Leakage-Resilient Inner-Product Functional Encryption in the Bounded-Retrieval Model

Linru Zhang[1], Xiangning Wang[1], Yuechen Chen[1], and Siu-Ming Yiu[1]

Department of Computer Science, The University of Hong Kong, HKSAR, China
{lrzhang,xnwang,ycchen,smyiu}@cs.hku.hk

Abstract. We propose a leakage-resilient inner-product functional encryption scheme (IPFE) in the bounded-retrieval model (BRM). This is the first leakage-resilient functional encryption scheme in the BRM. In our leakage model, an adversary is allowed to obtain at most l-bit knowledge from each secret key. And our scheme can flexibly tolerate arbitrarily leakage bound l, by only increasing the size of secret keys, while keeping all other parts small and independent of l.

Technically, we develop a new notion: Inner-product hash proof system (IP-HPS). IP-HPS is a variant of traditional hash proof systems. Its output of decapsulation is an inner-product value, instead of the encapsulated key. We propose an IP-HPS scheme under DDH-assumption. Then we show how to make an IP-HPS scheme to tolerate l'-bit leakage, and we can achieve arbitrary large l' by only increasing the size of secret keys. Finally, we show how to build a leakage-resilient IPFE in the BRM with leakage bound $l = \frac{l'}{n}$ from our IP-HPS scheme.

Keywords: Inner-Product Functional Encryption · Bounded-Retrieval Model · Hash Proof System.

1 Introduction

Leakage-resilient Cryptography. In traditional cryptography model, security usually relies on complete privacy of the secret values, such as secret keys and randomness. For many cryptographic systems in such a model, even if a single bit of these secrets is leaked, then the security will totally lose. However, it is often unrealistic to avoid all kinds of leakage of the secret values. Actually, developments of side channel attacks [38, 41–43] have found that the adversary is possible to obtain partial information of these secret values by capturing the physical nature of cryptographic operations. Cryptographic systems should be proven secure against the largest possible class of potential adversaries. Therefore, a new topic of modern cryptography: *leakage-resilient cryptography* appeared.

Leakage-resilient cryptography was introduced to provide formal security guarantees even the adversary can obtain some information of the secret values. There have been lots of studies on *leakage-resilient cryptography*, including public key encryption [5, 6, 14, 15, 26, 46], identity-based encryption [16, 20, 44, 58], attribute-based encryption [57, 59], signatures [13, 40] and so on.

The first step of achieving leakage-resilience is to decide an appropriate model of what information of secrets the adversary can learn. If the adversary can learn anything of the secret keys, then it is impossible to design a secure cryptographic system. So we have to restrict the power of the adversary. We may bound the amount of leakages the adversary can obtain in the following models.

Relative-leakage model. In this model, the secret key size is chosen in the same way as in standard cryptographic systems, which is based on the security parameter. We bound a leakage-ratio $0 < \mu < 1$, then we allow the adversary to obtain $\mu|\mathsf{sk}|$ bits from a secret key with bit-length $|\mathsf{sk}|$. In this model, no matter what the secret key size is, the adversary can get some imperfect reading of the secret key.

Bounded-retrieval model. The *bounded-retrieval model* (BRM) [23,29] is a generalization of the *relative-leakage model*. In this setting, the leakage bound l is decided by external factors, and we can resist such attacks by increasing the length of the secret key, to dominate l. Thus, we hope that the size of secret key can be set flexibly depending on the security parameter and the leakage bound l. When l is extremely large, it is desirable that we can resist such attacks by only increasing the length of the secret key without affecting efficiencies of others, such as public key size, encryption time, decryption time and even master secret key size in the case of IBE. The BRM is to ensure that all efficiency parameters other than the secret key size only depend on the security parameter, and not on the leakage bound l.

Functional Encryption. As another new tide of modern cryptography, *functional encryption* (FE) [12,49] was proposed to address the "all-or-nothing" issue of traditional *public key encryption* (PKE). That is, the decryption result of traditional PKE is the plaintext if the secret key sk matches the public key pk, or nothing otherwise. Traditional PKE is found to be insufficient for many emerging applications in which users are only allowed to obtain a function value of the ciphertext without any other information about the ciphertext. Roughly speaking, considering a functional encryption scheme for a functionality $F(k, x)$, where k is in the key space and x is in the plaintext space, the authority with the master secret key can generate secret key sk_k for each value k. Given a ciphertext of x, the user who holds sk_k can only learn $F(k, x)$ and nothing else except possible the length of x. Before the definition of FE appears, there were many works to overcome the "all-or-nothing" barrier. These works, including identity-based encryption (IBE) [11, 34, 39, 51, 54], attribute-based encryption (ABE) [37, 55] and predicate encryption (PE) [52], are considered as special cases of FE.

After proposing the definition of FE, researchers started to build FE schemes for general circuits, Turing machines and some very powerful functions [7,33,35,36,56]. But these FE schemes either have bounded collusion, or have to rely on powerful, but impractical and not well studied assumptions (indistinguishable obfuscation (IO) and its variants, or polynomial hardness of simple assumptions on multi-linear maps). Attacks were identified for some constructions that are based on IO and multi-linear maps [8, 18, 19, 21].

Functional encryption for inner-product (IPFE) from standard assumptions. Many works try to build efficient schemes for specific functions from well studied standard assumptions in recent years [1–3, 9, 60]. Most of them started their work from inner-product, which is simple but very useful. More precisely, given an encrypted vector $\boldsymbol{x}$ from message space $\mathcal{X}$ and a secret key $\mathsf{sk}_{\boldsymbol{y}}$ based on vector $\boldsymbol{y}$ in the key space $\mathcal{K}$, the decryption algorithm will output the inner-product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ without revealing any other information about $\boldsymbol{x}$ except the length of it. One of practical applications of IPFE is to calculate the *weighted mean*, a useful tool to describe the main features of a collection of information in statistics, and to protect the privacy of the data set which is used to calculate the weighted mean.

Leakage-resilient functional encryption in the BRM. While there are many existing results about PKE and IBE in the BRM, designing FE schemes in the BRM seems not easy. When considering the security model of FE, unlike traditional PKE, [49] showed that simulation-based security (SIM-security) is not always achievable for FE. So Indistinguishability-based security (IND-

security) is widely used in FE research. It is a folklore in the literature that there is a restriction in IND-security that all secret key queries for function $F(k,\cdot)$ should ensure that $F(k,x_0) = F(k,x_1)$, where x_0, x_1 are the challenge ciphertexts.

However, this restriction causes that the IND-security of FE is weak in the sense that a trivially insecure scheme for a certain functionality can be proved IND-secure [12, 49]. One possible way to enhance the IND-security is to allow the adversary to get some knowledge about the secret keys for functions $F(k,\cdot)$ where $F(k,x_0) \neq F(k,x_1)$. More precisely, the adversary is allowed to make *leakage query* to such secret keys to collect some information. Of course, if an adversary can get unrestricted information about the secret key, i.e., it can learn the secret key for $F(k,\cdot)$ where $F(k,x_0) \neq F(k,x_1)$ totally, then it can distinguish whether the challenge ciphertext is an encryption of x_0 or x_1 easily. Thus, we must place some restrictions on the type or amount of information that the adversary can learn through leakage queries. Therefore, it is the time to build FE schemes which is still IND-secure even the adversary can obtain a bounded amount of leakage to such secret keys.

The only related work [53] considered leakage-resilient FE for general functions in the relative-leakage model. They presented a leakage-resilient CCA-secure generic construction for single-key and single-ciphertext functional encryption via hash proof system(HPS), one-time lossy filter and garbled circuits. But the power of the adversary in this work is very limited since queries for one secret key and one ciphertext can be made. And another drawback is that when the system tries to tolerate a larger amount of leakage, the efficiencies of all parts become lower. Therefore, it is insufficient for practical application of functional encryption.

1.1 Our results

Towards practical functional encryption, we focus our research on leakage-resilient IPFE from standard assumptions in the BRM. We use the indistinguishability-based security model together with a leakage query oracle to describe its security. Any adversary can access the leakage query oracle with some secret keys and functions certain times before seeing the challenge ciphertext as long as for each key $\mathrm{sk}_{\boldsymbol{y}}$, the total number of bits output by the leakage query oracle is at most the leakage bound l (i.e., $\sum_f |f(\mathrm{sk}_{\boldsymbol{y}})| \leq l$, where $|f(\mathrm{sk}_{\boldsymbol{y}})|$ is the bit-length of $f(\mathrm{sk}_{\boldsymbol{y}})$).

