# Hierarchical functional encryption for linear transformations

Shiwei ZHANG

Yi MU

Guomin YANG
*Singapore Management University*, gmyang@smu.edu.sg

Xiaofen WANG

## Citation

# Hierarchical Functional Encryption
# for Linear Transformations

Shiwei Zhang[1]([✉]), Yi Mu[1], Guomin Yang[1], and Xiaofen Wang[2,3]

[1] Institute of Cybersecurity and Cryptology, School of Computing
and Information Technology, University of Wollongong, Wollongong, Australia
{sz653,ymu,gyang}@uow.edu.au
[2] The Center for Cyber Security, University of Electronic Science
and Technology of China, Chengdu 611731, Sichuan, China
wangxuedou@sina.com
[3] Guangxi Key Laboratory of Trusted Software,
Guilin University of Electronic Technology, Guilin 541004, Gunagxi, China

**Abstract.** In contrast to the conventional all-or-nothing encryption, functional encryption (FE) allows partial revelation of encrypted information based on the keys associated with different functionalities. Extending FE with key delegation ability, hierarchical functional encryption (HFE) enables a secret key holder to delegate a portion of its decryption ability to others and the delegation can be done hierarchically. All HFE schemes in the literature are for general functionalities and not very practical. In this paper, we focus on the functionality of linear transformations (i.e. matrix product evaluation). We refine the definition of HFE and further extend the delegation to accept multiple keys. We also propose a generic HFE construction for linear transformations with IND-CPA security in the standard model from hash proof systems. In addition, we give two instantiations from the DDH and DCR assumptions which to the best of our knowledge are the first practical concrete HFE constructions.

**Keywords:** Hierarchical · Functional encryption · Matrix product · Hash proof system

## 1 Introduction

Encryption can provide confidentiality and privacy for our sensitive information in a variety of ways. Typically, conventional encryption is in an all-or-nothing fashion that an entity is able to either access all the encrypted information or nothing, excepting the message length. In contrast, functional encryption [8] allows partial revelation of encrypted information based on the keys associated with different functionalities. Precisely, Alice can encrypt some message $m$ under Bob's public key, and send the ciphertext to a public domain where Charlie can access. Later, Bob issues a secret key for a function $f$ to Charlie using a master secret key corresponding to Bob's public key. As a result, Charlie is able to learn

$f(m)$ from the ciphertext and the secret key received from Bob but nothing else. Usually, the function $f$ is a universal function for a function class $\mathcal{F}$ indexed by a function key $k$ from a key space $K$ s.t. $\mathcal{F} = \{f_k \mid k \in K\}$. In this case, Bob sends a secret key for a function key $k$ to Charlie who learns $f(k, m)$ or simply $f_k(m)$.

When Bob can control the amount of the information that Charlie can reveal, it leads to a natural question of whether Charlie is able to further let other people (e.g. David) obtain a narrower portion of his decryption ability. Hierarchical functional encryption (HFE) [5,9,11] gives an affirmative answer. In HFE, Charlie with a secret key associated with a function $f$ is able to generate a new secret key associated with a composed function $f' \circ f$ for David without the help of Bob. Upon receiving the key from Charlie, David can learn $f'(f(m))$ but nothing else of the original message $m$. It is worth noting that the above delegation process is repeatable that David can delegate a portion of his decryption ability to other people, making the hierarchy grow deeper and deeper.

In this paper, we focus on HFE for the functionality of matrix product evaluation, which implies the function class of linear transformation via transformation matrices. More precisely, the functionality is defined as $f(\mathbf{A}, \mathbf{X}) = f_\mathbf{A}(\mathbf{X}) = \mathbf{A}\mathbf{X}$ where the matrix $\mathbf{A}$ is the transformation matrix (i.e. the function key) and the matrix $\mathbf{X}$ is the message to be encrypted. It is easy to see that such a functionality is a natural generalisation of the functionality of inner product evaluation [1,2,4,19].

There are a few practical applications of HFE for the functionality of matrix product evaluation, including but not limited to descriptive statistics. Differing from the functional encryption for inner product evaluation [1,2,4,19], our proposed HFE allows the evaluation to be done in a levelled manner. We take the calculation of the weighted sum as an example. Suppose Alice is a researcher in a consulting firm conducting marketing research on the demand of various products and the company will sell Alice's results to different clients. Once the research results are ready, Alice encrypts the demand level of different areas for each product (e.g. $\mathbf{X} = \begin{bmatrix} 2\ 1\ 9\ 0\ 6\ 2\ 5\ 6\ 1 \end{bmatrix}^\top$ for the demand of coffee in different regions) under her company's public key. As the manager of Alice's company, Bob has the master secret key and decides to sell Alice's result in different levels and at different prices. Suppose Charlie wants to buy from Bob some information he is interested in. He does not want all the details but a overall score for the demand of coffee, and he is more interested in the areas near his store, i.e. those areas will have a higher weight. Hence, Charlie buys a secret key of a weighted matrix (e.g. $\mathbf{A} = \begin{bmatrix} 0\ 1\ 2\ 3\ 4\ 3\ 2\ 1\ 0 \end{bmatrix}$) from Bob for a low price so that he can learn a summary of the market demand (e.g. $\mathbf{A}\mathbf{X} = 65$ for coffee).

With HFE, Bob is also able to sell Alice's result to some regional resellers (or proxies), e.g. David. Unlike Charlie, David wants the details of some regions and a summary for other regions. Thus he buys a secret key of a weighted matrix (e.g. $\mathbf{A} = \left[\begin{array}{c|c} \mathbf{I}_5 & \mathbf{0}_{5,4} \\ \hline \mathbf{0}_{1,5} & 5\ 4\ 1\ 2 \end{array}\right]$) from Bob for a higher price so that he learns Alice's partial result (e.g. $\mathbf{A}\mathbf{X} = \begin{bmatrix} 2\ 1\ 9\ 0\ 6\ 38 \end{bmatrix}^\top$ for coffee). Later, David can

resell what he obtained from Bob to other clients in his region. For example, another store manager Eve in David's region is interested in Alice's result, and decides to buy some market information for her nearby regions from David. In the reselling process, David uses his secret key for $\mathbf{A}$ to generate a new secret key for $\mathbf{C} = \mathbf{BA}$ based on Eve's demand (e.g. $\mathbf{B} = \begin{bmatrix} 20 & 15 & 10 & 5 & 2 & 0 \end{bmatrix}$ and thus $\mathbf{C} = \begin{bmatrix} 20 & 15 & 10 & 5 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}$). Once Eve gets the key, she can learn $\mathbf{CX}$ from Alice's research (e.g. $\mathbf{CX} = 157$ for coffee). On the other hand, David can also resell his data obtained from Bob to other resellers in different levels and at different prices.

There are many potential applications of HFE for linear transformation, such as those related to descriptive statistics as demonstrated above. In this paper, we introduce and formalise this useful cryptographic primitive and present a generic construction of it. We also show two practical instantiations of the generic construction based on some standard assumptions.

## 1.1  Related Work

The formal study of functional encryption (FE) was initiated by Boneh et al. [8] with syntax and security model defined for general functionality. In this paper, we focus on the public-key FE. Other than predicate encryption [15] (a subclass of FE) and theoretical constructions [14] for arbitrary functionality, Abdalla et al. [2] focused on the functionality of inner product evaluation that only the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ of the encrypted vector $\boldsymbol{y}$ is revealed with a secret key for the vector $\boldsymbol{x}$. Furthermore, the authors challenged to construct practical schemes with such a functionality, and proposed an s-IND-CPA secure generic construction from any s-IND-CPA public key encryption (PKE) schemes with the properties of randomness reuse, linear key homomorphism, and linear ciphertext homomorphism under shared randomness. Precisely, s-IND-CPA (or selective IND-CPA) is a weaker notion of the standard indistinguishability under chosen plaintext attacks (IND-CPA) where the adversary is required to submit the target plaintexts before receiving the public key from the challenger. Based on [2], Abdalla et al. [1] enhanced their previous generic construction [2] to the standard IND-CPA security from any s-IND-CPA PKE schemes with the properties of linear key homomorphism, linear ciphertext homomorphism under shared randomness, $\ell$-public-key reproducibility, and $\ell$-ciphertext reproducibility. The authors also showed several instantiations from various s-IND-CPA PKE schemes based on Decisional Diffie-Hellman (DDH) assumption [13], Decisional Composite Residuosity (DCR) assumption [10,17], and Learning With Error (LWE) assumption [18]. Independent from [1], Agrawal et al. [4] constructed function encryption schemes for inner products directly from DDH, DCR, and LWE assumptions instead of a generic construction, and obtained better efficiency. It is worth noting that the proofs of the DDH construction and the DCR construction in [4] are implicitly built on Hash Proof System (HPS) [12] as the HPS makes the secret keys simulatable and IND-CPA security achievable. In contrast to [4], Zhang et al. [19] recently provided another framework of constructing functional encryption for inner products (FE-IP) explicitly from HPS with the properties of

key linearity, hash linearity and diversity. Also, the authors further improved the security from IND-CPA to indistinguishability under adaptive chosen ciphertext attacks (IND-CCA).

