{\mathsf{msk},b}(k^0, k^1, \vec{m}^0, \vec{m}^1)$: Given $k^0, k^1 \in \mathcal{K}$, this oracle checks whether k satisfies $\mathcal{F}(k^0, m_t^0) = \mathcal{F}(k^1, m_t^1)$ for all $t \in [T]$. If so, the oracle returns $SK_k \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, k^b)$. Otherwise it returns nothing.*
4. *$\mathcal{C}$ gives $\{c_t = \mathsf{FE.Enc}(\mathsf{msk}, m_t^b)\}_{t \in [T]}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

We define the advantage of $\mathcal{A}$ in this game by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE}\text{-}T\text{-SEL-IND-FP}} = |\Pr[b = b'] - \frac{1}{2}|.$$

We say the FE scheme satisfies the T-SEL-IND-FP security if there is no PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE}\text{-}T\text{-SEL-IND-FP}}$ is non-negligible.

Throughout the paper, FE means sFE unless otherwise noted.

3 Abstract Description of Our Scheme

In this section, we present our functional encryption scheme for degree-d polynomial funtionality and show that it satisfies T-Sel-Sec security. To follow the common way to write polynomial, we denote the funtionality $\mathcal{F}$ of FE scheme as $\mathcal{F}(F, \vec{x}) = F(\vec{x})$ for $F \in \mathcal{K}$ and $\vec{x} \in \mathcal{M}_{\mathsf{FE}}$.

We remark that we can only consider homogeneous polynomials, since every polynomials with ℓ variables can be expressed as a homogeneous polynomial with degree $\leq 2\ell$. For instance, the polynomial $F(x_1, x_2, x_3) = x_1 x_2 x_3 + 2x_1$ can be represented by $G(x_1, x_2, x_3, 1, 1)$ where

$$G(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 x_3 + 2x_1 x_4 x_5.$$

3.1 Generalized Encryption Schemes

Our functional encryption scheme is constructed by combining two abstract scheme; LDE and RMod. We first give the description of each scheme, and properties for each scheme to satisfy.

LDE : An Encryption Scheme with Linear Decryption Circuit. Let $\mathcal{R}$ be a ring. An encryption scheme with linear decryption circuit LDE is defined over the following spaces.

- The message space $\mathcal{M}_{\mathsf{LDE}}$,
- The secret key space $\mathcal{R}^n$ (or a subset of $\mathcal{R}^n$),

– The ciphertext space $\mathcal{R}^n$.

We require that LDE includes an encoding algorithm $\mathsf{LDE.Ecd}$ that transforms an element $m \in \mathcal{M}_{\mathsf{LDE}}$ into an element $\mu \in \mathcal{R}$. We also have an decoding algorithm $\mathsf{LDE.Dcd}$ that *uniquely* determines an element $m \in \mathcal{M}_{\mathsf{LDE}}$, given an element $\mu \in \mathcal{R}$.

Now LDE is defined by a family of algorithms

$$(\mathsf{LDE.Param}, \mathsf{LDE.SkGen}, \mathsf{LDE.Enc}, \mathsf{LDE.Dec}) :$$

– $\mathsf{LDE.Param}(1^\lambda)$: Output public parameter $\mathsf{pp} = \mathcal{R}$.
– $\mathsf{LDE.SkGen}(\mathsf{pp})$: Given public parameter pp, output the secret key $\vec{\mathsf{sk}} \in \mathcal{R}^n$.
– $\mathsf{LDE.Enc}(\vec{\mathsf{sk}}, m)$: Given a secret key $\vec{\mathsf{sk}} \in \mathcal{R}^n$ and a message $m \in \mathcal{M}_{\mathsf{LDE}}$, output a ciphertext $\vec{c} \in \mathcal{R}^n$.
– $\mathsf{LDE.Dec}(\vec{\mathsf{sk}}, \vec{c})$: Given a secret key $\vec{\mathsf{sk}} \in \mathcal{R}^n$ and a ciphertext $\vec{c} \in \mathcal{R}^n$, compute the inner product $\mu = \langle \vec{\mathsf{sk}}, \vec{c} \rangle \in \mathcal{R}$, and output $m \leftarrow \mathsf{LDE.Dcd}(\mu)$.

We further assume that $\mathsf{LDE.Ecd}$ preserves ring operations, and then the tensor product $\otimes : \mathcal{R}^* \times \mathcal{R}^* \to \mathcal{R}^*$ implies a multiplication between two ciphertexts, while extending ciphertext space and the secret key space $\mathcal{R}^* = \biguplus_{n \geq 1} \mathcal{R}^n$ (disjoint union). More precisely, for two ciphertext $\vec{c}_1$ and $\vec{c}_2$ having secret key $\vec{\mathsf{sk}}_1$ and $\vec{\mathsf{sk}}_2$, we define $\vec{c}_{\mathsf{Mult}} = \vec{c}_1 \otimes \vec{c}_2$; it holds that

$$\langle \vec{\mathsf{sk}}_1 \otimes \vec{\mathsf{sk}}_2, \vec{c}_1 \otimes \vec{c}_2 \rangle = \langle \vec{\mathsf{sk}}_1, \vec{c}_1 \rangle \cdot \langle \vec{\mathsf{sk}}_2, \vec{c}_2 \rangle.$$

Note that $\mathsf{LDE.Dec}$ algorithm is naturally extended to extended ciphertexts and secret keys.

– $\mathsf{LDE.Dec}(\vec{\mathsf{sk}}', \vec{c}')$: Given a secret key $\vec{\mathsf{sk}}' \in \mathcal{R}^*$ and a (extended) ciphertext $\vec{c}' \in \mathcal{R}^*$ having same length, compute the inner product $\mu = \langle \vec{\mathsf{sk}}', \vec{c} \rangle \in \mathcal{R}$, and output $m \leftarrow \mathsf{LDE.Dcd}(\mu)$.

Finally, since our main concern is ciphertexts of the form $\otimes_{i=1}^d \vec{c}_i$ where $\vec{c}_i$ having same length n, we especially denote by n_d the length of $\otimes_{i=1}^d \vec{c}_i$. In the above multiplication, we have $n_d = n^d$.

Remark. If the secret key is of the form $\vec{\mathsf{sk}} = (1, \mathsf{sk}, \cdots, \mathsf{sk}^{n-1}) \in \mathcal{R}^n$, one can define another multiplication for $\vec{c} = (c_i) \in \mathcal{R}^{n_1}$ and $\vec{c}_2 = (c'_i) \in \mathcal{R}^{n_2}$ by

$$\vec{c}_{\mathsf{Mult}} = \left(\sum_{i+j=k} c_i \cdot c'_j \right)_{2 \leq k \leq n_1 + n_2} \in \mathcal{R}^{n_1 + n_2 - 1}.$$

This results in $n_d = 1 + d \cdot (n-1)$, which is $O(d \cdot n)$. It can be easily checked that n'-length ciphertexts are decrypted by $(1, \mathsf{sk}, \cdots, \mathsf{sk}^{n'-1}) \in \mathcal{R}^{n'}$.

RMod : $\mathcal{R}$-module Encryption Scheme. Let $\mathcal{R}$ be a ring. We use the following spaces.

- The message space $\mathcal{R}$.
- The secret key space $\mathcal{R}$.
- The randomness space $\mathcal{R}$.
- The ciphertext space M^2, where $(M, \oplus, \odot)$ is an $\mathcal{R}$-module.

Then an *$\mathcal{R}$-module encryption scheme* RMod refers to a family of algorithms

$$(\mathsf{RMod.Param}, \mathsf{RMod.SkGen}, \mathsf{RMod.Enc}, \mathsf{RMod.Dec})$$

which are defined as below.

- $\mathsf{RMod.Param}(1^\lambda)$: Output public parameter $\mathsf{pp} = \mathcal{R}$.
- $\mathsf{RMod.SkGen}(\mathsf{pp})$: Given public parameter pp, output a secret key $\mathsf{sk} = s \in \mathcal{R}$.
- $\mathsf{RMod.Enc}(\mathsf{sk}, m)$: This algorithm is represented by a subalgorithm $\mathsf{RMod.E}$ defined by following :
  - $\mathsf{RMod.E}(\mathsf{sk}, m, r)$: Given a secret key $\mathsf{sk} \in \mathcal{R}$, message $m \in \mathcal{R}$, and randomness $r \in \mathcal{R}$, output an element $c \in M$.

  Given a secret key $\mathsf{sk} \in \mathcal{R}$ and a message $m \in \mathcal{R}$, sample $r \in \mathcal{R}$, and output

  $$\vec{c} = (\mathsf{RMod.E}(1, 0, r), \mathsf{RMod.E}(\mathsf{sk}, m, r)) \in M^2.$$
- $\mathsf{RMod.Dec}(\mathsf{sk}, \vec{c})$: Given a secret key $\mathsf{sk} \in \mathcal{R}$ and ciphertext $\vec{c} \in M^2$, output the message $m \in \mathcal{R}$.

The algorithm $\mathsf{RMod.E}$ has following properties.

