From Quadratic Functions to Polynomials: Generic Functional Encryption from Standard Assumptions

Linru Zhang[1], Yuechen Chen[1], Jun Zhang[2], Meiqi He[1], and Siu-Ming Yiu[1(✉)]

[1] Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong SAR, China
{lrzhang,ycchen,mqhe,smyiu}@cs.hku.hk
[2] Educational Technology Department, Shenzhen University, Shenzhen, Guangdong, China
zhjun@connect.hku.hk

Abstract. The "all-or-nothing" notion of traditional public-key encryptions is found to be insufficient for many emerging applications in which users are only allowed to obtain a functional value of the ciphertext without any other information about the ciphertext. Functional encryption was proposed to address this issue. However, existing functional encryption schemes for generic circuits either have bounded collusions or rely on not well studied assumptions. Recently, Abdalla *et al.* started a new line of work that focuses on specific functions and well-known standard assumptions. Several efficient schemes were proposed for inner-product and quadratic functions. There are still a lot of unsolved problems in this direction, in particular, whether a generic FE scheme can be constructed for quadratic functions and even higher degree polynomials. In this paper, we provide affirmative answers to these questions. First, we show an IND-secure generic functional encryption scheme against adaptive adversary for quadratic functions from standard assumptions. Second, we show how to build a functional encryption scheme for cubic functions (the first in the literature in public-key setting) from a functional encryption scheme for quadratic functions. Finally, we give a generalized method that transforms an IND-secure functional encryption scheme for degree-m polynomials to an IND-secure functional encryption scheme for degree-$(m+1)$ polynomials.

1 Introduction

Background. Traditional public-key encryption (PKE) allows a user who owns a secret key sk to decrypt a ciphertext CT encrypted with a public key pk. The decryption result is the plaintext of CT if sk matches pk, or nothing otherwise. For many emerging applications, for example, a owner may store her encrypted data in cloud and allow different users to query different functional values of the plaintext without revealing the plaintext, this all-or-nothing concept is insufficient. Functional encryption (FE) was proposed to address this issue, which

C. Carlet et al. (Eds.): C2SI 2019, LNCS 11445, pp. 142–167, 2019.
https://doi.org/10.1007/978-3-030-16458-4_10

enables users to obtain a functional value of a plaintext without any other information about the plaintext. In general, consider a functional encryption scheme for a functionality $F(k, x)$, where $k \in K$ (the key space) and $x \in X$ (the plaintext space). The authority with the master key can generate secret keys sk_k for values k. Given a ciphertext of x, the key holder of sk_k can only learn $F(k, x)$ and nothing else except possibly the length of x.

Before FE was formally defined in [11,26], there were a lot of schemes proposed to overcome the "all-or-nothing" barrier of the traditional public-key encryption. These schemes, including identity-based encryption (IBE) [8,9,28], attribute-based encryption (ABE) [27], searchable encryption [1], and predicate encryption [12,23], are considered as special cases of FE[1].

While there are many exciting results in these special cases, designing FE schemes seems to be more difficult. Existing FE schemes that work for arbitrary circuits either have bounded collusions [21,22], or have to rely on powerful, but impractical and not well understood assumptions (indistinguishable obfuscation (IO) and its variants, or polynomial hardness of simple assumptions on multi-linear maps) [13,19,20,31]. Attacks were identified for some constructions that are based on IO and multi-linear map [5,14,15,17].

A Remark on Security Definition. Unlike traditional PKE, [26] showed that simulation-based security (SIM-security) is not always achievable for FE. Indistinguishability-based security (IND-security) is widely used in FE research. We also focus on IND-secure FE schemes. Roughly speaking, IND-security states that the adversary who has the secret keys for functions $f_1, \ldots, f_n$ cannot distinguish which of the challenge messages m_0 or m_1 has been encrypted, under the condition that $f_i(m_0) = f_i(m_1)$, $i \in [1..n]$.

Functional Encryption from Standard Assumptions. Recently, instead of focusing on generic functions, researchers started to design efficient schemes for specific functions using well studied standard assumptions. Abdalla *et al.* [2] started this line of work by proposing an IND-secure (against selective adversary) functional encryption for inner product (IPFE) based on the decisional Diffie-Hellman assumption. Precisely, given an encrypted vector $\boldsymbol{x}$ in the message space X and a key sk_y based on vector $\boldsymbol{y}$ in the key space K, the decryption algorithm will output the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ without revealing any other information about x except the length of it. [4] improved the framework of [2] to achieve IND-security against adaptive adversary, also from standard assumptions. A generic (i.e., one that can instantiate from any PKE scheme) construction of IPFE is given in [3]. The scheme is IND-secure against adaptive adversary.

[1] Some classify FE schemes into public index schemes and private index schemes based on the definition of predicate encryption, in which, the message x consists of two parts $(\mathcal{I}, m)$, where $\mathcal{I}$ is an index (e.g. a set of attributes) and m is the actual message. If $\mathcal{I}$ is publicly revealed by the ciphertext and only m is hidden, the corresponding scheme is referred as public index FE, which is commonly known as attributed-based encryption. The scheme is called private index scheme if both $\mathcal{I}$ and m are hidden.

The next step from linear functionality (inner product) is to consider quadratic functions. Note that if one does not care the size of a ciphertext, it is easy to have a generic FE scheme for quadratic function using an inner product scheme as illustrated by the following example. Let $f(x) = 2{x_1}^2 + 3{x_2}^2$. We can encrypt every pair of x_i, x_j to obtain $\boldsymbol{x} = ({x_1}^2, x_1x_2, x_2x_1, {x_2}^2)$. With the vector $\boldsymbol{y} = (2, 0, 0, 3)$, we can easily compute f as the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. However, the size of the ciphertext will be $O(n^2)$. Earlier this year, two FE schemes with linear size of ciphertext were proposed in [6]. One is IND-secure against selective adversary based on standard assumptions and the other is IND-secure against adaptive adversary in the generic group model. However, both schemes are not generic and cannot instantiate from any PKE scheme. The question whether it is possible to design a *generic* FE scheme for quadratic functions with linear size ciphertext is still open. And the same question applies to higher finite-degree polynomials[2]. Besides theoretical interest, there are real applications for function encryption for polynomials. For example, cubic functions can be used to calculate volumes; the distance d between two points $\boldsymbol{x}, \boldsymbol{y}$ in L^p space is defined as a p-degree polynomial: $d^p = |x_1 - y_1|^p + |x_2 - y_2|^p + \ldots + |x_n - y_n|^p$ with applications in data mining for high-dimensional data points; and in statistics, measures of central tendency and statistical dispersion, such as mean, median, and standard deviation, are also defined in terms of L^p metrics.

1.1 Our Contributions

In this paper, we provide affirmative answers to the above questions. We focus on the polynomial functionality over Z_p. We list our contributions as follows.

(1) We propose the first generic FE scheme for quadratic functions $LQFE = (Setup, Encrypt, KeyGen, Decrypt)$ with linear-size ciphertexts in the public-key setting. $LQFE$ is proved to be IND-secure against adaptive adversary. Generic functional encryption (proposed in [3]) means that such FE scheme can be instantiated from any PKE scheme with some properties.
(2) We derive a generalized method that transforms an IND-secure degree-m polynomial FE scheme to an IND-secure against selective adversary degree-$(m + 1)$ polynomial FE scheme. We illustrate our method based on our quadratic FE scheme to derive the first FE scheme for cubic functions, $CFE = (Setup, Encrypt, KeyGen, Decrypt)$, in the public-key setting having linear size ciphertext under standard assumptions. Actually, any FE scheme for quadratic functions can be used to build a FE scheme for cubic functions in our construction. For example, [6] can be used in CFE, but then the resulting cubic FE scheme is not generic.

[2] Very recently (in June, 2018), [16] provides a polynomial functional encryption scheme with linear ciphertext size. Their scheme is in private-key setting while our scheme is in public-key setting.

1.2 Overview of Our Techniques

In this section, we highlight some of the core ideas underlying our schemes. The details of our schemes will be given in later sections.

Our FE Scheme for Quadratic Functions over Z_p. Our construction is a generic construction, i.e., any public-key encryption scheme that has some structural and homomorphic properties can be used to instantiate it. These properties are similar to the requirements in [3]. Our scheme is efficient in communication and storage size: public keys and ciphertexts are both linear in the size of the encrypted vectors.

To simplify the notation, we may omit the security parameter k in the expression, i.e., instead of writing $f(k, x)$, we may just write $f(x)$ if it is clear from its context. A quadratic function $f(\boldsymbol{x})$ can be represented[3] as $f(\boldsymbol{x}) = \boldsymbol{x}^T F \boldsymbol{x}$, where $F \in Z_p^{(n+1)\times(n+1)}$ is a matrix with elements $f_{i,j}$, $\boldsymbol{x}$ is a column vector $(x_0, \ldots, x_n) \in Z_p^{n+1}$ and $x_0 = 1$. For our previous example, $f(x) = 2{x_1}^2 + 3{x_2}^2$, we can have a column vector $\boldsymbol{x} = (1, x_1, x_2)$ and $F = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$. The input to our scheme is a ciphertext of a vector $\boldsymbol{x}$, and decryption allows one to obtain $\boldsymbol{x}^T F \boldsymbol{x}$ with the given matrix F.

Our construction works over symmetric bilinear groups G_1, G_2 and G_T and two bilinear maps $e_1(g_1, g_1) = g_T : G_1 \times G_1 \to G_T, e_2(g_2, g_2) = g_T : G_2 \times G_2 \to G_T$, where g_1, g_2 and g_T are generators of G_1, G_2 and G_T. The order of G_1, G_2 and G_T is a prime p. The prime order ensures the existence of g_2 (g_1 can be any generator of G_1). The initial idea of the construction is to encrypt each $x_i, i \in \{0, 1, \ldots, n\}$ (denoted by $[n]$ in the rest of the paper) by using the selected PKE scheme ε under certain public key PK_i in group G_1. Then, we can get $\varepsilon.Enc(f_{i,j} x_i x_j)$ with public key PK'_{ij} in group G_T which depends on PK_i and PK_j by computing $e(ct_{\boldsymbol{x},i}, ct_{\boldsymbol{x},j})^{f_{i,j}}$, where $ct_{\boldsymbol{x},i}$ denotes the encrypted x_i. After summing them up and decrypting, we can obtain $f(\boldsymbol{x}) = \sum_{i,j=0}^{n} f_{i,j} x_i x_j = \boldsymbol{x}^T F \boldsymbol{x}$.

However, the result of $e(ct_{\boldsymbol{x},i}, ct_{\boldsymbol{x},j})^{f_{i,j}}$ includes not only $\varepsilon.Enc(f_{i,j} x_i x_j)$ in G_T, but also some noisy terms in G_T. The challenge is to carefully design the secret keys and ciphertexts to eliminate these noisy terms while guaranteeing the security.