There are several extension works [3,5,6] on functional encryption. As one of those extensions, Hierarchical functional encryption (HFE) enables delegation capability, which is initially mentioned in [5], generalising the notions of hierarchical identity-based encryption [7] and hierarchical predicate encryption [16] for more expressive access controls. In particular, [5,9,11] define HFE as a normal FE with an extra delegation algorithm that takes a function key $\mathsf{SK}_f$ and a function $f'$ and outputs a function key $\mathsf{SK}_{f'\circ f}$. As mentioned above, the newly generated key $\mathsf{SK}_{f'\circ f}$ allows revelation of $f'(f(m))$ but nothing else from the encrypted message $m$. In the literature, Ananth et al. [5] and Chandran et al. [11] purposed general purpose HFE constructions direct from indistinguishable obfuscation ($i\mathcal{O}$) with fixed depth where the delegation process can only be proceeded for a fixed number of times. As an improvement, Brakerski et al. [9] proposed a generic transformation from any general purpose FE to a general purpose HFE with unbounded depth. The transformation requires a private-key encryption scheme and a puncturable pseudorandom function family, and does not rely on $i\mathcal{O}$. As far as we know, all the HFE schemes in the literature are general purposed and not very practical.

## 1.2   Our Contribution

In this paper, we refine and simplify the definition of *hierarchical functional encryption* (HFE) originally from [5,9,11]. We merge the delegation algorithm $\mathsf{SK}_{f'\circ f} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, f')$ with the key generation algorithm $\mathsf{SK}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$ to form a new key generation algorithm $\mathsf{SK}_{f'\circ f} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_f, f')$, making the master secret key equivalent to a secret key of an identity function $\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{SK}_{\mathsf{id}}$. As a result, our definition of HFE consists of four algorithms instead of five algorithms, and it is compatible with the IND-CPA security model defined for the normal functional encryption, which is much simpler than the previously defined model for HFE. In terms of the evolution of encryption, our definition of HFE is a more natural generalisation of hierarchical identity-based encryption (HIBE) [7].

As an extension of HFE, we introduce the notion of *extended hierarchical functional encryption* (eHFE), allowing delegation from multiple secret keys. Precisely, in eHFE, the key generation algorithm takes $n$ secret keys $\mathsf{SK}_{f_1}, \ldots, \mathsf{SK}_{f_n}$ for functions $f_1, \ldots, f_n$ correspondingly and a delegation function $f'$, and it generates a new secret key $\mathsf{SK}_f$ for a function $f$ such that $f(x) = f'(f_1(x), \ldots, f_n(x))$.

We derive the definition of *hierarchical function encryption for linear transformation* (HFE-LT) from HFE (similarly, eHFE-LT from eHFE) for the functionality of matrix product evaluation. Different from the previous general results which are of theoretical interest, we propose a generic and practical construction of HFE-LT. It is worth noting that our construction has unbounded depth in

delegation (Table 1). Since the HFE-LT scheme cannot be trivially constructed from function encryption for inner products, our generic construction is explicitly built from hash proof systems (HPS) with key linearity, hash linearity and diversity, and achieves IND-CPA security in the standard model. As an extension, we also adapt our generic HFE-LT construction to a generic eHFE-LT construction. We provide two practical HFE-LT instantiations based on the Decisional Diffie-Hellman (DDH) assumption and the Decisional Composite Residuosity (DCR) assumption. In the DDH instantiation, the decryption space is limited to be of polynomial size so that the decryption can be done in polynomial time. However, in the DCR instantiation, there is no such limitation. In addition, both instantiations can be extended to the eHFE-LT setting.

**Table 1.** Comparison of different schemes

| Scheme | Functionality | Assumption | Hierarchical | Depth | Multi-key |
|---|---|---|---|---|---|
| [5] | Generic | $i\mathcal{O}$ | Yes | Constant | No |
| [11] | Generic | $i\mathcal{O}$ + HIBE | Yes | Fixed | No |
| [9] | Generic | Generic FE | Yes | Unbounded | No |
| [1] | Inner product | s-IND-CPA PKE: DDH/DCR/LWE | No | × | No |
| [4] | Inner product | DDH/DCR/LWE | No | × | No |
| Our HFE-LT | Matrix product | Diverse HPS: DDH/DCR | Yes | Unbounded | No |
| Our eHFE-LT | Matrix product | Diverse HPS | Yes | Unbounded | Yes |

### 1.3 Paper Organisation

The rest of this paper is organised as follows. It is strongly suggested to read the preliminaries in Sect. 2, especially the notations used in this paper. In Sect. 3, we review the subset membership problems, hash proof system, and functional encryption. As our main contribution, we refine HFE and define HFE-LT in Sect. 3.1. The generic construction from HPS is proposed in Sect. 3.2 and follows the security proof in Sect. 3.3. In Sect. 3.4, we adapt HFE to eHFE in terms of the definition, the generic construction, and the security proof. After that, we propose our concrete HFE-LT constructions instantiated from DDH and DCR assumptions in Sect. 4. Finally, the conclusion is addressed in Sect. 5.

## 2 Preliminaries

### 2.1 Notations

Let $\in_R$ denote random sampling that $x \in_R X$ means that $x$ is uniformly and randomly chosen from a set or distribution $X$. Let $\circ$ denote the function composition that $(g \circ f)(x) = g(f(x))$, and id denote the identity function that $\mathsf{id}(x) = x$ and $f \circ \mathsf{id} = f$. Let $\lfloor x \rfloor$ denote the floor function of a real number $x$.

In the group computation, the function sca denotes the inverse operation to the scalar multiplication that $a = \text{sca}_x(b) \iff b = ax$ where $X$ is an additive group generated by $x$, $a \in \mathbb{Z}_{|X|}$, and $b \in X$. It is an analogue of the logarithm operation in a multiplicative group.

In the matrix computation, let $\mathbf{X}_{m \times n}$ be a matrix of size $m \times n$, and $\mathbf{I}_n$ be the identity matrix of size $n$. If the size is clear in the context, $\mathbf{X}$ and $\mathbf{I}$ are used for short with size omitted. Let $\top$ denote matrix transpose. In addition to the standard matrix addition and multiplication operations, we define the following notations for better representations. Let $f$ be a function that takes an element from $X$ with other fixed parameters as input, $\mathbf{X} \in X^{m \times n}$ be a matrix, and $\overset{\text{def}}{=}$ denote equal by definition. We define that

$$
\mathbf{X} \overset{\text{def}}{=} \begin{bmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{bmatrix}, \quad f(\mathbf{X}, \dots) \overset{\text{def}}{=} \begin{bmatrix} f(X_{1,1}, \dots) & \cdots & f(X_{1,n}, \dots) \\ \vdots & \ddots & \vdots \\ f(X_{m,1}, \dots) & \cdots & f(X_{m,n}, \dots) \end{bmatrix}.
$$

If $f$ is an additive homomorphism that $f(a) + f(b) = f(a + b)$, we have $f(\mathbf{A}) + f(\mathbf{B}) = f(\mathbf{A} + \mathbf{B})$ where $\mathbf{A}$ and $\mathbf{B}$ are two matrices with the same size. Besides that, we can have more complex scalar multiplication as

$$
\begin{aligned}
\mathbf{A}f(\mathbf{B}) &= \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{bmatrix} \begin{bmatrix} f(B_{1,1}) & \cdots & f(B_{1,l}) \\ \vdots & \ddots & \vdots \\ f(B_{n,1}) & \cdots & f(B_{n,l}) \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{n} A_{1,i}f(B_{i,1}) & \cdots & \sum_{i=1}^{n} A_{1,i}f(B_{i,l}) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} A_{m,i}f(B_{i,1}) & \cdots & \sum_{i=1}^{n} A_{m,i}f(B_{i,l})) \end{bmatrix} \\
&= \begin{bmatrix} f(\sum_{i=1}^{n} A_{1,i}B_{i,1}) & \cdots & f(\sum_{i=1}^{n} A_{1,i}B_{i,l}) \\ \vdots & \ddots & \vdots \\ f(\sum_{i=1}^{n} A_{m,i}B_{i,1}) & \cdots & f(\sum_{i=1}^{n} A_{m,i}B_{i,l})) \end{bmatrix} = f(\mathbf{AB})
\end{aligned}
$$

where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times l$ matrix. Similarly, we have $f(\mathbf{A})\mathbf{B} = f(\mathbf{AB})$. In addition, the symbol $(\mathbf{A} \mid \mathbf{B})$ denotes an argumented matrix of two matrices $\mathbf{A}$ and $\mathbf{B}$ with the same row size. It can be generalised as $(\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_n) \overset{\text{def}}{=} ((\cdots ((\mathbf{A}_1 \mid \mathbf{A}_2) \mid \mathbf{A}_3) \mid \cdots) \mid \mathbf{A}_n)$.