- (Module Operations in M) : For any $a \in \mathcal{R}$ and E ciphertexts

  $$c_1 \leftarrow \mathsf{RMod.E}(s_1, m_1, r), c_2 \leftarrow \mathsf{RMod.E}(s_2, m_2, r),$$

  it holds that

  $$\begin{aligned} c_1 \oplus c_2 &= \mathsf{RMod.E}(s_1 + s_2, m_1 + m_2, r), \\ a \odot c_1 &= \mathsf{RMod.E}(a \cdot s_1, a \cdot m_1, r) \end{aligned}$$
- (Randomness deletion with zero secret key) : For any random $r_1, r_2 \in \mathcal{R}$ and $m \in \mathcal{R}$,

  $$\mathsf{RMod.E}(0, m, r_1) = \mathsf{RMod.E}(0, m, r_2).$$

  With this condition, one can use $\mathsf{RMod.E}(1, 0, r)$ to generate $\mathsf{RMod.E}(s', m', r)$ without knowing r. Note that anyone having pp can compute $\mathsf{RMod.E}(0, 1, 0)$, and hence we have

  $$\begin{aligned} &(s' \odot \mathsf{RMod.E}(1, 0, r)) \oplus (m' \odot \mathsf{RMod.E}(0, 1, 0)) \\ &= (\mathsf{RMod.E}(s', 0, r)) \oplus (\mathsf{RMod.E}(0, m', 0)) \\ &= (\mathsf{RMod.E}(s', 0, r)) \oplus (\mathsf{RMod.E}(0, m', r)) \\ &= \mathsf{RMod.E}(s', m', r) \end{aligned}$$

Security Models for LDE and RMod. The security model for LDE and RMod directly follows the one for symmetric key encryption. We state the definition of T-IND security model and IND security model for symmetric key encryption in Appendix A.

3.2 Hybrid Construction of FE Scheme from LDE and RMod

In the scheme description, the operator $\langle\, \cdot\, ,\, \cdot\, \rangle : \mathcal{R}^{n_d} \times \mathcal{R}^{n_d} \to \mathcal{R}$ is focused on the general inner product in $\mathcal{R}^{n_d}$. Meanwhile, we can consider another inner product between a vector in $\mathcal{R}^{n_d}$ and M^{n_d} via operations $\oplus, \odot$. We denote this inner product by $\langle\, \cdot\, ,\, \cdot\, \rangle_M$. In other words, for $(c_i)_{i \in [n_d]} \in \mathcal{R}^{n_d}$ and $(\mathsf{Ct}_i)_{i \in [n_d]} \in M^{n_d}$, we define

$$\langle (c_i), (\mathsf{Ct}_i) \rangle_M = \bigoplus_{i \in [n_d]} c_i \odot \mathsf{Ct}_i.$$

Now we are ready to construct a functional encryption scheme. Our functional encryption scheme can evaluate any degree d polynomial $F : \mathcal{M}_{\mathsf{FE}}^{\ell} \to \mathcal{M}_{\mathsf{FE}}$. We observe F can be represented by

$$F(\vec{m}) = F(m_1, \cdots, m_\ell) = \sum_{I \in \mathcal{I}} F_I \cdot \vec{m}^{(I)}$$

for some $F_I \in \mathcal{R}$, where the set $\mathcal{I}$ consists of all the size d multi-subsets of $[\ell]$.

Our hybrid functional encryption scheme

$$\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$$

with $\mathsf{pp} = \{\mathcal{R}, \ell, d\}$ is defined as below.

- $\mathsf{FE.Setup}(1^\lambda, \ell, d)$:
  - $\mathsf{FE.Param}(1^\lambda)$: Run $\mathsf{LDE.pp}(1^\lambda)$ and $\mathsf{RMod.pp}(1^\lambda)$ to get $\mathsf{pp}_{\mathsf{LDE}}$ and $\mathsf{pp}_{\mathsf{RMod}}$, respectively. Then construct $\mathsf{pp}_{\mathsf{FE}}$ from $\{\mathsf{pp}_{\mathsf{LDE}}, \mathsf{pp}_{\mathsf{RMod}}\}$. Output $\mathsf{pp}_{\mathsf{FE}}$.
  - $\mathsf{FE.SkGen}(\mathsf{pp}_{\mathsf{FE}}, \ell, d)$ Set $\mathsf{pp}_{\mathsf{LDE}} = \mathsf{pp}_{\mathsf{RMod}} = \mathcal{R}$ from $\mathsf{LDE.Param}$ and $\mathsf{RMod.Param}$ with security parameter λ.
    Run $\mathsf{LDE.SkGen}$ to get $\vec{\mathsf{sk}}_{\mathsf{LDE}}$ and run $\mathsf{RMod.SkGen}$ for each $(I, i) \in \mathcal{I} \times [n_d]$ to get secret keys $\{s_{I,i} = \mathsf{RMod.SkGen}(\mathsf{pp}_{\mathsf{RMod}})\}_{(I,i) \in \mathcal{I} \times [n_d]}$. Output

$$\mathsf{msk} = \{\mathsf{sk}_{\mathsf{LDE}}, \{\vec{s}_I = (s_{I,1}, \cdots, s_{I,n_d})\}_{I \in \mathcal{I}}\}.$$

- $\mathsf{FE.KeyGen}(\mathsf{msk}, F)$: Sample a random element $r \in \mathcal{R}$ and compute $\mathsf{evk}_0 = \mathsf{RMod.E}(1, 0, r)$. Then for each $I \in \mathcal{I}$, compute secret key

$$\otimes_{i=1}^{d} \vec{\mathsf{sk}}_{\mathsf{LDE}} := (\mathsf{sk}_1^d, \cdots, \mathsf{sk}_{n_d}^d).$$

  Now run $\mathsf{RMod.E}$ algorithm with same randomness r to get

$$\mathsf{evk}_{I,i} = \mathsf{RMod.E}(s_{I,i}, F_I \cdot \mathsf{sk}_i^d, r)$$

  for $i \in [n_d]$. Output $\mathsf{sk}_F = \{\mathsf{evk}_0, \{\vec{\mathsf{evk}}_I = (\mathsf{evk}_{I,1}, \cdots, \mathsf{evk}_{I,n_d})\}_{I \in \mathcal{I}}\}$.

– $\mathsf{FE.Enc}(\mathsf{pp}, \mathsf{msk}, \vec{m} = (m_1, \cdots, m_\ell) \in \mathcal{M}_{\mathsf{FE}}^\ell)$: Compute $c_i = \mathsf{LDE.Enc}(\mathsf{sk}_{\mathsf{LDE}}, m_i)$ for $i \in [\ell]$ and $c_0 = \sum_{I \in \mathcal{I}} \langle \vec{s}_I, \otimes_{i \in I} c_i \rangle$. Output $\mathsf{Ct} = \{c_0, \vec{c} = (c_1, \cdots, c_\ell)\}$.

– $\mathsf{FE.Dec}(\mathsf{sk}_F, \mathsf{Ct})$: Compute

$$C = \bigoplus_{I \in \mathcal{I}} \langle \otimes_{i \in I} c_i, \vec{\mathsf{evk}}_I \rangle_M.$$

Then compute $\mu = \mathsf{RMod.Dec}(c_0, (\mathsf{evk}_0, C)) \in \mathcal{R}$ and output $\mathsf{LDE.Dcd}(\mu) \in \mathcal{M}_{\mathsf{FE}}$.

We remark that, since the coefficients F_I are encrypted by the RMod encryption scheme, our scheme naturally provides function privacy.

Correctness. We denote $\vec{c}_I = \otimes_{i \in I} \vec{c}_i$. Then for each index $I \in \mathcal{I}$, we know

$$\vec{c}_I = \mathsf{LDE.Enc}(\otimes_{i=1}^d \vec{\mathsf{sk}}_{\mathsf{LDE}}, \vec{m}^{(I)}) \in \mathcal{R}^{n_d},$$

which means $\langle \otimes_{i=1}^d \vec{\mathsf{sk}}_{\mathsf{LDE}}, \vec{c}_I \rangle = \mathsf{LDE.Ecd}(m^{(I)})$ for some error $e_I \in R$. Therefore, it holds that

$$\begin{aligned}\langle \vec{c}_I, \vec{\mathsf{evk}}_I \rangle_M &= \langle \vec{c}_I, (\mathsf{RMod.E}(s_{I,i}, F_I \cdot \mathsf{sk}_i^d, r))_{i \in n_d} \rangle_M \\ &= \mathsf{RMod.E}(\langle \vec{s}_I, \vec{c}_I \rangle, F_I \cdot \langle \vec{\otimes}_{i=1}^d \vec{\mathsf{sk}}_{\mathsf{LDE}}, \vec{c}_I \rangle, r) \\ &= \mathsf{RMod.E}(\langle \vec{s}_I, \vec{c}_I \rangle, F_I \cdot \mathsf{LDE.Ecd}(\vec{m}^{(I)}), r).\end{aligned}$$

Hence we have

$$\begin{aligned}C &= \bigoplus_{I \in \mathcal{I}} \langle \vec{c}_I, \vec{\mathsf{evk}}_I \rangle_M \\ &= \bigoplus_{I \in \mathcal{I}} \mathsf{RMod.E}(\langle \vec{s}_I, \vec{c}_I \rangle, F_I \cdot \mathsf{LDE.Ecd}(\vec{m}^{(I)}), r) \\ &= \mathsf{RMod.E}(c_0, \mathsf{LDE.Ecd}(F(\vec{m})), r).\end{aligned}$$

Hence $\mu = \mathsf{RMod.Dec}(c_0, (\mathsf{evk}_0, C)) = \mathsf{LDE.Ecd}(F(\vec{m}))$, and then $\mathsf{LDE.Dcd}(\mu)$ gives $F(\vec{m})$.