As our main contribution, our leakage-resilient IPFE scheme and its security proof build on *hash proof system* [22]. [1] [5, 46] showed how to use a *hash proof system* (HPS) to construct leakage-resilient PKE and IBE schemes. An HPS can be viewed as a *key encapsulation mechanism* (KEM) with specific structure. A KEM includes a key generation algorithm to generate public key and secret key, an encapsulation algorithm to generate a pair of ciphertext and encapsulated key, and a decapsulation algorithm which uses the secret key to recover the encapsulated key from a ciphertext.

An HPS is a KEM with the following properties: (1) An HPS includes an invalid-encapsulation algorithm to generate invalid ciphertexts. And the invalid ciphertexts are computationally indistinguishable from those valid ciphertexts generated by a valid-encapsulation algorithm. (2) The output of decapsulation algorithm with input a fixed invalid ciphertext and a secret key is related to the random numbers used to generate the invalid ciphertext and the secret key. The main benefit of using HPS to construct encryption scheme is that, when proving the security, after switching the

[1] [10] showed how to construct an IPFE scheme from projective hash functions. But in their construction, the projective hash function is considered as a building block which is not related to the functionality in IPFE. And the way they build the construction is just like building an IPFE scheme from a PKE scheme. So it is difficult to build connection between the leakage-resilience of IPFE and the smoothness of hash functions.

valid ciphertext into invalid ciphertext in the first step, we can argue the leakage using information-theoretic analysis.

However, existing HPS such as IB-HPS in [5] cannot be applied to our cases directly. Recall that IPFE requires that the decryption result only reveals an inner-product value of two vectors and nothing else. When we convert an HPS into an encryption scheme, we usually use the encapsulated key as a mask to hide the plaintext in the encryption algorithm, and recover the plaintext from ciphertext by running decapsulation algorithm to get the encapsulated key. But when applying to FE, if the decapsulation algorithm of the underlying HPS still outputs the encapsulated directly, then the decryption of FE will reveal the plaintext vector, other than an inner-product value only. In order to guarantee the security of resulting IPFE scheme, some modifications are needed on the underlying HPS definition. Here, we develop the notion *Inner-product hash proof system*(IP-HPS), which can yield an IPFE scheme. Different from other HPS, in an IP-HPS scheme, the valid/invalid encapsulation algorithms will take a vector as input and will output a ciphertext and a encapsulated key $\boldsymbol{k}$. The key generation algorithm will output a secret key for a vector $\boldsymbol{y}$. And the decapsulation algorithm will output an inner-product value of $\boldsymbol{y}$ and the encapsulated key $\boldsymbol{k}$. Actually, this is the first hash proof system whose output of decapsulation is not the encapsulated key itself. This modification ensures that we can get a secure IPFE from IP-HPS very easily, by simply using the encapsulated key as a one-time pad to encrypt a message. As a benefit of it, we can move our focus from leakage-resilience property of IPFE to a *leakage-smoothness* property of IP-HPS. Leakage-smoothness states that the distribution of encapsulated key derived from an invalid ciphertext and secret keys is almost uniform over the key space, even if the adversary can obtain at most l' bits information about the secret keys, where l' is a pre-determined leakage bound. We prove the following theorem:

Theorem 1 (informal). *Given a l'-leakage-smooth IP-HPS, we can get a $l = \frac{l'}{n}$-leakage-resilient IPFE. And when the IP-HPS scheme meets the efficiency requirements of the BRM, the resulting IPFE scheme also meets the efficiency requirements of the BRM.*

Now, our goal is to design a l'-leakage-smooth IP-HPS, which meets the efficiency requirements of the BRM. As the first step to do it, we would like to design an IP-HPS scheme from simple assumptions, without the requirements of leakage-smoothness and efficiency. We build an IP-HPS Π_1 over $\mathbb{Z}_p$ from an IPFE scheme [3] based on DDH assumption. Notice that the key generation algorithm in the IPFE scheme [3] is deterministic, while in HPS, we require that the secret key is generated randomly. Thus, in the key generation algorithm of Π_1, we first choose a random number and form a new vector by concatenating $\boldsymbol{y}$ and the random number. Then we run the key generation algorithm of the IPFE scheme with input the new vector, and thus the new secret key $\mathrm{sk}_{\boldsymbol{y}}$ is related to the random number we chosed. Then, we study a property called 0-universality of the decapsulation algorithm in Π_1. The 0-universality ensures that it is impossible that any two distinct secret keys for the same vector $\boldsymbol{y}$ will decapsulate an invalid ciphertext to the same value. With these properties, we show that we are able to convert Π_1 into an l'-leakage-smooth IP-HPS for arbitrarily large leakage-bound l':

Theorem 2 (informal). *Given Π_1, we can get an l'-leakage-smooth IP-HPS Π_2 for arbitrarily large leakage bound l', and Π_2 meets the efficiency requirements of the BRM.*

Firstly, we find that the leakage amplification method of IB-HPS in [5], which can be viewed as *parallel-repetition* with small public key size, cannot be applied to our cases here. In IB-HPS, the output of the decapsulation is already the encapsulated key, then the leakage-smoothness of

their scheme can be proved from the 0-universality by leftover-hashing lemma [47]. Thus the only thing they need to do is to amplify the leakage bound while meeting the efficiency requirements of the BRM. However, in IP-HPS, the output of decapsulation is an inner-product value between the encapsulated key and the vector in the secret key, so we need at least n secret keys to determine an encapsulated key. Then, we cannot find the relation between leakage-smoothness and universality very easily. Thus, our task is to convert an IP-HPS with 0-universality of decapsulation algorithm into an leakage-smoothness IP-HPS for arbitrarily large leakage bound and meets the efficiency requirements of the BRM.

Although the leakage amplificaion method cannot be applied directly, there are some ideas we can borrow. We introduce a key-size parameter m, which gives us flexibility in the size of secret key and will depend on the desired leakage bound l'. And also, due to the efficiency requirements, the encapsulation will choose only target on a small subset from $\{1, ..., m\}$, and show that the size of the subset (denote by η) is independent of l'. Then, recall that we need n secret keys to recover one encapsulated key. In order to finish the proof of leakage-smoothness, the key generation will take an invertible $n \times n$ matrix Y as input and the encapsulation algorithm will output n ciphertexts wihch shares the same encapsulated key.

In the proof, we use a similar idea with *approximately universal hashing* defined in [5], where we only insist that two secret keys generated by running the key generation algorithm with the same input Y which are different enough are unlikely to result in a same encapsulated key. Then we obtaion the leakage-smoothness by applying a variant of leftover-hash lemma, and show our scheme meets the efficiency requirements of the BRM by giving a lower bound of η, which is independent of the leakage bound l'.

We sum up our results in the following:
(1) Give the definition of IP-HPS, together with a series of properties. And propose an IP-HPS construction Π_1 from DDH assumption.
(2) Show how to build a l'-leakage-smooth IP-HPS Π_2 from our IP-HPS Π_1 for arbitrarily large l', and meets the efficiency requirements of the BRM.
(3) Develop the security definition for a leakage-resilient IPFE scheme with leakage bound l, and the definition of leakage-resilient IPFE in the BRM. Then show how to build a leakage-resilient IPFE scheme Π_3 in BRM from our leakage-smooth IP-HPS Π_2.

1.2 Related works: Leakage-resilient cryptography

There are several models in the research line of leakage-resilience. [45] started the line of formal modeling of side-channel attacks by proposing the first model *only computation leaks information*. In this model, a function of only the bits accessed is leaked when the cryptographic system is called each time. Stream ciphers [31, 50] and signature schemes [32] were proposed under this model. However, this model cannot capture many types of leakage-attack, such as *cold-boot attack* [38], in which all memory contents can leak information regardless whether it is accessed.