Notice that the bilinear maps $e_1(\cdot, \cdot)$ and $e_2(\cdot, \cdot)$ are public. If the adversary takes the ciphertexts generated by g_1 (or g_2) as input of $e_1(\cdot, \cdot)$ (or $e_2(\cdot, \cdot)$), it can get new ciphertexts in G_T which are generated by g_T (denote as ct^*). If some parts of ciphertexts in the encryption algorithm are also generated by g_T, then combining such parts of ciphertexts and ct^* would leak information. To avoid this attack, we use the trick that when we need to encrypt something in group G_T, instead of using g_T based public key, we use the public key that is

[3] Note that [6] uses a slightly more general representation with two vectors: $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T F \boldsymbol{y}$.

based on g_T^q and q is kept secret. And when encrypting vectors in group G_1 (or G_2), we still use g_1 (or g_2) based public key. Therefore, without knowing q, the adversary cannot convert these ciphertexts in G_1 and G_2 to the new ciphertexts which are generated by g_T^q based public key. And the bilinear maps $e_1(\cdot,\cdot)$ and $e_2(\cdot,\cdot)$ cannot help the adversary any more.

The details of this construction can be found in Sect. 3.

Our FE Scheme for Cubic Functions over Z_p and Its Generalization. Our cubic FE scheme is based on any FE scheme for quadratic functions. When building from our generic scheme for quadratic functions, the cubic FE scheme is also generic. The scheme is also efficient: public keys and ciphertexts are both linear in the size of the encrypted vectors.

Similarly, a cubic functionality can be represented as $f(\boldsymbol{x}) = \sum_{i,j,k\in[n]} f_{i,j,k}x_ix_jx_k = \boldsymbol{x}^T \begin{pmatrix} \boldsymbol{x}^T A_0 \boldsymbol{x} \\ \boldsymbol{x}^T A_1 \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^T A_n \boldsymbol{x} \end{pmatrix}$, where $A_0, \ldots, A_n \in Z_p^{(n+1)\times(n+1)}$, $\boldsymbol{x} = (x_0, \ldots, x_n) \in Z_p^{n+1}$ and $x_0 = 1$. After encrypting a vector $\boldsymbol{x}$, the decryption of our scheme is expected to output $\sum_{i,j,k\in[n]} f_{i,j,k}x_ix_jx_k = \sum_{i\in[n]}(x_i\boldsymbol{x}^T A_i \boldsymbol{x})$ with the given coefficients $\{f_{i,j,k}\}_{i,j,k\in[n]}$. Note that there is a requirement for this representation to work and this requirement is easy to satisfy, see Sect. 4.1 for more details.

The initial idea of the construction is to divide the ciphertexts into two parts. The first part will look like $(r_i, t_i^{-1}x_i)W^{-1}$ and the second part will look like $W(a_i, QFE.Enc(x_i))^T$, where $W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \in Z_p^{2\times 2}, w_{12} = u_1^2, w_{22} = u_2^2$ is an invertible matrix and $\boldsymbol{r}, \boldsymbol{a}$ are random vectors in Z_p^{n+1}. The second part can be further divided into two parts, one is $w_{11}a_i, w_{21}a_i$, and the other is $QFE.Enc(u_1\boldsymbol{x}), QFE.Enc(u_2\boldsymbol{x})$. For decryption, we call $QFE.Dec(sk_{t_iA_i}, ct)$ to get $w_{12}t_i\boldsymbol{x}^T A_i\boldsymbol{x}$ and $w_{22}t_i\boldsymbol{x}^T A_i\boldsymbol{x}$. The second part becomes $W(a_i, t_i\boldsymbol{x}^T A_i\boldsymbol{x})$. We can multiply both parts of ciphertexts and get $a_ir_i + x_i\boldsymbol{x}^T A_i\boldsymbol{x}$. At last, we sum these up and minus $\sum_{i=0}^n a_ir_i$ to get the final value $f(\boldsymbol{x})$. If we have a FE scheme for degree-m polynomial and use degree-m polynomial $f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})$ instead of matrices in this construction, then we can get a new degree-$(m+1)$ polynomial FE scheme, i.e., this construction can be generalized.

The details of this construction can be found in Sects. 4 and 5.

2 Preliminaries

In this section, we recall some basic definitions that we will use in the remaining sections.

2.1 Bilinear Map

Here we review some facts related to bilinear groups with efficiently computable bilinear maps in [30].

Let G and G_T be two multiplicative cyclic groups of prime order p. Let g be a generator of G and e be a bilinear map, $e : G \times G \to G_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in G$ and $a, b \in Z_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that G is a bilinear group if the group operation in G and the bilinear map $e : G \times G \to G_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 Functional Encryption

Following Boneh *et al.* [11], we first define the notion of functionality and then the functional encryption scheme FE for functionality $\mathcal{F}$.

Definition 1 (Functionality). *A functionality $\mathcal{F}$ defined over $(K \times M)$ is a function $F : K \times M \to \Sigma \cap \{\perp\}$, where K is the key space, M is the message space and Σ is the output space and $\perp$ is a special string not contained in Σ. Notice that the functionality is undefined when the key is not in the key space or the message is not in the message space.*

Definition 2 (Functional encryption scheme). *For a functionality $\mathcal{F}$, a functional encryption scheme FE for $\mathcal{F}$ is a tuple $FE = (Setup, KeyGen, Encrypt, Decrypt)$ of 4 algorithms:*

1. *$Setup(1^\lambda)$ outputs public key and master secret keys (mpk, msk) for security parameter λ.*
2. *$KeyGen(msk, k)$, on input a master secret key msk and key $k \in K$ outputs secret key sk_k.*
3. *$Encrypt(mpk, m)$, on input public key mpk and message $m \in M$ outputs ciphertext Ct.*
4. *$Decrypt(mpk, Ct, sk_k)$ outputs $y \in \Sigma \times \{\perp\}$*

The correctness requirement is ensured: for all $(mpk, msk) \leftarrow Setup(1^\lambda)$, all $k \in K$ and $m \in M$, for $sk_k \leftarrow KeyGen(msk, k)$ and $Ct \leftarrow Encrypt(mpk, m)$, we have $Decrypt(mpk, Ct, sk_k) = F(k, m)$ whenever $F(k, m) \neq \perp$, except with negligible probability.

Now, we give the IND-FE-CPA and s-IND-FE-CPA security for functional encryption schemes.

Definition 3 (Indistinguishable-based security). *For a functional encryption scheme $FE = (Setup, KeyGen, Encrypt, Decrypt)$ for functionality $\mathcal{F}$, defined over (K, M), we define security against chosen-plaintext attacks (IND-FE-CPA security) via the security game depicted on Table 1. Firstly, the challenger performs **proc Initialize** and returns mpk to the adversary. The adversary can submit function queries f to the challenger, and the challenger returns the output of **proc KeyGen** to the adversary. The adversary can also submit*

two message m_0^*, m_1^* *to the challenger, and in response, the challenger returns the output of* ***proc LR*** *to the adversary. Finally, the adversary outputs* b' *and the challenger runs* ***proc Finalize*** *to test whether* $b = b'$.

We say that FE *is secure against chosen-plaintext attacks if*

$$|Pr[Exp_{FE,\lambda}^{ind-fe-cpa-0}] - Pr[Exp_{FE,\lambda}^{ind-fe-cpa-1}]| = negl(\lambda).$$

We also define selective security against chosen-plaintext attacks (s-IND-FE-CPA security) when the challenge message m_0^* *and* m_1^* *have to be chosen before the start of the game (see Table 2).*

Table 1. Game $Exp_{FE,\lambda}^{ind-cpa-b}$ define IND-FE-CPA security of FE

proc Initialize(λ)	**proc LR**(m_0^*, m_1^*)
$(mpk, msk) \leftarrow_R Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R Encrypt(mpk, m_b^*)$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(k)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup \{k\}$	If $\exists k \in \mathcal{F}$ s.t. $f(k, m_0^*) \neq f(k, m_1^*)$
$sk_k \leftarrow KeyGen(msk, k)$	then return false
Return sk_k	Return (b'=b)

Table 2. Game $Exp_{FE,\lambda}^{s-ind-cpa-b}$ define s-IND-FE-CPA security of FE

proc Initialize(λ, m_0^*, m_1^*)	**proc LR**()
$(mpk, msk) \leftarrow_R Setup(1^\lambda)$	$Ct* \leftarrow_R Encrypt(mpk, m_b^*)$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(k)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup \{k\}$	If $\exists k \in \mathcal{F}$ s.t. $f(k, m_0^*) \neq f(k, m_1^*)$
$sk_k \leftarrow KeyGen(msk, k)$	then return false
Return sk_k	Return (b'=b)

3 A Generic Functional Encryption Scheme for Quadratic Functions

Now, we present a generic functional encryption scheme for any quadratic functionality over Z_p $LQFE = (Setup, KeyGen, Encrypt, Decrypt)$ based on any public-key encryption scheme $\varepsilon = (Setup, Encrypt, Decrypt)$ that has the some structural and homomorphic properties. These properties are similar to the requirement in [3] and are shown in Supporting Material A. We prove that $LQFE$ is IND-secure against adaptive adversary. Before showing the construction, we give the definitions of quadratic functionality over Z_p.

Quadratic Functionality over Z_p. For any quadratic function,

$$f(\boldsymbol{m}) = t_0 + \sum_{i=1}^{n} t_i m_i + \sum_{i,j=1}^{n} t_{i,j} m_i m_j, m_i \in Z_p^n,$$

it can be transformed as $f(\boldsymbol{x}) = \boldsymbol{x}^T F \boldsymbol{x}$ by setting $\boldsymbol{x} = (1, \boldsymbol{m}) \in Z_p^{n+1}$ and the upper triangular matrix $F = (f_{i,j}) \in Z_p^{(n+1)\times(n+1)}$ where: $f_{1,1} = t_0, f_{1,i} = t_{i-1}$ for all $i \in [2, n+1]$, $f_{i,j} = 0$ for all $i > j$, and $f_{i,j} = t_{i-1,j-1}$ for all $i \in [2, n+1]$ and $i \leq j$. So we define the quadratic functionality over Z_p that $\mathcal{F}(F, \boldsymbol{x}) = \boldsymbol{x}^T F \boldsymbol{x}$.

3.1 Our FE Scheme for Quadratic Functions over Z_p

Before describing the scheme in full details, we give an informal description of our key ideas. We break the process that get $\varepsilon.Enc(\boldsymbol{x}^T F \boldsymbol{x})$ (which can be used to get the final result $\boldsymbol{x}^T F \boldsymbol{x}$ by calling $\varepsilon.Decrypt$) into following parts:

Firstly, we get $\varepsilon.Enc(f_{i,j} x_i x_j)$ by computing $e_1(ct_{\boldsymbol{x},i}, ct_{\boldsymbol{x},j})$, where $e_1(\cdot,\cdot)$ is a symmetric bilinear map. Secondly, we get $\varepsilon.Enc(\sum_{i,j} f_{i,j} x_i x_j)$ by using the homomorphic properties of ε to sum them up. Then, we eliminate these noisy items by designing secret keys and other ciphertexts. Finally, we set a secret number q to prevent the new attack that the adversary can convert some ciphertexts from one group to another group with the help of bilinear maps $e_1(\cdot,\cdot)$ and $e_2(\cdot,\cdot)$.