3.3 Security Proof

In this section, we mainly work on the index set $\mathcal{I} \times [n_d]$ for evk's and ciphertext space. For the ease of reading, we denote the cardinality of set $\mathcal{I} \times [n_d]$ by L, and we also denote the index $(I, i) \in \mathcal{I} \times [n_d]$ by $j \in [L]$. In other words, a set(or vector) $(v_{I,i})_{(I,i)} \in \mathcal{R}$ is identically denoted as $(v_j)_{j \in [L]}$.

Additionally, in this proof we only use the inner product in the space $\mathcal{R}^L$. We denote this inner product by $\langle \cdot, \cdot \rangle_L : \mathcal{R}^L \times \mathcal{R}^L \to \mathcal{R}$, which is defined as

$$\langle \vec{a}, \vec{b} \rangle_L = \sum_{(I,i) \in N} a_{I,i} \cdot b_{I,i} = \sum_{j \in [L]} a_j \cdot b_j \text{ for any } \vec{a}, \vec{b} \in \mathcal{R}^L.$$

The security of our hybrid scheme FE is based of the security of LDE and RMod. We state this fact formally as following theorem.

Theorem 1. *For $\ell < T < |\mathcal{I}_{\ell,d}| \cdot n_d$, assume that the* LDE *scheme satisfies T-IND security and the* RMod *scheme satisfies* IND-CPA *security. Then our secret key functional encryption scheme* FE *constructed as Section 3.2 is T-SEL-IND-FP secure.*

The proof of the Theorem 1 is given in the rest of this section. Our proof is based on the hybrid argument. First, we define the games $\textbf{Game}_i$ for $i = 0, 1, 2,$ and 3. In each game, there exists an adversary $\mathcal{A}$ and plays a game with a challenger $\mathcal{C}$. We denote the advantage of $\mathcal{A}$ in $\textbf{Game}_i$ by $\mathsf{Adv}_{\mathcal{A}}^{\textbf{Game}_i}$.

$\textbf{Game}_0$ In this game, $\mathcal{C}$ and $\mathcal{A}$ plays the original selective IND-CPA game for FE scheme. We recall the Definition 3 by changing variables to fix with our definition of FE scheme.

1. $\mathcal{C}$ generates $\mathsf{pp} \leftarrow \mathsf{FE.Param}(1^\lambda)$ and gives pp to $\mathcal{A}$.
2. $\mathcal{A}$ chooses $M^0 = \{\vec{m}_1^0, \cdots, \vec{m}_T^0\}, M^1 = \{\vec{m}_1^1, \cdots, \vec{m}_T^1\} \in (R_2^\ell)^T$ and gives them to $\mathcal{C}$.
3. $\mathcal{C}$ generates $\mathsf{msk} = \{\vec{\mathsf{sk}}_{\mathsf{LDE}}, \{\vec{s}_I = (s_{I,i})_{i\in[n_d]}\}_{I\in\mathcal{I}}\} \leftarrow \mathsf{FE.SkGen}(\mathsf{pp})$.
4. $\mathcal{C}$ chooses random bit $b \leftarrow \{0,1\}$ and $\mathcal{A}$ earns the oracle $\mathsf{FE.KeyGenO}_{\mathsf{msk},b}(\cdot, \cdot, M^0, M^1)$ that can be used polynomially many times. This oracle works as follows.
   - $\mathsf{FE.KeyGenO}_{\mathsf{msk},b}(F^0, F^1, M^0, M^1)$: Given $F^{b'}(\vec{x}) = \sum_{I\in\mathcal{I}} F_I^{b'} \vec{x}^{(I)}$ for $b' \in \{0,1\}$, the oracle checks whether $F^0(\vec{m}_t^0) = F^1(\vec{m}_t^1)$ for all $t \in [T]$ or not. If not, the oracle returns nothing. Otherwise, the oracle chooses a random element $r_F \leftarrow \mathcal{R}$ and computes the followings :

$$\mathsf{evk}_0 = \mathsf{RMod.E}(\ 1,\ 0,\ r_F) \in M$$
$$\vec{\mathsf{sk}}_{\mathsf{LDE}}^{(d)} = (\mathsf{sk}_i^{(d)})_{i\in[n_d]} = \otimes_{i=1}^d \vec{\mathsf{sk}}_{\mathsf{LDE}} \in \mathcal{R}^{n_d}$$
$$\mathsf{evk}_{I,i} = \mathsf{RMod.E}(\ s_{I,i}, F_I^b \cdot \mathsf{sk}_i^{(d)},\ r_F) \text{ for } (I,i) \in \mathcal{I} \times [n_d]$$

   The oracle defines $\vec{\mathsf{evk}}_I = (\mathsf{evk}_{I,i})_{i\in n_d} \in M^{n_d}$ for $I \in \mathcal{I}$ and returns $\mathsf{sk}_F = \{\mathsf{evk}_0, \{\vec{\mathsf{evk}}_I\}_{I\in\mathcal{I}}\}$ to $\mathcal{A}$.
5. $\mathcal{C}$ computes $\mathsf{Ct}_t = (c_{t,0}, \vec{c}_t = (c_{t,i} = \mathsf{FE.Enc}(\mathsf{msk}, m_{t,i}^b))_{i\in[\ell]})$ for $t \in [T]$. Then $\mathcal{C}$ gives $\mathsf{Ct} = \{\mathsf{Ct}_t\}_{t\in[T]}$ to $\mathcal{A}$.
6. $\mathcal{A}$ outputs $b' \in \{0,1\}$.

By the definition, we have $\mathsf{Adv}_{\mathcal{A}}^{\textbf{Game}_0} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE}-T\text{-SEL-IND-FP}}$.

$\textbf{Game}_1$ In this game, the challenger $\mathcal{C}$ has an additional RMod ciphertext $\vec{\mathsf{Ct}}_{\mathsf{RMod}} \in M^2$, which is an encryption of 0. In precise,

$$\vec{\mathsf{Ct}}_{\mathsf{RMod}} = (\mathsf{Ct}_{\mathsf{RMod},0}, \mathsf{Ct}_{\mathsf{RMod},1}) = (\mathsf{RMod.E}(1,0,r), \mathsf{RMod.E}(\mathsf{sk}_{\mathsf{RMod}}, 0, r))$$

for some unknown RMod secret key $\mathsf{sk}_{\mathsf{RMod}} \in \mathcal{R}$ and a random $r \in \mathcal{R}$. Compared to $\textbf{Game}_0$, the steps 3, 4, and 5 are modified as follows :

– Step 3: Before $\mathcal{C}$ generates secret keys of FE and give pp to $\mathcal{A}$, $\mathcal{C}$ further precompute some conatants for the step 4. First $\mathcal{C}$ chooses $\mathsf{pp} \leftarrow \mathsf{FE.Param}(1^\lambda)$ and generates $\vec{\mathsf{sk}}_{\mathsf{LDE}} \leftarrow \mathsf{LDE.SkGen}(\mathsf{pp})$ only. Then $\mathcal{C}$ randomly chooses a bit $b \leftarrow \{0,1\}$ and computes $\vec{c}_t = (c_{t,i} = \mathsf{LDE.Enc}(\vec{\mathsf{sk}}_{\mathsf{LDE}}, m_i^b)_{i\in[\ell]})$ for $t \in [T]$. Let $\vec{c}_I^{\,t} = (c_{I,i}^t)_{i\in[n_d]} = \otimes_{i\in I} c_{t,i} \in \mathcal{R}^{n_d}$ and $c_j^t = c_{I,i}^t$. Now $\mathcal{C}$ randomly samples a nonzero vector $\vec{v}_1 = (v_{1,j})_{j\in[L]}$ in the set

$$\mathcal{H} = \{\vec{y} \in \mathcal{R}^L : \langle \vec{y}, (c_j^t)_j \rangle_L = 0 \text{ for all } t \in [T]\}.$$

Note that $\mathcal{H}$ is a orthogonal space of the set $\{(c_j^t)_j\}_{t\in[T]}$ in inner product space $(\mathcal{R}^L, \langle\,\cdot\,,\,\cdot\,\rangle_L)$. From the assumption $T < L$, $\mathcal{H}$ is not a trivial space. Then selects $\mathcal{C}$ chooses random $L-1$ number of polynomials $\{s_k\}_{2\le k\le L}$ in $\mathcal{R}$ and also find vectors $\vec{v}_k = (v_{k,j})_{j\in[L]} \in \mathcal{R}^L$ for $2 \le k \le L$ so that the matrix $V = (v_{j,k})_{1\le j,k\le L} \in (\mathcal{R})^{L\times L}$ is invertible.
Now it's ready. $\mathcal{C}$ gives pp to $\mathcal{A}$.

– Step 4: In this step, the oracle $\mathsf{KeyGenO}_{\mathsf{msk},b}$ computes evk's differently. Instead of the original evk_0 and $\vec{\mathsf{evk}}_I$ for $I \in \mathcal{I}$, the oracle defines

$$\begin{aligned}\mathsf{evk}_0 &= \mathsf{Ct}_{\mathsf{RMod},0}\\ \mathsf{evk}_{I,i} &= \mathsf{evk}_j\\ &= (v_{1,j} \odot \mathsf{Ct}_{\mathsf{RMod},1}) \oplus \Big(\bigoplus_{k=2}^{L} (s_k \cdot v_{k,j}) \odot \mathsf{Ct}_{\mathsf{RMod},0}\Big) \oplus (\mathsf{sk}_j \odot \mathsf{RMod.E}(0,1,0))\end{aligned}$$

for all $(I,i) = j \in [L]$ where $\mathsf{sk}_j = F_I^b \cdot \mathsf{sk}_i^d$ when (I,i) corresponds to j. Then $\mathsf{KeyGenO}$ returns $\mathsf{sk}_F = \{\mathsf{evk}_0, \{\vec{\mathsf{evk}}_I = (\mathsf{evk}_{I,i})_{i\in[n_d]}\}\}$.

