11-2016

Achieving IND-CCA security for functional encryption for inner products

Shiwei ZHANG

Yi MU

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

Citation

ZHANG, Shiwei; MU, Yi; and YANG, Guomin. Achieving IND-CCA security for functional encryption for inner products. (2016). *Proceedings of the 12th International Conference, Beijing, China, 2016 November 4-6*. 119-139.
Available at: https://ink.library.smu.edu.sg/sis_research/7367

Achieving IND-CCA Security for Functional Encryption for Inner Products

Shiwei Zhang(✉), Yi Mu(✉), and Guomin Yang(✉)

School of Computing and Information Technology,
Centre for Computer and Information Security Research,
University of Wollongong, Wollongong, Australia
{sz653,ymu,gyang}@uow.edu.au

Abstract. Functional encryption allows the authorised parties to reveal partial information of the plaintext hidden in a ciphertext while in conventional encryption decryption is all-or-nothing. Focusing on the functionality of inner product evaluation (i.e. given vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, calculate $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$), Abdalla et al. (PKC 2015) proposed a functional encryption scheme for inner product functionality (FE-IP) with s-IND-CPA security. In some recent works by Abdalla et al. (eprint: Report 2016/11) and Agrawal et al. (CRYPTO 2016), IND-CPA secure FE-IP schemes have also been proposed. In order to achieve Indistinguishable under Chosen Ciphertext Attacks (IND-CCA security) for FE-IP, in this paper, we propose a generic construction of FE-IP from hash proof systems. We prove the constructed FE-IP is IND-CCA secure, assuming the hardness of the subset membership problem. In addition, we give an instantiation of our generic construction from the DDH assumption.

Keywords: IND-CCA · Functional encryption · Inner product · Hash proof system

1 Introduction

Encryption provides information confidentiality such that messages are hidden and can only be revealed by authorised parties. In traditional encryption, accessing to the plaintext is in an all-or-nothing manner. Precisely, Alice encrypts a message using Bob's encryption key and sends the ciphertext to Bob. Later, Bob can decrypt the ciphertext to read the message using his decryption key while a malicious interceptor Eve gets no information about the encrypted message. Whereas in functional encryption, it is possible for different authorised parties to reveal different partial information of the plaintext from a ciphertext by granting them different secret keys. It is also possible to control the information leaked from the ciphertexts. In detail, a functional encryption enables the authorised receivers to reveal the output of a functionality $F(k, x)$ from a ciphertext containing the plaintext x and a secret key associated with a function key value k.

In this paper, we focus on the functional encryption for the functionality of inner product evaluation where $F(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. A direct application of

DOI: 10.1007/978-3-319-54705-3_8

such a functional encryption scheme is privacy-preserving descriptive statistics such as calculating the weighted mean or sum of a list of integers. For instance, suppose in a high school the subject grades of each student are stored in a vector $\boldsymbol{y}$ which is encrypted under the school manager Alice's public key. As a university admission officer, Bob wants to offer scholarship to those students who are excellent at mathematics, physics, English, and good at other subjects. To ensure good students can get the scholarship, Alice decides to assist Bob in identifying the candidates. At the same time, Alice does not want to reveal the grades of all the students to Bob for privacy reasons. With functional encryption for inner-products, Alice can generate a secret key for Bob, which is associated with a vector $\boldsymbol{x} = (10, 8, 8, 5, 5)$ that represents the weight for different subjects (i.e. 10 for mathematics, 8 for physics and English, and 5 for other subjects). Later, Bob can run the decryption algorithm to get the weighted sum of each student's subject grades, and nothing else. For example, Charlie has a grade vector $\boldsymbol{y} = (90, 70, 80, 50, 60)$. The function $F(\boldsymbol{x}, \boldsymbol{y})$ gives $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 10 \times 90 + 8 \times 70 + 8 \times 80 + 5 \times 50 + 5 \times 60 = 2650$. In this case, Bob can only learn the result 2650 but nothing else about $\boldsymbol{y}$.

The security of functional encryption for inner products is defined by the notion of IND-CPA in [2]. However, such a security notion is not strong enough to cover the following variation of the above scenario. In order to restrict Bob's ability to calculate the weighted sum of each student, Alice gives the security key for the vector $\boldsymbol{x}$ to her colleague David instead of directly giving it to Bob. Consequently, Bob can only get the result of $F(\boldsymbol{x}, \boldsymbol{y})$ from David by sending the ciphertext to him, and David can reject any queries related to those students whose grades are below a threshold. If the scheme is malleable, then Bob can modify a rejected ciphertext such that it can pass the threshold.

In this paper, we aim to build functional encryption for inner products with IND-CCA security, which is stronger than IND-CPA and can withstand the attack described above.

1.1 Related Work

The notion of *functional encryption* is introduced by Lewko et al. [14] and later formally defined by Boneh et al. [7]. In [7], the security of functional encryption is naturally defined via *indistinguishability*-based security (IND-security) where an adversary cannot distinguish which message x_0 or x_1 is encrypted in the ciphertext with oracles provided according to the attacking model. However, the IND-security is not sufficient for the general functional encryption [7,16], and thus simulation-based security (SIM-security) has been proposed. However, the SIM-security is only achievable in the programmable random oracle model.

For generic functionality, Goldwasser et al. [12] proposed a function encryption scheme for circuits. Later in [11], the functionality is further extended to accept multiple inputs such that it is able to compute $F(k, x_1, \ldots, x_n)$ instead of $F(k, x)$. Since the construction for generic functionalities is very inefficient for practical use, the construction for specific functionality has been the main focus. It is worth noting there is a subclass of functional encryption named *Predicate*

Encryption [13]. Its message space X consists of two subspaces, index space I and payload space M. For a predicate $P : K \times I \rightarrow \{0, 1\}$, the functionality $F : K \times X \rightarrow M \cup \{\perp\}$ is defined as $F(k, (ind, m)) = m$ if $P(k, ind) = 1$ or $F(k, (ind, m)) = \perp$ otherwise. More subclasses of the functionalities can be derived from the Predicate Encryption class, including but not limited to Identity-Based Encryption [6], Attribute-Based Encryption [15,18], Hidden Vector Encryption [8], Inner Product Encryption [14,15], and Deterministic Finite Automata (DFA) Based Encryption (Functional Encryption for Regular Languages) [19]. Another notable functional encryption is searchable encryption [5] where $F(k, x) = 1$ if $k = x$ or $F(k, x) = 0$ otherwise where k and x are the keywords embedded in the trapdoor and ciphertext, respectively.

Recently, Abdalla et al. [2] investigated a new functionality $F(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$, i.e. to calculate the inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ where $\boldsymbol{x}$ is embedded in the secret key and $\boldsymbol{y}$ is embedded in the ciphertext. Unlike Inner Product Encryption [14,15] where inner product is used for access control, the new functionality here is to compute the actual inner product value. In [2], Abdalla et al. proposed a functional encryption for inner products scheme, which is selectively secure against chosen-plaintext attacks (s-IND-CPA). The scheme is generic as it can be constructed from any s-IND-CPA secure public key encryption, which is secure under randomness reuse and has linear key homomorphism and linear ciphertext homomorphism under shared randomness. Based on the generic construction, two instantiations are given from Decisional Diffie-Hellman (DDH) assumption and Learning With Error (LWE) assumption respectively. In some recent works [1,3], FE-IP schemes with IND-CPA security were also proposed. Specifically, Abdalla et al. [1] proposed another generic construction with IND-CPA security from any s-IND-CPA secure public key encryption with the same requirements as in [2]. They also showed that the IND-CPA security and Non-Adaptive Simulation (NA-SIM) security are equivalent for inner product functionality. Furthermore, an instantiation from Decisional Composite Residuosity (DCR) assumption is also proposed in [1].

In addition to the confidentiality, a notion called *function privacy* [4] has also been investigated for functional encryption which means an adversary should not be able to distinguish k (or F_k as $F(k, x) = F_k(x)$) from a secret key sk_k. However, the scheme [4] with function privacy is proposed in the private key setting while normal functional encryption schemes [1,2,19] are in the public key setting.

1.2 Our Contribution

In this paper, we define the notion of *Indistinguishablilty under adaptive Chosen Ciphertext Attacks* (i.e. IND-CCA, or more precisely IND-CCA2 security) for the general functional encryption. We also present the precise definition of functional encryption for the inner product functionality. In particular, we show that the secret keys for the functions $\langle \boldsymbol{x}_1, \cdot \rangle, \cdots, \langle \boldsymbol{x}_n, \cdot \rangle$ implies the secret key for the function $\langle \boldsymbol{x}', \cdot \rangle$ where $\boldsymbol{x}' \in \mathrm{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

As the main contribution of this paper, we propose under certain conditions an IND-CCA secure functional encryption for inner products (FE-IP) scheme from hash proof systems, assuming the hardness of the subset membership problem. In the generic construction, we require two hash proof systems Ξ_1 and Ξ_2 with some special properties as the building blocks. In detail, Ξ_1 is required to be *diverse* (Definition 10) and have *key linearity* (Definition 8) and *hash linearity* (Definition 9). For Ξ_2, we require it to be *universal*$_2$ (Definition 7) and have *hash linearity*. We show that those special properties are not hard to achieve. In [10], Cramer and Shoup constructed hash proof systems from a diverse group system $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$. We show that their constructions have the key linearity and the hash linearity. If the hash codomain Π of the underlying diverse group system has prime order, the constructed hash proof system has the property of *diversity*. In other words, we can generically construct an IND-CCA secure FE-IP scheme from a diverse group system $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ when $|\Pi|$ is prime.