Now, here comes the formal description of our construction. Let's consider a PKE scheme $\varepsilon = (Setup, Encrypt, Decrypt)$ with the properties defined above. We define our functional encryption scheme for quadratic functions over Z_p $LQFE = (Setup, KeyGen, Encrypt, Decrypt)$ as follows.

$Setup(1^\lambda)$: $G_1, G_2, G_T \leftarrow \varepsilon.Setup(1^\lambda)$, the order of G_1, G_2, G_T is a prime p. $e_1(g_1, g_1) = g_T \leftarrow \mathcal{G}^{1^\lambda}$ is a bilinear map $G_1 \times G_1 \rightarrow G_T$, where g_1, g_T are generators of G_1, G_T. Similarly, $e_2(g_2, g_2) = g_T \leftarrow \mathcal{G}^{1^\lambda}$ is a bilinear map $G_2 \times G_2 \rightarrow G_T$, where g_2, g_T are generators of G_2, G_T. Call ε's key generation algorithm to generate $n+1$ independent secret keys pairs $s_1, ..., s_n, sk$ sharing the same public parameters $params$ and $\boldsymbol{t} = (t_1, ...t_n) \in Z_p^n$. Let $sk_i = s_i + t_i sk, i \in [n]$. Choose $q \leftarrow_R Z_p^*$, then the algorithm sets $g_T' = g_T^q, PK = PKGen(g_1, qsk), PK_i = PKGen(g_1, qsk_i)$. Return $mpk := (params, PK, \{PK_i\}_{i\in[n]}, g_1, g_2, g_T, g_T', e(\cdot,\cdot))$ and $msk := (sk, \boldsymbol{s}, \boldsymbol{t}, q)$.

$KeyGen(msk, \{u_{i,j}\}_{i,j\in[n]})$: on input master secret key msk and the coefficients of quadratic function $f(\boldsymbol{x})$, the algorithm first outputs a random matrix $F \in Z_p^{(n+1)\times(n+1)}$, where $f_{i,j} + f_{j,i} = u_{i.j}$. Then, the algorithm computes $sk_{F,1} = q^2 \sum_{i,j\in[n]} f_{i,j}(s_i + t_i sk)(s_j + t_j sk)$. For $i \in [n]$, computes $sk_{F,2,i} = g_2^{q\sum_{j\in[n]} f_{i,j} sk_j}$ and $\hat{sk}_{F,2,i} = g_1^{q\sum_{j\in[n]} f_{i,j} sk_j}$. For $j \in [n]$, computes $sk_{F,3,j} = g_2^{q\sum_{i\in[n]} f_{i,j} sk_i}$ and $\hat{sk}_{F,3,j} = g_1^{q\sum_{i\in[n]} f_{i,j} sk_i}$.
Return $sk_F = (sk_{F,1}, \{sk_{F,2,i}, \hat{sk}_{F,2,i}\}_{i\in[n]}, \{sk_{F,3,j}, \hat{sk}_{F,3,j}\}_{j\in[n]})$

$Encrypt(\boldsymbol{x}, mpk)$: on input master public key mpk and message $\boldsymbol{x} = (x_0, ..., x_n) \in Z_p^{(n+1)}$, chooses shared randomness r and $\boldsymbol{a} = (a_0, ..., a_n)$ in $Z_p^{*(n+1)}$, and computes $ct_0 = \varepsilon.C(r^2, g_T), ct_{x,i} = \varepsilon.E(pk_i.x_i, r)$. For $i \in [n]$, sets $ct_{a,x,i} = \varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r)$ and $ct_{a,i} = \varepsilon.E(pk(g_1, 0), a_i, r)$
Return $ct_x = (ct_0, \{ct_{x,i}, ct_{a,i}, ct_{a,x,i}\}_{i\in[n]})$
$Decrypt(ct_x, sk_F, mpk)$: on input master public key mpk, ciphertext $ct_x = (ct_0, \{ct_{x,i}, ct_{a,i}, ct_{a,x,i}\}_{i\in[n]})$ and secret key sk_F for matrix $F \in Z_p^{(n+1)\times(n+1)}$, returns the output of

$$\varepsilon.Decrypt(sk_{F,1}, ct_0, \frac{(\prod_{i,j\in[n]} e_1(ct_{x,i}, ct_{x,j})^{f_{i,j}})(\prod_{i\in[n]} e_1(ct_{a,i}, \hat{sk}_{F,2,i}))(\prod_{j\in[n]} e_1(ct_{a,j}, \hat{sk}_{F,3,j}))}{(\prod_{i\in[n]} e_2(ct_{a,x,i}, sk_{F,2,i}))(\prod_{j\in[n]} e_2(ct_{a,x,j}, sk_{F,3,j}))})$$

Correctness of Our Scheme: We divide the decryption algorithm into the following parts:

$$\begin{aligned}
\mathbf{I} &= \prod_{i,j\in[n]} e_1(ct_{x,i}, ct_{x,j})^{f_{i,j}} \\
&= \prod_{i,j\in[n]} [\varepsilon.E(pk(g_T, q^2 sk_i sk_j), x_i x_j, r^2)\varepsilon.E(pk(g_T, qsk_i)^{x_j}, \\
&\qquad 0, r)\varepsilon.E(pk(g_T, qsk_j)^{x_i}, 0, r)]^{f_{i,j}} \\
&= \varepsilon.E(pk(g_T, q^2 \sum_{i,j\in[n]} (f_{i,j} sk_i sk_j)), \\
&\quad \sum_{i,j\in[n]} f_{i,j} x_i x_j, r^2)\varepsilon.E(pk(g_T, q \sum_{i,j\in[n]} f_{i,j}(x_j sk_i + x_i sk_j)), 0, r). \\
\mathbf{II} &= (\prod_{i\in[n]} e_2(ct_{a,x,i}, sk_{F,2,i}))(\prod_{j\in[n]} e_2(ct_{a,x,j}, sk_{F,3,j})) \\
&= \varepsilon.E(pk(g_T, q \sum_{i,j\in[n]} x_i f_{i,j} sk_j), q \sum_{i,j\in[n]} a_i f_{i,j} sk_j, r)\varepsilon.E(pk(g_T, \\
&\quad q \sum_{i,j\in[n]} x_j f_{i,j} sk_i), q \sum_{i,j\in[n]} a_j f_{i,j} sk_i, r) \\
&= \varepsilon.E(pk(g_T, q \sum_{i,j\in[n]} (x_i f_{i,j} sk_j + x_j f_{i.j} sk_i)), q \sum_{i,j\in[n]} f_{i,j}(a_i sk_j + a_j sk_i), r) \\
\mathbf{III} &= (\prod_{i\in[n]} e_1(ct_{a,i}, \hat{sk}_{F,2,i}))(\prod_{j\in[n]} e_1(ct_{a,j}, \hat{sk}_{F,3,j})) \\
&= \prod_{i\in[n]} \varepsilon.E(pk(g_T, 0), a_i q \sum_{j\in[n]} f_{i,j} sk_j, r) \prod_{j\in[n]} \varepsilon.E(pk(g_T, 0), a_j q \sum_{i\in[n]} f_{i,j} sk_i, r) \\
&= \varepsilon.E(pk(g_T, 0), q \sum_{i,j\in[n]} (a_i f_{i,j} sk_j + a_j f_{i,j} sk_i), r)
\end{aligned}$$

So, we can get

$$\begin{aligned} &LQFE.Decrypt(ct_x, sk_F, mpk) = \varepsilon.Decrypt(sk_{F,1}, ct_0, \frac{\mathbf{I}\cdot\mathbf{III}}{\mathbf{II}}) \\ &= \varepsilon.Decrypt(sk_{F,1}, ct_0, \varepsilon.E(pk(g_T, sk_{F,1}), \boldsymbol{x}^T F\boldsymbol{x}, r^2)) \\ &= \boldsymbol{x}^T F\boldsymbol{x} \end{aligned}$$

Theorem 1. *If the underlying PKE ε has message space, ciphertext space and secret key space of the same order p, if it is IND-CPA and satisfies the properties defined in Sect. 3.1, then LQFE is IND-FE-CP against adaptive adversary.*

The proof of Theorem 1 can be found in Supporting Material B.

4 From Quadratic FE to Cubic FE over Z_p

In this section, we show how to transform our generic FE scheme for quadratic functionality to a generic FE scheme for cubic functionality over Z_p. The method in this section can be generalized to realize a degree-$(m+1)$ polynomial FE from a degree-m polynomial FE, which will be discussed in Sect. 5.

Let $CFE = (Setup, KeyGen, Encrypt, Decrypt)$ be a FE for cubic functionality based on a FE scheme for quadratic functionality $QFE = (Setup, KeyGen, Encrypt, Decrypt)$ which is s-IND-FE-CPA secure. Firstly, we give the definition of cubic functionality over Z_p that is used in our scheme.

4.1 Cubic Functionality over Z_p

For any cubic function $f(\boldsymbol{x}) = \sum_{i,j,k=0}^{n} f_{i,j,k} x_i x_j x_k$, where $x_0 = 1$, we can find a set of matrices $A_0, \ldots, A_n$, s.t. $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} \boldsymbol{x}^T A_0 \boldsymbol{x} \\ \boldsymbol{x}^T A_1 \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^T A_n \boldsymbol{x} \end{pmatrix}$. Actually, there exists more than one set $\{A_i\}_{i\in[n]}$ that can satisfy this equation.

For security reasons, we define the cubic functionality over Z_p $\mathcal{F}$ in our functional encryption scheme as follows, where $\mathcal{M} \in Z_p^{n+1}$ is the message space:

Definition 4 *(cubic functionality).* *For any $f_{i,j,k} \in Z_p$, let $\mathcal{F}(\{f_{i,j,k}\}_{i,j,k\in[n]}, \boldsymbol{x}) = \sum_{i,j,k=0}^{n} f_{i,j,k} x_i x_j x_k\}$. For $\forall f(\boldsymbol{x}) \in \mathcal{F}, \boldsymbol{x_0}, \boldsymbol{x_1} \in \mathcal{M}$ and $f(\boldsymbol{x_0}) = f(\boldsymbol{x_1})$, there exists an algorithm $ALG(f)$ which could find a set of matrices $A_0, \ldots, A_n \in Z_p^{(n+1)\times(n+1)}$, s.t. $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} \boldsymbol{x}^T A_0 \boldsymbol{x} \\ \boldsymbol{x}^T A_1 \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^T A_n \boldsymbol{x} \end{pmatrix}$ and $\forall i \in [n], \boldsymbol{x_0} A_i \boldsymbol{x_0} = \boldsymbol{x_1} A_i \boldsymbol{x_1}$.*

This requirement is due to the fact that in any quadratic FE, only if $f(\boldsymbol{x_0}) = f(\boldsymbol{x_1})$, any probabilistic poly(n)-time (PPT) adversary cannot distinguish the ciphertexts of $\boldsymbol{x_0}$ and $\boldsymbol{x_1}$ by IND-security definition of FE.