– Step 5: Using $\{\vec{c}_t\}_{t\in[T]}$ which are already computed in Step 3, $\mathcal{C}$ computes $c_{t,0}$ as

$$c_{t,0} = \sum_{k=2}^{L} s_k \langle \vec{v}_k, (c_j^t)_j \rangle_L \text{ for } t \in [T].$$

Then $\mathcal{C}$ defines $\mathsf{Ct}_t = (c_{t,0}, \vec{c}_t)$ for $t \in [T]$ and gives $\{\mathsf{Ct}_t\}_{t\in[T]}$ to $\mathcal{A}$.

Lemma 1. $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_1} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_0}$.

Proof. By the properties of $\mathsf{RMod.E}$, evk_j can be written as $\mathsf{RMod.E}(s'_j, \mathsf{sk}_j, r)$ where $\vec{s}\,' = (s'_j)_{j\in[L]} = \mathsf{sk}_{\mathsf{RMod}} \cdot \vec{v}_1 + \sum_{k=2}^{L} s_k \cdot \vec{v}_k = V \cdot \big(\mathsf{sk}_{\mathsf{RMod}}\ s_2 \cdots s_L\big)^T$ for each $j \in [L]$. We claim that $c_{t,0}$ satisfies the condition $c_{t,0} = \langle \vec{s}\,', (c_j^t)_j \rangle_L$ for all $t \in [T]$. This is true since

$$\begin{aligned}\langle \vec{s}\,', (c_j^t)_j \rangle_L &= \langle \mathsf{sk}_{\mathsf{RMod}} \cdot \vec{s}_1 + \sum_{k=2}^{L} a_k \cdot \vec{s}_k, (c_j^t)_j \rangle_L\\ &= \mathsf{sk}_{\mathsf{RMod}} \cdot \langle \vec{s}_1, (c_j^t)_j \rangle_L + \langle \sum_{k=2}^{L} a_k \cdot \vec{s}_k, (c_j^t)_j \rangle_L\\ &= \sum_{k=2}^{L} a_k \langle \vec{s}_k, (c_j^t)_j \rangle_L\\ &= c_{t,0}\end{aligned}$$

from the fact $\vec{s}_1 \in \mathcal{H}$. Therefore the only difference between $\mathbf{Game}_0$ and $\mathbf{Game}_1$ is that the vector of RMod secret keys $\vec{s}$ is changed to $\vec{s}'$. Since $\mathsf{sk}_{\mathsf{RMod}}$ and $s'_k s$ are randomly sampled in $\mathcal{R}$ and the matrix V is invertible, $\vec{s}'$ is also chosen randomly, same as in $\mathbf{Game}_0$. In other words, $\{\vec{s}_j\}_{j\in[L]}$ and $\{\vec{s}'_j\}_{j\in L}$ belong to the same distribution in $\mathcal{R}^L$. Therefore the difference between $\mathbf{Game}_0$ and $\mathbf{Game}_1$ doesn't affect the advantage of $\mathcal{A}$. □

$\mathbf{Game_2}$ In this game, the only difference with $\mathbf{Game}_1$ is that the challenger $\mathcal{C}$ is given RMod ciphertext $\vec{\mathsf{Ct}}_{\mathsf{RMod}} \in M^2$ is an encryption of random $a \in \mathcal{R}$, not encryption of zero. In precise,

$$\vec{\mathsf{Ct}}_{\mathsf{RMod}} = (\mathsf{Ct}_{\mathsf{RMod},0}, \mathsf{Ct}_{\mathsf{RMod},1}) = (\mathsf{RMod.E}(1,0,r), \mathsf{RMod.E}(\mathsf{sk}_{\mathsf{RMod}}, a, r)).$$

Lemma 2. $|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_1}| \le \mathsf{Adv}_{\mathcal{B}_{\mathsf{RMod}}}^{\mathsf{RMod}-\text{IND-CPA}}$ *for some adversary* $\mathcal{B}_{\mathsf{RMod}}$ *in* IND-CPA*game of the* RMod *scheme.*

Proof. The only difference between $\mathbf{Game}_2$ and $\mathbf{Game}_1$ is the given RMod ciphertext. Hence if there exists an adversary $\mathcal{B}_{\mathsf{RMod}}$ who can distinguish $\mathbf{Game}_2$ and $\mathbf{Game}_1$ can also distinguish $\mathsf{RMod.E}(\mathsf{sk}_{\mathsf{RMod}}, 0, r)$ and $\mathsf{RMod.E}(\mathsf{sk}_{\mathsf{RMod}}, a, r)$ by the difference of advantages between $\mathbf{Game}_2$ and $\mathbf{Game}_1$. Therefore the difference of advantages between $\mathbf{Game}_2$ and $\mathbf{Game}_1$ for any adversary $\mathcal{A}$ must be smaller than the advantage of $\mathcal{B}_{\mathsf{RMod}}$ in IND-CPA game for RMod. □

$\mathbf{Game_3}$ In this game, the only difference with $\mathbf{Game}_2$ is Step 4. The Step 4 is replaced as follows:

– Step 4 : From the knowledge of $C = \{\vec{c}_t\}_{t\in[T]}$, (F^0, F^1) and (M^0, M^1), $\mathcal{C}$ computes the subspace of $\mathcal{R}^L$ by

$$\mathcal{H}_{C,F^0,M^0} = \{\vec{y} \in \mathcal{R}^L : \langle \vec{y}, (c_j^t)_{j\in[L]} \rangle_L = F^0(\vec{m}_t^0) \text{ for } t \in [T]\}.$$

Note that $\mathcal{H}_{C,F^0,M^0}$ is not a trivial subspace in $\mathcal{R}^L$ since $T < L$. Hence $\mathcal{C}$ can sample random nonzero vector $\vec{t} = (t_j)_{j\in[L]} \in \mathcal{H}_{C,F^0,M^0}$. Then $\mathcal{C}$ samples $s_j \in \mathcal{R}$ for $j \in [L]$, $r_F \in \mathcal{R}$ and defines evk_0, $\mathsf{evk}_{I,i} = \mathsf{evk}_j$ as

$$\begin{aligned} \mathsf{evk}_0 &= \mathsf{RMod.E}(1, 0, r_F) \\ \mathsf{evk}_j &= \mathsf{RMod.E}(s_j, t_j, r_F) \text{ for } j \in [L]. \end{aligned}$$

The other computations are the same.

Lemma 3. $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_3} = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_2}$.

Proof. Recall that $\mathsf{sk}_j = F_I \cdot \mathsf{sk}_i^d$ when the index j corresponds to (I, i). In $\mathbf{Game}_2$, evk_j is equal to $\mathsf{RMod.E}(s'_j, \mathsf{sk}_j + a \cdot s_{1,j}, r)$ for all $j \in [L]$. On the other hand, in $\mathbf{Game}_3$, evk_j is equal to $\mathsf{RMod.E}(s'_j, t_j + a \cdot s_{1,j}, r)$ for all $j \in [L]$. Since $\vec{s}_1$ is chosen to satisfy $\langle \vec{s}_1, (c_j^t)_j \rangle_L = 0$ for all $t \in [T]$, we can check that $(\mathsf{sk}_j)_j + a \cdot \vec{s}_1 \in \mathcal{H}_{C,F^0 M^0}$

regardless of choice of $\vec{s}_1 \in \mathcal{H}$. Conversely, every elements in $\mathcal{H}_{C,F^0,M^0}$ can be expressed as $(\mathsf{sk}_j)_j + a \cdot \vec{s}$ for some $a \in \mathcal{R}$ and $\vec{s} \in \mathcal{H}$. We remark that the vector $\vec{t} \in \mathcal{H}_{C,F^0,M^0}$ works as the secret key of LDE whatever b is since $F^0(\vec{m}_t^0) = F^1(\vec{m}_t^1)$ for all $t \in [T]$. Therefore the distrubition of $(\mathsf{sk}_j)_j + a \cdot \vec{s}_1$ and $\vec{t}$ is the same. This fact implies that $\mathbf{Game}_2$ and $\mathbf{Game}_3$ are indistinguishable. $\square$

Lemma 4. $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{Game}_3} \leq \mathsf{Adv}_{\mathcal{B}_{\mathsf{LDE}}}^{\mathsf{LDE}-T\text{-IND}}$ *for some adversary* $\mathcal{B}_{\mathsf{LDE}}$ *in* T-IND *game of the* LDE *scheme.*

Proof. Note that one can efficiently compute the set $\mathcal{H}_{\vec{c},F(\vec{m}^b)}$ if $\vec{c}, \vec{m}^b$ and F are given. Using this set, we claim that there exists an adversary $\mathcal{B}_{\mathsf{LDE}}$ who plays an T-IND game about LDE scheme with challenger $\mathcal{C}'$ using the adversary $\mathcal{A}$ in $\mathbf{Game}_3$ as an oracle. The detail of the game between $\mathcal{C}'$ and $\mathcal{B}_{\mathsf{LDE}}$ is as follows.