In addition, we propose a concrete IND-CCA secure FE-IP scheme from DDH assumption as an instantiation of our generic construction. Note that if we remove the NIZK proof part of Definition 5, the resulting scheme is exactly the same as the schemes in [1,3]. Thus the efficiency is the same as [1,3].

1.3 Paper Organisation

The rest of this paper is organised as follows. Beginning with Sect. 2, we review the subset membership problem, the definition and the security model of the functional encryption, and introduce the IND-CCA security model. In Sect. 3, we review the hash proof system and its construction, define new properties of HPS, and show that the existing construction has the new defined properties. After that, we give out a precise definition of FE-IP and a generic construction of IND-CCA secure FE-IP with security proof in Sect. 4. In addition, an instantiation of our generic construction from DDH assumption is provided in Sect. 5. Finally, the conclusion is addressed in Sect. 6.

2 Preliminaries

2.1 Subset Membership Problems

In this subsection, we review a problem class named *Subset Membership Problems* (SMP) defined by Cramer and Shoup [10]. Some standard problems such as Quadratic Residuosity and Decisional Diffie-Hellman problems belong to the SMP problem class.

Definition 1 (Subset Membership Problem). *Let X, L, W be three non-empty sets, and $R \subset X \times W$ be a binary relation such that $L \subset X$ and $\forall x \in X, \exists w \in W, (x, w) \in R \iff x \in L$. In other words, w is a witness of x if $x \in L$. Let $\Lambda = (X, L, W, R)$, $x \in_R L$, and $x' \in_R X \setminus L$ where $x \in_R L$ means that x is randomly chosen from L. Giving two probability distributions $\mathcal{D}_L = \{(\Lambda, x)\}$*

and $\mathcal{D}_{X\setminus L} = \{(\Lambda, x')\}$, *there is an algorithm* $\mathcal{A}$ *can distinguish* $\mathcal{D}_L$ *and* $\mathcal{D}_{X\setminus L}$ *with advantage:*

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

A subset membership problem is computational hard if and only if the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SMP}}$ is negligible.

2.2 Functional Encryption

In this subsection, we review the definition of functional encryption in [7].

Definition 2 (Functional Encryption). *Let* $F : K \times X \rightarrow \{0,1\}^*$ *be a function where* K *is the function key space and* X *is the message space. A functional encryption (FE) for a functionality* 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^λ *as input, and generates system-wide parameters and a key pair of the master secret key* MSK *and the public key* PK*.*
- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$*: The randomised secret key generation algorithm takes a master secret key* MSK *and a function key* $k \in K$ *as input, and generates a secret key* SK *for the functionality* F_k*.*
- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x)$*: The randomised encryption algorithm takes a public key* PK *and a plaintext* x *as input, and calculates a ciphertext for it.*
- $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*: The (probably) deterministic decryption algorithm takes a secret key* SK *of the functionality* F_k *and a ciphertext containing* x*. It outputs a value* D*, which is equivalent to the output of* $F(k, x)$*.*

In this paper, we consider the indistinguishability-based security and enhance the IND-CPA security model defined in [7] to the *Indistinguishability under adaptive Chosen Ciphertext Attacks* (IND-CCA) as the generalisation of the IND-CCA2 security [17] for public key encryption schemes [9]. The difference is that the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$ is not allowed in the IND-CPA game at any stage. The IND-CCA game (Game 1) is defined as follows where an adaptive adversary $\mathcal{A}$ tries to distinguish a ciphertext from two chosen plaintexts x_0 and x_1.

```
$\mathsf{Game}^{\lambda}_{\text{IND-CCA}}$ :
    $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(x_0, x_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Decrypt}}}(\mathsf{PK})$
    $b \in_R \{0,1\}$
    $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b)$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{Decrypt}}}(C)$

$\mathcal{O}_{\mathsf{KeyGen}}$ :
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$
    return $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$

$\mathcal{O}_{\mathsf{Decrypt}}$ :
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{C'\}$
    $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, k)$
    return $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C')$
```

Game 1: IND-CCA

1. The challenger $\mathcal{S}$ runs $\mathsf{Setup}(1^\lambda)$ to generate a key pair $(\mathsf{MSK}, \mathsf{PK})$, and passes the public key PK to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ can adaptively query the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ for the secret key SK of a function F_k from the challenger $\mathcal{S}$. The restriction is that $\mathcal{A}$ can only query the secret keys for the functionality F_k such that $F(k, x_0) = F(k, x_1)$ where x_0 and x_1 are the target plaintexts in the next step. Otherwise, the game is trivial since $\mathcal{A}$ can simply win the game by testing $\mathsf{Decrypt}(\mathsf{SK}_k, C) \stackrel{?}{=} F(k, x_0)$. Besides that, the adversary $\mathcal{A}$ can also ask the challenger $\mathcal{S}$ for decrypting a ciphertext C' of x to obtain the output of $F(k, x)$ for any $k \in K$ via the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$.
3. At some point, the adversary $\mathcal{A}$ outputs two target plaintexts x_0 and x_1.
4. The challenger $\mathcal{S}$ randomly selects a bit $b \in_R \{0, 1\}$, and generates a target ciphertext $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_b)$. Then $\mathcal{S}$ passes C to the adversary $\mathcal{A}$.
5. The adversary $\mathcal{A}$ can continue to query the oracle $\mathcal{O}_{\mathsf{KeyGen}}$ with the same restriction as before, and the oracle $\mathcal{O}_{\mathsf{Decrypt}}$ with the restriction that $\mathcal{A}$ cannot query the target ciphertext C since $\mathcal{A}$ can win the game trivially by testing $\mathcal{O}_{\mathsf{Decrypt}}(k, C) \stackrel{?}{=} F(k, x_0)$ for some $k \in K$ such that $F(k, x_0) \neq F(k, x_1)$.
6. Eventually, the adversary $\mathcal{A}$ outputs a bit b', and $\mathcal{A}$ wins if $b = b'$.

The advantage of $\mathcal{A}$ winning the Game 1 is

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} = \left| \Pr\left[b = b' \,\middle|\, C \notin \mathcal{C} \wedge \left(\forall k \in \mathcal{K},\ F(k, x_0) = F(k, x_1) \right) \right] - \frac{1}{2} \right|$$

Definition 3 (IND-CCA Security). *A FE scheme is Indistinguishable under adaptive Chosen Ciphertext Attacks (IND-CCA) if* $\mathsf{Adv}_{\mathcal{A}}^{IND\text{-}CCA}$ *is a negligible function for all adversary* $\mathcal{A}$ *winning the Game 1 in polynomial time.*

3 Hash Proof System

In this section, we review the *hash proof system* (HPS) introduced by Cramer and Shoup [10]. We also extend their HPS with some extra properties so that we can use it to construct our scheme.

3.1 Definition

Definition 4 (Hash Proof System). *Let X be a non-empty set, and L be a $\mathcal{NP}$ language with a witness space W and a binary relation R such that $L = \{x \in X \mid \exists w : (x, w) \in R\}$. A hash proof system (HPS) 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^λ as input, and specifies an instance of X, L, W and R, using X as the hash domain. It also defines a secret hash key space K, a public hash key space S, and a hash codomain Π. After that, it packs all descriptions as the system public 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 system parameter* param *as input, and outputs a randomly chosen hash key* $\mathsf{SK} \in_R K$*.*
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: The deterministic public hash key generation algorithm takes a secret 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 value* $x \in X$*, 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 key* $PK \in S$*, a value* $x \in L$ *and its witness* $w \in W$ *as input, and generate an equivalent hash value* $\pi = \mathsf{Hash}(SK, x) \in \Pi$ *of* x *such that* $\mathsf{PK} = \mathsf{PKGen}(SK)$*.*

As a basic property, a HPS should be correct.

Definition 5 (Correctness). *A hash proof system is correct if the following statement is always true.*

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

Furthermore, some useful security properties of a HPS are required.

Definition 6 (Universal). *A hash proof system is universal if the following probability is negligible for all* $\mathsf{PK} \in S$*,* $x \in X \setminus L$ *and* $\pi \in \Pi$*.*

$$\mathsf{Adv}^{Universal} = \Pr[\mathsf{Hash}(\mathsf{SK}, x) = \pi \mid \mathsf{PKGen}(\mathsf{SK}) = \mathsf{PK}]$$

Definition 7 (Universal$_2$). *A hash proof system is universal$_2$ if the following probability is negligible for all* $\mathsf{PK} \in S$*,* $x^* \in X$*,* $x \in X \setminus (L \cup \{x^*\})$ *and* $\pi^*, \pi \in \Pi$*.*

$$\mathsf{Adv}^{Universal_2} = \Pr[\mathsf{Hash}(\mathsf{SK}, x) = \pi \mid \mathsf{Hash}(\mathsf{SK}, x^*) = \pi^* \wedge \mathsf{PKGen}(\mathsf{SK}) = \mathsf{PK}]$$

If a hash proof system is universal$_2$ and $|X| > 1$, it is also universal. Besides all above properties defined in [10], we require three extra properties to construct our schemes.