## 2.2   Subset Membership Problems

Subset Membership Problem (SMP) is a problem class introduced by Cramer and Shoup [12]. Many standard problems can be classified to SMP, including DDH and DCR problems.

**Definition 1 (Subset Membership Problem).** *Let $L \subset X, W$ be three non-empty sets, and $R = \{(x, w) \mid x \in L\} \subset X \times W$ be a binary relation where $w$ is a witness of a word $x$ in the language $L$. Let $\Lambda = (X, L, W, R)$ be a problem*

instance, $x \in_R L$, and $x' \in_R X \setminus L$. *Given two probability distributions* $\mathcal{D}_L = \{(\Lambda, x)\}$ *and* $\mathcal{D}_{X \setminus L} = \{(\Lambda, x')\}$, *there is an algorithm* $\mathcal{A}$ *that distinguishes* $\mathcal{D}_L$ *and* $\mathcal{D}_{X \setminus L}$ *with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SMP}} = \left| \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_L)] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{X \setminus L})] \right|$$

An SMP is computational hard if and only if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SMP}}$ is negligible for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$.

### 2.3   Hash Proof System

In this subsection, the hash proof system introduced by Cramer and Shoup [12] and extended by Zhang et al. [19] is reviewed.

**Definition 2 (Hash Proof System).** *A hash proof system (HPS) associated with subset membership problems consists of the following five polynomial time algorithms:*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$: *The randomised system setup algorithm takes a security parameter* $1^\lambda$ *as input, and specifies an SMP instance* $\Lambda = (X, L, W, R)$ *where the hash domain is* $X$. *The algorithm further specifies a secret hash key space* $K$, *a public hash key space* $S$, *and a hash codomain* $\Pi$. *After that, the algorithm packs all above descriptions and publishes a system-wide parameter* $\mathsf{param} = (X, L, W, R, K, S, \Pi)$.
- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$: *The randomised secret hash key generation algorithm takes a system parameter* $\mathsf{param}$, *and generates a random hash key* $\mathsf{SK} \in_R K$.
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$: *The deterministic public hash key generation algorithm takes a secret hash key* $\mathsf{SK} \in K$ *as input, and maps it to a public hash key* $\mathsf{PK} \in S$.
- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x)$: *The deterministic private evaluation algorithm takes a secret hash key* $\mathsf{SK} \in K$ *and a word* $x \in X$ *as inputs, and outputs a hash value* $\pi \in \Pi$ *of* $x$.
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x, w)$: *The deterministic public evaluation algorithm takes a public hash key* $\mathsf{PK} \in S$ *and a word* $x$ *in the language* $L$ *along with a corresponding witness* $w \in W$ *such that* $(x, w) \in R$, *and output a hash value* $\pi \in \Pi$ *of* $x$.

A HPS is required to be correct that the private evaluation algorithm $\mathsf{Hash}$ and the public evaluation algorithm $\mathsf{PHash}$ are equivalent when $x \in L$.

**Definition 3 (Correctness).** *A HPS is correct if*

$$\forall \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda), \quad \forall \mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param}), \quad \mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK}),$$
$$\forall (x, w) \in R, \quad \mathsf{Hash}(\mathsf{SK}, x) = \mathsf{PHash}(\mathsf{PK}, x, w).$$

The original definition of HPS from [12] is not sufficient for our schemes, and the following extended properties introduced by [19] are required.

**Definition 4 (Key Linearity).** *A HPS is linear key homomorphic if $K$ and $S$ are additive abelian groups and*

$$\forall a, b \in K, \quad \mathsf{PKGen}(a) + \mathsf{PKGen}(b) = \mathsf{PKGen}(a + b) \in S.$$

**Definition 5 (Hash Linearity).** *A HPS is linear hash homomorphic if $K$ and $\Pi$ are additive abelian groups and*

$$\forall a, b \in K, \quad \forall x \in X, \quad \mathsf{Hash}(a, x) + \mathsf{Hash}(b, x) = \mathsf{Hash}(a + b, x) \in \Pi.$$

**Definition 6 (Diversity).** *A HPS is diverse if*

$$\exists \pi \in \Pi \setminus \{0\}, \quad \forall x \in X \setminus L, \quad \exists \mathsf{SK} \in K, \quad \mathsf{PKGen}(\mathsf{SK}) = 0 \wedge \mathsf{Hash}(\mathsf{SK}, x) = \pi$$

*We call such a element $\pi$ as an element derived from the diversity.*

### 2.4 Functional Encryption

The definition and the security model of functional encryption by Boneh et al. [8] are reviewed as follows.

**Definition 7 (Functional Encryption).** *Let $f : K \times X \to Y$ be a universal function for a function class $\mathcal{F} = \{f_k \mid k \in K\}$ indexed by a function key space $K$, mapping the message space $X$ to the revelation space $Y$. A functional encryption (FE) for a function class $\mathcal{F}$ consists of the following four polynomial time algorithms:*

- *$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$: The randomised system setup algorithm takes a security parameter $1^\lambda$ as input, and generates system-wide parameters and a random pair of a master secret key $\mathsf{MSK}$ and a public key $\mathsf{PK}$.*
- *$\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$: The (probably) randomised secret key generation algorithm takes a master secret key $\mathsf{MSK}$ and a function key $k \in K$ as inputs, and computes a secret key $\mathsf{SK}$ for the function $f_k$.*
- *$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x)$: The randomised encryption algorithm takes a public key $\mathsf{PK}$ and a message $x \in X$ as inputs, and generates a ciphertext $C$ of the message $x$.*
- *$y \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$: The (probably) deterministic decryption algorithm takes a secret key $\mathsf{SK}$ for a function $f_k$ and a ciphertext $C$ of a message $x$ as inputs. The algorithm reveals $y = f_k(x) \in Y$ from the ciphertext $C$, and outputs it.*

Indistinguishability-based security is considered in this paper. Precisely, we consider the Indistinguishability under Chosen Plaintext Attacks (IND-CPA security) formulated by Boneh et al. [8]. In the IND-CPA game formally defined as follows, an adaptive adversary $\mathcal{A}$ tries to distinguish a target ciphertext from two messages $x_0$ and $x_1$ chosen by $\mathcal{A}$.

**Setup phase.** The challenger $\mathcal{S}$ runs the system setup algorithm $\mathsf{Setup}(1^\lambda)$ to generate a key pair $(\mathsf{MSK}, \mathsf{PK})$, and passes the public key $\mathsf{PK}$ to $\mathcal{A}$.

**Pre-challenge phase.** $\mathcal{A}$ can adaptively query the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ with a function key $k \in K$ to obtain a secret key $\mathsf{SK}$ for the function $f_k$. At the same time, $\mathcal{S}$ stores the queried function key $k$ in the list $\mathcal{K}$. The restriction is that $\mathcal{A}$ can only query the function key $k$ such that $f_k(x_0) = f_k(x_1)$ where $x_0$ and $x_1$ are the messages chosen by $\mathcal{A}$ in the challenge phase. Otherwise, winning the game is trivial since $\mathsf{Decrypt}(\mathsf{SK}, C) \neq f_k(x_{1-b})$.

**Challenge phase.** At some point, $\mathcal{A}$ outputs two messages $x_0$ and $x_1$. $\mathcal{S}$ randomly picks $b \in_R \{0, 1\}$, and generates a target ciphertext $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b)$. After that, $\mathcal{S}$ passes the target ciphertext $C$ to $\mathcal{A}$.

**Post-challenge phase.** $\mathcal{A}$ can further query the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ as in the pre-challenge phase with the same restriction.

**Guessing phase.** Eventually, $\mathcal{A}$ outputs an educated guess $b'$. If $b = b'$, $\mathcal{A}$ wins.

The advantage of the adversary $\mathcal{A}$ winning the IND-CPA game is

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} = \left| \Pr\left[ b = b' \mid \forall k \in \mathcal{K}, \, f_k(x_0) = f_k(x_1) \right] - \frac{1}{2} \right|$$

**Definition 8 (IND-CPA Security).** *An FE scheme is Indistinguishable under Chosen Plaintext Attacks (IND-CPA) if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}}$ for all adversary $\mathcal{A}$ winning the IND-CPA game in the polynomial time is a negligible function.*

## 3    Hierarchical Functional Encryption for Linear Transformations

### 3.1    Definition

In [5,9,11], the hierarchical functional encryption (HFE) is defined as a normal functional encryption with an extra delegation algorithm $\mathsf{Delegate}$. The (probably) randomised algorithm $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, f')$ takes a secret key $\mathsf{SK}_f$ for a function $f : X \to Z \subset Y$, and an another function $f' : Z \to Z' \subset Y$ as inputs where $Z, Z'$ are the images of the functions $f, f'$ correspondingly. It generates a new secret key $\mathsf{SK}_{f' \circ f}$ for the composed function $f' \circ f : X \to Z' \subset Y$. By observing the key generation algorithm $\mathsf{SK}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$ and the delegation algorithm $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, f')$, we find that the algorithm $\mathsf{KeyGen}$ is actually a special case of the algorithm $\mathsf{Delegate}$ when the master secret key $\mathsf{MSK}$ is considered as a secret key $\mathsf{SK}_{\mathsf{id}}$ of an identity function. More precisely, we have $\mathsf{KeyGen}(\mathsf{MSK}, k) \overset{\text{def}}{=} \mathsf{Delegate}(\mathsf{SK}_{\mathsf{id}}, f_k) \to \mathsf{SK}_{f_k \circ \mathsf{id}} = \mathsf{SK}_{f_k}$ with $\mathsf{MSK} \overset{\text{def}}{=} \mathsf{SK}_{\mathsf{id}}$. Therefore, the duplicated algorithm can be removed, and it becomes a more natural generalisation of hierarchical identity based encryption (HIBE) [7] as there is no $\mathsf{Delegate}$ algorithm in HIBE. The refined HFE is formalised as follows.