1. $\mathcal{C}'$ runs $\mathsf{LDE.Param}$ and $\mathsf{LDE.SkGen}$ to get $(\mathsf{pp}, \mathsf{sk})$ and gives pp to $\mathcal{B}_{\mathsf{LDE}}$. Then $\mathcal{B}_{\mathsf{LDE}}$ gives pp to $\mathcal{A}$.
2. $\mathcal{A}$ chooses $M^0 = (\vec{m}_t^0)_{t\in[T]}, M^1 = (\vec{m}_t^1)_{t\in[T]} \in (\mathcal{R}^\ell)^T$ and gives this pair to $\mathcal{B}_{\mathsf{LDE}}$. Then $\mathcal{B}_{\mathsf{LDE}}$ gives m^0, m^1 to $\mathcal{C}$.
3. $\mathcal{C}'$ randomly chooses a bit $b \leftarrow \{0,1\}$ and computes $c_{t,i} = \mathsf{LDE.Enc}(\mathsf{sk}_{\mathsf{LDE}}, m_{t,i}^b)$ for $(t,i) \in [T] \times [\ell]$. $\mathcal{C}'$ defines $\vec{c}_t = (c_{t,i})_{i\in[\ell]}$ for $t \in [T]$ and gives $\{\vec{c}_t\}_{t\in[T]}$ to $\mathcal{B}_{\mathsf{LDE}}$. Then $\mathcal{B}_{\mathsf{LDE}}$ gives $\{\vec{c}_t\}_{t\in[T]}$ to $\mathcal{A}$.
4. $\mathcal{B}_{\mathsf{LDE}}$ randomly fixes $\mathsf{msk}' = \{s_{I,i}\}_{(I,i)\in\mathcal{I}\times n_d}$ sampled in $\mathcal{R}$. Then $\mathcal{B}_{\mathsf{LDE}}$ gives the oracle $\mathsf{KeyGenO}'_{\mathsf{msk}'}(\cdot, \cdot, M^0, M^1, \{c_t\}_{t\in[T]})\}$ to $\mathcal{A}$. This oracle is defined as follows:
   – $\mathsf{KeyGenO}'_{\mathsf{msk}'}(F^0, F^1, M^0, M^1, \{c_t\}_{t\in[T]})\}$: Given $F^{b'} : \vec{x} \mapsto \sum_{I\in\mathcal{I}} F_I^{b'} \vec{x}^{(I)}$ for $b' \in \{0,1\}$, the oracle checks whether $F^0(\vec{m}_t^0) = F^1(\vec{m}_t^1)$ for all $t \in [T]$. If not, then this oracle returns nothing. Otherwise, the oracle computes the subspace $\mathcal{H}_{C,F^0,M^0}$ of $\mathcal{R}^L$ by $C = \{\vec{c}_t\}_{t\in T}, F^0$ and M^0. (since $F^0(\vec{m}_t^0) = F^1(\vec{m}_t^1)$ for all $t \in [T]$, $\mathcal{H}_{C,F^0,M^0}$ is not depend on b). Then the oracle samples $\vec{y} = (y_I = (y_{I,i})_{i\in n_d})_{I\in\mathcal{I}} \leftarrow \mathcal{H}_{\vec{c},F(\vec{m}^b)}$. For random $r_F \in \mathcal{R}$, the oracle computes the followings:
$$\begin{aligned} \mathsf{evk}_0 &= \mathsf{RMod.E}(1, 0, r_F) \\ \mathsf{evk}_{I,i} &= \mathsf{RMod.E}(s_{I,i}, y_{I,i}, r_F) \text{ for } (I,i) \in \mathcal{I} \times [n_d]. \end{aligned}$$
     The oracle returns $\mathsf{sk}_F = \{\mathsf{evk}_0, \vec{\mathsf{evk}}_I = (\mathsf{evk}_{I,i})_{i\in[n_d]}\}$.
5. $\mathcal{B}_{\mathsf{LDE}}$ gives $\{\vec{c}_t\}_{t\in[T]}$ to $\mathcal{A}$.
6. $\mathcal{A}$ returns a bit b' to $\mathcal{B}_{\mathsf{LDE}}$, then $\mathcal{B}_{\mathsf{LDE}}$ also returns b' to $\mathcal{C}'$.

In the above game, $\mathcal{A}$ has the roll same as $\mathbf{Game}_3$. Thus if $\mathcal{A}$ returns the right answer $b' = b$, then $\mathcal{B}$ also wins the T-IND game for LDE. Therefore the advantage of $\mathcal{B}_{\mathsf{LDE}}$ in T-IND game for LDE is greater than $\mathcal{A}$ in $\mathbf{Game}_3$. $\square$

Conclusion of Proof By the Lemma 1, 2, 3, and 4, we can conclude that for any adversary $\mathcal{A}$, there exists adversaries $\mathcal{B}_{\mathsf{RMod}}$ and $\mathcal{B}_{\mathsf{LDE}}$ such that
$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE}-T\text{-SEL-IND-FP}} \leq \mathsf{Adv}_{\mathcal{B}_{\mathsf{RMod}}}^{\mathsf{RMod}\text{-IND-CPA}} + \mathsf{Adv}_{\mathcal{B}_{\mathsf{LDE}}}^{\mathsf{LDE}-T\text{-IND}}.$$
This result completes the proof of Theorem 1.

4 Concrete Schemes

In this section, we provide concrete constructions of functional encryption built upon the framework of the previous section. In particular, we use a RLWE-based somewhat homomorphic encryption (SHE) [11] as LDE scheme, and ElG encryption as RMod scheme. The use of RLWE-based SHE yields FE ciphertext length $2\cdot\ell+1$, and note that the ciphertext length is independent from the degree d. Although our construction can deal with every degree d, we here only consider the degree 3 for simplicity.

4.1 FE from RLWE and ElG

We give a detailed description of our FE scheme, which has message space $\mathcal{M}_{\mathsf{FE}} = R_2 = \mathbb{Z}_2[X]/\langle X^N+1\rangle$, and key space

$$\mathcal{K} = \{F : R_2^\ell \to R_2 : F(\vec{\boldsymbol{x}}) = \sum_{I\in\mathcal{I}} F_I\cdot \boldsymbol{x}^{(I)} \text{ for } \vec{\boldsymbol{x}}\in R_2^\ell, F_I\in\mathbb{Z}\}.$$

We also consider a cyclic group $\langle g\rangle$ of prime order p. For an element $\boldsymbol{a} = \sum_{i=0}^{N-1} a_i\cdot X^i \in R_p$, we denote $[\boldsymbol{a}]_g = (g^{a_0},\cdots,g^{a_{N-1}})\in G^N$. With this notation, G^N becomes a R_p-module equipped with the following operations.

- (Addition) $\oplus : G^N\times G^N\to G^N$ by $[\boldsymbol{a}]_g\oplus[\boldsymbol{b}]_g := [\boldsymbol{a}+\boldsymbol{b}]_g$
- (Scalar multiplication) $\odot : R\times G^N\to G^N$ by $\boldsymbol{b}\odot[\boldsymbol{a}]_g := [\boldsymbol{b}\cdot\boldsymbol{a}]_g$.

The RLWE Scheme We specify RLWE-based encryption scheme, which is used for LDE. It is defined over the following spaces.

- The message space R_2,
- The secret key space R_p^*,
- The ciphertext space R_p^*.

Then, RLWE-based scheme consists of algorithms:

$$(\mathsf{RLWE.Param}, \mathsf{RLWE.SkGen}, \mathsf{RLWE.Enc}, \mathsf{RLWE.Mult}, \mathsf{RLWE.Dec}).$$

- $\mathsf{RLWE.Param}(1^\lambda)$: Output public parameters $\mathsf{pp} = \{p, N, \mathcal{D}_{\mathbb{Z},\alpha p}\}$ where $\mathcal{D}_{\mathbb{Z},\alpha p}$ is an error distribution.
- $\mathsf{RLWE.SkGen}(\mathsf{pp})$: Sample a random $\mathsf{sk}\in R$, and let $\vec{\mathsf{sk}} = (1,\mathsf{sk})\in R^2$.
- $\mathsf{RLWE.Enc}(\vec{\mathsf{sk}}, \boldsymbol{m}, \mathsf{pp})$: Sample a random polynomial $\boldsymbol{a}\in R_p$, and an error polynomial $\boldsymbol{e}\in R$ having each coefficients sampled from $\mathcal{D}_{\mathbb{Z},\alpha p}$. Return ciphertext
$$\vec{\boldsymbol{c}} = (-\boldsymbol{a}\cdot\boldsymbol{sk}+\boldsymbol{m}+2\boldsymbol{e},\boldsymbol{a})\in R^2.$$
- $\mathsf{RLWE.Dec}(\vec{\mathsf{sk}} = (1,\boldsymbol{sk}), \vec{\boldsymbol{c}}\in R^n, \mathsf{pp})$: Compute
$$\boldsymbol{\mu} := \langle\vec{\boldsymbol{c}}, (1,\boldsymbol{sk},\cdots,\boldsymbol{sk}^{n-1})\rangle\in R$$
and return $\bar{\boldsymbol{\mu}} \bmod 2\in\mathcal{M}_{\mathsf{LDE}}$.