Definition 8 (Key Linearity). *A hash proof system has linear key homomorphism if* K *and* S *are abelian groups and*

$$\forall \mathsf{SK}_1, \mathsf{SK}_2 \in K, \quad \mathsf{PKGen}(\mathsf{SK}_1) + \mathsf{PKGen}(\mathsf{SK}_2) = \mathsf{PKGen}(\mathsf{SK}_1 + \mathsf{SK}_2) \in S.$$

Particularly, if a HPS has key linearity, we have $\mu \cdot \mathsf{PKGen}(\mathsf{SK}) = \mathsf{PKGen}(\mu \cdot \mathsf{SK})$ for all $\mathsf{SK} \in K$ and $\mu \in \mathbb{Z}$.

Definition 9 (Hash Linearity). *A hash proof system has linear hash homomorphism if* K *and* Π *are abelian groups and*

$$\forall \mathsf{SK}_1, \mathsf{SK}_2 \in K, \forall x \in X, \mathsf{Hash}(\mathsf{SK}_1, x) + \mathsf{Hash}(\mathsf{SK}_2, x) = \mathsf{Hash}(\mathsf{SK}_1 + \mathsf{SK}_2, x) \in \Pi.$$

Similar to the key linearity, we have $\mu \cdot \mathsf{Hash}(\mathsf{SK}, x) = \mathsf{Hash}(\mu \cdot \mathsf{SK}, x)$ for all $\mathsf{SK} \in K$, $x \in X$, and $\mu \in \mathbb{Z}$.

Definition 10 (Diversity). *A hash proof system is diverse if there exists $\pi \in \Pi$ such that $\pi \neq 0$ and for all $x \in X \setminus L$, there exists $\mathsf{SK} \in K$ such that $\mathsf{Hash}(\mathsf{SK}, x) = \pi$ and $\mathsf{PKGen}(\mathsf{SK}) = 0$. Formally,*

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

3.2 Construction

In this subsection, we review the Cramer-Shoup constructions of HPS from universal projective hashing derived from diverse group systems [10]. We start with the definition of the group system, then the constructions of the universal projective hashing. In the end, we show that the reviewed constructions have *key linearity*, *hash linearity*, and *diversity*. For notational convenience, we use addition for the group operations.

Definition 11 (Group System). *Let X, Π be two finite abelian groups, and L be a $\mathcal{NP}$ language with a witness space W and a binary relation R such that $L = \{x \in X \mid \exists w : (x, w) \in R\}$. Let Φ be a finite abelian group of homomorphism $\phi : X \to \Pi$ such that for all $\phi, \phi' \in \Phi$, $x \in X$, and $a \in \mathbb{Z}$, we have $(\phi \pm \phi')(x) = \phi(x) \pm \phi'(x)$ and $(a\phi)(x) = a\phi(x) = \phi(ax)$. If $\phi = 0 \in \Phi$, we have $\phi(x) = 0 \in \Pi$ for all $x \in X$. Let $\mathcal{H}$ be a subgroup of Φ. Then $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ is a group system.*

Definition 12 (Diverse Group System). *A group system $\mathbf{G}$ is diverse if there exists $\phi \in \Phi$ such that $\phi(L) = \langle 0 \rangle$ and $\phi(x) \neq 0$ for all $x \in X \setminus L$.*

Construction 1 (Projective Hash Families from Group Systems). *Let $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ be a group system and $(g_1, \ldots, g_d) \in L$ be a generator of L. A projective hash family $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ defined in [10] can be constructed from $\mathbf{G}$ by setting $\{\phi = H_k \mid k \in K\} = \mathcal{H}$ with uniform distribution, $S = \Pi^d$, and $\alpha : K \to S$ that $\alpha(k) = (\phi(g_1), \ldots, \phi(g_d)) = (H_k(g_1), \ldots, H_k(g_d))$. To hash $x \in X$, it simply calculates $H_k(x) = \phi(x) \in \Pi$. If $x \in L$ that $x = \sum_{i=1}^{d} w_i g_i$ where $(w_1, \ldots, w_d) \in W$ is the witnesses of x, it can alternatively calculates $H_k(x) = \phi(\sum_{i=1}^{d} w_i g_i) = \sum_{i=1}^{d} w_i \phi(g_i) \in \Pi$, using $\alpha(k)$ and $(w_1, \ldots, w_d)$.*

Construction 2 (HPS from Projective Hash Families). *Let $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ be a projective hash family where L is a $\mathcal{NP}$ language with a witness space W and a binary relation R such that $L = \{x \in X \mid \exists w : (x, w) \in R\}$. A hash proof system $\Xi = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ can be constructed as follows.*

- $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$*: return* $\mathsf{param} = (X, L, W, R, K, S, \Pi)$.
- $\mathsf{SK} \leftarrow \mathsf{SKGen}(\mathsf{param})$*: return* $k \in_R K$.
- $\mathsf{PK} \leftarrow \mathsf{PKGen}(\mathsf{SK})$*: return* $\alpha(k)$.

- $\pi \leftarrow \mathsf{Hash}(\mathsf{SK}, x)$*: return* $H_k(x)$*.*
- $\pi \leftarrow \mathsf{PHash}(\mathsf{PK}, x, w)$*: return* $H_k(x)$ *computed using* $\alpha(k)$ *and a witness* w *of* x *without the actual* k*.*

Let Ξ be a hash proof system constructed from a group system $\mathbf{G}$ by combining Constructions 1 and 2. From [10], Ξ is *universal* if $\mathbf{G}$ is *diverse*. Furthermore, we need a universal$_2$ HPS, which can be derived from a universal projective hash family.

Construction 3 (Universal$_2$ Projective Hash Families). *Let* $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ *be a universal projective hash family,* p *be the smallest prime dividing* $|X \setminus L|$*, and* $\Gamma : X \times E \to \mathbb{Z}_p^n$ *be an injective map. A universal*$_2$ *projective hash family* $\hat{\mathbf{H}} = (\hat{H}, K^{n+1}, X \times E, L \times E, \Pi, S^{n+1}, \hat{\alpha})$ *can be constructed that* $\hat{k} = (k_0, \ldots, k_n) \in K^{n+1}$*,* $\hat{\alpha}(k) = (\alpha(k_0), \ldots, \alpha(k_n)) \in S^{n+1}$*, and* $\hat{H}_{\hat{k}} = H_{k_0}(x) + \langle \Gamma(x, e), (H_{k_1}(x), \ldots, H_{k_n}(x)) \rangle$ *for all* $x \in X$ *and* $e \in E$*.*

Let Ξ_2 be a hash proof system constructed from a group system $\mathbf{G}$ by combining Constructions 1, 3 and 2 in sequence. From [10], Ξ_2 is *universal*$_2$ if $\mathbf{G}$ is *diverse*.

Besides the above security properties, we find that HPS from a group system $\mathbf{G} = (\mathcal{H}, X, L, \Pi)$ has some extra properties. Let K be a finite abelian group of order $|\mathcal{H}|$. Since H_k is uniformly distributed over $\mathcal{H}$ by randomly choosing $k \in K$ in Construction 1, we have that H is a bijection for K and $\mathcal{H}$. Thus we have $H_{k_1} + H_{k_2} = H_{k_1+k_2} \in \mathcal{H}$ for all $k_1, k_2 \in K$.

Theorem 1 (Key Linearity). *Let* Ξ *be a universal HPS and* Ξ_2 *be a universal*$_2$ *HPS as constructed above from a group system* $\mathbf{G}$*. The HPSs* Ξ *and* Ξ_2 *have key linearity.*

Proof As Ξ and Ξ_2 share the same mapping $\alpha : K \to \Pi^d$, we show the linearity of α that for all $k_1, k_2 \in K$,

$$\begin{aligned}
\alpha(k_1) + \alpha(k_2) &= (H_{k_1}(g_1), \ldots, H_{k_1}(g_d)) + (H_{k_2}(g_1), \ldots, H_{k_2}(g_d)) \\
&= (H_{k_1}(g_1) + H_{k_2}(g_1), \ldots, H_{k_1}(g_d) + H_{k_2}(g_d)) \\
&= ((H_{k_1} + H_{k_2})(g_1), \ldots, (H_{k_1} + H_{k_2})(g_d)) \\
&= (H_{k_1+k_2}(g_1), \ldots, H_{k_1+k_2}(g_d)) = \alpha(k_1 + k_2)
\end{aligned}$$

From the linearity of α, we directly have the key linearity of Ξ.

$$\begin{aligned}
&\mathsf{PKGen}(\mathsf{SK}_1) + \mathsf{PKGen}(\mathsf{SK}_2) \\
&= \alpha(\mathsf{SK}_1) + \alpha(\mathsf{SK}_2) = \alpha(\mathsf{SK}_1 + \mathsf{SK}_2) = \mathsf{PKGen}(\mathsf{SK}_1 + \mathsf{SK}_2)
\end{aligned}$$

For Ξ_2, we show the key linearity as follows where $\mathsf{SK}_1 = (k_{1,0}, \ldots, k_{1,n}), \mathsf{SK}_2 = (k_{2,0}, \ldots, k_{2,n}) \in K^{n+1}$.

$$\begin{aligned}
&\mathsf{PKGen}(\mathsf{SK}_1) + \mathsf{PKGen}(\mathsf{SK}_2) \\
&= (a(k_{1,0}), \ldots, a(k_{1,n})) + (a(k_{2,0}), \ldots, a(k_{2,n})) \\
&= (a(k_{1,0}) + a(k_{2,0}), \ldots, a(k_{1,n}) + a(k_{2,n}))) \\
&= (a(k_{1,0} + k_{2,0}), \ldots, a(k_{1,n} + k_{2,n})) = \mathsf{PKGen}(\mathsf{SK}_1 + \mathsf{SK}_2)
\end{aligned}$$