The message space is decided by how many cubic functions we want to include in the function space. If we just want to use a few fractions of all cubic functions, then the message space could be larger. In the other side, brute-force FE scheme cannot ensure linear-size ciphertext, which is important when constructing FE scheme for higher degree polynomials.

The following theorem shows a special case that if we only have two messages, $\boldsymbol{y_0}, \boldsymbol{y_1}$, such that $f(\boldsymbol{y_0}) = f(\boldsymbol{y_1})$, all cubic functions meet this requirement[4].

Theorem 2. *Given* $\boldsymbol{y_0}, \boldsymbol{y_1} \in Z_p^{n+1}$, $f(\boldsymbol{x}) = \sum_{i \geq j \geq k=0}^{n} f_{i,j,k} x_i x_j x_k$ *and* $f(\boldsymbol{y_0}) = f(\boldsymbol{y_1})$. *There exists* $A_0, \ldots, A_n \in Z_p^{(n+1)\times(n+1)}$, *s.t.* $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} \boldsymbol{x}^T A_0 \boldsymbol{x} \\ \boldsymbol{x}^T A_1 \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^T A_n \boldsymbol{x} \end{pmatrix}$ *and* $\forall i \in [n], \boldsymbol{y_0} A_i \boldsymbol{y_0} = \boldsymbol{y_1} A_i \boldsymbol{y_1}$.

The proof of Theorem 2 can be found in Supporting Material C.

For a given message space $\mathcal{M}$, a natural approach to test whether a cubic function $f(\boldsymbol{x})$ is in $\mathcal{F}$ is that: (1) Find all vector pairs $(\boldsymbol{a}, \boldsymbol{b})$ s.t. $f(\boldsymbol{a}) = f(\boldsymbol{b})$ and sets $\mathcal{V} = \{(\boldsymbol{a}, \boldsymbol{b}) | f(\boldsymbol{a}) = f(\boldsymbol{b})\}$. (2) For each pairs $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{V}$, find the system of linear equations and get the solutions $\mathcal{S}_i = \{(A_0, A_1, \ldots, A_n)\}$, where $i = 1, \ldots, |\mathcal{V}|$. (3) Check whether $\mathcal{S}_1 \cap \mathcal{S}_2 \cap \ldots \cap \mathcal{S}_n = \emptyset$.

4.2 Our FE Scheme for Cubic Functions over Z_p

Before describing the scheme in full detail, we give an informal exposition of our key ideas. We use a random invertible matrix $W \in Z_p^{2\times 2}$ and some random vectors $\boldsymbol{a}, \boldsymbol{r}$ to construct our ciphertexts, which consist of two major parts.

The first part of the ciphertexts seems like $(r_i, t_i^{-1} x_i) W^{-1}$. The second part of the ciphertext seems like $W(a_i, QFE.Enc(x_i))^T$, where QFE is a FE scheme for quadratic functions. Notice that the second part of ciphertexts are not generated directly. The outputs of algorithm ALG is randomized and not unique, so the matrices A_i are not known by the adversary. When doing decryption, firstly, we get $W(a_i, t_i \boldsymbol{x}^T A_i \boldsymbol{x})$ by calling $QFE.Decrypt$. Then, we get $\sum_i a_i r_i + x_i \boldsymbol{x}^T A_i \boldsymbol{x}$ by multiply two parts of ciphertexts and sum them up. Finally, we get the final result $f(\boldsymbol{x}) = \sum_{i,j,k} f_{i,j,k} x_i x_j x_k = \sum_i x_i \boldsymbol{x}^T A_i \boldsymbol{x}$ by substraction $\sum_i a_i r_i$.

Now, here comes the formal description of our construction. Let's consider a quadratic FE scheme $QFE = (Setup, KeyGen, Encrypt, Decrypt)$.

[4] A more detailed analysis needs to be carried out to see how practical this requirement is although the requirement and our construction method represent a step towards constructing secure FE schemes for polynomials.

For cubic functions $\mathcal{F}$, we define our functional encryption scheme $CFE = (Setup, KeyGen, Encrypt, Decrypt)$ as follows:

$Setup(1^\lambda, 1^n)$: $(mpk1, msk1) \leftarrow QFE.Setup(1^\lambda, 1^n)$, Randomly choose $\boldsymbol{t} = (t_0, \ldots, t_n) \leftarrow_R Z_p^{n+1}$. Return $mpk := (mpk1, \boldsymbol{t})$ and $msk := (msk1)$.
$KeyGen(mpk, msk, f = \{f_{i,j,k}\}_{i,j,k\in[n]})$: Call $ALG(f)$ as defined above to obtain the set of matrices $A_0, \ldots, A_n \leftarrow_R ALG(f)$. Then computes $sk_{A_i} = QFE.KeyGen\ (mpk1, msk1, t_i A_i)$. Return $sk_F := \{sk_{A_i}\}_{i\in[n]}$.
$Encrypt(\boldsymbol{x}, mpk)$: Choose a matrix $W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ from $Z_p^{*(2\times 2)}$, where $w_{12} = u_1^2, w_{22} = u_2^2$ and $WW^{-1} = I$. Randomly choose $\boldsymbol{r} = (r_0, \ldots, r_n), \boldsymbol{a} = (a_0, \ldots, a_n) \in Z_p^{n+1}$. Then computes $Ct_{u_1 x} = QFE.Encrypt(mpk1, u_1\boldsymbol{x}), Ct_{u_2 x} = QFE.Encrypt(mpk1, u_2\boldsymbol{x})$. And sets $Ct_{w,x,i} = (r_i, t_i^{-1}x_i)W^{-1} = (r_i w_{11}^{-1} + t_i^{-1} w_{21}^{-1} x_i, r_i w_{12}^{-1} + t_i^{-1} w_{22}^{-1} x_i)$, $Ct_{a,1,i} = w_{11}a_i, Ct_{a,2,i} = w_{21}a_i, Ct_{ar} = \sum_{i=0}^{n} a_i r_i$. Return $Ct_x = (Ct_{u_1 x}, Ct_{u_2 x}, \{Ct_{w,x,i}, Ct_{a,i,1}, Ct_{a,2,i}\} i \in [n], Ct_{a,r})$.
$Decrypt(Ct_x, sk_F, mpk)$: Return:

$$\left[\sum_{i=0}^{n} Ct_{w,x,i} \cdot \begin{pmatrix} Ct_{a,1,i} + QFE.Decrypt(sk_{A_i}, Ct_{u_1 x}, mpk) \\ Ct_{a,2,i} + QFE.Decrypt(sk_{A_i}, Ct_{u_2 x}, mpk) \end{pmatrix}\right] - Ct_{a,r}$$

Correctness of Our Scheme:

$$\begin{aligned}
&Decrypt(Ct_x, sk_F, mpk) \\
=&\sum_{i=0}^{n} (r_i w_{11}^{-1} + t_i^{-1} x_i w_{21}^{-1}, r_i w_{12}^{-1} + t_i^{-1} x_i w_{22}^{-1}) \begin{pmatrix} w_{11}a_i + w_{12} t_i \boldsymbol{x}^T A_i \boldsymbol{x} \\ w_{21}a_i + w_{22} t_i \boldsymbol{x}^T A_i \boldsymbol{x} \end{pmatrix} - \sum_{i=0}^{n} a_i r_i \\
=&\sum_{i=0}^{n} (r_i, t_i^{-1} x_i) W^{-1} W \begin{pmatrix} a_i \\ t_i \boldsymbol{x}^T A_i \boldsymbol{x} \end{pmatrix} - \sum_{i=0}^{n} a_i r_i \\
=&\sum_{i=0}^{n} (a_i r_i + x_i \cdot \boldsymbol{x}^T A_i \boldsymbol{x}) - \sum_{i=0}^{n} a_i r_i \\
=&\boldsymbol{x}^T \begin{pmatrix} \boldsymbol{x}^T A_0 \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^T A_n \boldsymbol{x} \end{pmatrix}
\end{aligned}$$

Theorem 3. *If the underlying functional encryption scheme for quadratic functions QFE has message space and secret key space of the same order p, if it is s-IND-CPA secure, then CFE is s-IND-CPA secure.*

The proof of Theorem 3 can be found in Supporting Material C.

5 Generalization: From Degree-m Polynomial FE to Degree-$(m+1)$ Polynomial FE

Let $f(\boldsymbol{x}) = \sum_{q_1,\ldots,q_{m+1}=0}^{n} (f_{q_1,\ldots,q_{m+1}} \Pi_{i=1}^{m+1} x_{q_i})$, where $x_0 = 1$, be a degree-$(m+1)$ polynomial function, we can find more than one such sets $\{f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\}$

such that, $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} f_0(\boldsymbol{x}) \\ f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix}$, where $f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})$ are degree-m polynomial functions.

So our transformation scheme in the above section can be generalized to a degree-$(m+1)$ polynomial FE scheme from any degree-m polynomial FE. For security, the restriction in cubic functionality is also needed in our degree-$(m+1)$ polynomial functionality. Let $\mathcal{M} \in Z_p^{n+1}$ be the message space, then our degree-$(m+1)$ polynomial functionality $\mathcal{F}_{m+1}$ is defined as follows:

Definition 5 *(degree-$(m+1)$ polynomial functionality). For any $f_{q_1,\ldots,q_{m+1}} \in Z_p$, let $\mathcal{F}(\{f_{q_1,\ldots,q_{m+1}}\}_{q_1,\ldots,q_{m+1}\in[n]}, \boldsymbol{x}) = \sum_{q_1,\ldots,q_{m+1}=0}^{n}(f_{q_1,\ldots,q_{m+1}} \Pi_{i=1}^{m+1} x_{q_i})\}$. $\forall f(\boldsymbol{x}) \in \mathcal{F}_{m+1}$, $\boldsymbol{x_0}, \boldsymbol{x_1} \in \mathcal{M}$ and $f(\boldsymbol{x_0}) = f(\boldsymbol{x_1})$, there exists as algorithm ALG(f) which could find a set of degree-m polynomial functions $f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})$, s.t. $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} f_0(\boldsymbol{x}) \\ f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix}$ and $\forall i \in [n], f_i(\boldsymbol{x_0}) = f_i(\boldsymbol{x_1})$.*

Similar to our cubic functionality definition, two restrictions on the set of degree-m polynomials $f_0, \ldots, f_n$ are implied by the degree-$(m+1)$ functionality definition. One is that the degree-$(m+1)$ polynomial $f(\boldsymbol{x})$ can be written as $f(\boldsymbol{x}) = \sum_{i=0}^{n}(x_i f_i(\boldsymbol{x}))$. The other is that for any two vectors $\boldsymbol{x_0}, \boldsymbol{x_1} \in \mathcal{M}$, if $f(\boldsymbol{x_0}) = f(\boldsymbol{x_1})$, then the outputs of each degree-m polynomials $f_i, i \in [n]$ on inputs $\boldsymbol{x_0}, \boldsymbol{x_1}$ are also the same. In the following theorem, we also show that when the message space $\mathcal{M}$ is very small, $\mathcal{F}_{m+1}$ contains all degree-$(m+1)$ polynomials.