– $\mathsf{RLWE.Mult}(\vec{\boldsymbol{c}} \in R_p^{n_1}, \vec{\boldsymbol{c}}' \in R_p^{n_2}, \mathsf{pp})$: Write $\vec{\boldsymbol{c}} = (\boldsymbol{c}_1, \cdots, \boldsymbol{c}_{n_1})$ and $\vec{\boldsymbol{c}}' = (\boldsymbol{c}'_1, \cdots \boldsymbol{c}'_{n_2})$. Output

$$\vec{\boldsymbol{c}}_{\mathsf{Mult}} = \left(\sum_{i+j=k} \boldsymbol{c}_i \cdot \boldsymbol{c}'_j \right)_{2 \le k \le n_1+n_2} \in R_p^{n_1+n_2-1}.$$

One can check that RLWE scheme can be viewed as LDE scheme, by defining $\mathsf{RLWE.Ecd}(\boldsymbol{m})$ to return $\boldsymbol{m} + 2\boldsymbol{e}$ with $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z},\alpha p}$ and $\boldsymbol{\mu} \mod 2 \leftarrow \mathsf{RLWE.Dcd}(\boldsymbol{\mu})$.

ElG Encryption Scheme The ElG encryption scheme is defined over the following spaces.

– The message space R_p
– The secret key space R_p
– The randomness space R_p
– The ciphertext space G^N

Then, the ElG encryption scheme consists of algorithms.

$$(\mathsf{ElG.Param}, \mathsf{ElG.SkGen}, \mathsf{ElG.Enc}, \mathsf{ElG.Dec}),$$

which are described as follows.

– $\mathsf{ElG.Param}(1^\lambda)$: Generate a group G of order p and its generator g. Output public parameter $\mathsf{pp} = \{g, p, N\}$.
– $\mathsf{ElG.SkGen}(\mathsf{pp})$: Given public parameter pp, sample $s \leftarrow R_p$. Output a secret key $\mathsf{sk} = s = \sum_{i=0}^{N} s_i \cdot X^i$.
– $\mathsf{ElG.Enc}(\mathsf{sk}, \boldsymbol{m} = \sum_{i=0}^{N} m_i \cdot X^i \in R_p)$: Sample an integer $\boldsymbol{r} \leftarrow R_p$ Compute $c_0 = [\boldsymbol{r}]_g$ and $\vec{c} = (c_i)$ with $c_i = [\boldsymbol{r} \cdot s_i + m_i]_g$, and output $(\vec{c}_0, \vec{c}) \in G^N \times G^N$.
– $\mathsf{ElG.Dec}(\mathsf{sk}, (\vec{c}_0, \vec{c}))$: Compute $\vec{c} - \boldsymbol{s} \odot \vec{c}_0 = [\boldsymbol{m}']_g \in G^N$. Then the i-th entry of the vector is of the form $[m'_i]_g \in G$. Then recover m'_i by solving discrete logarithm with a pair $([m'_i]_g, g)$ for each $i \in [N]$. Finally output $\sum_{i=0}^{N-1} m'_i \cdot X^i$

Now we argue that the ElG scheme can be understood as RMod scheme where the R_p module M is G^N. First, by defining $\mathsf{ElG.E}(\boldsymbol{s}, \boldsymbol{m}, \boldsymbol{r}) \coloneqq [\boldsymbol{r} \cdot \boldsymbol{s} + \boldsymbol{m}]_g$, we can understand $(\vec{c}_0, \vec{c}) \in G^N \times G^N$ by

$$(\mathsf{ElG.E}(\boldsymbol{1}, \boldsymbol{0}; \boldsymbol{r}), \mathsf{ElG.E}(\boldsymbol{s}, \boldsymbol{m}, \boldsymbol{r}))$$

. We then see that, the ElG scheme supports R_p module operations. Indeed, we have

$$\begin{aligned}
\mathsf{ElG.E}(\mathsf{sk}, \boldsymbol{m}, \boldsymbol{r}) \oplus \mathsf{ElG.E}(\mathsf{sk}', \boldsymbol{m}', \boldsymbol{r}) &= ([\boldsymbol{r} \cdot \mathsf{sk}_i + m_i]_g) \oplus ([\boldsymbol{r} \cdot \mathsf{sk}'_i + m'_i]_g) \\
&= ([\boldsymbol{r} \cdot (\mathsf{sk}_i + \mathsf{sk}'_i) + (m_i + m'_i)]_g) \\
&= \mathsf{ElG.E}(\mathsf{sk} + \mathsf{sk}', \boldsymbol{m} + \boldsymbol{m}', \boldsymbol{r}), \\
\boldsymbol{b} \odot \mathsf{ElG.E.}(\mathsf{sk}, \boldsymbol{m}, \boldsymbol{r}) &= \boldsymbol{b} \odot ([\boldsymbol{r} \cdot \mathsf{sk}_i + \boldsymbol{m}_i]_g) \\
&= ([\boldsymbol{r} \cdot \boldsymbol{b} \cdot \mathsf{sk}_i + \boldsymbol{b} \cdot m_i]_g) \\
&= \mathsf{ElG.E}(\boldsymbol{b} \cdot \mathsf{sk}, \boldsymbol{b} \cdot m, \boldsymbol{r}).
\end{aligned}$$

Remark. In the original ElG scheme, both the message space and the secret space are defined over $\mathbb{Z}_p$. Instead, we use the variant of ElG scheme defined over R_p. This modified ElG scheme has a natural reduction from the existing ElG scheme and still satisfies the semantic security. The reduction can be found in the appendix C.

Functional Encryption for Degree-3 Polynomial Now we obtain a concrete functional encryption by substituting LDE into RLWE, and RMod into ElG in the abstract FE scheme of the previous section.

We denote the message vector length ℓ. Note that since we use degree 3 and RLWE scheme having ciphertext in R_p^2, we know $\#(\mathcal{I})$ and n_d is less than ℓ^3 and smaller than $4 \cdot \ell^3$, respectively, in the RLWE and ElG scheme. Our hybrid functional encryption scheme

$$\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$$

with $\mathsf{pp} = \{p, N, \ell, B\}$ is defined as follows:

- $\mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$:
  1. Run $\mathsf{RLWE.skGen}(\mathsf{pp}, N)$ to get secret key $\vec{\mathsf{sk}} = (1, \boldsymbol{sk})$.
  2. For each $(I, i) \in \mathcal{I} \times [4]$, run $\mathsf{ElG.skGen}(\mathsf{pp}, N)$ to get a set of secret keys $\boldsymbol{s}_{I,i}$ and set $\vec{\boldsymbol{s}}_I := (\boldsymbol{s}_{I,1}, \boldsymbol{s}_{I,2}, \boldsymbol{s}_{I,3}, \boldsymbol{s}_{I,4})$
  3. Output $\mathsf{msk} = \{\vec{\mathsf{sk}}, \{\vec{\boldsymbol{s}}_I\}_{I \in \mathcal{I}}\}$.

- $\mathsf{FE.KeyGen}(\mathsf{msk}, F, \mathsf{pp})$:
  1. Choose $\boldsymbol{r} \leftarrow R_p$ and compute $\mathsf{evk}_0 = [\boldsymbol{r}]_g \in G^N$
  2. For each $(I, i) \in \mathcal{I} \times [4]$, set $\mathsf{evk}_{I,i} = [\boldsymbol{r} \cdot \boldsymbol{s}_{I,i} + F_I \cdot \boldsymbol{sk}^{i-1}]_g \in G^N$ and set $\vec{\mathsf{evk}}_I := (\mathsf{evk}_{I,1}, \mathsf{evk}_{I,2}, \mathsf{evk}_{I,3}, \mathsf{evk}_{I,4}) \in G^{N \times 4}$.
  3. Output $\{\mathsf{evk}_0, \{\vec{\mathsf{evk}}_I\}_{I \in \mathcal{I}}\}$.

- $\mathsf{FE.Enc}(\mathsf{msk}, \vec{\boldsymbol{m}} = (\boldsymbol{m}_i)_{i \in [\ell]}, \mathsf{pp})$:
  1. Sample $\boldsymbol{t}_{i,0} \leftarrow R_p$ and $\boldsymbol{e}_i \in R_p$ having coefficients sampled from $D_{\mathbb{Z},\alpha p}$. Set
  $$\vec{\boldsymbol{c}}_i = (\boldsymbol{t}_{i,1}, \boldsymbol{t}_{i,0}) = (\boldsymbol{t}_{i,0}, -\boldsymbol{t}_{i,0} \cdot \boldsymbol{sk} + \boldsymbol{m}_i + 2\boldsymbol{e}_i) \in R_p^2$$
  for $i \in [\ell]$ and let $\vec{\boldsymbol{c}} = (\vec{\boldsymbol{c}}_1, \cdots, \vec{\boldsymbol{c}}_\ell)$.
  2. For each $I \in \mathcal{I}$, compute a vector
  $$\vec{\boldsymbol{C}}_{I=(i,j,k)} = (\boldsymbol{C}_{3,(i,j,k)}, \boldsymbol{C}_{2,(i,j,k)}, \boldsymbol{C}_{1,(i,j,k)}, \boldsymbol{C}_{0,(i,j,k)}) \in R_p^4$$
  with
  $$\begin{aligned}
  \boldsymbol{C}_{3,(i,j,k)} &= \boldsymbol{t}_{i,1} \cdot \boldsymbol{t}_{j,1} \cdot \boldsymbol{t}_{k,1} \\
  \boldsymbol{C}_{2,(i,j,k)} &= \boldsymbol{t}_{i,0} \cdot \boldsymbol{t}_{j,1} \cdot \boldsymbol{t}_{k,1} + \boldsymbol{t}_{i,1} \cdot \boldsymbol{t}_{j,0} \cdot \boldsymbol{t}_{k,1} + \boldsymbol{t}_{i,1} \cdot \boldsymbol{t}_{j,1} \cdot \boldsymbol{t}_{k,0} \\
  \boldsymbol{C}_{1,(i,j,k)} &= \boldsymbol{t}_{i,1} \cdot \boldsymbol{t}_{j,0} \cdot \boldsymbol{t}_{k,0} + \boldsymbol{t}_{i,0} \cdot \boldsymbol{t}_{j,1} \cdot \boldsymbol{t}_{k,0} + \boldsymbol{t}_{i,0} \cdot \boldsymbol{t}_{j,0} \cdot \boldsymbol{t}_{k,1} \\
  \boldsymbol{C}_{0,(i,j,k)} &= \boldsymbol{t}_{i,0} \cdot \boldsymbol{t}_{j,0} \cdot \boldsymbol{t}_{k,0}
  \end{aligned}$$
  and evaluate
  $$\boldsymbol{c}_0 = \sum_{I \in \mathcal{I}} \langle \vec{\boldsymbol{s}}_I, \vec{\boldsymbol{C}}_I \rangle.$$