In order to capture these attacks, many works try to study about *relative-leakage model*, in which a proportion of secret values can be leaked. The public-key encryption schemes [4, 46], signature schemes [40], and IBE schemes [20] were proposed under this model. *Bounded-retrieval model* was proposed by [23, 29]. In this model, the amount of information can be leaked is bounded by an external parameter, and this leakage bound can be very large. Further, it requires that the efficiencies of other parts of cryptographic system (except the length of secret key) should be independent from the leakage bound. Many works [5, 17, 30, 48] proposed different cryptographic systems under this model. *Auxiliary inputs model* was introduced by [27], in which an adversary is given auxiliary

input $h(s)$, and it is computationally hard to find s (the secret values) from $h(s)$. Symmetric encryption schemes [27], public-key encryption schemes [24] and IBE schemes [58] were proposed under this model. *Continual leakage model* was introduced by [15, 25], where there is a notion of time periods and secret values will be updated at the end of each time period. In this model, an adversary is allowed to obtain a bounded amount of information of secret values in each time period, but there is no limitation on the total amount of information it can obtain in all time periods. Public-key encryption schemes [15], IBE schemes [15, 44, 58], ABE schemes [44, 57][2] and signature schemes [15, 25] were proposed under this model.

2 Preliminaries

Notations. Let $[n]$ denote set $\{1, \ldots, n\}$. For vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, let $\boldsymbol{x}||\boldsymbol{y}$ be their concatenation. For a set S, define U_S be the uniform distribution over S. Similarly, let U_v be the uniform distribution over $\{0,1\}^v$.

2.1 Functional encryption (FE)

We define FE and its indistinguishable security here. Following [12], we start by defining the notion of functionality and then that of functional encryption scheme for functionality $\mathcal{F}$.

Definition 1 (Functionality and FE scheme). *A functionality $\mathcal{F}$ defined over $(\mathcal{K}, \mathcal{X})$ is a function $\mathcal{F} : \mathcal{K} \times \mathcal{X} \to \Sigma \cup \{\perp\}$, where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the message space and Σ is the output space and $\perp$ is a special string not contained in Σ. Notice that the functionality is undefined for when either the key is not in the key space or the message is not in the message space.*
A FE scheme for functionality $\mathcal{F}$ consists of 4 PPT algorithms just like FE: (Setup, KeyGen,Encrypt, Decrypt)*. The algorithms have the following syntax.*

- $\text{Setup}(1^\lambda)$*: It takes the security parameter λ as input, and produces the master public key* mpk *and the master secret key* msk*. The following algorithms implicitly include* mpk *as input.*
- $\text{KeyGen}(\text{msk}, k)$*: It uses the master secret key* msk *and key $k \in \mathcal{K}$ to sample a secret key* sk_k*.*
- $\text{Encrypt}(\text{mpk}, x)$*: It uses the master public key* mpk *and a message $x \in \mathcal{X}$ to generate a ciphertext* ct_x*.*
- $\text{Decrypt}(\text{sk}_k, \text{ct}_x)$*: It takes a ciphertext* ct_x *and a secret key* sk_k *as input and outputs $\mathcal{F}(k, x)$*

Correctness. For any (mpk, msk) generated by $\text{Setup}(1^\lambda)$, any $k \in \mathcal{K}$ and $x \in \mathcal{X}$, we have:

$$\mathbf{Pr}\left[\mathcal{F}(k,x) \neq \gamma \middle| \begin{array}{c} \text{sk}_k \leftarrow \text{KeyGen}(\text{msk}, k) \\ \text{ct}_x \leftarrow \text{Encrypt}(\text{mpk}, x), \quad \gamma = \text{Decrypt}(\text{ct}_x, \text{sk}_k) \end{array}\right] \leq \text{negl}(\lambda) \ .$$

Indistinguishable security. We define the *indistinguishable security game*, parameterized by a security parameter λ as the following game between an adversary $\mathcal{A}$ and a challenger in Table 1. The *advantage* of an adversary $\mathcal{A}$ in the indistinguishable security game is defined by $Adv_{FE,\mathcal{A}}^{FE-IND}(\lambda) := |\mathbf{Pr}[\mathcal{A}\ wins] - \frac{1}{2}|$.

[2] In [57], it said that they discovered leakage-resilient functional encryption scheme for regular languages based on composite-order pairing groups in continual memory leakage (CML) model. However, in a functional encryption scheme for regular languages, a secret key sk_M is associated with a deterministic finite automata M, and a ciphertext ct encrypts a message m and is associated with an arbitrary length string w. A user holds sk_M is able to decrypt the ciphertext ct if and only if M accepts the string w. Notice that the decryption result is still m or nothing, so it actually can be viewed as a ABE scheme for wider classes of functionality.

Table 1: FE-IND(λ)

Setup: The challenger computes $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{Setup}(1^\lambda)$ and sends mpk to the adversary $\mathcal{A}$.
Query 1: The adversary $\mathcal{A}$ can adaptively ask the challenger for the following queries: *Secret key query*: On input $k \in \mathcal{K}$, the challenger replies with sk_k.
Challenge: The adversary $\mathcal{A}$ chooses two vectors $x_0, x_1 \in \mathcal{X}$ subject to the restriction that for all k that the adversary have make the *secret key query* in **Query 1**, it holds that $\mathcal{F}(k, x_0) = \mathcal{F}(k, x_1)$. The challenger chooses $b \leftarrow \{0,1\}$ uniformly at random and computes $\mathrm{ct}_b \leftarrow \mathrm{Encrypt}(\mathrm{mpk}, x_b)$ and gives ct_b to the adversary $\mathcal{A}$.
Query 2: The adversary can make *secret key query* for arbitrary k as long as $\mathcal{F}(k, x_0) = \mathcal{F}(k, x_1)$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and wins if $b' = b$.

Definition 2 (IND-secure FE). *A FE scheme is IND-secure, if (1) it satisfies the* correctness*, and (2) the advantage of any PPT adversary $\mathcal{A}$ in the indistinguishable security game is* $Adv^{FE-IND}_{FE,\mathcal{A}}(\lambda) = \mathrm{negl}(\lambda)$.

Inner-product functionality. Here, we are interested in the *inner-product functionality* over the field $\mathbb{Z}_p$ defined in [1]. It is a family of functionalities with key space $\mathcal{K}_n$ and message space $\mathcal{X}_n$ both consisting of vectors in $\mathbb{Z}_p$ of length n: for any $\boldsymbol{y} \in \mathcal{K}_n, \boldsymbol{x} \in \mathcal{X}_n$, the functionality $\mathcal{F}(\boldsymbol{y}, \boldsymbol{x}) = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$.

3 Inner product hash proof system(IP-HPS)

3.1 Definitions

To construct a leakage-resilient IPFE scheme, we introduce the notion, IP-HPS, together with the required properties. An *Inner product hash proof system* (IP-HPS) consists of 5 PPT algorithms just like IB-HPS. The algorithms have the following syntax. ($\mathcal{M}$ is the message space and $\mathcal{K}$ is the encapsulated-key space.)

- $\mathrm{Setup}(1^\lambda, 1^n)$: It takes the security parameter λ and n as input, and produce the *master public key* mpk and the *master secret key* msk. The following algorithms implicitly include mpk as input.
- $\mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y})$: It uses msk and a vector $\boldsymbol{y} \in \mathcal{K}$ with length n to sample a secret key $\mathrm{sk}_{\boldsymbol{y}}$.
- $\mathrm{Encap}(\boldsymbol{z})$: This is the *valid* encapsulation algorithm. It uses $\boldsymbol{z} \in \mathcal{M}$ to output a valid ciphertext $\mathrm{ct}_{\boldsymbol{z}}$ and a encapsulated key $\boldsymbol{k}$.
- $\mathrm{Encap}^*(\boldsymbol{z})$: This is the *invalid* encapsulation algorithm. It uses $\boldsymbol{z} \in \mathcal{M}$ to output only an invalid ciphertext $\mathrm{ct}_{\boldsymbol{z}}$.
- $\mathrm{Decap}(\mathrm{ct}_{\boldsymbol{z}}, \mathrm{sk}_{\boldsymbol{y}}, \boldsymbol{y})$: This is the decapsulation algorithm(deterministic). It takes a ciphertext as input and outputs an inner product of the encapsulated key and $\boldsymbol{y}$: $\langle \boldsymbol{k}, \boldsymbol{y} \rangle$.