Theorem 2 (Hash Linearity). *Let $\varXi$ be a universal HPS and $\varXi_2$ be a universal$_2$ HPS as constructed above from a group system* $\mathbf{G}$*. The HPSs $\varXi$ and $\varXi_2$ have hash linearity.*

Proof. Starting from $\varXi$, we show the hash linearity that

$$\begin{aligned}&\mathsf{Hash}(SK_1, x) + \mathsf{Hash}(SK_2, x)\\ &= H_{\mathsf{SK}_1}(x) + H_{\mathsf{SK}_2}(x) = H_{\mathsf{SK}_1+\mathsf{SK}_2}(x) = \mathsf{Hash}(\mathsf{SK}_1 + \mathsf{SK}_2, x)\end{aligned}$$

Then we show the hash linearity of $\varXi_2$ as follows where $\mathsf{SK}_1 = (k_{1,0}, \ldots, k_{1,n})$ and $\mathsf{SK}_2 = (k_{2,0}, \ldots, k_{2,n}) \in K^{n+1}$.

$$\begin{aligned}&\mathsf{Hash}(\mathsf{SK}_1, (x, e)) + \mathsf{Hash}(\mathsf{SK}_2, (x, e))\\ &= H_{k_{1,0}}(x) + \langle \varGamma(x, e), (H_{k_{1,1}}(x), \ldots, H_{k_{1,n}}(x))\rangle\\ &\quad + H_{k_{2,0}}(x) + \langle \varGamma(x, e), (H_{k_{2,1}}(x), \ldots, H_{k_{2,n}}(x))\rangle\\ &= (H_{k_{1,0}}(x) + H_{k_{2,0}}(x)) + \langle \varGamma(x, e), (H_{k_{1,1}}(x) + H_{k_{2,1}}(x), \ldots, H_{k_{1,n}}(x) + H_{k_{2,n}}(x))\rangle\\ &= H_{k_{1,0}+k_{2,0}}(x) + \langle \varGamma(x, e), (H_{k_{1,1}+k_{2,1}}(x), \ldots, H_{k_{1,n}+k_{2,n}}(x))\rangle\\ &= \mathsf{Hash}(\mathsf{SK}_1 + \mathsf{SK}_2, (x, e))\end{aligned}$$

Theorem 3 (Diversity). *Let $\varXi$ be a universal HPS as constructed above from a group system* $\mathbf{G}$*. The HPS $\varXi$ is diverse if $|\varPi|$ is prime and* $\mathbf{G}$ *is diverse.*

Proof Since $\mathbf{G}$ is diverse, we have that there exists $k \in K$ such that $\mathsf{Hash}(k, x) = 0$ for all $x \in L$ and $\mathsf{Hash}(k, x^*) \neq 0$ for all $x^* \in X \setminus L$. Let $\pi = \mathsf{Hash}(k, x^*) \neq 0$. Since $\varPi$ is a prime order cyclic group, π is a generator of $\varPi$ and thus for all $\pi' \in \varPi$, $\pi' = \mu \cdot \pi$ for some $\mu \in \mathbb{Z}_{|\varPi|}$. By Theorem 2, we have that for all $x' \in X \setminus L$,

$$\begin{aligned}&\pi' = \mathsf{Hash}(k, x') \iff \mu \cdot \pi = \mathsf{Hash}(k, x')\\ &\iff \pi = \mu^{-1} \cdot \mathsf{Hash}(k, x') \iff \pi = \mathsf{Hash}(\mu^{-1} \cdot k, x')\end{aligned}$$

Hence, for a fixed π, there exists a secret key $\mu^{-1} \cdot k$ that hashes x' to π for all $x' \in X$. Let w be a witness of $x \in L$. Recall the construction of $\varXi$ that $\mathsf{Hash}(k, x) = H_k(x)$ and $\mathsf{PKGen}(k) = \alpha(k) = (H_k(g_1), \ldots, H_k(g_d))$. Since $g_1, \ldots, g_d \in L$ and $H_k(x) = 0$ for all $x \in L$, we have $\mathsf{PKGen}(k) = 0$. Therefore, by Theorem 1, we have $\mathsf{PKGen}(\mu^{-1} \cdot k) = \mu^{-1} \cdot \mathsf{PKGen}(k) = 0$ and complete the proof.

4 Functional Encryption for Inner Products

4.1 Definition

Let $\mathbb{G}_x$, $\mathbb{G}_y$, $\mathbb{G}_z$ be three abelian groups where there exists an efficient inner product computation $\langle \cdot, \cdot \rangle : \mathbb{G}_x \times \mathbb{G}_y \to \mathbb{G}_z$. The functional encryption for inner products is associated with a functionality $F : \mathbb{G}_x^\delta \times \mathbb{G}_y^\delta \to \mathbb{G}_z$, mapping two δ-dimension vectors into a single group $\mathbb{G}_z$ such that $F(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{\delta} \langle x_i, y_i \rangle$ for all $\boldsymbol{x} = (x_1, \ldots, x_\delta) \in \mathbb{G}_x^\delta$ and $\boldsymbol{y} = (y_1, \ldots, y_\delta) \in \mathbb{G}_y^\delta$. Based on the functionality F, we derive the syntax of the functional encryption from Definition 2.

Definition 13 (Functional Encryption for Inner Products). *A functional encryption for inner products (FE-IP) scheme for a functionality $F : \mathbb{G}_x^\delta \times \mathbb{G}_y^\delta \to \mathbb{G}_z$ consists of the following four polynomial time algorithms:*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta)$*: The randomised system setup algorithm takes a security parameter 1^λ and a unary value 1^δ that specifies the maximum vector dimension as input. Then it generates system-wide parameters and a key pair* $(\mathsf{PK}, \mathsf{MSK})$*.*
- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{x})$*: The randomised secret key generation algorithm takes a master secret key* MSK *and a vector $\boldsymbol{x} \in \mathbb{G}_x^d$ with dimension d. If $d > \delta$, the extra dimensions of $\boldsymbol{x}$ is discarded. If $d < \delta$, the vector $\boldsymbol{x}$ is reconstructed to the dimension δ by filling an additive identity element 0 (i.e. $\boldsymbol{x}' = (\boldsymbol{x}, \underbrace{0, \ldots, 0}_{\delta - d})$).*

  After that, the algorithm generates a secret key SK *for the (modified) vector $\boldsymbol{x}$ with dimension δ.*
- $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \boldsymbol{y})$*: The randomised encryption takes a public key* PK *and a vector $\boldsymbol{y} \in \mathbb{G}_y^d$ with dimension d. If $d \neq \delta$, the ciphertext may still be constructed. However, it may not be decrypted properly. Hence, the same modification to $\boldsymbol{x}$ in the algorithm* KeyGen *is applied to $\boldsymbol{y}$. After that, the algorithm generates a ciphertext C for the (modified) vector $\boldsymbol{y}$ with dimension δ.*
- $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$*: The deterministic decryption algorithm takes a secret key* SK *for $\boldsymbol{x}$ and a ciphertext C of $\boldsymbol{y}$, and computes $D = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{G}_z$. If the decryption fails, the algorithm outputs a special symbol $\perp$.*

Before introducing the security model of FE-IP, we review the inner product functionality along with the vector space first.

Due to the linearity that $\langle x_0 + x_1, y \rangle = \langle x_0, y \rangle + \langle x_1, y \rangle$ for all $x_0, x_1 \in \mathbb{G}_x$ and $y \in \mathbb{G}_y$, we have $\mu \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \mu \boldsymbol{x}, \boldsymbol{y} \rangle$ for all $\mu \in \mathbb{Z}$, $\boldsymbol{x} \in \mathbb{G}_x^\delta$, and $\boldsymbol{y} \in \mathbb{G}_y^\delta$. Thus the ability of the secret key for a vector $\boldsymbol{x}$ is not only to calculate $F(\boldsymbol{x}, \boldsymbol{y})$ but also to compute $F(\mu \boldsymbol{x}, \boldsymbol{y}) = \langle \mu \boldsymbol{x}, \boldsymbol{y} \rangle = \mu \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \mu F(\boldsymbol{x}, \boldsymbol{y})$, which is equivalent to the ability of the secret key for the vector $\mu \boldsymbol{x}$ for all $\mu \in \mathbb{Z}$. In other words, the key generation algorithm KeyGen actually generates a secret key for a vector space $\mathrm{span}(\boldsymbol{x})$ linearly spanned by $\boldsymbol{x}$ instead of a single vector $\boldsymbol{x}$. Generally, given multiple secret keys for a vector set $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, we are able to compute $F(\boldsymbol{x}, \boldsymbol{y})$ for all $\boldsymbol{x} \in \mathrm{span}(S)$. It is possible since $F(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^n \mu_i F(\boldsymbol{x}_i, \boldsymbol{y})$ where $\boldsymbol{x} = \sum_{i=1}^n \mu_i \boldsymbol{x}_i$. Notably, if we obtain secret keys for a vector set S such that $\mathrm{span}(S) = \mathbb{G}_x$ (e.g. S contains δ linearly dependent vectors), we have the same ability of the master secret key without compromising it.