3. Output $\mathsf{Ct} = \{\boldsymbol{c}_0, \vec{\boldsymbol{c}}\}$.

– $\mathsf{Dec}(\mathsf{sk}_F, \mathsf{Ct}, \mathsf{pp})$:
1. Compute a vector $\vec{\boldsymbol{C}}_I$ from Ct.
2. Compute
$$\vec{C} = \bigoplus_{I \in \mathcal{I}} \langle \otimes_{i \in I} \vec{\boldsymbol{c}}_i, \vec{\mathbf{evk}}_I \rangle \in G^N.$$
3. Compute $\vec{C} - \boldsymbol{c}_0 \odot \mathsf{evk}_0 = [\boldsymbol{\mu}]_g \in G^N$. Then recover μ by solving discrete logarithm, and output $\mu \mod 2$.

In the decryption process, one has to solve discrete logarithm of $[\boldsymbol{\mu}]_g$. Note that $\boldsymbol{\mu}$ would be $F(\vec{\boldsymbol{m}}) + 2\boldsymbol{e}_F$ for some error polynomial $\boldsymbol{e}_F$, and the following lemma gives one bound for $\|\boldsymbol{e}_F\|_\infty$.

Lemma 5. *Assuming the error distribution $\mathcal{D}_{\mathbb{Z},\alpha p}$ is bounded by T, we have $\mu = F(\vec{\boldsymbol{m}}) + 2\boldsymbol{e}_F \in R_p$ with $\|\boldsymbol{e}_F\|_\infty = O(3^{\log N+1} \cdot T^4)$.*

Thus we have to try about $O(3^{\log N+1} \cdot T^4)$ candidates for each coefficient, and therefore $\mathsf{FE.Dec}$ costs at most $O(N \cdot 3^{\log N+1} \cdot T^4)$ times.

5 Conclusion

We propose a functional encryption scheme with linear ciphertext size that supports degree d polynomials. Unfortunately, the proposed scheme cannot support arbitrarily large degree d, but this is enough to have some important consequences for current open problems.

Our main idea is to generically combine two abstract schemes LDE and RMod where LDE scheme has a decryption procedure consisting of inner product of ciphertext and secret key, and RMod scheme supports R-module operations among ciphertexts and a decryption with B-bounded noise. We also give a concrete construction of such abstract framework, by using a variant of SHE scheme for LDE scheme and ElG for RMod scheme.

It would be an interesting question to find another candidate for LDE or RMod schemes, which can yield more efficiency or larger degree capacity. For example, as a main drawback of our concrete scheme, we need to find a discrete logarithm for some group element in the decryption procedure. Although the exponent would be set small so that it can be solved in polynomial time, this is definitely not an easy task. Since this discrete logarithm comes from the decryption of ElG scheme used as RMod scheme, another scheme for RMod than ElG can speed up the decryption of the functional encryption.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Annual International Cryptology Conference*, pages 205–222. Springer, 2005.

2. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer, 2015.
3. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Annual Cryptology Conference*, pages 333–362. Springer, 2016.
4. M. Albrecht, S. Bai, and L. Ducas. A subfield lattice attack on overstretched ntru assumptions. In *Annual Cryptology Conference*, pages 153–178. Springer, 2016.
5. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Annual International Cryptology Conference*, pages 67–98. Springer, 2017.
6. A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 470–491. Springer, 2015.
7. D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 2004.
8. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer, 2004.
9. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.
10. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.
11. Z. Brakerski, V. Vaikuntanathan, and C. Gentry. Fully homomorphic encryption without bootstrapping. In *Proc. of ITCS'12*. Citeseer, 2012.
12. J. H. Cheon, P. Fouque, C. Lee, B. Minaud, and H. Ryu. Cryptanalysis of the new CLT multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 509–536, 2016.
13. J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 2015.
14. J. H. Cheon, J. Jeong, and C. Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.
15. J. H. Cheon and J. Kim. A hybrid scheme of public-key encryption and somewhat homomorphic encryption. *IEEE transactions on information forensics and security*, 10(5):1052–1063, 2015.
16. J. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Cryptanalysis of GGH15 multilinear maps. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 607–628, 2016.
17. J. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, pages 41–58, 2017.

18. J. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.
19. J. Coron, T. Lepoint, and M. Tibouchi. New multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 267–286, 2015.
20. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 1–17, 2013.
21. R. Gay. Functional encryption for quadratic functions, and applications to predicate encryption. Cryptology ePrint Archive, Report 2016/1106, 2016. https://eprint.iacr.org/2016/1106.
22. C. Gentry. Practical identity-based encryption without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 445–464. Springer, 2006.
23. C. Gentry, S. Gorbunov, and S. Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 498–527, 2015.
24. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.
25. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.
26. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.
27. Y. Hu and H. Jia. Cryptanalysis of ggh map. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 537–565. Springer, 2016.
28. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 146–162. Springer, 2008.
29. A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234. ACM, 2012.
30. E. Miles, A. Sahai, and M. Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Annual Cryptology Conference*, pages 629–658. Springer, 2016.
31. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.
32. A. Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

A Security Definition for Symmetric Key Encryption

In this section, we will define the indistinguishable security for chosen plaintext attack for symmetric key encryption. We note that we are focusing on the symmetric encryption because we construct secret key functional encryption, so we don't need the public key anymore, even if there already exists the public key in concrete scheme, such as RLWE, ElG, *etc.* For the symmetric key scheme Sym which consists of four algorithms $(\mathsf{Param}, \mathsf{SkGen}, \mathsf{Enc}, \mathsf{Dec})$, we define T-message IND game (in short, T-IND game) for an integer T as follows.

Definition 4 (T-IND Game for Symmetric Key Encryption). *There are two participants in T-IND game. An adversary $\mathcal{A}$ wants to show that he can break the T-IND security of the scheme and a challenger $\mathcal{C}$ wants to make sure the $\mathcal{A}$'s argument is correct. For each purpose, they proceed the following game.*

1. *$\mathcal{C}$ generates $\mathsf{pp} \leftarrow \mathsf{Param}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{SkGen}(\mathsf{pp})$ and gives pp to $\mathcal{A}$.*
2. *$\mathcal{A}$ chooses two set of messages $\vec{m}^0 = (m_1^0, \cdots, m_T^0)$ and $\vec{m}^1 = (m_1^1, \cdots, m_T^1)$. Then $\mathcal{A}$ gives $(\vec{m}^0, \vec{m}^1)$ to $\mathcal{C}$.*
3. *$\mathcal{C}$ chooses random bit $b \leftarrow \{0,1\}$ and gives $\vec{c} = (c_t = \mathsf{Enc}(\mathsf{sk}, m_t^b))_{t \in [T]}$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

We define the advantage of $\mathcal{A}$ in this game by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sym}-T\text{-IND}} = |\Pr[b = b'] - \frac{1}{2}|.$$

We say the symmetric key encryption scheme satisfies the T-IND security if there is no PPT adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sym}-T\text{-IND}}$ is non-negligible.

Recall that our LDE and RMod schemes have the same concept with symmetric key encryptions, so the T-IND security definition for LDE and RMod can be defined from Definition 4.

B Proof for Lemma 5

Recall that each entry of $\vec{\boldsymbol{c}}_i \in R_p^2$ satisfies $\langle \vec{\boldsymbol{c}}_i, \mathsf{sk} \rangle = \boldsymbol{m}_i + 2\boldsymbol{e}_i$, with $\|\boldsymbol{e}_i\|_\infty \le T$. Then for every $\vec{\boldsymbol{c}}_i \otimes \vec{\boldsymbol{c}}_j$, it holds that

$$\langle \vec{\boldsymbol{c}}_i \otimes \vec{\boldsymbol{c}}_j, \mathsf{sk} \otimes \mathsf{sk} \rangle = \boldsymbol{m}_1 \cdot \boldsymbol{m}_2 + 2\boldsymbol{e}_{\mathsf{Mult}},$$

with $\|\boldsymbol{e}_{\mathsf{Mult}}\|_\infty = O(2N \cdot T^2)$. We can inductively show that $\vec{\boldsymbol{c}}^{(I)} \in R_p^{d+1}$ has $O((2N)^{\log d} \cdot T^d)$ error size.