Correctness. Given $\mathrm{msk}, \mathrm{mpk}$ from $\mathrm{Setup}(1^\lambda, 1^n)$ and $\boldsymbol{y}$ with length n, we have:

$$\mathbf{Pr}\left[\langle \boldsymbol{k}, \boldsymbol{y} \rangle \neq \gamma \middle| \begin{array}{c} \mathrm{sk}_{\boldsymbol{y}} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y}) \\ (\mathrm{ct}_{\boldsymbol{z}}, \boldsymbol{k}) \leftarrow \mathrm{Encap}(\boldsymbol{z}), \quad \gamma = \mathrm{Decap}(\mathrm{ct}_{\boldsymbol{z}}, \mathrm{sk}_{\boldsymbol{y}}, \boldsymbol{y}) \end{array}\right] \leq \mathrm{negl}(\lambda) \ .$$

The correctness requires that a ciphertext generated by Encap can be correctly decapsulated to the corresponding inner-product of the encapsulated key and the vector $\boldsymbol{y}$ in the secret key.

Valid/Invalid Ciphertext Indistinguishiability. Given the same input, the valid ciphertext generated by Encap and the invalid ciphertext generated by Encap^* should be computationally indistinguishable. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we define the following experiment for an IP-HPS Π in Table 2:

Table 2: V/I-IND(λ, n)

Setup: The challenger computes $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$ and sends mpk to the adversary $\mathcal{A}$.
Query 1: The adversary $\mathcal{A}$ can adaptively ask the challenger for the following queries: *Secret key query*: On input $\boldsymbol{y} \in \mathcal{K}$, the challenger replies with $\text{sk}_{\boldsymbol{y}}$.
Challenge: The adversary $\mathcal{A}$ chooses a vector $\boldsymbol{z} \in \mathcal{M}$ and sends it to the challenger. The challenger computes $\text{ct}_0 \leftarrow \text{Encap}(\boldsymbol{z})$ and $\text{ct}_1 \leftarrow \text{Encap}^*(\boldsymbol{z})$. The challenger chooses $b \leftarrow \{0,1\}$ uniformly at random and gives ct_b to the adversary $\mathcal{A}$.
Query 2: The adversary can make *secret key query* for arbitrary $\boldsymbol{y}$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and wins if $b' = b$.

The challenger computes $\text{sk}_{\boldsymbol{y}} \leftarrow \text{KeyGen}(\text{msk}, \boldsymbol{y})$ the first time that $\boldsymbol{y}$ is queried and responds to all future queries on the same $\boldsymbol{y}$ with the same $\text{sk}_{\boldsymbol{y}}$.

Definition 3. *A PPT adversary $\mathcal{A}$ is* admissible *if it makes at most n key queries with linear independent vectors. Then, we say that an IP-HPS Π is* adaptively secure *if for any admissible adversary $\mathcal{A}$, the advantage satisfies:* $\text{Adv}^{ind}_{\Pi,\mathcal{A}}(\lambda, n) := |\mathbf{Pr}[\text{Exp}^{ind}_{\Pi,\mathcal{A}}(\lambda, n) = 1] - \frac{1}{2}|$.

The valid/invalid ciphertext indistinguishability requires that the valid and invalid ciphertexts are computationally indistinguishiable even if an adversary can obtain one secret key per vector for at most n linear independent vectors. We explain why there is a restriction of numbers of key queries here. By the requirement of HPS, the secret keys should be related to some random numbers chosen by the key generation algorithm at each running. As a result of it, the output of decapsulation which takes an invalid ciphertext and a secret key as input is dependent on the random numbers used to generate the secret key. However, the output of decapsulation with a valid ciphertext is always the real inner-product value. For example, the adversary first makes 2 key queries with $\boldsymbol{y}_1 = (1, 0, ..., 0)$ and $\boldsymbol{y}_2 = (2, 0, ..., 0)$. If the ciphertext ct is a valid one, then $\text{Decap}(\text{ct}, \text{sk}_{\boldsymbol{y}_2}) - \text{Decap}(\text{ct}, \text{sk}_{\boldsymbol{y}_1}) = 2k_1 - k_1 = k_1 = \text{Decap}(\text{ct}, \text{sk}_{\boldsymbol{y}_1})$. However, if the ciphertext ct is an invalid one, then $\text{Decap}(\text{ct}, \text{sk}_{\boldsymbol{y}_2}) - \text{Decap}(\text{ct}, \text{sk}_{\boldsymbol{y}_1}) \neq \text{Decap}(\text{ct}, \text{sk}_{\boldsymbol{y}_1})$ since the random numbers used to generate $\text{sk}_{\boldsymbol{y}_2}$ and $\text{sk}_{\boldsymbol{y}_1}$ are different. Thus, the adversary can distinguish whether one ciphertext is valid or invalid. Note that, during the challenge phase, the adversary can choose *any* vector $\boldsymbol{z}$ from the message space, since there is only one vector is chosen in the **Challenge** stage, instead of 2 vectors in the definiciton of IND-security of IPFE.

We still need the following information theoretic properties, as in [5].

Definition 4 (ρ-Universality). *A family $\mathcal{H}$, consisting of (deterministic) functions $h(\cdot)$, is* ρ-universal *if for any $x_1 \neq x_2$, we have $\mathbf{Pr}_{h \leftarrow \mathcal{H}}[h(x_1) = h(x_2)] \leq \rho$. Then, an IP-HPS Π is* ρ-universal *if: fix* mpk, msk *from* $\text{Setup}(1^\lambda, 1^n)$*, two vectors $\boldsymbol{y}$ and $\boldsymbol{z}$, $\{\text{Decap}(\text{ct}, \cdot, \boldsymbol{y}) | \text{ct} \leftarrow \text{Encap}^*(\boldsymbol{z})\}$ is a ρ-universal hash family.*

Definition 5 (Smoothness and Leakage-smoothness). *Define an $n \times n$ invertible matrix $Y := [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]$. Define the statistical distance $\mathbf{SD}(X, Y) := \frac{1}{2} \sum_w |\mathbf{Pr}[X = w] - \mathbf{Pr}[Y = w]|$. We say*

that an IP-HPS Π is smooth *if, for any fixed values of* $\mathrm{mpk}, \mathrm{msk}$ *from* $\mathrm{Setup}(1^\lambda, 1^n)$*, any fixed* Y *and* $\boldsymbol{z} \in \mathcal{M}$*, we have*

$$\mathbf{SD}\left((\mathrm{ct}, \boldsymbol{k}), (\mathrm{ct}, \boldsymbol{k}')\right) \leq \mathrm{negl}(\lambda),$$

where $\mathrm{ct} \leftarrow \mathrm{Encap}^*(\boldsymbol{z})$*,* $\boldsymbol{k}' \leftarrow U_{\mathcal{K}}$ *and* $\boldsymbol{k}$ *is sampled by first choosing* $\mathrm{sk}_{\boldsymbol{y}_i} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y}_i)$ *for each* i *and then computing* $\boldsymbol{k}^T := [\mathrm{Decap}(\mathrm{ct}, \mathrm{sk}_{\boldsymbol{y}_1}), \ldots, \mathrm{Decap}(\mathrm{ct}, \mathrm{sk}_{\boldsymbol{y}_n})]Y^{-1}$.

An IP-HPS Π is l-leakage-smooth *if, for any (possible randomized and inefficient) function f with at most l-bit output, we have*

$$\mathbf{SD}\left((\mathrm{ct}, f(\{\mathrm{sk}_{\boldsymbol{y}_i}\}_{i=1}^n), \boldsymbol{k}), (\mathrm{ct}, f(\{\mathrm{sk}_{\boldsymbol{y}_i}\}_{i=1}^n), \boldsymbol{k}')\right) \leq \mathrm{negl}(\lambda),$$

where $\mathrm{ct}, \boldsymbol{k}', \boldsymbol{z}, \boldsymbol{k}$ *and each* $\mathrm{sk}_{\boldsymbol{y}_i}$ *are sampled as above.*