**Definition 9 (Hierarchical Functional Encryption).** *A hierarchical functional encryption (HFE) for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$ consists of the following four polynomial time algorithms:*

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda), \qquad\qquad \mathsf{SK}_{f' \circ f} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_f, f'),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x), \qquad y = f(x) \leftarrow \mathsf{Decrypt}(\mathsf{SK}_f, C).$$

*While other three algorithms work as in Definition 7, the refined key generation algorithm* KeyGen *takes a secret key for function $f \in \mathcal{F} : X \to Z \subset Y$, and a function $f' : Z \to Z' \subset Y$ as inputs such that $f_k = f' \circ f \in \mathcal{F}$ for some $k \in K$. The algorithm* KeyGen *outputs a secret key for the function $f_k$. The master secret key is defined as a secret key for the identity function (*$\mathsf{MSK} \overset{\text{def}}{=} \mathsf{SK}_{\mathsf{id}}$*) and thus it can be directly used in the decryption algorithm* Decrypt.

*Remark 1.* The secret key holders should be careful in the secret key generation. If the delegation function $f'$ is invertible, the newly generated secret key $\mathsf{SK}_{f' \circ f} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_f, f')$ is equivalent to the original secret key $\mathsf{SK}_f$ since

$$\mathsf{KeyGen}(\mathsf{SK}_{f' \circ f}, {f'}^{-1}) = \mathsf{SK}_{f'^{-1} \circ f' \circ f} = \mathsf{SK}_{\mathsf{id} \circ f} = \mathsf{SK}_f.$$

As the above operations can be done in the ideal world, it is not considered as a security issue in the real world, even with a secure scheme.

Since the syntax is similar to the normal FE, the definition of the IND-CPA security (Definition 8) can be re-used for HFE. When the adversary $\mathcal{A}$ query the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ for functions $f$ and $f'$ to obtain a new secret key $\mathsf{SK}_{f' \circ f}$ in HFE, it can be resolved by querying the original oracle $\mathcal{O}_{\mathsf{KeyGen}}$ with a function key $k$ since $f_k = f' \circ f \in \mathcal{F}$ for some $k \in K$. Using the security model of FE for HFE, we implicitly require that the secret key and the delegated key have the same distribution.

With the refined definition of HFE, we derive the syntax of our HFE for Linear Transformations.

**Definition 10 (Hierarchical Functional Encryption for Linear Transformations).** *Let $\mathbb{R}$ be a ring, $K = \{\mathbf{A} \mid \mathbf{A} \in \mathbb{R}^{i \times \delta}, i \in \mathbb{Z}^+\}$, $X = \mathbb{R}^{\delta \times \gamma}$, and $Y = \{\mathbf{Y} \mid \mathbf{Y} \in \mathbb{R}^{i \times \gamma}, i \in \mathbb{Z}^+\}$. The universal function $f : K \times X \to Y$ is defined as $f_{\mathbf{A}} : \mathbf{X} \mapsto \mathbf{AX}$. In short, the transformation function $f_{\mathbf{A}}$ is simply denoted by the internal transformation matrix $\mathbf{A}$. A hierarchical functional encryption for linear transformation (HFE-LT) is an HFE for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$, consisting of the following four polynomial time algorithms:*

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma), \qquad \mathsf{SK}_{\mathbf{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}}, \mathbf{B}),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X}), \qquad \mathbf{Y} = \mathbf{AX} \leftarrow \mathsf{Decrypt}(\mathsf{SK}_{\mathbf{A}}, C).$$

It is clear that the system setup algorithm Setup specifies the dimensions of $K$, $X$, and $Y$ by the extra inputs $\delta$ and $\gamma$. Again, the master secret key is a secret

key for the identity matrix that $\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{SK_I}$. Since the properties of ring are not fully used in matrix multiplication, the above definition can be extended to any algebraic structure (even different structures for $K$, $X$, and $Y$) as long as $(\mathbf{BA})\mathbf{X} = \mathbf{B}(\mathbf{AX})$ with all operations valid. In addition to the key generation algorithm $\mathsf{KeyGen}$, if the delegation matrix $\mathbf{B}$ is invertible, the newly generated key is equivalent to the original key since $\mathsf{KeyGen}(\mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B}), \mathbf{B}^{-1}) = \mathsf{SK_{B^{-1}BA}} = \mathsf{SK_A}$.

## 3.2   Construction

In this subsection, we propose a generic construction of HFE-LT from HPS. It is strongly recommended that the readers should read Sect. 2.1 in advance since our construction is based on the notations defined in Sect. 2.1.

Let $\varXi = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ be a diverse HPS associated with an SMP instance $\varLambda = (X, L, W, R)$ and further spaces $(K, S, \varPi)$. The HPS $\varXi$ is required to have hash linearity for completeness and key linearity for soundness. Let $\xi \in \varPi \setminus \{0\}$ be an element derived from the diversity of the HPS $\varXi$, and $n$ be the order of the group $\varPi' = \{a\xi \mid a \in \mathbb{Z}\}$. Our hierarchical functional encryption for linear transformation with $\mathbb{R} = \mathbb{Z}_n$ works as follows[1].

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma)$: Given a security parameter $1^\lambda$ and the size $\delta \times \gamma$ of message matrices, the system setup algorithm generates a system-wide parameter $\mathsf{param} \leftarrow \varXi.\mathsf{Setup}(1^\lambda)$. Then the algorithm generates a key matrix $\mathbf{K_I}$ of size $\delta \times \gamma$ as the core part of the secret key of the identity matrix $\mathbf{I}_\delta \in \mathbb{Z}_n^{\delta \times \delta}$ by invoking $\varXi.\mathsf{SKGen}(\mathsf{param})$ for $\delta \times \gamma$ times. After that, the algorithm generates the corresponding public keys $\mathbf{P} = \varXi.\mathsf{PKGen}(\mathbf{K_I})$. The full secret key for the identity matrix is packed as $\mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I})$. Finally, the algorithm publishes $\mathbf{P}$ as the public key $\mathsf{PK}$, and keeps $\mathsf{SK_I}$ as the master secret key $\mathsf{MSK}$.

$$\mathsf{param} \leftarrow \varXi.\mathsf{Setup}(1^\lambda), \quad k_{i,j} \leftarrow \varXi.\mathsf{SKGen}(\mathsf{param}),$$

$$\mathbf{K_I} = \begin{bmatrix} k_{1,1} & \cdots & k_{1,\gamma} \\ \vdots & \ddots & \vdots \\ k_{\delta,1} & \cdots & k_{\delta,\gamma} \end{bmatrix} \in K^{\delta \times \gamma}, \quad \mathbf{P} = \varXi.\mathsf{PKGen}(\mathbf{K_I}), \quad \mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I}).$$

return $(\mathsf{PK}, \mathsf{MSK}) = (\mathbf{P}, \mathsf{SK_I})$.

- $\mathsf{SK_{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B})$: In the key generation algorithm, the algorithm recognises the secret key $\mathsf{SK_A}$ for the matrix $\mathbf{A}$. If the secret key is not in the form of $\mathsf{SK_A} = (\mathbf{A}, \mathbf{K_A}) \in \mathbb{Z}_n^{m \times \delta} \times K^{m \times \gamma}$ for some $m \in \mathbb{Z}^+$, the algorithm returns $\bot$ for failure indication. The algorithm also checks the validity of the parameter $\mathbf{B} \in \mathbb{Z}_n^{m' \times m}$ for some $m' \in \mathbb{Z}^+$. If all parameters are valid and compatible, the algorithm computes and returns a secret key

---

[1] From the key linearity and hash linearity, we have that $|K| \geq |\varPi| \geq |\varPi'| = n$, and $n$ could be maximised by summing two or more elements derived from the diversity if those elements generate different groups.

$\mathsf{SK_{BA}} = (\mathbf{BA}, \mathbf{BK_A}) \in \mathbb{Z}_n^{m' \times \delta} \times K^{m' \times \gamma}$ for the matrix $\mathbf{BA}$. Remark that $\mathbf{BK_A} = \mathbf{BAK_I}$.