Since δ secret keys for linearly independent vectors are equivalent to the master secret key, the function key space K (recall Definition 2) is reduced to the size of δ, which is polynomial bounded. Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\delta$ be a basis of $\mathbb{G}_x$. Intuitively, one may think that the "brute force" construction in [7] becomes practical by encrypting the output of $F(\boldsymbol{v}_1, \boldsymbol{y}), \ldots, F(\boldsymbol{v}_\delta, \boldsymbol{y})$ instead of the vector $\boldsymbol{y}$ where the resulting ciphertext size is $\Theta(\delta)$. However, it is not true since $\mathrm{span}(\boldsymbol{v}_1 + \boldsymbol{v}_2) \neq \mathrm{span}(\boldsymbol{v}_1, \boldsymbol{v}_2)$ where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are independent. Hence, a proper construction is still required.

Besides that, if $\mathbb{G}_x = \mathbb{G}_y = \mathbb{G}_z = \mathbb{F}$ are the same field, it is impossible to hide $\boldsymbol{x}$ in the public key setting, given the secret key for the vector $\boldsymbol{x}$. Since it is in the public key setting, δ linearly independent vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\delta$ can be chosen and encrypted freely. By decrypting the above ciphertexts with the secret key for the vector $\boldsymbol{x} = (x_1, \ldots, x_\delta)$, we can obtain the results $\{D_i = F(\boldsymbol{x}, \boldsymbol{y}_i)\}_{i=1\ldots\delta}$. After that, we can calculate $\boldsymbol{x}$ by solving the following matrix equation in polynomial time.

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_\delta \end{bmatrix} \boldsymbol{x}^\top = \begin{bmatrix} D_1 \\ \vdots \\ D_\delta \end{bmatrix}$$

Hence, it is impossible to achieve *function privacy*. As a side effect, it is "safe" to provide $\boldsymbol{x}$ along with the secret key in the key generation algorithm KeyGen.

For the security model, the definition of the IND-CPA security and the IND-CCA security can be derived from the security model of the general functional encryption. The difference is that the setup algorithm is required to take an additional parameter 1^δ.

4.2 Generic Construction from Hash Proof Systems

In this subsection, we describe the key ideas to construct an IND-CCA secure FE-IP scheme. Then we present our FE-IP scheme from Hash Proof Systems.

In a FE-IP scheme, the plaintext vector $\boldsymbol{y}$ should be encrypted in a raw form that can be recovered instead of being encrypted as the output of the function $F(\boldsymbol{x}, \boldsymbol{y})$ so that it can be manipulated by arbitrary $\boldsymbol{x}$ to compute $F(\boldsymbol{x}, \boldsymbol{y})$. In order to achieve the IND-CCA security, the sender who encrypts the message shall provide a non-interactive zero knowledge (NIZK) proof that it knows the decryption in terms of the raw form of the plaintext vector $\boldsymbol{y}$ [17]. This is the essential idea of achieving the IND-CCA security. On the other hand, the decryption of a functional encryption involves two parts: the function evaluation and the authorisation to that function evaluation. Obviously, we have to do manipulation first then decryption instead of decryption first then manipulation since the receiver should only be able to compute $F(\boldsymbol{x}, \boldsymbol{y})$ but not $\boldsymbol{y}$ itself. For inner product functionality, the ciphertext of the plaintext vector $\boldsymbol{y}$ is manipulated into the ciphertext of $F(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ by using the vector $\boldsymbol{x}$ and the ciphertext homomorphism. Later, the receiver can decrypt the resulted ciphertext to obtain $F(\boldsymbol{x}, \boldsymbol{y})$, given the authorisation to $F(\boldsymbol{x}, \cdot)$. Before decryption, the receiver also needs to verify the NIZK proof attached to the ciphertext. Using the hash proof system as the NIZK proof system (with auxiliary input), the receiver is required to use the secret key of the HPS to verify the proof. If we consider attacks from outsiders only, it is safe to give the secret key to the end users. However, from the Game 1, we allow attacks from insiders. Therefore, we cannot give the secret key directly to the users. Otherwise, they can generate the proofs without knowing the witnesses. To solve this problem, we limit the scheme to serve at most η users and generate a vector $\boldsymbol{\beta}$ of secret keys for the proof system with dimension η. For each user, we randomly pick a vector $\boldsymbol{s}$ and compute the inner product

$\langle \boldsymbol{s}, \boldsymbol{\beta} \rangle$ as the secret key for the end user. When encrypting, the sender generates the proof using the individual public keys directly derived for $\boldsymbol{\beta}$ as η proof parts. To verify, the receiver assembles the proof parts using $\boldsymbol{s}$ and checks with its secret key $\langle \boldsymbol{s}, \boldsymbol{\beta} \rangle$. Since both $\boldsymbol{\beta}$ and $\boldsymbol{s}$ have the dimension η, it is impossible to compute $\boldsymbol{\beta}$ with $\eta - 1$ pairs of $(\boldsymbol{s}, \langle \boldsymbol{s}, \boldsymbol{\beta} \rangle)$ (i.e. $\boldsymbol{\beta}$ is statistically indistinguishable with a random vector). If all η users collude together, they can obtain $\boldsymbol{\beta}$ and generate proofs without witnesses but it is meaningless to launch attacks against themselves.

Unfortunately, we are not able to construct a generic FP-IP scheme for arbitrary $\mathbb{G}_x$, $\mathbb{G}_y$, and $\mathbb{G}_z$. Due to the definition of hash linearity (Definition 9) of HPS, we have to make $\mathbb{G}_x = \mathbb{G}_y = \mathbb{G}_z = \mathbb{Z}_\rho \subset \mathbb{Z}$. To build our FE-IP scheme, we need a diverse HPS with key linearity and hash linearity, a universal$_2$ HPS with hash linearity. Note that the key linearity is used in the proof only. Formally, we present our construction as follows.

Construction 4 (FE-IP from HPS). *Let $\varXi_1 = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ be a diverse HPS associated with spaces $(X, L, W, R, K, S, \varPi)$ and $\varXi_2 = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Hash}, \mathsf{PHash})$ be a universal$_2$ HPS associated with space $(X \times \varPi^\delta, L \times \varPi^\delta, W, R, K', S', \varPi')$ where δ is passed as the input of the algorithm Setup. Both $\varXi_1$ and $\varXi_2$ are required to have the hash linearity. $\varXi_1$ is required to have the key linearity for the security proof. Let $\chi \in \varPi$ derived from the diversity property of $\varXi_1$ that $\chi \neq 0$ and $\forall x \in X \setminus L, \exists \mathsf{SK} \in K, \varXi_1.\mathsf{Hash}(\mathsf{SK}, x) = \chi \wedge \varXi_1.\mathsf{PKGen}(\mathsf{SK}) = 0$. We use $\mu = \chi^{-1}(\pi)$ to denote the calculation of $\mu \in \mathbb{Z}_\rho$ such that $\mu \cdot \chi = \pi \in \varPi$. Our functional encryption scheme for the functionality $F : \mathbb{Z}_\rho^\delta \times \mathbb{Z}_\rho^\delta \to \mathbb{Z}_\rho$ works as follows.*

- $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\delta, 1^\eta)$: *Given a security parameter 1^λ, a maximum vector size 1^δ and a maximum user size 1^η, the algorithm generates system-wide parameters $\mathsf{param}_1 \leftarrow \varXi_1.\mathsf{Setup}(1^\lambda)$ and $\mathsf{param}_2 \leftarrow \varXi_2.\mathsf{Setup}(1^\lambda)$. The algorithm generates two secret key vectors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\delta) \in K^\delta$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_\eta) \in K'^\eta$ where $\alpha_i \leftarrow \varXi_1.\mathsf{SKGen}(\mathsf{param}_1)$, $\beta_i \leftarrow \varXi_2.\mathsf{SKGen}(\mathsf{param}_2)$. After that, it generates corresponding public keys $\boldsymbol{A} = (A_1, \ldots, A_\delta) \in S^\delta$ and $\boldsymbol{B} = (B_1, \ldots, B_\eta) \in S'^\eta$ where $A_i = \varXi_1.\mathsf{PKGen}(\alpha_i)$, $B_i = \varXi_2.\mathsf{PKGen}(\beta_i)$. Next, the algorithm packs the public key $PK = (\boldsymbol{A}, \boldsymbol{B})$ and the master secret key $MSK = (\boldsymbol{\alpha}, \boldsymbol{\beta})$. Finally, the algorithm publishes PK and keeps MSK private.*

$$\begin{aligned} &\mathsf{param}_1 \leftarrow \varXi_1.\mathsf{Setup}(1^\lambda), \quad \mathsf{param}_2 \leftarrow \varXi_2.\mathsf{Setup}(1^\lambda) \\ &\mathit{For}\ i = 1 \ldots \delta, \quad \alpha_i \leftarrow \varXi_1.\mathsf{SKGen}(\mathsf{param}_1), \quad A_i = \varXi_1.\mathsf{PKGen}(\alpha_i) \\ &\mathit{For}\ i = 1 \ldots \eta, \quad \beta_i \leftarrow \varXi_2.\mathsf{SKGen}(\mathsf{param}_2), \quad B_i = \varXi_2.\mathsf{PKGen}(\beta_i) \end{aligned}$$

  return $(\mathsf{PK}, \mathsf{MSK}) = ((\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{\alpha}, \boldsymbol{\beta}))$.
- $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{x})$: *To generate a secret key for the vector $\boldsymbol{x}$, the algorithm randomly selects a vector $\boldsymbol{s} = (s_1, \ldots, s_\eta) \in_R \mathbb{Z}_\rho^\eta$ and calculates K_1 and K_2 as follows.*