3.2 Construction of IP-HPS Π_1

- $\mathrm{Setup}(1^\lambda, 1^n)$: It chooses a cyclic group $\mathbb{G}$ of prime order $p > 2^\lambda$, together with generators $g, h \leftarrow \mathbb{G}$. Write $h = g^w$. Then, $\forall i \in [n+1]$, sample $s_i, t_i \leftarrow_R \mathbb{Z}_p$, s.t. $s_{n+1} + wt_{n+1} \neq 0 \mod p$ Compute $h_i = g^{s_i} h^{t_i}, i \in [n+1]$. It outputs $(\mathrm{msk} := \{(s_i, t_i)\}_{i=1}^{n+1}), \mathrm{mpk} := \left(\mathbb{G}, g, h, \{h_i\}_{i=1}^{n+1}\right)$.
- $\mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y})$: It generates a key for the vector $\boldsymbol{y}$. Sample $u \leftarrow \mathbb{Z}_p$ and then define $\boldsymbol{y}^* := \boldsymbol{y}||u$. Output $\mathrm{sk}_{\boldsymbol{y}} := (\mathrm{sk}_{\boldsymbol{y}}(1) = \langle \boldsymbol{s}, \boldsymbol{y}^* \rangle, \mathrm{sk}_{\boldsymbol{y}}(2) = \langle \boldsymbol{t}, \boldsymbol{y}^* \rangle, u)$.
- $\mathrm{Encap}(\boldsymbol{z})$: The input vector $\boldsymbol{z}$ has length $n+1$. It samples $r \leftarrow \mathbb{Z}_p$ and $\boldsymbol{x} \leftarrow \mathbb{Z}_p^n$. Define $\boldsymbol{x}^* := \boldsymbol{x}||0$ with length $n+1$. Let $C = g^r, D = h^r, E_i = g^{\frac{x_i^*}{z_i}} h_i^{\frac{r}{z_i}}, \forall i \in [n+1]$. Output $\mathrm{ct}_{\boldsymbol{z}} := (C, D, \{E_i\}_{i=1}^{n+1}, \boldsymbol{z}), \boldsymbol{k} := \boldsymbol{x}$.
- $\mathrm{Encap}^*(\boldsymbol{z})$: First sample $r, r' \leftarrow \mathbb{Z}_p$ with $r \neq r'$, and $\boldsymbol{x} \leftarrow \mathbb{Z}_p^n$. Define $\boldsymbol{x}^* := \boldsymbol{x}||0$ with length $n+1$. Let $C = g^r, D = h^r, E_i = g^{\frac{x_i^*}{z_i}} h_i^{\frac{r'}{z_i}}, \forall i \in [n+1]$. Output $\mathrm{ct}_{\boldsymbol{z}} := (C, D, \{E_i\}_{i=1}^{n+1}, \boldsymbol{z})$.
- $\mathrm{Decap}(\mathrm{ct}_{\boldsymbol{z}}, \mathrm{sk}_{\boldsymbol{y}}, \boldsymbol{y})$: Calculate $E_{\boldsymbol{y}} := \frac{\prod_{i=1}^{n+1} E_i^{y_i^* z_i}}{C^{\mathrm{sk}_{\boldsymbol{y}}(1)} D^{\mathrm{sk}_{\boldsymbol{y}}(2)}}$ Then output $\log_g(E_{\boldsymbol{y}})$.

Similar with [3], the decryption algorithm requires to compute a discrete logarithm. As the analysis in [3], there are some methods to reduce the cost of this operation. We state the following theorem to study the properties of Π_1, and the proof is shown in Appendix A.

Theorem 3. *Under DDH assumption, the above IP-HPS construction Π_1 satisfies correctness, valid/invalid ciphertext indistinguishability, and* 0*-universality.*

4 Leakage-smoothness of IP-HPS

The next step is to construct an IP-HPS scheme, which is l'-leakage-smooth for arbitrarily large l', and meets the efficiency requirements of the BRM. The l'-leakage-smoothness states that the scheme is still smooth even if the adversary can get some information about secret keys with the output length is less than l' bits. This property offers the chance to make our final IPFE scheme become leakage-resilient for arbitrarily large leakage bound. The efficiency requirements of the BRM states that except the length of secret keys, all other parts of the system should be independent of the leakage bound l'. This requirement ensures that our final IPFE scheme also meets the efficiency requirements of the BRM.

The main idea is: (1) introduce a key-size parameter m, which gives us flexibility in the size of secret key and will depend on the desired leakage bound l'. For each input vector $\boldsymbol{y}$ of the key

generation algorithm, project it into m new vectors in the same vector space, and then generate secret key for each new vector. (2) In order to meet the efficiency requirements, the encapsulation will choose only η indices from $\{1, ..., m\}$, denoted as a vector $\boldsymbol{w}$; and the decapsulation will only use these η secret keys. Here η is a parameter to be determined later and is independent of m. (3) In the proof of leakage-smoothness, we need to use the same random numbers to generate the n secret keys for each vector $\boldsymbol{y}_i$. So the key generation algorithm will take n linear independent vectors as input (denoted as an invertible matrix Y). (4) Since the key generation algorithm will output n secret keys for n vectors, the encapsulation algorithm will also run n times to get n ciphertexts. These n ciphertexts shares the same encapsulated key $\boldsymbol{k}$. The i-th ciphertext can be decapsulated by the i-th secret key.

Before showing our construction, we talk about why a simple extension of leakage amplification of IB-HPS in [5] cannot be applied here:

On one hand, in IB-HPS, the output of the decapsulation algorithm is already the encapsulated key. So, in their definition of leakage-smoothness, it only needs one secret key to compute an encapsulated key. However, in IP-HPS, the output of the decapsulation algorithm is just an inner-product value between the encapsulated key and the vector $\boldsymbol{y}$ in the secret key. So, in order to determine an encapsulated key, we need at least n secret keys for n linear independent vectors, which makes our leakage-smoothness definition and proof become more complicated.

On the other hand, in an IB-HPS, the inputs of KeyGen and Encap (Encap$*$) have the same parameter: *identity*. This brings lots of convenience for decapsulation, since the output of decapsulation algorithm only need to be reasonable when the identity in the ciphertext is the same as the identity in the secret key. While in IP-HPS, there is no relation between the inputs of KeyGen and Encap/Encap$*$, and the outputs of decapsulation algorithm need to be reasonable for **all** possible inputs of vectors.

We start with our IP-HPS scheme $\Pi_1 = (\text{Setup}, \text{KeyGen}_1, \text{Encap}_1, \text{Encap}_1^*, \text{Decap}_1)$, and then construct an IP-HPS scheme $\Pi_2 = (\text{Setup}, \text{KeyGen}_2, \text{Encap}_2, \text{Encap}_2^*, \text{Decap}_2)$ where the number of secret keys associated with one vector(i.e. m) can be arbitrarily large. Then we will obtain the property of l'-leakage-smoothness for arbitrary l' without losing efficiency.

Let $\mathcal{M}$ be a family of $n \times n$ invertable matrices and let $|\mathcal{M}| = m$. Define functions $H_1, H_2 : \mathbb{Z}_p^n \times [m] \to \mathbb{Z}_p^n$: $H_1(\boldsymbol{y}, \alpha) := M_\alpha^T \boldsymbol{y}, \quad H_2(\boldsymbol{y}, \alpha) := M_\alpha^{-1} \boldsymbol{y}$. They are both one-to-one for $\mathbb{Z}_p^n$.

Define $\Pi_2 = (\text{Setup}, \text{KeyGen}_2, \text{Encap}_2, \text{Encap}_2^*, \text{Decap})$ as follows:

- $\text{Setup}(1^\lambda, 1^n)$: The Setup algorithm is the same as that of Π_1.
- $\text{KeyGen}_2(\text{msk}, Y)$: Let $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]$ be invertible. First sample $u[1], \ldots, u[m] \leftarrow \mathbb{Z}_p$. For all $\alpha \in [m], i \in [n]$, let $\text{sk}_{\boldsymbol{y}_i}[\alpha] := \Big(\text{sk}_{\boldsymbol{y}_i}[\alpha](1) = \langle \boldsymbol{s}, \boldsymbol{y}_i^*[\alpha] \rangle, \text{sk}_{\boldsymbol{y}_i}[\alpha](2) = \langle \boldsymbol{t}, \boldsymbol{y}_i^*[\alpha] \rangle, u[\alpha], i\Big)$. Here we set $\boldsymbol{y}_i^*[\alpha] := H_1(\boldsymbol{y}_i, \alpha) || u[\alpha]$. Let $\text{sk}_{\boldsymbol{y}} := (\text{sk}_{\boldsymbol{y}}[1], \ldots, \text{sk}_{\boldsymbol{y}}[m])$ and then output $\text{sk}_Y := (\text{sk}_{\boldsymbol{y}_1}, \ldots, \text{sk}_{\boldsymbol{y}_n})$.
- $\text{Encap}_2(\boldsymbol{z})$: $\boldsymbol{z}$ is a vector in $\mathbb{Z}_p^{n+1}$. First sample a vector $\boldsymbol{k} \in \mathbb{Z}_p^n$. This algorithm will run the following steps for n times. In step i:
  (1)sample $\boldsymbol{w}_i \leftarrow [m]^\eta$ and $\boldsymbol{\theta}_i \leftarrow \mathbb{Z}_p^\eta$.
  (2)For each $\alpha \in [\eta]$, sample $\boldsymbol{k}_i[\alpha] \leftarrow \mathbb{Z}_p^n$ s.t. $\sum_{\alpha=1}^{\eta} \theta_i[\alpha] \boldsymbol{k}_i[\alpha] = \boldsymbol{k}$; and $r_i[\alpha] \leftarrow \mathbb{Z}_p$.
  (3)Let $\boldsymbol{k}^*[\alpha] := H_2(\boldsymbol{k}[\alpha], w_i[\alpha]) || 0$ with length $n+1$. Let $C_i[\alpha] = g^{r_i[\alpha]}$, $D_i[\alpha] = h^{r_i[\alpha]}$ and $E_{ij}[\alpha] = g^{\frac{k_j^*[\alpha]}{z_j}} h_j^{\frac{r_i[\alpha]}{z_j}}$ (Recall that h_j is from mpk and $h_j = g^{s_j} h^{t_j}$).