Theorem 4. *Given $\boldsymbol{y_0}, \boldsymbol{y_1} \in Z_p^{n+1}$, $f(\boldsymbol{x}) = \sum_{q_1,\ldots,q_{m+1}=0}^{n}(f_{q_1,\ldots,q_{m+1}} \Pi_{i=1}^{m+1} x_{q_i})$ and $f(\boldsymbol{y_0}) = f(\boldsymbol{y_1})$. There exists a set of degree-m polynomials $f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})$, s.t., $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} f_0(\boldsymbol{x}) \\ f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix}$ and $\forall i \in [n], f_i(\boldsymbol{y_0}) = f_i(\boldsymbol{y_1})$.*

Proof. The proof is similar with the proof of Theorem 3.

Intuitively, when m increasing, the number of sets $\{f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\}$ where $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} f_0(\boldsymbol{x}) \\ f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix}$ is increased. Then, the number of equations which are generated by the second restriction $\forall i \in [n], f_i(\boldsymbol{y_0}) = f_i(\boldsymbol{y_1})$ are not change (this system of equations always consists of $n+1$ equations). When $m > 2$, after putting these two restrictions together, it is easier to find a feasible solution than the system of linear equations (3) in the proof of Theorem 3.

Therefore, one can choose a smaller message space $\mathcal{M}$ to achieve larger functionality space $\mathcal{F}_{m+1}$. Or choose smaller functionality space $\mathcal{F}_{m+1}$ to get larger message space $\mathcal{M}$.

5.1 Our FE Scheme for Degree-$(m+1)$ Polynomial over Z_p

Let us consider a degree-m polynomial FE scheme $mFE = (Setup, KeyGen, Encrypt, Decrypt)$. We define our functional encryption scheme for a degree-$(m+1)$ polynomial $\mathcal{F}_{m+1}$ $(m+1)FE = (Setup, KeyGen, Encrypt, Decrypt)$ as follows:

$Setup(1^\lambda, 1^n)$: $(mpk1, msk1) \leftarrow mFE.Setup(1^\lambda, 1^n)$, Randomly choose $\boldsymbol{t} = (t_0, \ldots, t_n) \leftarrow_R Z_p^{n+1}$. Return $mpk := (mpk1, \boldsymbol{t})$ and $msk := (msk1)$.
$KeyGen(mpk, msk, f = \{f_{q_1,\ldots q_{m+1}}\}_{q_1,\ldots,q_{m+1}\in[n]})$: Call $ALG(f)$ as defined above to obtain the set of m degree polynomial $f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}) \leftarrow_R ALG(f)$. Then computes $sk_{f_i} = mFE.KeyGen(mpk1, msk1, t_i f_i)$. Return $sk_F := \{sk_{f_i}\}_{i\in[n]}$.
$Encrypt(\boldsymbol{x}, mpk)$: Choose a matrix $W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ from $Z_p^{*(2\times 2)}$, where $w_{12} = u_1^m, w_{22} = u_2^m$ and $WW^{-1} = I$. Randomly choose $\boldsymbol{r} = (r_0, \ldots, r_n), \boldsymbol{a} = (a_0, \ldots, a_n) \in Z_p^{n+1}$. Then computes $Ct_{u_1 x} = mFE.Encrypt(mpk1, u_1\boldsymbol{x}), Ct_{u_2 x} = mFE.Encrypt(mpk1, u_2\boldsymbol{x})$. And sets $Ct_{w,x,i} = (r_i, t_i^{-1}x_i)W^{-1} = (r_i w_{11}^{-1} + t_i^{-1} w_{21}^{-1} x_i, r_i w_{12}^{-1} + t_i^{-1} w_{22}^{-1} x_i)$, $Ct_{a,1,i} = w_{11}a_i, Ct_{a,2,i} = w_{21}a_i, Ct_{ar} = \sum_{i=0}^{n} a_i r_i$. Return $Ct_x = (Ct_{u_1 x}, Ct_{u_2 x}, \{Ct_{w,x,i}, Ct_{a,i,1}, Ct_{a,2,i}\} i \in [n], Ct_{a,r})$.
$Decrypt(Ct_x, sk_F, mpk)$: Return:

$$\left[\sum_{i=0}^{n} Ct_{w,x,i} \cdot \begin{pmatrix} Ct_{a,1,i} + mFE.Decrypt(sk_{A_i}, Ct_{u_1 x}, mpk) \\ Ct_{a,2,i} + mFE.Decrypt(sk_{A_i}, Ct_{u_2 x}, mpk) \end{pmatrix}\right] - Ct_{a,r}$$

The correctness can be easily extended from the proof in our cubic FE scheme.

Theorem 5. *If the underlying functional encryption scheme for a degree-m polynomials, mFE, has message space and secret key space of the same order p, if it is s-IND-CPA secure, then $(m+1)FE$ is s-IND-CPA secure.*

The proof of this theorem can be easily extended from the proof of Theorem 4.

Efficiency Analysis. Notice that in this scheme, the size of secret keys in a degree-$(m+1)$ polynomial FE will become n times the size of secret keys in the degree-m polynomial FE, $m \geq 2$. However, we show that it would not induce more cost in practice. In a cloud application scenario, the data owner stores the encrypted data in a server, and the user has a $m+1$-degree polynomial f and wants to ask the function value $f(\boldsymbol{x})$. So the user sends all coefficients of monomial in f to the Key Generator, and the Key Generator will return a corresponding secret key sk_f. Then the user sends sk_f to the server, and the

server does decryption and returns $f(\boldsymbol{x})$. The size of coefficients is $O(n^{m+1})$. In almost existing public key FE schemes from standard assumptions [2–4,6], the coefficients are used in the Decryption algorithm. So the size of message that the user sends to the server is also $O(n^{m+1})$ in their schemes (the message includes two parts: the coefficients and sk_f). The expansion of the secret key in our transformation scheme is a factor n, but the expansion of the size of coefficients is also a factor n (from degree-m to degree-$(m+1)$). So under the condition that the coefficients are used in the Decryption algorithm, the expansion of secret key can be bounded by the expansion of the size of coefficients. Therefore, our transformation scheme does not lose any efficiency.

6 Conclusions and Discussion

In this paper, we show that constructing generic FE schemes for quadratic functions, cubic functions and finite degree polynomials are achievable. In summary, our generic FE scheme for quadratic functions is IND-secure against adaptive adversary with a linear size ciphertext. This generic scheme can be instantiated from any PKE schemes that satisfy a few structural and homomorphic properties. Our generic FE scheme for cubic functions from our quadratic FE scheme is the first effective scheme for cubic functions. The transformation can be generalized to higher degree polynomials. In particular, we show how to transform an IND-secure degree-m polynomial FE scheme to an IND-secure degree-$(m+1)$ polynomial FE scheme.

There are still quite a number of open questions in this topic. When we use an IND-secure degree-m polynomial FE mFE to build a degree-$(m+1)$ polynomial FE scheme, a natural restriction, i.e., the cubic functionality, on such a degree-$(m+1)$ polynomial appears. It seems that the question of building generic FE scheme for any finite-degree polynomials without this restriction is not feasible. Another question is about function privacy (also known as function hiding, studied in [10,18,24,29]). Intuitively, function privacy requires that decryption keys reveal essentially nothing on their corresponding function. However, in almost existing FE constructions from standard assumptions, the functions are actually a part of their secret keys, i.e., the coefficients are directly used in decryption algorithms without any protection. The question of building functional hiding FE scheme for polynomials from standard assumptions still remains open.

Finally, we conclude the paper by giving a remark on "Indistinguishable obfuscation from Functional encryption." Very recently, some papers [7,25] show that it is possible to construct indistinguishability obfuscation (IO) from FE. They showed that IO can be obtained from constant degree graded encoding schemes or subexponentially-secure weakly-succinct FE for functions in NC^1. In fact, our transformation meets the weakly-succinct condition and may provide a new direction on constructing IO from standard assumptions, although FE for polynomials (i.e., arithmetic circuits) seems not strong enough to get IO yet.

Acknowledgement. This project is partially supported by the Collaborative Research Fund (CRF) of RGC of Hong Kong (Project No. CityU C1008-16G).

Supporting Material

A Requirements of PKE

Our framework constructs functional encryption scheme for quadratic functions $QFE = (Setup, KeyGen, Encrypt, Decrypt)$ from a public-key encryption scheme $\varepsilon = (Setup, Encrypt, Decrypt)$. In order to prove the correctness and security of the new scheme, we need some structural and homomorphic properties on ε as defined below.

Structure. ε's secret keys and public keys are elements of a group G (with generator g_1), and the message space is $M_x \subset Z$. We require the ciphertexts to consist of two parts $c_0 = C(g_1, r)$ and $ct_1 = E(pk, x, r)$, where $pk(g_1, sk)$ is the public key in G corresponding to the secret key sk. The first part c_0 corresponds to some commitment $C(g_1, r)$ of the randomness r used for the encryption. The second part ct_1 is the encryption of x with randomness r. Computing a from $E(pk(g, 0), a, r)$ can be reduced to some difficult problems.

We also split the **Setup** algorithm for convenience in the following two algorithms to sample secret keys, and to sample corresponding public keys:

$SKGen(1^\lambda)$ takes in input the security parameter and sample a secret key sk from the secret key space according to the same distribution induced by **Setup**. $PKGen(sk, \tau)$ takes in input a secret key sk and parameters τ, and generates a public key pk corresponding to sk according to the distribution induced by τ. We will omit τ when it is clear from the context.

Linear Key Homomorphism. We say that a PKE has linear key homomorphism if for any two secret keys $sk_1, sk_2 \in G$ and any $y_1, y_2 \in Z_p$, the linear combination formed by $y_1 sk_1 + y_2 sk_2$ can be computed efficiently only using public parameters, the secret keys and the coefficients. And this combination $y_1 sk_1 + y_2 sk_2$ also functions as a secret key to a public key that can be computed as $pk_1^{y_1} \cdot pk_2^{y_2}$, where pk_1 (resp. pk_2) is a public key corresponding to sk_1 (resp. sk_2).