$$\boldsymbol{s} \in_R \mathbb{Z}_\rho^\eta, \quad K_1 = \langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle, \quad K_2 = \langle \boldsymbol{s}, \boldsymbol{\beta} \rangle.$$

  return $\mathsf{SK} = (\boldsymbol{x}, \boldsymbol{s}, K_1, K_2)$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \boldsymbol{y})$: *To encrypt a vector* $\boldsymbol{y}$, *the algorithm randomly samples a word* $l \in L$ *with a witness* $w \in W$. *Then the algorithm computes the ciphertext part* $\boldsymbol{C} = (C_1, \ldots, C_\delta)$ *where* $C_i = y_i\chi + \Xi_1.\mathsf{PHash}(A_i, l, w)$. *After that, the algorithm computes the proof part* $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_\eta)$ *where* $\pi_i = \Xi_2.\mathsf{PHash}(B_i, (l, \boldsymbol{C}), w)$. *Finally, the algorithm packs the word, the ciphertext part, and the proof part as one single ciphertext.*

$$\begin{aligned}&(l, w) \in_R R\\ &\textit{For } i = 1 \ldots \delta, \quad C_i = y_i\chi + \Xi_1.\mathsf{PHash}(A_i, l, w)\\ &\textit{For } i = 1 \ldots \eta, \quad \pi_i = \Xi_2.\mathsf{PHash}(B_i, (l, \boldsymbol{C}), w)\end{aligned}$$

return $C = (l, \boldsymbol{C}, \boldsymbol{\pi})$.

– $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$: *To decrypt, the algorithm assembles the proof parts* $D_2 = \langle \boldsymbol{s}, \boldsymbol{\pi}\rangle$. *If* $D_2 \neq \Xi_2.\mathsf{Hash}(K_2, (l, \boldsymbol{C}))$, *the algorithm outputs* $D = \perp$ *to reject the ciphertext. Otherwise, the algorithm assembles the ciphertext part* $D_1 = \langle \boldsymbol{x}, \boldsymbol{C}\rangle$. *Then the algorithm decrypts the resulted ciphertext* $D^* = D_1 - \Xi_1.\mathsf{Hash}(K_1, l)$. *Finally, the algorithm extracts the result* $D = \chi^{-1}(D^*)$.

$$\begin{aligned}&D_2 = \langle \boldsymbol{s}, \boldsymbol{\pi}\rangle, \quad D_2 \overset{?}{=} \Xi_2.\mathsf{Hash}(K_2, (l, \boldsymbol{C})),\\ &D_1 = \langle \boldsymbol{x}, \boldsymbol{C}\rangle, \quad D^* = D_1 - \Xi_1.\mathsf{Hash}(K_1, l).\end{aligned}$$

return $D = \chi^{-1}(D^*)$.

Theorem 4. *The Construction 4 is correct.*

Proof. We verify the correctness by verifying the decryption algorithm.

$$\begin{aligned}D_2 &= \langle \boldsymbol{s}, \boldsymbol{\pi}\rangle = \sum_{i=1}^{\eta} s_i\pi_i = \sum_{i=1}^{\eta} s_i\Xi_2.\mathsf{PHash}(B_i, (l, \boldsymbol{C}), w) = \sum_{i=1}^{\eta} s_i\Xi_2.\mathsf{Hash}(\beta_i, (l, \boldsymbol{C}))\\ &= \Xi_2.\mathsf{Hash}(\sum_{i=1}^{\eta} s_i\beta_i, (l, \boldsymbol{C})) = \Xi_2.\mathsf{Hash}(\langle \boldsymbol{s}, \boldsymbol{\beta}\rangle, (l, \boldsymbol{C})) = \Xi_2.\mathsf{Hash}(K_2, (l, \boldsymbol{C})).\\ D_1 &= \langle \boldsymbol{x}, \boldsymbol{C}\rangle = \sum_{i=1}^{\delta} x_iC_i = \sum_{i=1}^{\delta} x_i\big(y_i\chi + \Xi_1.\mathsf{PHash}(A_i, l, w)\big)\\ &= \sum_{i=1}^{\delta} x_iy_i\chi + \sum_{i=1}^{\delta} x_i\Xi_1.\mathsf{Hash}(\alpha_i, l) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle\chi + \Xi_1.\mathsf{Hash}(\sum_{i=1}^{\delta} x_i\alpha_i, l)\\ &= \langle \boldsymbol{x}, \boldsymbol{y}\rangle\chi + \Xi_1.\mathsf{Hash}(\langle \boldsymbol{x}, \boldsymbol{\alpha}\rangle, l) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle\chi + \Xi_1.\mathsf{Hash}(K_1, l).\end{aligned}$$

After verifying D_2 and computing D_1, we compute D as follows and complete the verification.

$$D^* = D_1 - \Xi_1.\mathsf{Hash}(K_1, l) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle\chi, \quad D = \chi^{-1}(D^*) = \chi^{-1}(\langle \boldsymbol{x}, \boldsymbol{y}\rangle\chi) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle.$$

In Construction 4, we require the calculation of χ^{-1}, which may not be computed in polynomial time. If the decryption space $|\{\langle \boldsymbol{x}, \boldsymbol{y}\rangle\}|$ is polynomial bounded, we can do decryption in an alternative way. In this variation, D_1, D_2, and D^* are computed and checked as normal. To calculate D, we check $D\chi \overset{?}{=} D^*$ for each possible $D \in |\{\langle \boldsymbol{x}, \boldsymbol{y} \rangle\}|$. Since $|\{\langle \boldsymbol{x}, \boldsymbol{y} \rangle\}|$ is polynomial-sized, the checking algorithm can be done in polynomial time.

In terms of the maximum number η of users, it is not a defect and the IND-CCA model is still suitable for Construction 4. Since the number of users is polynomial sized, the value of η is also polynomial sized. By observing Construction 4, we have the following size table.

Item	PK	MSK	SK	C	D
Size	$\delta\|S\| + \eta\|S'\|$	$\delta\|K\| + \eta\|K'\|$	$(\delta + \eta)\|\mathbb{Z}_\rho\| + \|K\| + \|K'\|$	$\|L\| + \delta\|\Pi\| + \eta\|\Pi'\|$	$\|\mathbb{Z}_\rho\|$

As long as the value of η is polynomial sized, all the elements in Construction 4 are polynomial sized. Hence, the limitation on the maximum user number is no longer an issue.

4.3 Security Proof

Theorem 5. *The proposed FE-IP scheme (Construction 4), allowing at most η users, is IND-CCA secure (Definition 3) if the language (X, L, W, R) associated with both the underlying diverse HPS Ξ_1 and universal$_2$ HPS Ξ_2 satisfies a hard subset membership problem (Definition 1).*

Proof. Having a glance at the security proof, we leverage the diversity property of the HPS Ξ_1 to prove the ciphertext indistinguishability of our construction. At the same time, we exploit the universal$_2$ property of the HPS Ξ_2 to finalise the IND-CCA security in terms of dealing the decryption oracle.

In detail, we show that an algorithm $\mathcal{S}$ (i.e. simulator) can be constructed to solve subset membership problems in polynomial time with non-negligible probability if an adversary $\mathcal{A}$ can win the Game 1 with non-negligible probability, querying the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ at most $\eta - 1$ times and the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$ for at most q times. As explained in Sect. 4.2, it is meaningless to obtain all secret keys of η users and this is the reason why we let the adversary $\mathcal{A}$ query the key generation oracle at most $\eta - 1$ times. Although we limit the maximum number of the key generation queries, we do not limit the maximum number of the decryption queries to a function of η. Therefore, the proof is still in a valid IND-CCA model.

Let $(\Lambda = (X, L, W, R), x^*)$ be an instance of subset membership problems challenged to the simulator $\mathcal{S}$ for distinguishing whether $x^* \in L$ or $x^* \in X \setminus L$ where x^* is sampled from L or $X \setminus L$ with equal probability. To simulate the Game 1, the simulator $\mathcal{S}$ runs the algorithm Setup as normal to generate a key pair $(\mathsf{PK}, \mathsf{MSK}) = ((\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{\alpha}, \boldsymbol{\beta}))$, and passes the public key PK to the adversary $\mathcal{A}$. Since the simulator $\mathcal{S}$ has the master secret key MSK, it can answer the oracles $\mathcal{O}_{\mathsf{KeyGen}}$ and $\mathcal{O}_{\mathsf{Decrypt}}$ as normal using the master secret key MSK.

The adversary $\mathcal{A}$ is restricted to query the secret keys for $\boldsymbol{x}$ to $\mathcal{O}_{\mathsf{KeyGen}}$ such that $\langle \boldsymbol{x}, \boldsymbol{y}_0 \rangle \neq \langle \boldsymbol{x}, \boldsymbol{y}_1 \rangle$ where $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$ are the target vectors output by the adversary $\mathcal{A}$ in the next phase. In other words, the adversary $\mathcal{A}$ can only ask the secret keys for $\boldsymbol{x}$ such that $\langle \boldsymbol{x}, \boldsymbol{y}_0 - \boldsymbol{y}_1 \rangle = 0$.