- $C \leftarrow$ Encrypt$(\mathsf{PK}, \mathbf{X})$: To encrypt a matrix $\mathbf{X} \in \mathbb{Z}_n^{\delta \times \gamma}$, the algorithm randomly samples a word $x$ in the language $L$ with a witness $w$ such that $(x, w) \in R$. After that, the algorithm computes the core part of the ciphertext $\mathbf{C} = \mathbf{X}\xi + \Xi.\mathsf{PHash}(\mathbf{P}, x, w) \in \Pi^{\delta \times \gamma}$. The full ciphertext is $C = (x, \mathbf{C})$.

$$(x, w) \in_R R, \quad \mathbf{C} = \mathbf{X}\xi + \Xi.\mathsf{PHash}(\mathbf{P}, x, w).$$

return $C = (x, \mathbf{C})$.

- $\mathbf{Y} \leftarrow$ Decrypt$(\mathsf{SK_A}, C)$: To decrypt a ciphertext $C = (x, \mathbf{C}) \in X \times \Pi^{\delta \times \gamma}$ with a secret key $\mathsf{SK_A} = (\mathbf{A}, \mathbf{K_A}) \in \mathbb{Z}_n^{m \times \delta} \times K^{m \times \gamma}$, the algorithm computes an intermediate value $\mathbf{D} = \mathbf{AC} - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \in \Pi^{m \times \gamma}$. After that, the algorithm find the scalar of $\mathbf{D}$ with the base $\xi$ as the final decryption result $\mathbf{Y} = \mathrm{sca}_\xi(\mathbf{D}) \in \mathbb{Z}_n^{m \times \gamma}$.

$$\mathbf{D} = \mathbf{AC} - \Xi.\mathsf{Hash}(\mathbf{K_A}, x).$$

return $\mathbf{Y} = \mathrm{sca}_\xi(\mathbf{D})$.

We show that our construction is complete by verifying the decryption algorithm. Starting from the intermediate value $\mathbf{D}$, we have

$$\begin{aligned}
\mathbf{D} &= \mathbf{AC} - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) = \mathbf{A}(\mathbf{X}\xi + \Xi.\mathsf{PHash}(\mathbf{P}, x, w)) - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{AX}\xi + \mathbf{A}\Xi.\mathsf{PHash}(\mathbf{P}, x, w) - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{AX}\xi + \mathbf{A}\Xi.\mathsf{Hash}(\mathbf{K_I}, x) - \Xi.\mathsf{Hash}(\mathbf{K_A}, x) \\
&= \mathbf{AX}\xi + \Xi.\mathsf{Hash}(\mathbf{AK_I}, x) - \Xi.\mathsf{Hash}(\mathbf{AK_I}, x) = \mathbf{AX}\xi.
\end{aligned}$$

Then the completeness of our construction is verified by

$$\mathbf{Y} = \mathrm{sca}_\xi(\mathbf{D}) = \mathrm{sca}_\xi(\mathbf{AX}\xi) = \mathbf{AX}.$$

### 3.3   Security Proof

**Theorem 1.** *The HFE-LT construction in Sect. 3.2 is IND-CPA secure if the SMP instance $\Lambda$ associated with the underlying diverse HPS $\Xi$ is hard.*

*Proof.* In this proof, we require the underlying HPS $\Xi$ to have diversity instead of smoothness introduced by Cramer and Shoup [12], which is used to prove the security of an IND-CPA public key encryption scheme. The reason is that the smoothness only prevents the information leakage of the hashing value from the public hash keys but not from the secret hash keys. In other words, the adversary may be able to distinguish ciphertexts via secret keys obtained from the key generation algorithm KeyGen of the HFE-LT scheme. If we follow the proof in [12] that the hash values are replaced with random values, the adversary can recognise this game modification with overwhelming probability by running the decryption algorithm since the adversary finds that Decrypt$(\mathsf{SK_A}, C) \neq \mathbf{AX}_0$ and Decrypt$(\mathsf{SK_A}, C) \neq \mathbf{AX}_1$ where $C$ is the challenge ciphertext of $\mathbf{X}_0$ or $\mathbf{X}_1$.

To prove the theorem, we show that an simulator $\mathcal{S}$ can be constructed to solve the SMP instance $\Lambda = (X, L, W, R)$ in polynomial time with non-negligible probability if an adversary $\mathcal{A}$ can win the IND-CPA game with non-negligible probability.

Let $(\Lambda, x^*)$ be the actual subset membership problem challenged to the simulator $\mathcal{S}$. The objective of the simulator $\mathcal{S}$ is to distinguish whether $x^* \in L$ or $x^* \in X \setminus L$ with $x^*$ sampled from $L$ or $X \setminus L$ with equal probability. Following the IND-CPA game (defined in Sect. 2.4), the simulator $\mathcal{S}$ runs the system setup algorithm Setup to generate a key pair $(\mathsf{PK}, \mathsf{MSK}) = (\mathbf{P}, \mathsf{SK_I})$, and passes the public key $\mathsf{PK}$ to the adversary $\mathcal{A}$ in the setup phase. During the pre-challenge phase, the simulator $\mathcal{S}$ invokes the key generation algorithm KeyGen with the master secret key $\mathsf{MSK} = \mathsf{SK_I}$ to answer the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ directly. The restriction for the adversary $\mathcal{A}$ is that $\mathcal{A}$ can only query the secret keys for $\mathbf{A}$ such that $\mathbf{AX}_0 = \mathbf{AX}_1$ where $\mathbf{X}_0$ and $\mathbf{X}_1$ are the target message matrices output by $\mathcal{A}$ in the challenge phase.

At some point, the adversary $\mathcal{A}$ outputs two target message matrices $\mathbf{X}_0$ and $\mathbf{X}_1$, changing the game state to the challenge phase. The simulator $\mathcal{S}$ tosses a random coin $b \in_R \{0, 1\}$, and computes the target ciphertext $C^* = (x^*, \mathbf{C}^*)$ different to the encryption algorithm Encrypt where

$$\mathbf{C}^* = \mathbf{X}_b \xi + \Xi.\mathsf{Hash}(\mathbf{K_I}, x^*).$$

After that, the target ciphertext $C^*$ is sent to the adversary $\mathcal{A}$. In the post-challenge phase, the adversary $\mathcal{A}$ is allowed to access the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ as before with the same restriction. Eventually, the adversary $\mathcal{A}$ outputs a bit $b'$ in the guessing phase. If $b = b'$, the adversary $\mathcal{A}$ wins, and the simulator $\mathcal{S}$ outputs 1. Otherwise, the simulator $\mathcal{S}$ outputs 0 instead. Finally, the simulator $\mathcal{S}$ halts and completes the simulation.

After the simulation, we analyse the probabilities in the style of [12,19]. Let $E_L$ be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in L$, and $E_{X \setminus L}$ be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in X \setminus L$. The advantage $\mathsf{Adv}_{\mathcal{S}}^{\mathrm{SMP}}$ of solving the subset membership problem is

$$\mathsf{Adv}_{\mathcal{S}}^{\mathrm{SMP}} \geq |\Pr[1 \leftarrow \mathcal{S} \mid x^* \in L] - \Pr[1 \leftarrow \mathcal{S} \mid x^* \in X \setminus L]| = \left|\Pr[E_L] - \Pr[E_{X \setminus L}]\right| \tag{1}$$

For the case of $x^* \in L$, the simulation is perfect since the algorithms $\Xi.\mathsf{Hash}$ and $\Xi.\mathsf{PHash}$ are equivalent. From the IND-CPA game, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{IND\text{-}CPA}} = \left|\Pr[E_L] - \frac{1}{2}\right|. \tag{2}$$

For the case of $x^* \in X \setminus L$, we show that the hidden bit $b$ is independent from the adversary $\mathcal{A}$'s view that

$$\Pr[E_{X \setminus L}] = \frac{1}{2} \tag{3}$$

Since the element $\xi$ is an element derived from the the diversity of the HPS $\Xi$, we have that there exists a $k \in K$ such that $\mathsf{PKGen}(k) = 0$ and $\Xi.\mathsf{Hash}(k, x^*) = \xi$

where $k$ is not required to be efficiently computable. Let $\mathbf{\Gamma} = (\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k \in K^{\delta \times \gamma}$. We have

$$
\begin{aligned}
\varXi.\mathsf{Hash}(\mathbf{\Gamma}, x^*) &= \varXi.\mathsf{Hash}((\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k, x^*) = (\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot \varXi.\mathsf{Hash}(k, x^*) \\
&= (\mathbf{X}_b - \mathbf{X}_{1-b})\xi = \mathbf{X}_b\xi - \mathbf{X}_{1-b}\xi.
\end{aligned}
$$