C Reduction from DDH to Ring-ElG

Note that for fixed group G of prime order p with generator g, DDH assumption is that distinguishing following two distributions is hard :

$$D_1 = \{(g^x, g^y, g^{xy}) : x, y \leftarrow \mathbb{Z}_p\}, D_2 = \{(g^x, g^y, g^{xy+r}) : x, y, r \leftarrow \mathbb{Z}_p\}.$$

We prove that IND-CPA security for Ring-ElG in our scheme defined as Section 4.1, by proving following theorem.

Theorem 2. *Assuming* DDH *assumption, ring-Elgamal scheme ring-*ElG *is* IND-CPA *secure.*

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\text{Ring-}\mathsf{ElG}\text{-IND-CPA}}$ is non-negligible. For a random bit $b \leftarrow \{0,1\}$, suppose $([x]_g, [y]_g, T)$ sampled from D_b is given. Then now we roll the challenger in IND-CPA game with $\mathcal{A}$. The game proceeds as follows:

1. From security parameter λ, we choose an integer N and give (g, p, N) to $\mathcal{A}$.
2. We randomly choose $\boldsymbol{g} = \sum_{i=0}^{N-1} g_i X^i \leftarrow R_p$.
3. For each message query $\boldsymbol{m} = \sum_{i=0}^{N-1} m_i X^i$ given by $\mathcal{A}$, we randomly choose $\boldsymbol{t} = \sum_{i=0}^{N-1} t_i X^i$ from $\mathbb{R}_p$ and compute $g_i' = \sum_{j+k=i} g_j t_k - \sum_{j+k=N+i} g_j t_k \mod p$ for $i = 0, \cdots, N-1$. Note that $\boldsymbol{g}(X)\boldsymbol{t}(X) = \sum_{i=0}^{N-1} g_i'(X)$ in R_p. Now return
$$c_{\boldsymbol{m},0} = \boldsymbol{t} \odot [y]_g,\ \vec{c}_{\boldsymbol{m}} = ((g_0' \odot T) \oplus [m_0]_g, (g_1' \odot T) \oplus [m_1]_g, \cdots, (g_{N-1}' \odot T) \oplus [m_{N-1}]_g)$$
to $\mathcal{A}$.
4. For ciphertext query, $\mathcal{A}$ gives $\boldsymbol{f}^0 = \sum_{i=0}^{N-1} f_i^0 X^i$ and $\boldsymbol{f}^1 = \sum_{i=0}^{N-1} f_i^1 X^i$ to us, then we randomly choose a bit $b' \leftarrow \{0,1\}$, $\boldsymbol{t}' = \sum_{i=0}^{N-1} t_i' X^i \leftarrow R_p$ and compute $g_i'' = \sum_{j+k=i} g_j t_k' - \sum_{j+k=N+i} g_j t_k' \mod p$ for $i = 0, \cdots, N-1$. Note that $\boldsymbol{g}(X)\boldsymbol{t}'(X) = \sum_{i=0}^{N-1} g_i''(X)$ in R_p. Now return
$$c_0 = \boldsymbol{t}' \odot [y]_g,\ \vec{c} = ((g_0'' \odot T) \oplus [f_0^b]_g, (g_1'' \odot T) \oplus [f_1^b]_g, \cdots, (g_{N-1}'' \odot T) \oplus [f_{N-1}^b]_g)$$
to $\mathcal{A}$. to $\mathcal{A}$.
5. $\mathcal{A}$ returns a bit b''.

We know that $T = [xy + r]_g$ for some r, where r is 0 or chosen from random in $\mathbb{Z}_p$. Also we can easily check that $(c_{\boldsymbol{m},0}, \vec{c}_{\boldsymbol{m}}) = ([y\boldsymbol{t}]_g, \mathsf{ElG.E}(x\boldsymbol{g}, \boldsymbol{m} + r\boldsymbol{g}, y\boldsymbol{t}))$ and $c_0, \vec{c} = ([y\boldsymbol{t}']_g, \mathsf{ElG.E}(x\boldsymbol{g}, \boldsymbol{f}^b + r\boldsymbol{g}; y\boldsymbol{t}'))$. Since $\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{g}$ are randomly chosen, $\mathcal{A}$ can return the valid output.

If $r = 0$, then $\mathcal{A}$ will return $b'' = b'$ with advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ElG}\text{-IND-CPA}}$. Other if r is random, then $\mathcal{A}$ can know only the information of $\boldsymbol{f}^b + r\boldsymbol{g}$, which tells no information about $\boldsymbol{f}^b$. Thus $\mathcal{A}$ will return $b'' = b'$ with advantage 0. Therefore, we choose $b = 1$ if $b'' = b'$ and $b = 0$ otherwise, then our advantage for DDH is the half of $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ElG}\text{-IND-CPA}}$, which is non-negligible. This contradicts to our assumption.

D Another variant of RLWE

One can preserve the ciphertext length in RLWE scheme, by publishing an additional key Switch that switches $\vec{c}_{\mathsf{Mult}} \in R_p^3$ into $\vec{c}'_{\mathsf{Mult}} \in R_p^2$. To be precise, the Setup and Mult process are changed as following:

– $\mathsf{Setup}'(1^\lambda, \mathsf{pp})$: Sample a random $\boldsymbol{s} \in R$, and let $\mathsf{sk} = (\boldsymbol{s}, 1) \in R_p^2$. Publish

$$\mathsf{Switch} = \{\mathsf{Switch}_i := \mathsf{RLWE.Enc}(\mathsf{sk}, 2^i \cdot \boldsymbol{s}^2)\}.$$

– $\mathsf{Mult}'(\vec{c}, \vec{c}' \in R_p^2, \mathsf{Switch})$: Write $\vec{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ and $\vec{c}' = (\boldsymbol{c}'_1, \boldsymbol{c}'_2)$. Decompose $\boldsymbol{c}_1 \cdot \boldsymbol{c}'_1 \in R_p$ by $\sum_i = 2^i \cdot d_i$ with $d_i \in R_2$. Output

$$\vec{c}'_{\mathsf{Mult}} = (\boldsymbol{c}_1 \cdot \boldsymbol{c}'_2 + \boldsymbol{c}'_1 \cdot \boldsymbol{c}_2,\ \boldsymbol{c}_2 \cdot \boldsymbol{c}'_2) + \sum_i \mathsf{Switch}_i \cdot d_i.$$

To build a functional encryption with this RLWE version, we need to add Switch to $\mathsf{FE.Enc}(\mathsf{msk}, \vec{m})$. In other words, the ciphertext would contains Switch additional to ciphertext $\vec{c}$. However, in this case, we have to assume *circular security*, since the additional public information Switch is a sort of encryption of sk by sk.

In the original RLWE scheme, it can avoid the circular security by assuming many secret key $\mathsf{sk}_k = (1, \boldsymbol{s}_k)$, and providing many

$$\mathsf{Switch}^{(k)} = \{\mathsf{Switch}_i^{(k)} := \mathsf{RLWE.Enc}(\mathsf{sk}_{k+1}, 2^i \cdot \boldsymbol{s}_k^2)\}.$$

Then, given two ciphertexts encrypted with sk_k, we have a ciphertext $\vec{c}_{\mathsf{Mult}}$ encrypted with sk_{k+1}. However, in this case, the ciphertext CT in our FE scheme gets longer along with degree d, which is undesirable for our goal.

E Another candidate for LDE : LTV SHE scheme [29]

As another LDE encryption candidate, we introduce hybrid functional encryption using the LTV scheme [29].

– $\mathsf{Setup}(1^\lambda, \mathsf{pp})$: Sample a polynomial $\boldsymbol{f} \in R$ whose each coefficient is sampled from $\mathcal{D}_{\mathbb{Z},\alpha p}$, and let $\mathsf{sk} = 2\boldsymbol{f} + 1 \in R_p$. Repeat until sk is invertible in R_p.
– $\mathsf{Enc}(\mathsf{sk}, \boldsymbol{m} \in R_2)$: Sample a polynomial $\boldsymbol{g} \in R$ whose coefficients are sampled from $\mathcal{D}_{\mathbb{Z},\alpha p}$. Return ciphertext $\boldsymbol{c} = (2\boldsymbol{g} + 1) \cdot \mathsf{sk}^{-1} + 2\boldsymbol{e} + \boldsymbol{m} \in R_p$.
– $\mathsf{Dec}(\mathsf{sk}', \boldsymbol{c})$: Compute $\bar{\boldsymbol{m}} = \boldsymbol{c} \cdot \mathsf{sk}' \in R_p$, and return $\boldsymbol{m} = \bar{\boldsymbol{m}} \in R_2$.
– $\mathsf{Mult}(\boldsymbol{c}_1, \boldsymbol{c}_2)$: Output $\boldsymbol{c}_{\mathsf{Mult}} = \boldsymbol{c}_1 \cdot \boldsymbol{c}_2 \in R_p$. The corresponding $\mathsf{sk}_{\mathsf{Mult}}$ would be $\mathsf{sk}_{\mathsf{Mult}} = \mathsf{sk}_1 \cdot \mathsf{sk}_2 \in R_p$.

In LTV scheme, the ciphertext and secret key are always elements in R_p, different from RLWE scheme that having ciphertext and secret key in R_p^2, and it leads to FE ciphertext length $\ell + 1$, less than $2\ell + 1$ in RLWE.

However, In 2015, concurrently and independently Albrecht *et al.* and Cheon *et al.* proposed new attacks, called subfield attack [4,14]. It allows to find a secret key of the LTV scheme in subexponential time. As a result, the parameter of ring LWE-based scheme is set more efficiently than that of the LTV scheme in terms of the same security parameters.