(4)Set $\mathrm{ct}_{\boldsymbol{z}}[\alpha][i] = \left(C_i[\alpha], D_i[\alpha], \{E_{ij}[\alpha]\}_{j=1}^{n+1}\right)$.

Then Encap_2 outputs $\mathrm{ct}_{\boldsymbol{z}} = \left((\mathrm{ct}_{\boldsymbol{z}}[\alpha][i])_{\alpha\in[\eta],i\in[n]}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_n, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n, \boldsymbol{z}, \boldsymbol{k}\right)$.

- $\mathrm{Encap}_2^*(\boldsymbol{z})$: $\boldsymbol{z}$ is a vector in $\mathbb{Z}_p^{n+1}$. First sample a vector $k \in \mathbb{Z}_p^n$. This algorithm will run the following steps for n times. In step i:
  (1)sample $\boldsymbol{w}_i \leftarrow [m]^\eta$ and $\boldsymbol{\theta}_i \leftarrow \mathbb{Z}_p^\eta$.
  (2)For each $\alpha \in [\eta]$, sample $\boldsymbol{k}_i[\alpha] \leftarrow \mathbb{Z}_p^n$ s.t. $\sum_{\alpha=1}^{\eta} \theta_i[\alpha]\boldsymbol{k}_i[\alpha] = \boldsymbol{k}$; and $r_i[\alpha], r_i'[\alpha] \leftarrow \mathbb{Z}_p$ with $r_i[\alpha] \neq r_i'[\alpha]$.
  (3)Let $\boldsymbol{k}^*[\alpha] := H_2(\boldsymbol{k}[\alpha], w_i[\alpha])||0$ with length $n+1$. Let $C_i[\alpha] = g^{r_i[\alpha]}$, $D_i[\alpha] = h^{r_i[\alpha]}$ and $E_{ij}[\alpha] = g^{\frac{k_j^*[\alpha]}{z_j}} h_j^{\frac{r_i'[\alpha]}{z_j}}$ (Recall that h_j is from mpk and $h_j = g^{s_j}h^{t_j}$).
  (4)Set $\mathrm{ct}_{\boldsymbol{z}}[\alpha][i] = \left(C_i[\alpha], D_i[\alpha], \{E_{ij}[\alpha]\}_{j=1}^{n+1}\right)$.

  Then Encap_2^* outputs $\mathrm{ct}_{\boldsymbol{z}} = \left((\mathrm{ct}_{\boldsymbol{z}}[\alpha][i])_{\alpha\in[\eta],i\in[n]}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_n, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n, \boldsymbol{z}\right)$.
- $\mathrm{Decap}_2(\mathrm{ct}_{\boldsymbol{z}}, \mathrm{sk}_{\boldsymbol{y}})$: It outputs the inner product of $\boldsymbol{k}$ and $\boldsymbol{y}$. Parse $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ from $\mathrm{ct}_{\boldsymbol{z}}$ and i from $\mathrm{sk}_{\boldsymbol{y}}$. For each $\alpha \in [\eta]$, obtain $dec[\alpha][i] := \mathrm{Decap}_1(\mathrm{ct}_{\boldsymbol{z}}[\alpha][i], \mathrm{sk}_{\boldsymbol{y}}[w_{i\alpha}])$. Output $\sum_{\alpha=1}^{\eta} \theta_{i\alpha} \times dec[\alpha][i]$.

For the **leakage-smoothness** and **efficiency**, we propose Theorem 4. The proof of it is shown in Appendix B, together with the analysis of **correctness** and **valid/invalid ciphertext indistinguishability**. From Theorem 4, we can conclude that our IP-HPS scheme Π_2 is l'-leakage-smooth for arbitrarily large l', by choosing $m \geq \frac{l'+n\log p+2\lambda}{(1-\varepsilon)\log p}$.

Theorem 4. *For any $\varepsilon > 0$, there exists $\eta = O(\log p)$, s.t. for any polynomial $m(\lambda)$, the above construction of Π_2 from Π_1 is l'-leakage-smooth as long as: $l' \leq (1-\varepsilon)m\log p - n\log p - 2\lambda$.*

5 Leakage resilient inner-product functional encryption

We define the security for an Inner-product functional encryption (IPFE) scheme which is resistant to key leakage attacks in the bounded-retrieval model (BRM) and show how to use an leakage-smooth IP-HPS to construct such an IPFE scheme. Our security notion only allows leakage attacks against the secret keys of the various functions, but not the master secret key. And we only allow the adversary to perform leakage attacks before seeing the challenge ciphertext. As shown in [4, 6, 46], this limitation is inherent to encryption schemes since otherwise the leakage function can simply decrypt the challenge ciphertext and output its first bit.

5.1 Definitions

Indistinguishable security with leakage. We define the *indistinguishable security game*, parametrized by a security parameter λ, a parameter of vector length n and a leakage parameter l, as the following game between an adversary $\mathcal{A}$ and a challenger in Table 3.

A PPT adversary $\mathcal{A}$ is admissible if it makes leakage queries for at most n linear independent vectors in **Query 1**. The *advantage* of an admissible adversary $\mathcal{A}$ in the indistinguishable security game with leakage l is defined by $Adv_{\mathrm{IPFE},\mathcal{A}}^{\text{IPFE-IND}}(\lambda, n, l) := |\mathbf{Pr}[\mathcal{A}\ wins] - \frac{1}{2}|$.

Table 3: IPFE-IND(λ, n, l)

Setup: The challenger computes $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{Setup}(1^\lambda, 1^n)$ and sends mpk to the adversary $\mathcal{A}$. The challenger constructs a list $\mathcal{L}_{sk}$ to store the secret keys which are queried by the adversary, and a vector $\mathcal{R}$ to store the random numbers which are used to generate the secret keys.
Query 1: The adversary $\mathcal{A}$ can adaptively ask the challenger for: *Leakage query*: On input a vector $\boldsymbol{y} \in \mathcal{V}$, a PPT function f^*, if $\mathcal{L}_{sk}$ is empty, the challenger runs $\mathrm{sk}_{(\boldsymbol{y},1)} \leftarrow$ IPFE.KeyGen$(\mathrm{msk}, \boldsymbol{y}, 1)$, then stores the tuple $(r, 1)$ in $\mathcal{R}$, and the tuple $(\boldsymbol{y}, \mathrm{sk}_{(\boldsymbol{y},1)})$ in the list $\mathcal{L}_{sk}$. Else if $\boldsymbol{y}$ is not in the list $\mathcal{L}_{sk}$, then the challenger reads and deletes the tuple (r, τ) from $\mathcal{R}$ and generates $\mathrm{sk}_{(\boldsymbol{y},\tau+1)} \leftarrow$ IPFE.KeyGen$(\mathrm{msk}, \boldsymbol{y}, \tau+1)$ with randomness r. The challenger stores $(r, \tau+1)$ in $\mathcal{R}$ and the tuple $(\boldsymbol{y}, \mathrm{sk}_{(\boldsymbol{y},\tau+1)})$ in the list $\mathcal{L}_{sk}$. Else if $\boldsymbol{y}$ is in the list $\mathcal{L}_{sk}$, then the challenger reads the tuple $(\boldsymbol{y}, \mathrm{sk}_{(\boldsymbol{y},\tau)})$ from it. Then the challenger replies with $f^*(\mathrm{sk}_{(\boldsymbol{y},\tau^*)})$ if $\sum_{f \in \{f'\}_{\boldsymbol{y}} \cup \{f^*\}} |f(\mathrm{sk}_{(\boldsymbol{y},\tau^*)})| \leq l$, where $\{f'\}_{\boldsymbol{y}}$ denotes the set of functions that the adversary have queried with input $\mathrm{sk}_{(\boldsymbol{y},\tau^*)}$, and $|f(\mathrm{sk}_{(\boldsymbol{y},\tau^*)})|$ is the bit-length of the function value $f(\mathrm{sk}_{(\boldsymbol{y},\tau^*)})$.
Challenge: The adversary $\mathcal{A}$ chooses two vectors $\boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathcal{V}$ The challenger chooses $b \leftarrow \{0,1\}$ uniformly at random and computes $\mathrm{ct}_b \leftarrow \mathrm{Encrypt}(\boldsymbol{x}_b)$ and gives ct_b to the adversary $\mathcal{A}$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ and wins if $b' = b$.