Linear Ciphertext Homomorphism Under Shared Randomness. We say that a PKE has linear ciphertext homomorphism under shared randomness if it holds that $E(pk_1, x, r) \cdot E(pk_2, y, r) = E(pk_1 pk_2, x + y, r)$ and $E(pk(g^q, sk), x, r) = E(pk(g, sk), x, r)^q = E(pk(g, qsk), qx, r)$.

Computation Properties in Bilinear Map. Assume that $e(g_1^a, g_1^b) = g_T^{ab}$ is a bilinear map $G \times G \to G_T$, where g_T is a generator of G_T and $e(g_1, g_1) = g_T$. We require that

$$\begin{aligned} &e(E(pk_{(g_1, sk_1)}, x, a), E(pk_{(g_1, sk_2)}, y, b)) \\ &= E(pk_{(g_T, sk_1 sk_2)}, xy, ab) E(pk_{(g_T, sk_1)}, 0, a)^y E(pk_{(g_T, sk_2)}, 0, b)^x \end{aligned}$$

And for security, we define two properties via security game. More details can be referred to [3].

l-Public-Key-Reproducibility. For a public-key encryption scheme ε, we define l-public-key-reproducibility via the following security game:

Game $Exp_{\varepsilon,\lambda}^{l-ct-rep-b}(\mathcal{A})$
proc Initialize$(\lambda, \mathcal{M})$
$(sk, (\alpha_i, sk_i)_{i\in[l]}) \leftarrow_R \mathcal{D}(1^\lambda)$
If $b = 0$ then $(pk_i = \varepsilon.PKGen(\alpha_i sk + sk_i, \tau))_{i\in[l]}$
else $pk \leftarrow \varepsilon.PKGen(sk, \tau')$; $(pk_i = pk^{\alpha_i} \cdot \varepsilon.PKGen(sk_i.\tau_i))_{i\in[l]}$
Return $(pk_i, sk_i)_{i\in[l]}$
proc Finalize(b')
Return $(b' = b)$

with $\mathcal{D}$ samples tuples of the form $(sk, (\alpha_i, sk_i)_{i\in[l]})$ where sk and the sk_i's are sampled from $SKGen$, and the α_i's are in $\mathcal{T}$.

Then, we say that ε has $l-public-key-reproducibility$ if there exists $\tau, \tau'(\tau_i)_{i\in[l]}$ such that

$$|Pr[Exp_{\varepsilon,\lambda}^{l-pk-rep-0}(\mathcal{A} = 1)]| - |Pr[Exp_{\varepsilon,\lambda}^{l-pk-rep-1}(\mathcal{A}) = 1]| = negl(\lambda)$$

l-Ciphertext-Reproducibility. For a public-key encryption scheme ε, we define l-ciphertext-reproducibility via the following security game:

Game $Exp_{\varepsilon,\lambda}^{l-ct-rep-b}(\mathcal{A})$
proc Initialize$(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, sk_i)_{i\in[l]}) \leftarrow_R \mathcal{D}(1^\lambda)$
$sk \leftarrow \varepsilon.SKGen(1^\lambda); pk \leftarrow \varepsilon.PKGen(sk, \tau'); (pk_i \leftarrow \varepsilon.PKGen(sk_i, \tau_i))_{i\in[l]}$
$ct_0 = \varepsilon.C(r); ct = \varepsilon.E(pk, a, r)$
If $b = 0$ then $ct_i = ct^{\alpha_i} \cdot \varepsilon.E(pk_i, x_i, r)$
else $ct_i = ct^{\alpha_i} \cdot \varepsilon.E'(sk_i, x_i, ct_0, \tau_i)$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i\in[l]}, ct_0, (ct_i)_{i\in[l]})$
proc Finalize(b')
Return $(b' = b)$

where (1) $\mathcal{D}$ samples tuples of the form $(a, (\alpha_i, x_i, sk_i)_{i\in[l]})$, where sk_i's are sampled from $SKGen$, α_i's are in $\mathcal{T}$ and a and the x_i's are in $\mathcal{M}_x$. (2) E' is an algorithm that takes in input a secret key in H, a message in Z_p, a first part ciphertext $C(r)$ for some r in the randomness space, and the parameters needed to generate public keys, and output a second part ciphertext.

Then, we say that ε has $l-ciphertext-reproducibility$ if there exists τ', τ_i's and algorithm E' such that

$$|Pr[Exp_{\varepsilon,\lambda}^{l-ct-rep-0}(\mathcal{A} = 1)]| - |Pr[Exp_{\varepsilon,\lambda}^{l-ct-rep-1}(\mathcal{A}) = 1]| = negl(\lambda)$$

.

B Proofs in Our FE Scheme for Quadratic Functions

B.1 Proof of Theorem 1

Proof. We proof the security via a sequence of hybrid experiments, and then we show they are indistinguishable.

Hybrid H1: This is the IND-FE-CPA game:

proc Initialize(λ)	**proc LR**($\boldsymbol{x_0}, \boldsymbol{x_1}$)
$(mpk, msk) \leftarrow_R Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R Encrypt(mpk, \boldsymbol{x_b})$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

Hybrid H2: This is like H1 except that the master public key is generated by invoking the algorithm $H2.Setup$ defined as follows:
$H2.Setup(1^\lambda, 1^n)$: The algorithm samples $sk \leftarrow \varepsilon.SKGen(1^\lambda)$, for $i \in [n]$, PKE secret key $s_i \leftarrow \varepsilon.Setup(1^\lambda)$ and uniformly random scalar $t_i \leftarrow_R Z_P, q \leftarrow_R Z_p^*$ and a bilinear map $e(g_1, g_1) = g_T$, where g_1, g_T are generators of G_1, G_T. Similarly, $e_2(g_2, g_2) = g_T \leftarrow \mathcal{G}^{1^\lambda}$ is a bilinear map $G_2 \times G_2 \rightarrow G_T$, where g_2, g_T are generators of G_2, G_T. Then the algorithm sets: $PK = \varepsilon.PKGen(g_1, qsk, \tau), sk_i = s_i + t_i sk, g_T' = g_T^q$. $PK_{s_i} = \varepsilon.PKGen(g_1, qs_i, \tau_i)$ $PK_i = PK^{t_i} \cdot PK_{s_i}$, where τ is the same as used in the Setup algorithm, and τ_i is such that $PK^{t_i} \cdot PK_{s_i}$ is close to $\varepsilon.PKGen(g_1, qsk_i)$.
The algorithm returns $mpk := (params, PK, \{PK_i\}_{i\in[n]}, g_1, g_2, g_T, g_T', e(\cdot,\cdot))$ and $msk := (\boldsymbol{s}, \boldsymbol{t}, sk, q)$. Under the $l-public-key-reproducibility$ of ε, H1 and H2 are indistinguishable.

Hybrid H3: This is like H2 except that the challenge ciphertext is generated by invoking the algorithm $H3.Encrypt$ defined as follows:
$H3.Encrypt(msk, mpk, \boldsymbol{x})$: Choose shared randomness r and $\boldsymbol{a} = (a_1, ..., a_n)$ in Z_p, and computes
$ct_0 = \varepsilon.C(r^2, g_1), ct_1 = \varepsilon.E(PK, 0, r), ct_{a,i} = \varepsilon.E(pk(g_1, 0), a_i, r)$
For $i \in [n]$, $ct_{x,i} = ct_1^{t_i} \cdot \varepsilon.E(PK_{s_i}, x_i, r), ct_{a,x,i} = \varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r)$
By linear ciphertext-homomorphism of ε, H2 = H3.

Hybrid H4: This is like H3 except that the challenge ciphertext is generated by invoking the algorithm $H4.Encrypt$ defined as follows:
$H4.Encrypt(msk, mpk, Ct, \boldsymbol{x})$: Let $Ct = (Ct_0, Ct_1)$. Then the algorithm computes the ciphertext for $\boldsymbol{x}$ in the following way:
$ct_0 = \varepsilon.C(r^2, g_1), ct_{a,i} = \varepsilon.E(pk(g_1, 0), a_i, r)$. For $i \in [n]$, $ct_{x,i} = ct_1^{t_i} \cdot \varepsilon.E'(s_i, x_i, Ct_0, \tilde{r}), ct_{a,x,i} = \varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r)$, where $\varepsilon.E'$ is the alternative encryption algorithm defined in the $l-ciphertext-reproducibility$ game. $\tilde{r}$ is some randomness shared among all the invocation of $\varepsilon.E'$.

proc Initialize(λ)	**proc LR**($\boldsymbol{x_0}, \boldsymbol{x_1}$)
$(mpk, msk) \leftarrow_R H2.Setup(1^\lambda, 1^n)$	$Ct = \varepsilon.E(PK, 0)$
$\mathcal{F} \leftarrow \emptyset$	$Ct* \leftarrow_R H4.Encrypt(msk, mpk, Ct, \boldsymbol{x_b})$
Return mpk	Return $Ct*$
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

Under the $l-ciphertext-reproducibility$ of ε, H3 and H4 are indistinguishable.

Hybrid H5: This is like H4 except that the challenge ciphertext is generated by invoking the algorithm $H5.Encrypt$ defined as follows and Ct encrypts a random value in Z_p.
$H5.Encrypt(msk, mpk, Ct, \boldsymbol{x})$: Let $Ct = (Ct_0, Ct_1)$. Then the algorithm computes the ciphertext for $\boldsymbol{x}$ in the following way:
$ct_0 = \varepsilon.C(r^2, g_1), ct_{a,i} = \varepsilon.E(pk(g_1, 0), a_i, r)$.
For $i \in [n]$, $ct_{x,i} = ct_1^{t_i} \cdot \varepsilon.E'(s_i, x_i, Ct_0, \tilde{r}), ct_{a,x,i} = \varepsilon.E(pk(g_2, 1)^{x_i+t_i}, a_i, r)$

proc Initialize(λ)	**proc LR**($\boldsymbol{x_0}, \boldsymbol{x_1}$)
$(mpk, msk) \leftarrow_R H2.Setup(1^\lambda, 1^n)$	$Ct = \varepsilon.E(PK, 1)$
$\mathcal{F} \leftarrow \emptyset$	$Ct* \leftarrow_R H5.Encrypt(msk, mpk, Ct, \boldsymbol{x_b})$
Return mpk	Return $Ct*$
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

Under the s-IND-CPA security of ε, $\varepsilon.E(PK, 0)$ and $\varepsilon.E(pk, 1)$ are indistinguishable. Now, we need to show that $\varepsilon.E(pk(g_2, 1)^{x_i+t_i}, a_i, r)$ and $\varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r)$ are indistinguishable.