Based on $\varXi.\mathsf{Hash}(\mathbf{\Gamma}, x^*) = \mathbf{X}_b\xi - \mathbf{X}_{1-b}\xi$, we argue that the target ciphertext $C^*$ is not only a valid ciphertext of the message $\mathbf{X}_b$ under the key $\mathbf{K_I}$ but also a valid ciphertext of the message $\mathbf{X}_{1-b}$ under the key $\mathbf{K_I^*} = \mathbf{K_I} + \mathbf{\Gamma}$. More precisely, we have

$$
\begin{aligned}
\mathbf{C}^* &= \mathbf{X}_{1-b}\xi + \varXi.\mathsf{Hash}(\mathbf{K_I} + \mathbf{\Gamma}, x^*) = \mathbf{X}_{1-b}\xi + \varXi.\mathsf{Hash}(\mathbf{K_I}, x^*) + \varXi.\mathsf{Hash}(\mathbf{\Gamma}, x^*) \\
&= \mathbf{X}_{1-b}\xi + \varXi.\mathsf{Hash}(\mathbf{K_I}, x^*) + \mathbf{X}_b\xi - \mathbf{X}_{1-b}\xi = \mathbf{X}_b\xi + \varXi.\mathsf{Hash}(\mathbf{K_I}, x^*).
\end{aligned}
$$

Furthermore, we show that it is impossible for the adversary $\mathcal{A}$ to distinguish $\mathbf{K_I}$ and $\mathbf{K_I^*}$ from the public key $\mathbf{P}$ or the secret keys $\mathbf{K_A}$ obtained from the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$. Since $\varXi.\mathsf{PKGen}(k) = 0$ from the diversity, the keys $\mathbf{K_I}$ and $\mathbf{K_I^*}$ have the same public key

$$
\begin{aligned}
\mathbf{P} &= \varXi.\mathsf{PKGen}(\mathbf{K_I} + \mathbf{\Gamma}) = \varXi.\mathsf{PKGen}(\mathbf{K_I}) + \varXi.\mathsf{PKGen}(\mathbf{\Gamma}) \\
&= \varXi.\mathsf{PKGen}(\mathbf{K_I}) + \varXi.\mathsf{PKGen}((\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k) \\
&= \varXi.\mathsf{PKGen}(\mathbf{K_I}) + (\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot \varXi.\mathsf{PKGen}(k) = \varXi.\mathsf{PKGen}(\mathbf{K_I}).
\end{aligned}
$$

Since the adversary $\mathcal{A}$ is restricted that $\mathcal{A}$ can only query the secret keys for $\mathbf{A}$ such that $\mathbf{AX}_0 = \mathbf{AX}_1 \iff \mathbf{A}(\mathbf{X}_0 - \mathbf{X}_1) = 0$, the keys $\mathbf{K_I}$ and $\mathbf{K_I^*}$ generate the same secret key $\mathbf{K_A}$ for such a matrix $\mathbf{A}$ for the adversary $\mathcal{A}$ as

$$
\mathbf{K_A} = \mathbf{A}(\mathbf{K_I} + \mathbf{\Gamma}) = \mathbf{AK_I} + \mathbf{A\Gamma} = \mathbf{AK_I} + \mathbf{A}(\mathbf{X}_b - \mathbf{X}_{1-b}) \cdot k = \mathbf{AK_I}.
$$

Hence, the hidden bit $b$ is independent from the adversary $\mathcal{A}$'s view.

By combining Eqs. (1), (2) and (3), we have the following inequality and complete the proof.

$$
\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} \leq \mathsf{Adv}_{\mathcal{S}}^{\text{SMP}}.
$$

### 3.4   Extensions

In the Definition 9, the delegation is performed by one party with a secret key $\mathsf{SK}_f$ and a delegation function $f'$ so that the delegated party can learn $f'(f(x))$. In order not to be restricted to nested hierarchy, the HFE can be extended to allow delegation performed by multiple parties. Precisely, $n$ secret key holders with $\mathsf{SK}_{f_1}, \ldots, \mathsf{SK}_{f_n}$ can work together to generate a secret key $\mathsf{SK}_{f^*}$ for a function $f^*$ defined as $f^*(x) = f'(f_1(x), \ldots, f_n(x))$ where $f'$ is a delegation function, which takes $n$ inputs. The extended HFE is formalised as follows.

**Definition 11 (Extended Hierarchical Functional Encryption).** *An extended hierarchical functional encryption (eHFE) for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$ consists of the following four polynomial time algorithms:*

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda), \qquad \mathsf{SK}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{f_{k_1}}, \ldots \mathsf{SK}_{f_{k_n}}, f'),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x), \quad y = f(x) \leftarrow \mathsf{Decrypt}(\mathsf{SK}_f, C).$$

*While all other components work as in Definition 9, the extended key generation algorithm KeyGen takes $n$ secret keys for function $f_{k_i} \in \mathcal{F} : X \to Z_i \subset Y$, and a function $f' : Z_1 \times \cdots \times Z_n \to Z' \subset Y$ as inputs such that $f_k(x) \overset{\text{def}}{=} f'(f_{k_1}(x), \ldots, f_{k_2}(x))$ and $f_k \in \mathcal{F}$ for some $k \in K$. The algorithm KeyGen outputs a secret key for the function $f_k$. It is worth noting the number $n$ of parameters of the algorithm KeyGen is not fixed by the system setup algorithm Setup.*

Similar to HFE, the definition of the IND-CPA security (Definition 8) can also be applied to eHFE with the same method to resolve the queries to the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$. Based on Definitions 10 and 11, we derive the syntax of our extended HFE-LT.

**Definition 12 (Extended HFE-LT).** *Let $\mathbb{R}$ be a ring, $K = \{\mathbf{A} \mid \mathbf{A} \in \mathbb{R}^{i \times \delta}, i \in \mathbb{Z}^+\}$, $X = \mathbb{R}^{\delta \times \gamma}$, and $Y = \{\mathbf{Y} \mid \mathbf{Y} \in \mathbb{R}^{i \times \gamma}, i \in \mathbb{Z}^+\}$. The universal function $f : K \times X \to Y$ is defined as $f_{\mathbf{A}} : \mathbf{X} \mapsto \mathbf{AX}$. An extended hierarchical functional encryption for linear transformation (eHFE-LT) is an eHFE for a function class $\mathcal{F} = \{f_k : X \to Y \mid k \in K\}$, consisting of the following four polynomial time algorithms:*

$$(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma), \qquad \mathsf{SK}_{\mathbf{B}} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}_1}, \ldots, \mathsf{SK}_{\mathbf{A}_n}, \mathbf{T}),$$
$$C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X}), \quad \mathbf{Y} = \mathbf{AX} \leftarrow \mathsf{Decrypt}(\mathsf{SK}_{\mathbf{A}}, C).$$

*In the key generation algorithm KeyGen, the resulted transformation matrix is calculated as*

$$\mathbf{B} = \mathbf{T} \left( \mathbf{A}_1^\top \mid \cdots \mid \mathbf{A}_n^\top \right)^\top.$$

The construction of our eHFE-LT scheme is exactly the same as the HFE-LT scheme in Sect. 3.2 except the key generation algorithm. The idea of constructing the new key generation algorithm is simple due to the special structure of the secret keys that we combine the keys $\mathsf{SK}_{\mathbf{A}_1}, \ldots, \mathsf{SK}_{\mathbf{A}_\ell}$ to be a new key $\mathsf{SK}_{\mathbf{A}}$ where $\mathbf{A} = \left( \mathbf{A}_1^\top \mid \cdots \mid \mathbf{A}_\ell^\top \right)^\top$, and then runs the original key generation algorithm $\mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}}, \mathbf{T})$ to obtain the final key $\mathsf{SK}_{\mathbf{B}}$. More precisely, we have

– $\mathsf{SK}_{\mathbf{B}} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_{\mathbf{A}_1}, \ldots, \mathsf{SK}_{\mathbf{A}_\ell}, \mathbf{T})$: Given $\ell$ secret keys for $\mathbf{A}_i \in \mathbb{Z}_n^{m_i \times \delta}$, the algorithm checks the validity of $\mathbf{T} \in \mathbb{Z}_n^{m' \times \sum_{i=1}^{\ell} m_i}$ for some $m' \in \mathbb{Z}^+$. Then it computes

$$\mathbf{B} = \mathbf{T} \left( \mathbf{A}_1^\top \mid \cdots \mid \mathbf{A}_\ell^\top \right)^\top, \quad \mathbf{K}_{\mathbf{B}} = \mathbf{T} \left( \mathbf{K}_{\mathbf{A}_1}^\top \mid \cdots \mid \mathbf{K}_{\mathbf{A}_\ell}^\top \right)^\top.$$

return $\mathsf{SK}_{\mathbf{B}} = (\mathbf{B}, \mathbf{K}_{\mathbf{B}}) \in \mathbb{Z}_n^{m' \times \delta} \times K^{m' \times \gamma}$.

**Theorem 2.** *The proposed eHFE-LT construction in Sect. 3.4 is IND-CPA secure if the SMP instance $\Lambda$ associated with the underlying diverse HPS $\Xi$ is hard.*

*Proof.* The proof follows the same lines as in Sect. 3.3 and is omitted.