Now we give some explanation about the definition. All restrictions of the definiiton come from the definitions and proofs of properties of IP-HPS Π_2. Recall that there are only 3 items in the definition of leakage-smoothness: $(\mathrm{ct}, f(\{\mathrm{sk}_{\boldsymbol{y}_i}\}_{i=1}^n), \boldsymbol{k})$. The secret keys $\{\mathrm{sk}_{\boldsymbol{y}_i}\}_{i=1}^n$ used to compute the encapsulated key $\boldsymbol{k}$ do not appear in the equation directly. In order to use leakage-smoothness of Π_2 to prove the security of leakage-resilient IPFE scheme, for the secret keys used to compute $\boldsymbol{k}$, any adversary can only know a function value $f(\cdot)$, instead of the secret keys. And in the security proof, all secret keys generated in **Query 1** will be used to compute $\boldsymbol{k}$ Thus, we allow the adversary to make *leakage queries* on arbitrary vector $\boldsymbol{y}$, rather than making *secret key queries* on vectors $\boldsymbol{y}$ subject to the condition that $\langle \boldsymbol{x_0}, \boldsymbol{y} \rangle = \langle \boldsymbol{x_1}, \boldsymbol{y} \rangle$.

In the valid/invalid ciphertext indistinguishability definition of leakage-smooth IP-HPS Π_2, the adversary is allowed to make secret key query once for a $n \times n$ invertible matrix $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n]$ and get $\mathrm{sk}_Y = \{\mathrm{sk}_{\boldsymbol{y}_1}, ..., \mathrm{sk}_{\boldsymbol{y}_n}\} \leftarrow \Pi_2.\mathrm{KeyGen}(\mathrm{msk}, Y)$. In order to rely the security of leakage-resilient IPFE on the valid/invalid indistinguishability of Π_2, we have to require that there are at most n different linear independent vectors appearing in the *leakage query*. And such n secret keys should be generated from the same random numbers, and are corresponding to the 1-th,...,n-th parts of ciphertext respectively. In the definition, we use a parameter τ to indicate that $\mathrm{sk}_{(\boldsymbol{y},\tau)}$ is corresponding to τ-th part of the ciphertext. ($\mathrm{sk}_{(\boldsymbol{y},\tau)}$ generated by IPFE.KeyGen$(\mathrm{msk}, \boldsymbol{y}, \tau)$ can decrypt the τ-part of the ciphertext.)

Definition 6 (leakage-resilient IPFE). *An IPFE scheme is l-leakage-resilient, if (1) it satisfies the* correctness*, and (2) the advantage of any admissible PPT adversary $\mathcal{A}$ in the indistinguishable security game with leakage l is* $\mathrm{negl}(\lambda)$*. We define the leakage ratio of the scheme to be $\mu = \frac{l}{\hat{\beta}}$, where $\hat{\beta}$ is the number of bits needed to efficiently store secret key* $\mathrm{sk}_{\boldsymbol{y}}$*.*

Definition 7 (leakage-resilient IPFE in the BRM). *An IPFE scheme is adaptively leakage-resilient in the bounded retrieval model (BRM), if the scheme is adaptively leakage-resilient, and the master public key size, master secret key size, ciphertext size, encryption time, and decryption time (and the number of secret-key bits read by decryption) are independent of the leakage-bound*

l. More formally, there exist polynomials $\mathrm{mpksize}, \mathrm{msksize}, \mathrm{ctsize}, \mathrm{encTime}, \mathrm{decTime}$*, such that for any polynomial l and any* $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{KeyGen}(1^\lambda, 1^n, 1^l), \boldsymbol{x} \in \mathcal{V}, \mathrm{ct}_{\boldsymbol{x}} \leftarrow \mathrm{Encrypt}(\mathrm{mpk}, \boldsymbol{x})$*:*

- *Master public key size is* $|\mathrm{mpk}| \leq O(\mathrm{mpksize}(\lambda))$*, master secret key size is* $|\mathrm{msk}| \leq O(\mathrm{msksize}(\lambda))$*, and ciphertext size is* $|\mathrm{ct}_{\boldsymbol{x}}| \leq O(\mathrm{ctsize}(\lambda, |\boldsymbol{x}|))$.
- *Run-time of* $\mathrm{Encrypt}(\mathrm{mpk}, \boldsymbol{x})$ *is* $\leq O(\mathrm{encTime}(\lambda, |\boldsymbol{x}|))$.
- *Run-time of* $\mathrm{Decrypt}(\mathrm{sk}_{\boldsymbol{y}}, \boldsymbol{x})$*, and the number of bits of* $\mathrm{sk}_{\boldsymbol{y}}$ *accessed, are* $\leq O(\mathrm{encTime}(\lambda, |\boldsymbol{x}|))$.

5.2 Construction of Leakage-resilient IPFE

The construction of leakage-resilient IPFE from a leakage-smooth IP-HPS is very simple. Given an *l-leakage-smooth* IP-HPS scheme $\Pi = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encap}, \mathrm{Encap}^*, \mathrm{Decap})$ where the encapsulated key space is $\mathcal{K}$ and the message space is $\mathcal{M}$, we construct an IPFE scheme with the same vector space $\mathcal{V} = \mathcal{K}$. We show our construction in Table 4.

Recall that in our leakage-smooth IP-HPS scheme Π_2, the encapsulation algorithm will output n ciphertexts sharing the same encapsulated key $\boldsymbol{k}$, and the i-th ciphertext can be decapsulated by the i-th secret key. So in our leakage-resilient IPFE scheme, we will choose an index $\tau \in [n]$ in key generation algorithm to indicate which ciphertext it wants to decrypt with this secret key.

Table 4: The construction from an *l-leakage-smooth* IP-HPS scheme Π_2 to an IPFE scheme.

$\mathrm{Setup}(1^\lambda, 1^n)$: The Setup procedure is the same as Π_2.Setup.
$\mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y}, \tau)$: It chooses $n-1$ random vectors $\boldsymbol{y}_1, ..., \boldsymbol{y}_{\tau-1}, \boldsymbol{y}_{\tau+1}, \boldsymbol{y}_n$, such that $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_\tau = \boldsymbol{y}, ..., \boldsymbol{y}_n]$ is a $n \times n$ invertible matrix. It gets $(\mathrm{sk}_{\boldsymbol{y}_1}, \mathrm{sk}_{\boldsymbol{y}_2}, ..., \mathrm{sk}_{\boldsymbol{y}_n}) \leftarrow \Pi_2.\mathrm{KeyGen}(\mathrm{msk}, Y)$, and returns $\mathrm{sk}_{(\boldsymbol{y},\tau)} = \mathrm{sk}_{\boldsymbol{y}_\tau}$.
$\mathrm{Encrypt}(\boldsymbol{x})$: It chooses a random $\boldsymbol{z} \in \mathcal{M}$ and computes $(ct_{\boldsymbol{z}}, \boldsymbol{k}) \leftarrow \Pi_2.\mathrm{Encap}(\boldsymbol{z})$. It sets $c_1 = ct_{\boldsymbol{z}}, c_2 = \boldsymbol{k} + \boldsymbol{x}$. Output $\mathrm{ct}_{\boldsymbol{x}} = (c_1, c_2)$.
$\mathrm{Decrypt}(\mathrm{ct}_{\boldsymbol{x}}, \mathrm{sk}_{(\boldsymbol{y},\tau)})$: Parse $\mathrm{ct}_{\boldsymbol{x}} = (c_1, c_2)$ and output $\boldsymbol{y} \cdot c_2 - \Pi_2.\mathrm{Decap}(c_1, \mathrm{sk}_{(\boldsymbol{y},\tau)})$