If $\exists w(t_i) \in Z_p$, s.t. $\varepsilon.E(pk(g_2, t_i), 0, r) = \varepsilon.E(pk(g_2, 0), w(t_i), r)$, then

$$\begin{aligned}\varepsilon.E(pk(g_2,1)^{x_i+t_i}, a_i, r) &= \varepsilon.E(pk(g_2, x_i), a_i, r)\varepsilon.E(pk(g_2, t_i), 0, r)\\ &= \varepsilon.E(pk(g_2, x_i), a_i, r)\varepsilon.E(pk(g_2, 0), w(t_i), r)\\ &= \varepsilon.E(pk(g_2, x_i), a_i + w(t_i), r)\end{aligned}$$

and $\varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r) = \varepsilon.E(pk(g_2, x_i), a_i, r)$. We can refer to $\varepsilon.E(pk(g_2, 1)^{x_i+t_i}, a_i, r)$ as encryption of a random number, so the ciphertext is a random 'fake' ciphertext. According to the security of PKE ε and the equivalent between IND-security and semantic security of PKE, $\varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r)$ should be indistinguishable from a random number. Therefore $\varepsilon.E(pk(g_2, 1)^{x_i}, a_i, r)$ and $\varepsilon.E(pk(g_2, 1)^{x_i+t_i}, a_i, r)$ are indistinguishable.

Else, $\forall b \in Z_p$, $\varepsilon.E(pk(g_2,0),b,r) \neq \varepsilon.E(pk(g_2,t_i),0,r)$. If $\exists c,d \in Z_p$, $\varepsilon.E(pk(g_2,0),c,r) = \varepsilon.E(pk(g_2,0),d,r)$, then

$$\begin{aligned} c &= \varepsilon.Decrypt(sk, \varepsilon.C(g_2,r), \varepsilon.E(pk(g_2,0),c,r)) \\ &= \varepsilon.Decrypt(sk, \varepsilon.C(g_2,r), \varepsilon.E(pk(g_2,0),d,r)) \\ &= d \end{aligned}$$

Since $G_T = p$, we have that $G_T = \{\varepsilon.E(pk(g_2,0),b,r)\}_{b\in Z_p}$. So

$$\{\varepsilon.E(pk(g_2,0),b,r)\}_{b\in Z_p} \cap \varepsilon.E(pk(g_2,t_i),0,r) = G_T \cap \varepsilon.E(pk(g_2,t_i),0,r) \neq \emptyset$$

By contradiction, $\forall t_i \in Z_p, \exists b \in Z_p$, s.t. $\varepsilon.E(pk(g_2,0),b,r) = \varepsilon.E(pk(g_2,t_i),0,r)$

Therefore, H4 and H5 are indistinguishable.

Hybrid H6: This is like H5 except that the challenge ciphertext is generated by invoking the algorithm $H6.Encrypt$ defined as follows:
$H6.Encrypt(msk, mpk, \boldsymbol{x})$: The algorithm computes the ciphertext for $\boldsymbol{x}$ in the following way:
$ct_0 = \varepsilon.C(r^2, g_1), ct_1 = \varepsilon.E(PK, 1, r), ct_{a,i} = \varepsilon.E(pk(g_1,0), a_i, r)$.
For $i \in [n]$, $ct_{x,i} = ct_1^{t_i} \cdot \varepsilon.E'(s_i, x_i, Ct_0, \tilde{r}), ct_{a,x,i} = \varepsilon.E(pk(g_2,1)^{x_i+t_i}, a_i, r)$

proc Initialize(λ)	**proc LR**($\boldsymbol{x_0}, \boldsymbol{x_1}$)
$(mpk, msk) \leftarrow_R H2.Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R H6.Encrypt(msk, mpk, \boldsymbol{x_b})$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

Under the $l-ciphertext-reproducibility$ of ε, H5 and H6 are indistinguishable.

Hybrid H7: This is like H8 except that the challenge ciphertext is generated by invoking the algorithm $\varepsilon.Encrypt$

proc Initialize(λ)	**proc LR**($\boldsymbol{x_0}, \boldsymbol{x_1}$)
$(mpk, msk) \leftarrow_R H2.Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R \varepsilon.E(mpk, \boldsymbol{x_b} + \boldsymbol{t})$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

By linear ciphertext homomorphism of ε, H6 = H7.

Hybrid H8: This is like H7 except that the master public key is generated by invoking the algorithm $Setup$.

proc Initialize(λ)	**proc LR**($\boldsymbol{x_0}, \boldsymbol{x_1}$)
$(mpk, msk) \leftarrow_R Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R \varepsilon.E(mpk, \boldsymbol{x_b} + \boldsymbol{t})$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′ =b)

Under the *l−public−key−reproducibility* of ε, H7 and H8 are indistinguishable.

Advantage of Any PPT Adversary in H8: Notice that $\boldsymbol{t} + \boldsymbol{x_b} - \boldsymbol{x_{1-b}} \in Z_p^n$. Let $\boldsymbol{t'} = \boldsymbol{t} + \boldsymbol{x_b} - \boldsymbol{x_{1-b}}, s_i' = s_i + (\boldsymbol{x_{1-b}} - \boldsymbol{x_b})_i sk$. Then $(\boldsymbol{s'}, \boldsymbol{t'})$ equally likely as $(\boldsymbol{s}, \boldsymbol{t})$ that gives exactly the same view by replacing $\boldsymbol{x_b}$ by $\boldsymbol{x_{1-b}}$.
Moreover, when analyzing $sk_F \leftarrow FE.KeyGen(F, msk)$, since $s_i' + t_i' sk = s_i + x_{1,b,i} sk - x_{b,i} sk + (t_i + x_{b,i} - x_{1-b,i}) sk = s_i + t_i sk$, so the sk_F are same for $(\boldsymbol{s}, \boldsymbol{t})$ and $(\boldsymbol{s'}, \boldsymbol{t'})$. Therefore, the advantage of the adversary in this game is 0.

C Proofs in Our FE Scheme for Cubic Functions

C.1 Proof of Theorem 2

Proof. Let $a_i = f_{i,j,k}, i = j = k \neq 0$, $b_{i,k} = f_{i,j,k}, i = j \neq k, i, k \neq 0$, $c_{i,j,k} = f_{i,j,k}, i \neq j \neq k, i, j, k \neq 0$, $d_i = f_{i,j,k}, i = j \neq 0, k = 0$, $e_{i,j} = f_{i,j,k}, i \neq j \neq 0, k = 0$, $f_i = f_{i,j,k}, i \neq 0 j = k = 0$, $g = f_{i,j,k}, i = j = k = 0$ and

$$A_0 = \begin{pmatrix} a_{00}^0, \ldots, a_{0n}^0 \\ a_{10}^0, \ldots, a_{1n}^0 \\ \vdots \ddots \vdots \\ a_{n0}^0, \ldots, a_{nn}^0 \end{pmatrix}, \ldots, A_0 = \begin{pmatrix} a_{00}^n, \ldots, a_{0n}^n \\ a_{10}^n, \ldots, a_{1n}^n \\ \vdots \ddots \vdots \\ a_{n0}^n, \ldots, a_{nn}^n \end{pmatrix}.$$

Since $f(\boldsymbol{x}) = \boldsymbol{x}^T \begin{pmatrix} \boldsymbol{x}^T A_0 \boldsymbol{x} \\ \boldsymbol{x}^T A_1 \boldsymbol{x} \\ \vdots \\ \boldsymbol{x}^T A_n \boldsymbol{x} \end{pmatrix}$, we can get the following equations:

$$\begin{cases} a_{00}^0 = g \\ a_{ii}^i = a_i, i > 0 \\ a_{ii}^0 + (a_{0i}^i + a_{i0}^i) = d_i, i > 0 \\ (a_{i0}^0 + a_{0i}^0) + a_{00}^i = f_i, i > 0 \\ (a_{ij}^0 + a_{ji}^0) + (a_{0i}^j + a_{i0}^j) + (a_{j0}^i + a_{0j}^i) = e_{i,j}, i > j > 0 \\ (a_{ik}^i + a_{ki}^i) + a_{ii}^k = b_{i,k}, i > k > 0 \\ (a_{jk}^i + a_{kj}^i) + (a_{ik}^j + a_{ki}^j) + (a_{ij}^k + a_{ji}^k) = c_{i,j,k}, i > j > k > 0 \end{cases} \quad (1)$$

Since $\forall i \in [n], \boldsymbol{y_0} A_i \boldsymbol{y_0} = \boldsymbol{y_1} A_i \boldsymbol{y_1}$, where $\boldsymbol{y_u} = (y_{u0}, y_{u1}, \ldots, y_{un}), u = 1, 2$ we can get the following equations:

$$\begin{cases} \sum_{i=1}^{n}(a_{i0}^0 + a_{0i}^0)(y_{0i} - y_{1i}) + \sum_{i=1}^{n} a_{ii}^0(y_{0i}^2 - y_{1i}^2) + \sum_{i>j=1}^{n}(a_{ij}^0 + a_{ji}^0)(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0 \\ \vdots \\ \sum_{i=1}^{n}(a_{i0}^n + a_{0i}^n)(y_{0i} - y_{1i}) + \sum_{i=1}^{n} a_{ii}^n(y_{0i}^2 - y_{1i}^2) + \sum_{i>j=1}^{n}(a_{ij}^n + a_{ji}^n)(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0 \end{cases} \tag{2}$$

Putting Eqs. (1) and (2) together, we can get that:

$$\begin{cases} a_{00}^0 = g \\ a_{ii}^i = a_i, i > 0 \\ \sum_{i=1}^{n}(f_i - a_{00}^i)(y_{0i} - y_{1i}) + \sum_{i=1}^{n}(d_i - (a_{0i}^i + a_{i0}^i))(y_{0i}^2 - y_{1i}^2) + \\ \sum_{i>j=1}^{n}(e_{i,j} - (a_{0i}^j + a_{i0}^j) - (a_{j0}^i + a_{0j}^i))(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0, i > j > 0 \\ \sum_{i=1}^{n}(a_{i0}^1 + a_{0i}^1)(y_{0i} - y_{1i}) + \sum_{i=1}^{n}(b_{i,1} - (a_{i1}^i + a_{1i}^i))(y_{0i}^2 - y_{1i}^2) + \\ \sum_{i>j=1}^{n}(c_{i,j,1} - (a_{j1}^i + a_{1j}^i + a_{i1}^j + a_{1i}^j))(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0, i > j > 0 \\ \vdots \\ \sum_{i=1}^{n}(a_{i0}^n + a_{0i}^n)(y_{0i} - y_{1i}) + \sum_{i=1}^{n}(b_{i,n} - (a_{in}^i + a_{ni}^i))(y_{0i}^2 - y_{1i}^2) + \\ \sum_{i>j=1}^{n}(c_{i,j,n} - (a_{jn}^i + a_{nj}^i + a_{in}^j + a_{ni}^j))(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0, i > j > 0 \end{cases} \tag{3}$$

Now, we will show that the system of linear Eq. (3) is solvable, i.e., its coefficient matrix is full rank.