## 4    Instantiations

In this section, we instantiate our generic HFE-LT construction from the Decisional Diffie-Hellman (DDH) problem and from the Decisional Composite Residuosity (DCR) problem. Remark that the HFE-LT instantiations can be easily converted to eHFE-LT instantiations as discussed in Sect. 3.4. For better readability, we use multiplication in this section instead of addition in the previous sections for the group operations related to HPS. Therefore, all scalar multiplications on groups related to HPS are replaced by exponentiations.

### 4.1    HFE-LT from DDH

**Definition 13 (Decisional Diffie-Hellman problem).** *Let $\mathbb{G}$ be a cyclic multiplicative group of prime order $p$ where $|p| = \lambda$ and $\lambda$ is the security parameter. Let $a, b \in_R \mathbb{Z}_p$, $g, Z \in_R \mathbb{G}$. Given two probability distributions $\mathcal{D}_{\mathrm{DDH}} = \{(g, g^a, g^b, g^{ab})\}$ and $\mathcal{D}_R = \{(g, g^a, g^b, Z)\}$, there is an algorithm $\mathcal{A}$ that distinguishes $\mathcal{D}_{\mathrm{DDH}}$ and $\mathcal{D}_R$ with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DDH}} = |\Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathrm{DDH}})] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_R \setminus \mathcal{D}_{\mathrm{DDH}})]|$$

Let $g_1, g_2, x_1, x_2 \in_R \mathcal{D}, w \in_R \mathbb{Z}_p$. The above distributions can be represented as $\mathcal{D}_{\mathrm{DDH}} = \{(g_1, g_2, g_1^w, g_2^w)\}$ and $\mathcal{D}_R = \{g_1, g_2, x_1, x_2\}$. Thus fixed on $g_1, g_2$, the DDH problem is an SMP problem where $X = \mathbb{G}^2$, $L = \{g_1^w, g_2^w\} \subset X$, and $W = \mathbb{Z}_p$. The DDH problem is assumed hard that $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DDH}}$ is negligible. From [12], we review the corresponding HPS construction as follows with $K = \mathbb{Z}_p^2$, $S = \Pi = \mathbb{G}$.

- param $\leftarrow$ Setup($1^\lambda$): $g_1, g_2 \in_R \mathbb{G}$, return param $= (\mathbb{G}, g_1, g_2)$.
- SK $\leftarrow$ SKGen(param): return SK $= (k_1, k_2) \in_R \mathbb{Z}_p^2$.
- PK $\leftarrow$ PKGen(SK): return PK $= g_1^{k_1} g_2^{k_2}$.
- $\pi \leftarrow$ Hash(SK, $x$): $(x_1, x_2) \leftarrow x$, return $\pi = x_1^{k_1} x_2^{k_2}$.
- $\pi \leftarrow$ PHash(PK, $x$, $w$): return $\pi = \mathsf{PK}^w$.

From [19], the above HPS construction has key linearity, hash linearity, and diversity with $g_1$ as a derived element[2].

---

[2] All elements in $\mathbb{G}$ are elements derived from the diversity.

Let $\xi = g_1$. Our HFE-LT instantiation of $\mathbb{R} = \mathbb{Z}_p$ works as follows.

– $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma)$:

$$g_1, g_2 \in \mathbb{G}, \quad \mathbf{K_I} = \begin{bmatrix} (k_{1,1,1}, k_{1,1,2}) & \cdots & (k_{1,\gamma,1}, k_{1,\gamma,2}) \\ \vdots & \ddots & \vdots \\ (k_{\delta,1,1}, k_{\delta,1,2}) & \cdots & (k_{\delta,\gamma,1}, k_{\delta,\gamma,2}) \end{bmatrix} \in_R (\mathbb{Z}_p^2)^{\delta \times \gamma},$$

$$\mathbf{P} = \begin{bmatrix} P_{1,1} & \cdots & P_{1,\gamma} \\ \vdots & \ddots & \vdots \\ P_{\delta,1} & \cdots & P_{\delta,\gamma} \end{bmatrix} \text{ s.t. } P_{i,j} = g_1^{k_{i,j,1}} g_2^{k_{i,j,2}}, \quad \mathsf{SK_I} = (\mathbf{I_\delta}, \mathbf{K_I}).$$

return $(\mathsf{PK}, \mathsf{MSK}) = ((g_1, g_2, \mathbf{P}), \mathsf{SK_I})$.
– $\mathsf{SK_{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B})$: return $\mathsf{SK_{BA}} = (\mathbf{BA}, \mathbf{BK_A})$.
– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X})$:

$$w \in_R \mathbb{Z}_p, \quad x = (x_1, x_2) = (g_1^w, g_2^w), \quad \begin{bmatrix} X_{1,1} & \cdots & X_{1,\gamma} \\ \vdots & \ddots & \vdots \\ X_{\delta,1} & \cdots & X_{\delta,\gamma} \end{bmatrix} \leftarrow \mathbf{X} \in \mathbb{Z}_p^{\delta \times \gamma},$$

$$\mathbf{C} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,\gamma} \\ \vdots & \ddots & \vdots \\ C_{\delta,1} & \cdots & C_{\delta,\gamma} \end{bmatrix} \text{ s.t. } C_{i,j} = \xi^{X_{i,j}} P_{i,j}^w = g_1^{X_{i,j}} P_{i,j}^w.$$

return $C = (x, \mathbf{C})$.
– $\mathbf{Y} \leftarrow \mathsf{Decrypt}(\mathsf{SK_A}, C)$:

$$\begin{bmatrix} (k_{1,1,1}, k_{1,1,2}) & \cdots & (k_{1,\gamma,1}, k_{1,\gamma,2}) \\ \vdots & \ddots & \vdots \\ (k_{m,1,1}, k_{m,1,2}) & \cdots & (k_{m,\gamma,1}, k_{m,\gamma,2}) \end{bmatrix} \leftarrow \mathbf{K_A} \in (\mathbb{Z}_p^2)^{m \times \gamma},$$

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,\delta} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,\delta} \end{bmatrix} \leftarrow \mathbf{A} \in \mathbb{Z}_p^{m \times \delta},$$

$$\mathbf{D} = \begin{bmatrix} D_{1,1} & \cdots & D_{1,\gamma} \\ \vdots & \ddots & \vdots \\ D_{m,1} & \cdots & D_{m,\gamma} \end{bmatrix} \text{ s.t. } D_{i,j} = \frac{\prod_{l=1}^\delta C_{l,j}^{A_{i,l}}}{x_1^{k_{i,j,1}} x_2^{k_{i,j,2}}}.$$

return $\mathbf{Y} = \log_\xi \mathbf{D} = \log_{g_1} \mathbf{D}$.

Remark the calculation of $\log_{g_1} \mathbf{D}$ can be done in polynomial time if the decryption space $\{\mathbf{Y}\}$ is polynomial sized.

## 4.2   HFE-LT Instantiation from DCR

**Definition 14 (Decisional Composite Residuosity problem).** *Let $p, q$ be two safe primes such that $p = 2p' + 1$ and $q = 2q' + 1$ where $p', q'$ are two primes of length $\lambda$ bites and $\lambda$ is the security parameter. Let $N = pq$, and $N' = p'q'$. We have that $\mathbb{Z}_{N^2}^* = \mathbb{G}_N \mathbb{G}_{N'} \mathbb{G}_2 \mathbb{G}_T$ where $\mathbb{G}_T$ is a group generated by $-1 \pmod{N^2}$. Let $P = \mathbb{G}_{N'} \mathbb{G}_2 \mathbb{G}_T \subset \mathbb{Z}_{N^2}^*$, $x \in_R P$, and $x' \in_R \mathbb{Z}_{N^2}^* \setminus P$. Given two probability distributions $\mathcal{D}_P = \{(N, x)\}$ and $\mathcal{D}_{\mathbb{Z}_{N^2}^* \setminus P} = \{(N, x')\}$, there is an algorithm $\mathcal{A}$ that distinguishes $\mathcal{D}_P$ and $\mathcal{D}_{\mathbb{Z}_{N^2}^* \setminus P}$ with advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DCR}} = \left| \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_P)] - \Pr[1 \leftarrow \mathcal{A}(D \in_R \mathcal{D}_{\mathbb{Z}_{N^2}^* \setminus P})] \right|$$

The DCR problem is assumed to be hard with negligible advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DCR}}$. However, we do not use $(\mathbb{Z}_{N^2}^*, P)$ as $(X, L)$ for several technical reasons. Instead, we set $X = \mathbb{G}_N \mathbb{G}_{N'}$ and $L = \mathbb{G}_{N'} \subset X$. According to the full version of [12], the resulted SMP problem with $(X, L)$ is at least hard as the DCR problem. Slightly different from [12], we have the corresponding HPS construction as follows with $W = \{0, \ldots, \lfloor N/4 \rfloor\}$, $K = \{0, \ldots, \lfloor N^2/2 \rfloor\}$, $S = L = \mathbb{G}_{N'}$, and $\Pi = X = \mathbb{G}_N \mathbb{G}_{N'}$.