At some point, the adversary $\mathcal{A}$ outputs two target vectors $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$. Then the simulator $\mathcal{S}$ randomly chooses $b \in_R \{0,1\}$. After that, the simulator $\mathcal{S}$ computes the target ciphertext $C^* = (x^*, \boldsymbol{C}^*, \boldsymbol{\pi}^*)$ where

$$C_i^* = y_{b,i}\chi + \varXi_1.\mathsf{Hash}(\alpha_i, x^*), \quad \pi_i^* = \varXi_2.\mathsf{Hash}(\beta_i, (x^*, \boldsymbol{C}^*)).$$

After receiving the target ciphertext C^*, the adversary $\mathcal{A}$ can continue to query provided oracles as before with the restriction that $\mathcal{A}$ cannot query C^* to the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$. Eventually, the adversary $\mathcal{A}$ outputs a bit b'. If $b = b'$, the simulator $\mathcal{S}$ outputs 1, indicating that $\mathcal{A}$ wins the Game 1. Otherwise, the simulator $\mathcal{S}$ outputs 0. After that, the simulator $\mathcal{S}$ halts in order to complete the simulation.

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 $x^* \in X \setminus L$. Thus we have the advantage $\mathsf{Adv}_{\mathcal{S}}^{\text{SMP}}$ of solving the subset membership problem.

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

For the case of $x^* \in L$, the simulation is perfect since the algorithms ($\varXi_1.\mathsf{PHash}$, $\varXi_2.\mathsf{PHash}$) and ($\varXi_1.\mathsf{Hash}, \varXi_2.\mathsf{Hash}$) are equivalent. Thus we have

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

For the case of $x^* \in X \setminus L$, we modify the game to a new game such that the simulator $\mathcal{S}$ rejects all ciphertexts $C = (l, \boldsymbol{C}, \boldsymbol{\pi})$ where $l \in X \setminus L$ in the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$ in addition to those words, which cannot pass the proof verification. Let E_m be the event that $\mathcal{S}$ outputs 1 conditioned on $x^* \in X \setminus L$ in this modified game, and $E_\perp$ be the event that $l \in X \setminus L$ and $\langle \boldsymbol{s}, \boldsymbol{\pi} \rangle = \varXi_2.Hash(K_2, (l, \boldsymbol{C}))$. In other words, $E_\perp$ is the event that a ciphertext is rejected in the modified game but accepted in the original game. Since the original game and the modified game are identical until event $E_\perp$ occurs, we have

$$\left|\Pr[E_m] - \Pr[E_{X\setminus L}]\right| \le \Pr[E_\perp]. \tag{3}$$

Lemma 1. *The event $E_\perp$ occurs in negligible probability as long as $\varXi_2$ is a universal$_2$ HPS. More precisely, letting $\mathcal{A}$ query $\mathcal{O}_{\mathsf{Decrypt}}$ for at most q times, we have a upper bound of the probability that $E_\perp$ occurs.*

$$\Pr[E_\perp] \le q \cdot \mathsf{Adv}^{Universal_2} \tag{4}$$

Proof. Let $C = (l, \boldsymbol{C}, \boldsymbol{\pi})$ be a ciphertext submitted to the decryption oracle $\mathcal{O}_{\mathsf{Decrypt}}$.

- Suppose that $(l, \boldsymbol{C}) = (x^*, \boldsymbol{C}^*)$, the adversary $\mathcal{A}$ tries to find a $\boldsymbol{\pi} \neq \boldsymbol{\pi}^*$ such that $\langle \boldsymbol{s}, \boldsymbol{\pi} \rangle = \langle \boldsymbol{s}, \boldsymbol{\pi}^* \rangle$. Let $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi} - \boldsymbol{\pi}^* \neq \boldsymbol{0}$. Since $\boldsymbol{s}$ is independent from $\mathcal{A}$'s view, it is impossible to find a $\hat{\boldsymbol{\pi}}$ such that $\langle \boldsymbol{s}, \hat{\boldsymbol{\pi}} \rangle = 0$.

– Suppose that $(l, \boldsymbol{C}) \neq (x^*, \boldsymbol{C}^*)$, the adversary $\mathcal{A}$ tries to find a $\boldsymbol{\pi}$ such that $\langle \boldsymbol{s}, \boldsymbol{\pi} \rangle = \Xi_2.\mathsf{Hash}(K_2, (l, \boldsymbol{C}))$. Let $\hat{\pi}_i = \Xi_2.\mathsf{Hash}(\beta_i, (l, \boldsymbol{C}))$. Since

$$\begin{aligned}\langle \boldsymbol{s}, \boldsymbol{\pi} \rangle &= \Xi_2.\mathsf{Hash}(K_2, (l, \boldsymbol{C})) = \Xi_2.\mathsf{Hash}(\langle \boldsymbol{s}, \boldsymbol{\beta} \rangle, (l, \boldsymbol{C})) \\ &= \sum_{i=1}^{\eta} s_i \Xi_2.\mathsf{Hash}(\beta_i, (l, \boldsymbol{C})) = \langle \boldsymbol{s}, \hat{\boldsymbol{\pi}} \rangle\end{aligned}$$

and $\boldsymbol{s}$ is independent from $\mathcal{A}$'s view, we have $\langle \boldsymbol{s}, \boldsymbol{\pi} \rangle = \langle \boldsymbol{s}, \hat{\boldsymbol{\pi}} \rangle \iff \boldsymbol{\pi} = \hat{\boldsymbol{\pi}}$. As the scheme allows at most η users (i.e. $\mathcal{A}$ can query $\mathcal{O}_{\mathsf{KeyGen}}$ for at most $\eta - 1$ times), the adversary can get at most $\eta - 1$ pairs of $(\boldsymbol{s}, \langle \boldsymbol{s}, \boldsymbol{\beta} \rangle)$ where all $\boldsymbol{s}$ are linearly independent. In the worst case, the vector $\boldsymbol{\beta}$ is collapsed into a space of dimension 1 with size $|K'|$ but $\boldsymbol{\beta}$ is still uniformly distributed over that space. Let $\hat{\boldsymbol{s}} \in \mathbb{Z}_\rho^\eta$ such that $\hat{\boldsymbol{s}}$ is linearly independent with all $\boldsymbol{s}$ obtained by $\mathcal{A}$. Thus $k = \langle \hat{\boldsymbol{s}}, \boldsymbol{\beta} \rangle$ is independent from $\mathcal{A}$'s view and uniformly distributed over K'. If $\mathcal{A}$ find a $\boldsymbol{\pi}$ such that $\boldsymbol{\pi} = \hat{\boldsymbol{\pi}}$, we immediately have attacked the universal$_2$ property of Ξ_2 that $\langle \hat{\boldsymbol{s}}, \boldsymbol{\pi} \rangle = \Xi_2.\mathsf{Hash}(k, (l, \boldsymbol{C}))$. This completes the proof of Eq. (4).

Lemma 2. *The hidden bit b is independent from $\mathcal{A}$'s view that*

$$\Pr[E_m] = \frac{1}{2} \tag{5}$$

Proof. Thanks to the diversity property of Ξ_1, there exists a $r \in K$ such that $\Xi_1.\mathsf{Hash}(r, x) = \chi$ and $\Xi_1.\mathsf{PKGen}(r) = 0$. Note that we do not need to calculate r. Since $\chi \neq 0$, we have $r \neq 0$. Let $\boldsymbol{\gamma} = r \cdot (\boldsymbol{y}_b - \boldsymbol{y}_{1-b}) \in K^\delta$. Thus we have $\Xi_1.\mathsf{Hash}(\gamma_i, x^*) = (y_{b,i} - y_{1-b,i}) \cdot \Xi_1.\mathsf{Hash}(r, x^*) = y_{b,i}\chi - y_{1-b,i}\chi$. Recall the target ciphertext $\boldsymbol{C}^*$ where $C_i^* = y_{b,i}\chi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*)$. Although $\boldsymbol{C}^*$ is a ciphertext for $\boldsymbol{y}_b$, it can also be a ciphertext for $\boldsymbol{y}_{1-b}$ that

$$\begin{aligned}&C_i^* \\ &= y_{1-b,i}\chi + \Xi_1.\mathsf{Hash}(\alpha_i + \gamma_i, x^*) = y_{1-b,i}\chi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*) + \Xi_1.\mathsf{Hash}(\gamma_i, x^*) \\ &= y_{1-b,i}\chi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*) + y_{b,i}\chi - y_{1-b,i}\chi = y_{b,i}\chi + \Xi_1.\mathsf{Hash}(\alpha_i, x^*).\end{aligned}$$

Thus $\boldsymbol{C}^*$ is a ciphertext of $\boldsymbol{y}_b$ for $\boldsymbol{\alpha}$ or $\boldsymbol{y}_{1-b}$ for $\boldsymbol{\alpha}+\boldsymbol{\gamma}$ where $\boldsymbol{\alpha} \neq \boldsymbol{\alpha}+\boldsymbol{\gamma}$. Since the adversary $\mathcal{A}$ can only request the secret keys for $\boldsymbol{x}$ such that $\langle \boldsymbol{x}, \boldsymbol{y}_0 \rangle = \langle \boldsymbol{x}, \boldsymbol{y}_1 \rangle$, we have

$$\langle \boldsymbol{x}, \boldsymbol{\alpha} + \boldsymbol{\gamma} \rangle = \langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle + \langle \boldsymbol{x}, \boldsymbol{\gamma} \rangle = \langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle + \langle \boldsymbol{x}, r \cdot (\boldsymbol{y}_b - \boldsymbol{y}_{1-b}) \rangle = \langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle$$

Hence, the adversary $\mathcal{A}$ cannot distinguish $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha} + \boldsymbol{\gamma}$ from generated keys. Since $\mathsf{PKGen}(\alpha_i + \gamma_i) = \mathsf{PKGen}(\alpha_i) + (y_{b,i} - y_{1-b,i}) \cdot \mathsf{PKGen}(r) = \mathsf{PKGen}(\alpha_i)$, the adversary $\mathcal{A}$ cannot distinguish $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha} + \boldsymbol{\gamma}$ from public keys. Therefore, the hidden bit b is independent from $\mathcal{A}$'s view.