Notice that a_{00}^0 only occurs in the first equation of (3), each a_{ii}^i only occurs in one equation of $\{a_{ii}^i = a_i, i > 0\}$, each a_{00}^i only occurs in one equation of $\{\sum_{i=1}^{n}(f_i - a_{00}^i)(y_{0i} - y_{1i}) + \sum_{i=1}^{n}(d_i - (a_{0i}^i + a_{i0}^i))(y_{0i}^2 - y_{1i}^2) + \sum_{i>j=1}^{n}(e_{i,j} - (a_{0i}^j + a_{i0}^j) - (a_{j0}^i + a_{0j}^i))(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0, i > j > 0\}$, each $(a_{it}^j + a_{ti}^j)$ only occurs in one equation of $\{\sum_{i=1}^{n}(a_{i0}^t + a_{0i}^t)(y_{0i} - y_{1i}) + \sum_{i=1}^{n}(b_{i,n} - (a_{it}^i + a_{ti}^i))(y_{0i}^2 - y_{1i}^2) + \sum_{i>j=1}^{n}(c_{i,j,n} - (a_{jt}^i + a_{tj}^i + a_{it}^j + a_{ti}^j))(y_{0i}y_{0j} - y_{1i}y_{1j}) = 0, i > j > 0\}$. So the coefficient matrix is full rank.

C.2 Proof of Theorem 3

Proof. We proof security via a sequence of hybrid experiments, and then we show they are indistinguishable.

Hybrid H1: This is the s-IND-CPA game:

proc Initialize$(\lambda, \boldsymbol{x_0}, \boldsymbol{x_1})$	**proc LR()**
$(mpk, msk) \leftarrow_R Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R Encrypt(mpk, \boldsymbol{x_b})$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

Hybrid H2: This is like H1 except that the master public key is generated by invoking the algorithm $H2.Setup$ defined as follows:
$H2.Setup(1^\lambda, 1^n, \boldsymbol{x_0}, \boldsymbol{x_1})$: The algorithm samples $(mpk1, msk1) \leftarrow QFE.Setup(1^\lambda, 1^n)$. Randomly choose $\boldsymbol{t} = (t_0, \ldots, t_n) \leftarrow_R Z_p^{n+1}$. Then sets $t_i^0 = (\frac{x_{1i}}{x_{0i}} t_i^{-1})^{-1}$ and $t_i^1 = (\frac{x_{0i}}{x_{1i}} t_i^{-1})^{-1}$. Return $mpk := (mpk1, \boldsymbol{t^b})$ and $msk := (msk1)$.

Z_p is a field, so t_i^0 and t_i^1 are uniformly distributed in Z_p. Therefore, H2 and H1 are indistinguishable.

Hybrid H3: This is like H2 except that the challenge ciphertext is generated by invoking the algorithm $H3.Encrypt(mpk, \boldsymbol{x_b})$ defined as follows:
$H3.Encrypt(mpk, \boldsymbol{x_b})$: Choose a matrix $W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ from $Z_p^{*(2\times 2)}$, where $w_{12} = u_1^2, w_{22} = u_2^2$ and $WW^{-1} = I$. Randomly choose $\boldsymbol{r} = (r_0, \ldots, r_n), \boldsymbol{a} = (a_0, \ldots, a_n) \in Z_p^{n+1}$. Then computes $Ct_{u_1 x} = QFE.Encrypt(mpk1, u_1 \boldsymbol{x_{1-b}}, Ct_{u_2 x} = QFE.Encrypt(mpk1, u_2 \boldsymbol{x_{1-b}})$. Sets $Ct_{w,x,i} = (r_i, (t_i^b)^{-1} x_{bi}) W^{-1}$, and $Ct_{a,1,i} = w_{11} a_i, Ct_{a,2,i} = w_{21} a_i, Ct_{ar} = \sum_{i=0}^n a_i r_i$.
Return $Ct_x = (Ct_{u_1 x}, Ct_{u_2 x}, \{Ct_{w,x,i}, Ct_{a,i,1}, Ct_{a,2,i}\} i \in [n], Ct_{a,r})$.

Firstly, we show that $Ct'_{u_1 x} = QFE.Encrypt(mpk1, u_1 \boldsymbol{x_b})$ and $Ct_{u_1 x} = QFE.Encrypt(mpk1, u_1 \boldsymbol{x_{1-b}})$ are indistinguishable. For any $f \in \mathcal{F}$, exists a set of matrices $\{A_i\}_{i \in [n]}$, s.t. $f(\boldsymbol{x}) = \sum_{i=0}^n x_i \boldsymbol{x}^T A_i \boldsymbol{x}$ and $\boldsymbol{x_0}^T A_i \boldsymbol{x_0} = \boldsymbol{x_1}^T A_i \boldsymbol{x_1}, i \in [n]$. So,

$$(\boldsymbol{x_{1-b}})^T A_i (\boldsymbol{x_{1-b}}) = \boldsymbol{x_b}^T A_i \boldsymbol{x_b}, i \in [n]$$

By the s-IND-CPA of the QFE scheme, these two should be indistinguishable. Similarly, $Ct'_{u_2 x} = QFE.Encrypt(mpk1, u_2 \boldsymbol{x_b})$ and $Ct_{u_2 x} = QFE.Encrypt(mpk1, u_2 \boldsymbol{x_{1-b}})$ are also indistinguishable.

Then, we show that $Ct'_{w,x,i} = (r_i, t_i^{-1} x_{bi}) W^{-1}$ and $Ct_{w,x,i} = (r_i, (t_i^b)^{-1} x_{bi}) W^{-1}$ are indistinguishable. t_i and t_i^b are hidden by the matrix W, i.e. without knowing about W, the adversary cannot determine whether t_i or t_i^b is used in the encryption. So the only thing we should prove is that the adversary cannot recover W. When considering the ciphertexts $Ct_{a,1,i} = w_{11} a_i$ and $Ct_{a,2,i} = w_{21} a_i$, we find that there exists $\alpha \in Z_p$, s.t. $w_{11} = \alpha w_{21}$. So $Ct_{a,1,i} = w_{11} a_i = \alpha w_{21} a_i = \alpha Ct_{a,2,i}$. Actually, there are $n+1$ unknown values $(a_1, ..., a_n, w_{11})$ but only n effective equations, so w_{11} are not achievable. It is easy to see that w_{11} in other parts of ciphertext is also hidden by some random values.

Therefore, H3 and H2 are indistinguishable.

Hybrid H4: This is like H3 except that the challenge ciphertext is generated by invoking the algorithm $CFE.Encrypt$ as follows:

proc Initialize$(\lambda, \boldsymbol{x_0}, \boldsymbol{x_1})$	**proc LR()**
$(mpk, msk) \leftarrow_R Setup(1^\lambda, 1^n)$	$Ct* \leftarrow_R Encrypt(mpk, \boldsymbol{x_{1-b}})$
$\mathcal{F} \leftarrow \emptyset$	Return $Ct*$
Return mpk	
proc KeyGen(F)	**proc Finalize**(b')
$\mathcal{F} \leftarrow \mathcal{F} \cup F$	If $\exists F \in \mathcal{F}$ s.t. $f(F, \boldsymbol{x_0}) \neq f(F, \boldsymbol{x_1})$
$sk_F \leftarrow KeyGen(msk, F)$	then return false
Return sk_F	Return (b′=b)

In $H3.Encrypt$, $Ct'_{w,x,i} = (r_i, (t_i^b)^{-1} x_{bi}) W^{-1} = (r_i, \frac{x_{1-b,i}}{x_{bi}} t_i^{-1} x_{bi}) W^{-1} = (r_i, x_{1-b,i} t_i^{-1}) W^{-1}$. In $CFE.Encrypt$, $Ct'_{w,x,i} = (r_i, x_{1-b,i} t_i^{-1}) W^{-1}$. So H4 = H3.

Advantage of Any PPT Adversary in H4: In H4, the challenge ciphertext is a valid ciphertext for the message $\boldsymbol{x_{1-b}}$. So it gives the same view by replacing $\boldsymbol{x_b}$ by $\boldsymbol{x_{1-b}}$. Therefore, the advantage of any adversary in this game is 0.
Notice that we only consider the situation that $x_{0i} \neq 0, x_{1i} \neq 0, i \in [n]$. And the proof can be extended when considering 0. We need to modify the construction of $\boldsymbol{t^b}$ and $H3.Encrypt$ as follows:

1. If $x_{0i} = x_{1i} = 0$, then $t_i^0 = t_i^1 = t_i$, and $Ct_{w,x.i} = (r_i, (t_i^b)^{-1} x_{bi}) W^{-1}$.
2. If $x_{bi} = 0, x_{1-b,i} \neq 0$, then $t_i^b = x_{1-b,i} t_i^{-1} - x_{bi}$, and $Ct_{w,x,i} = (r_i, t_i^b + x_{bi}) W^{-1}$.
3. If $x_{bi} \neq 0, x_{1-b,i} = 0$, then $t_i^b = -x_{bi} t_i^{-1}$, and $Ct_{w,x,i} = (r_i, t_i^b + t_i^{-1} x_{bi}) W^{-1}$.

The remaining proof can be easily extended from our proof.

References

1. Abdalla, M., et al.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_13
2. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
3. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Better security for functional encryption for inner product evaluations. IACR Cryptology ePrint Archive, Report 2016/11 (2016)
4. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12

5. Apon, D., Döttling, N., Garg, S., Mukherjee, P.: Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. In: LIPIcs-Leibniz International Proceedings in Informatics, vol. 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
6. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 67–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_3
7. Bitansky, N., Lin, H., Paneth, O.: On removing graded encodings from functional encryption. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 3–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_1
8. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_14
9. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
10. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_26
11. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
12. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
13. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_3
14. Chen, Y., Gentry, C., Halevi, S.: Cryptanalyses of candidate branching program obfuscators. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 278–307. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_10
15. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_1
16. Cheon, J.H., Hong, S., Lee, C., Son, Y.: Polynomial functional encryption scheme with linear ciphertext size. IACR Cryptology ePrint Archive, Report 2018/585 (2018)
17. Coron, J.-S., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of GGH15 multilinear maps. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 607–628. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_21
18. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 164–195. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_7

19. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016)
20. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure attribute based encryption from multilinear maps. Cryptology ePrint Archive, Report 2014/622 (2014)
21. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Symposium on Theory of Computing Conference, STOC 2013, Palo Alto, California, USA, 1–4 June 2013, pp. 555–564. ACM (2013)
22. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
23. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
24. Kim, S., Lewi, K., Mandal, A., Montgomery, H.W., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. Cryptology ePrint Archive, Report 2016/440
25. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: Proceedings of the IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, Brunswick, New Jersey, USA, 9–11 October 2016, pp. 11–20. IEEE (2016)
26. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
27. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
28. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
29. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_27
30. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_4
31. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 678–697. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_33