- param $\leftarrow$ Setup($1^\lambda$): $\mu \in_R \mathbb{Z}_{N^2}^*, g = \mu^{2N} \pmod{N^2}$, return param $= (N, g)$.
- SK $\leftarrow$ SKGen(param): return SK $= k \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}$.
- PK $\leftarrow$ PKGen(SK): return PK $= g^k \pmod{N^2}$.
- $\pi \leftarrow$ Hash(SK, $x$): return $\pi = x^k \pmod{N^2}$.
- $\pi \leftarrow$ PHash(PK, $x = g^w, w$): return $\pi = \mathsf{PK}^w \pmod{N^2}$.

Remark that the value $N$ in Setup is generated as described in Defintion 14 and $g$ is a generator of $L = \mathbb{G}_{N'}$ with overwhelming probability.

The key linearity and hash linearity of the above HPS can be easily verified as $g^{k_1} g^{k_2} = g^{k_1 + k_2}$ and $x^{k_1} x^{k_2} = x^{k_1 + k_2}$. Let $\xi = 1 + N \pmod{N^2}$, which is a generator of $\mathbb{G}_N$ of order $N$. It is worth noting that $\xi^a = 1 + aN \pmod{N^2}$ and $\log_\xi x = \frac{x - 1 \mod N^2}{N}$ for all $a \in \mathbb{Z}_N$ and $x \in \mathbb{G}_N$ where the division does not mean the multiplicative inverse but the division of integers. We show that the above HPS has diversity and $\xi$ is a derived element such that $\forall x \in X \setminus L, \exists k \in K, \mathsf{PKGen}(k) = 1 \wedge \mathsf{Hash}(k, x) = \xi$ where 1 is the identity element 0 in Definition 6. Let $r = \left( \log_\xi x^{N'} \right)^{-1} \mod N$ where $x^{N'} \in \mathbb{G}_N$ and the multiplicative inverse is computable for all $x \in \mathbb{G}_N \mathbb{G}_{N'}$ with overwhelming probability. Let $k = rN'$. Since $g$ is a generator of $\mathbb{G}_{N'}$ of order $N'$, we have $\mathsf{PKGen}(k) = g^k = g^{rN'} = 1 \pmod{N^2}$. Since $r^{-1} = \log_\xi x^{N'} \iff x^{N'} = \xi^{r^{-1}}$, we have $\mathsf{Hash}(k, x) = x^{rN'} = (x^{N'})^r = (\xi^{r^{-1}})^r = \xi$. Hence, the diversity is verified with the derived element $\xi = 1 + N \pmod{N^2}$.

Our HFE-LT instantiation of $\mathbb{R} = \mathbb{Z}_N$ works as follows[3]. Let $\xi = 1 + N \pmod{N^2}$.

---

[3] We do not fully use the key space (i.e. $|K| = \lfloor N^2/2 \rfloor > N$).

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\gamma)$:

$$\mu \in_R \mathbb{Z}_{N^2}^*, \, g = \mu^{2N} \pmod{N^2}, \, \mathbf{K_I} = \begin{bmatrix} k_{1,1} & \cdots & k_{1,\gamma} \\ \vdots & \ddots & \vdots \\ k_{\delta,1} & \cdots & k_{\delta,\gamma} \end{bmatrix} \in_R \{0, \ldots, \lfloor N^2/2 \rfloor\}^{\delta \times \gamma},$$

$$\mathbf{P} = \begin{bmatrix} P_{1,1} & \cdots & P_{1,\gamma} \\ \vdots & \ddots & \vdots \\ P_{\delta,1} & \cdots & P_{\delta,\gamma} \end{bmatrix} \text{ s.t. } P_{i,j} = g^{k_{i,j}} \pmod{N^2}, \quad \mathsf{SK_I} = (\mathbf{I}_\delta, \mathbf{K_I}).$$

   return $(\mathsf{PK}, \mathsf{MSK}) = ((N, g, \mathbf{P}), \mathsf{SK_I})$.
- $\mathsf{SK_{BA}} \leftarrow \mathsf{KeyGen}(\mathsf{SK_A}, \mathbf{B})$: return $\mathsf{SK_{BA}} = (\mathbf{BA}, \mathbf{BK_A})$ where $\mathbf{BK_A}$ is computed over $\mathbb{Z}$.
- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \mathbf{X})$:

$$w \in_R \{0, \ldots, \lfloor N/4 \rfloor\}, \quad x = g^w \pmod{N^2}, \quad \begin{bmatrix} X_{1,1} & \cdots & X_{1,\gamma} \\ \vdots & \ddots & \vdots \\ X_{\delta,1} & \cdots & X_{\delta,\gamma} \end{bmatrix} \leftarrow \mathbf{X} \in \mathbb{Z}_N^{\delta \times \gamma},$$

$$\mathbf{C} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,\gamma} \\ \vdots & \ddots & \vdots \\ C_{\delta,1} & \cdots & C_{\delta,\gamma} \end{bmatrix} \text{ s.t. } C_{i,j} = \xi^{X_{i,j}} P_{i,j}^w = (1 + X_{i,j}N) P_{i,j}^w \pmod{N^2}.$$

   return $C = (x, \mathbf{C})$.
- $\mathbf{Y} \leftarrow \mathsf{Decrypt}(\mathsf{SK_A}, C)$:

$$\begin{bmatrix} A_{1,1} & \cdots & A_{1,\delta} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \cdots & A_{m,\delta} \end{bmatrix} \leftarrow \mathbf{A} \in \mathbb{Z}_N^{m \times \delta}, \quad \begin{bmatrix} k_{1,1} & \cdots & k_{1,\gamma} \\ \vdots & \ddots & \vdots \\ k_{m,1} & \cdots & k_{m,\gamma} \end{bmatrix} \leftarrow \mathbf{K_A} \in \mathbb{Z}^{m \times \gamma},$$

$$\mathbf{D} = \begin{bmatrix} D_{1,1} & \cdots & D_{1,\gamma} \\ \vdots & \ddots & \vdots \\ D_{m,1} & \cdots & D_{m,\gamma} \end{bmatrix} \text{ s.t. } D_{i,j} = \frac{\prod_{l=1}^{\delta} C_{l,j}^{A_{i,l}}}{x^{k_{i,j}}} \pmod{N^2},$$

$$\mathbf{Y} = \log_\xi \mathbf{D} = \begin{bmatrix} Y_{1,1} & \cdots & Y_{1,\gamma} \\ \vdots & \ddots & \vdots \\ Y_{m,1} & \cdots & Y_{m,\gamma} \end{bmatrix} \text{ s.t. } Y_{i,j} = \frac{D_{i,j} - 1 \mod N^2}{N}.$$

   return $\mathbf{Y}$.

## 5   Conclusion

In this paper, we revisited and simplified the definition of HFE, and further extended it to eHFE. We derived the notion of HFE-LT from HFE (and eHFE-LT from eHFE), allowing matrix product evaluation. Furthermore, we proposed a generic construction of HFE-LT (and eHFE-LT) with IND-CPA security in

the standard model from Hash Proof Systems with key and hash linearity, and diversity. To illustrate that our scheme is practical, we presented two concrete HFE-LT instantiations from the DDH and DCR assumptions.

This paper proposed a practical HFE construction for matrix product evaluation. It is still an open problem whether we could build a practical HFE for other functionalities and we leave it as our future work.

# References

1. Abdalla, M., Bourse, F., Caro, A.D., Pointcheval, D.: Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011 (2016)
2. Abdalla, M., Bourse, F., 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). doi:10.1007/978-3-662-46447-2_33
3. Abdalla, M., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425 (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). doi:10.1007/978-3-662-53015-3_12
5. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013)
6. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 470–491. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_20
7. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). doi:10.1007/11426639_26
8. 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). doi:10.1007/978-3-642-19571-6_16
9. Brakerski, Z., Segev, G.: Hierarchical functional encryption. Cryptology ePrint Archive, Report 2015/1011 (2015)
10. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003). doi:10.1007/978-3-540-40061-5_3
11. Chandran, N., Goyal, V., Jain, A., Sahai, A.: Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive, Report 2015/1017 (2015)
12. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_4

13. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. Cryptology ePrint Archive, Report 2013/451 (2013)
15. 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). doi:10.1007/978-3-540-78967-3_9
16. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_4
17. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_16
18. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: ACM Symposium on Theory of Computing - STOC 2005, pp. 84–93 (2005)
19. Zhang, S., Mu, Y., Yang, G.: Achieving IND-CCA security for functional encryption for inner products. In: Chen, K., Lin, D., Yung, M. (eds.) Inscrypt 2016. LNCS, vol. 10143, pp. 119–139. Springer, Cham (2017). doi:10.1007/978-3-319-54705-3_8