Combining Eqs. (3)–(5), we have

$$\left| \Pr[E_{X \setminus L}] - \frac{1}{2} \right| \leq q \cdot \mathsf{Adv}^{\text{Universal}_2}. \tag{6}$$

Combining Eqs. (1), (2) and (6), we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA}} \leq \mathsf{Adv}_{\mathcal{S}}^{\text{SMP}} + q \cdot \mathsf{Adv}^{\text{Universal}_2}. \tag{7}$$

From Eq. (7), we immediately have the theorem.

5 Instantiation from DDH

Definition 14 (Decisional Diffie-Hellman problem). *Let $\mathbb{G}$ be a cyclic group of prime order p, $a, b \in_R \mathbb{Z}_p$, and $g, T \in_R \mathbb{G}$. Giving two probability distributions $\mathcal{D}_{DDH} = \{(g, g^a, g^b, g^{ab})\}$ and $\mathcal{D}_{rand} = \{(g, g^a, g^b, T)\}$, there is an algorithm $\mathcal{A}$ can distinguish $\mathcal{D}_{DDH}$ and $\mathcal{D}_{rand}$ with advantage:*

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

Let $g_1 = g$ and $g_2 = g^a$. The Decisional Diffie-Hellman (DDH) problem is to distinguish $\mathcal{D}_{\text{DDH}} = \{(g_1, g_2, g_1^b, g_2^b)\}$ and $\mathcal{D}_{\text{rand}} = \{(g_1, g_2, g_1^b, T)\}$. In other words, the problem is to decide whether $\log_{g_1} X_1 = \log_{g_2} X_2$ where $X_1, X_2 \in \mathbb{G}$. Obviously, the DDH problem is a subset membership problem where $X = \mathbb{G}^2$, $L = (g_1^r, g_2^r) \subset X$ and $r \in \mathbb{Z}_p$. We assume the DDH problem is hard. That is, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{DDH}}$ is negligible.

We recall the universal projective hash family proposed by Cramer and Shoup [10] derived from a diverse group system based on the DDH problem. The key space is $K = \mathbb{Z}_p^2$. For the hash key $k = (s_1, s_2) \in K$, the projection key generation is $\alpha(k) = g_1^{s_1} g_2^{s_2} \in S = \Pi = \mathbb{G}$. To compute the hash value of $x = (X_1, X_2) \in X = \mathbb{G}^2$ with the hash key k, it computes $H_k(x) = X_1^{s_1} X_2^{s_2} \in \Pi$. To compute the have value of $x = (g_1^w, g_2^w) \in L$ with the projection key $\alpha(k)$ and a witness $w \in W = \mathbb{Z}_p$, it computes $H_k(x) = \alpha(k)^w \in \Pi$.

By applying Construction 2, we obtain a HPS Ξ_1. From Theorems 1–3 and $|\Pi| = |\mathbb{Z}_p| = p$, we have that Ξ_1 has key linearity and hash linearity, and is diverse. To ensure that the underlying group system is diverse, we show the existence of ϕ (or equivalent secret key k). Let $r \in \mathbb{Z}_p^+$ and $k = (r, -r\log_{g_2} g_1)$. For all $x = (g_1^w, g_2^w) \in L$, we have $H_k(x) = (g_1^w)^r (g_2^w)^{-r\log_{g_2} g_1} = g_1^0$. For all $x = (g_1^{w_1}, g_2^{w_2}) \in X \setminus L$, we have $H_k(x) = (g_1^{w_1})^r (g_2^{w_2})^{-r\log_{g_2} g_1} = g_1^{r(w_1 - w_2)}$. To simplify our instantiation, we choose $r = 1$ and $x = (g_1^2, g_2)$, and computes $\chi = \Xi_1.\mathsf{Hash}(k, x) = g_1$. By applying Constructions 2 and 3 we obtain another HPS Ξ_2, which is universal$_2$. From Theorem 2, we have that Ξ_2 has hash linearity. Note that we use a collision resistant hash function (CRHF) $H : \mathbb{G}^2 \times \mathbb{G}^\delta \rightarrow \mathbb{Z}_p$ instead of an injective map Γ when applying Construction 3.

With Ξ_1 and Ξ_2, we apply Construction 4 to construct a functional encryption scheme for the inner product functionality $F : \mathbb{Z}_p^\delta \times \mathbb{Z}_p^\delta \rightarrow \mathbb{Z}_p$.

Construction 5. *Our instantiation works as follows.*

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

$$H : \mathbb{G}^2 \times \mathbb{G}^\delta \rightarrow \mathbb{Z}_p, \quad g_1, g_2 \in_R \mathbb{G}.$$
$$\textit{For } i = 1 \ldots \delta, \quad \alpha_i = (\alpha_{i,1}, \alpha_{i,2}) \in_R \mathbb{Z}_p^2, \quad A_i = g_1^{\alpha_{i,1}} g_2^{\alpha_{i,2}}.$$
$$\textit{For } i = 1 \ldots \eta, \quad \beta_i = (\beta_{i,1}, \beta_{i,2}, \beta_{i,3}, \beta_{i,4}) \in_R \mathbb{Z}_p^4,$$
$$B_i = (B_{i,1}, B_{i,2}) = (g_1^{\beta_{i,1}} g_2^{\beta_{i,2}}, g_1^{\beta_{i,3}} g_2^{\beta_{i,4}}).$$

return $(\mathsf{PK}, \mathsf{MSK}) = ((\boldsymbol{A}, \boldsymbol{B}), (\boldsymbol{\alpha}, \boldsymbol{\beta}))$.

– $\mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{x})$:

$$\boldsymbol{s} \in_R \mathbb{Z}_p^\eta, \quad K_1 = (K_{1,1}, K_{1,2}) = \left(\sum_{i=1}^{\delta} x_i \alpha_{i,1}, \sum_{i=1}^{\delta} x_i \alpha_{i,2} \right)$$
$$K_2 = (K_{2,1}, K_{2,2}, K_{2,3}, K_{2,4}) = \left(\sum_{i=1}^{\eta} s_i \beta_{i,1}, \sum_{i=1}^{\eta} s_i \beta_{i,2}, \sum_{i=1}^{\eta} s_i \beta_{i,3}, \sum_{i=1}^{\eta} s_i \beta_{i,4} \right)$$

return $\mathsf{SK} = (\boldsymbol{x}, \boldsymbol{s}, K_1, K_2)$.

– $C \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \boldsymbol{y})$:

$$r \in_R \mathbb{Z}_p, \quad l = (u_1, u_2) = (g_1^r, g_2^r), \quad \textit{For } i = 1 \ldots \delta, \quad C_i = g_1^{y_i} \cdot A_i^r$$
$$h = H(u_1, u_2, C_1, \ldots, C_\delta), \quad \textit{For } i = 1 \ldots \eta, \quad \pi_i = (B_{i,1} \cdot B_{i,2}^h)^r$$

return $C = (l, \boldsymbol{C}, \boldsymbol{\pi})$.

– $D \leftarrow \mathsf{Decrypt}(\mathsf{SK}, C)$:

$$h = H(u_1, u_2, C_1, \ldots, C_\delta),$$
$$\prod_{i=1}^{\eta} \pi_i^{s_i} \overset{?}{=} u_1^{K_{2,1} + h \cdot K_{2,3}} u_2^{K_{2,2} + h \cdot K_{2,4}}, \quad D^* = \frac{\prod_{i=1}^{\delta} C_i^{x_i}}{u_1^{K_{1,1}} u_2^{K_{1,2}}}$$

return $D = \log_{g_1} D^*$.

As mentioned in Sect. 4.2, we can decrypt D in an alternative manner instead of calculating $\log_{g_1} D^*$ if $|\{D\}|$ is polynomial-sized.

6 Conclusion

In this paper, we reviewed the hash proof system (HPS) introduced by [10] and defined new properties of HPS. We found that the existing HPS constructions by [10] have those new properties. As the main contribution of this paper, we proposed an IND-CCA secure functional encryption for inner products, which can be generically constructed from a diverse HPS with key linearity and hash linearity, and a universal$_2$ HPS with hash linearity. Moreover, we constructed a concrete scheme from DDH assumption via our proposed generic construction.

One of our future work will be relaxing the scheme without limiting the number of user who can decrypt. Another future work will be finding new HPS, which has our defined properties, from other subset membership problems so that we can construct new functional encryption schemes under new assumptions.

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). http://eprint.iacr.org/
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. 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
4. 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
5. Boneh, D., Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_30
6. 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). doi:10.1007/3-540-44647-8_13
7. 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
8. 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). doi:10.1007/978-3-540-70936-7_29
9. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998). doi:10.1007/BFb0055717
10. 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
11. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_32
12. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 555–564 (2013)
13. 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
14. 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

15. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_22
16. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010). http://eprint.iacr.org/
17. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992). doi:10.1007/3-540-46766-1_35
18. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
19. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_14