Theorem 5. *Assume that we start with an l'-leakage-smooth IP-HPS Π_2, and for the challenge ciphertext* $\mathrm{ct}_b = (c_1, c_2)$ *and any* $\mathrm{sk}_{\boldsymbol{y}}$*, the adversary can only do* $\Pi_2.\mathrm{Decap}(c_1, \mathrm{sk}_{\boldsymbol{y}})$ *in a black-box way. Then the construction in Table 4 yields an $l = \frac{l'}{n}$-leakage-resilient IPFE.*

Here, the restriction on the computations of Π_2.Decap comes from the valid/invalid ciphertext indistinguishability analysis of Π_2. We use a series of games argument in our security proof, wihch begins with the real security game and ends with a game whose challenge ciphertext is independent of the bit b chosen by the challenger.

The formal proof of Theorem 5 can be found in Appendix C.

Theorem 6. *Using the l'-leakage-smooth IP-HPS construction Π_2 in Section 4, we can get an l-leakage-resilient IPFE scheme in the BRM with message space $\mathcal{V} = \mathbb{Z}_p^n$ and :*
(1) Master public-key size, master secret-key size, ciphertext-size and the number of secret-key bits read by decryption are the same as Π_2, and are independent of l.
(2) Encryption time consists of the Encap *time of Π_2 and the time of one vector addtion operation with length n. Decryption time consists of the* Decap *time of Π_2, the time of inner-product operation with vector length n, and a subtraction. Both the encryption time and decryption time are independent of l.*
(3) The leakage ratio is $\mu = \frac{1-\varepsilon}{3n}$, for sufficiently large values of the leakage-parameter l.

Proof. The first two statements are directly proved by the construction of l'-leakage-resilient IPFE scheme from a l'-leakage smooth IP-HPS. For the leakage ratio, by Theorem 4, we have $l = \frac{l'}{n} \leq \frac{(1-\varepsilon)m\log p - n\log p - 2\lambda}{n}$. We can write $m(l)$ is a function of l, and choose $m(l) \geq \frac{l'+n\log p+2\lambda}{(1-\varepsilon)\log p}$ is sufficient. Then the leakage ratio for a given l is defined as:

$$\mu = \frac{l}{3m(l)\log p} = \frac{(1-\varepsilon)l}{3nl + 3n\log p + 6\lambda} \ .$$

For sufficiently large l, the ratio is approximately $\frac{1-\varepsilon}{3n}$.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: IACR International Workshop on Public Key Cryptography. pp. 733–751. Springer (2015)
2. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 601–626. Springer (2017)
3. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Annual Cryptology Conference. pp. 333–362. Springer (2016)
4. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Theory of cryptography conference. pp. 474–495. Springer (2009)
5. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 113–134. Springer (2010)
6. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Annual International Cryptology Conference. pp. 36–54. Springer (2009)
7. Ananth, P., Sahai, A.: Functional encryption for turing machines. In: Theory of Cryptography Conference. pp. 125–153. Springer (2016)
8. Apon, D., Döttling, N., Garg, S., Mukherjee, P.: Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. In: LIPIcs-Leibniz International Proceedings in Informatics. vol. 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
9. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Annual International Cryptology Conference. pp. 67–98. Springer (2017)
10. Benhamouda, F., Bourse, F., Lipmaa, H.: Cca-secure inner-product functional encryption from projective hash functions. In: IACR International Workshop on Public Key Cryptography. pp. 36–66. Springer (2017)
11. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Annual international cryptology conference. pp. 213–229. Springer (2001)
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011)
13. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 89–108. Springer (2011)
14. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability. In: Annual Cryptology Conference. pp. 1–20. Springer (2010)
15. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 501–510. IEEE (2010)

16. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous ibe, leakage resilience and circular security from new assumptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 535–564. Springer (2018)
17. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Theory of Cryptography Conference. pp. 479–498. Springer (2007)
18. Chen, Y., Gentry, C., Halevi, S.: Cryptanalyses of candidate branching program obfuscators. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 278–307. Springer (2017)
19. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–12. Springer (2015)
20. Chow, S.S., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 152–161. ACM (2010)
21. Coron, J.S., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of ggh15 multilinear maps. In: Annual Cryptology Conference. pp. 607–628. Springer (2016)
22. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 45–64 (2002)
23. Di Crescenzo, G., Lipton, R., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Theory of Cryptography Conference. pp. 225–244. Springer (2006)
24. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Theory of Cryptography Conference. pp. 361–381. Springer (2010)
25. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 511–520. IEEE (2010)
26. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 613–631. Springer (2010)
27. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 621–630. ACM (2009)
28. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM journal on computing **38**(1), 97–139 (2008)
29. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Theory of Cryptography Conference. pp. 207–224. Springer (2006)
30. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07). pp. 227–237. IEEE (2007)
31. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 2008 49th Annual IEEE Symposium on Foundations of Computer Science. pp. 293–302. IEEE (2008)
32. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Theory of Cryptography Conference. pp. 343–360. Springer (2010)
33. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)
34. Gentry, C., Halevi, S.: Hierarchical identity based encryption with polynomially many levels. In: Theory of Cryptography Conference. pp. 437–456. Springer (2009)
35. 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. ACM (2013)
36. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Annual Cryptology Conference. pp. 162–179. Springer (2012)

37. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 89–98. Acm (2006)
38. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Communications of the ACM **52**(5), 91–98 (2009)
39. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 466–481. Springer (2002)
40. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 703–720. Springer (2009)
41. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Side channel cryptanalysis of product ciphers. In: European Symposium on Research in Computer Security. pp. 97–110. Springer (1998)
42. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Annual International Cryptology Conference. pp. 388–397. Springer (1999)
43. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Annual International Cryptology Conference. pp. 104–113. Springer (1996)
44. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Theory of Cryptography Conference. pp. 70–88. Springer (2011)
45. Micali, S., Reyzin, L.: Physically observable cryptography. In: Theory of Cryptography Conference. pp. 278–296. Springer (2004)
46. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM Journal on Computing **41**(4), 772–814 (2012)
47. Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences **52**(1), 43–52 (1996)
48. Nishimaki, R., Yamakawa, T.: Leakage-resilient identity-based encryption in bounded retrieval model with nearly optimal leakage-ratio. In: IACR International Workshop on Public Key Cryptography. pp. 466–495. Springer (2019)
49. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive **2010**, 556 (2010)
50. Pietrzak, K.: A leakage-resilient mode of operation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 462–482. Springer (2009)
51. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 457–473. Springer (2005)
52. Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: International Colloquium on Automata, Languages, and Programming. pp. 560–578. Springer (2008)
53. Wang, H., Chen, K., Liu, J.K., Hu, Z.: Leakage-resilient chosen-ciphertext secure functional encryption from garbled circuits. In: International Conference on Information Security Practice and Experience. pp. 119–140. Springer (2018)
54. Waters, B.: Efficient identity-based encryption without random oracles. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 114–127. Springer (2005)
55. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: International Workshop on Public Key Cryptography. pp. 53–70. Springer (2011)
56. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Annual Cryptology Conference. pp. 678–697. Springer (2015)
57. Yu, Z., Au, M.H., Xu, Q., Yang, R., Han, J.: Leakage-resilient functional encryption via pair encodings. In: Australasian Conference on Information Security and Privacy. pp. 443–460. Springer (2016)
58. Yuen, T.H., Chow, S.S., Zhang, Y., Yiu, S.M.: Identity-based encryption resilient to continual auxiliary leakage. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 117–134. Springer (2012)

59. Zhang, J., Chen, J., Gong, J., Ge, A., Ma, C.: Leakage-resilient attribute based encryption in prime-order groups via predicate encodings. Designs, Codes and Cryptography **86**(6), 1339–1366 (2018)
60. Zhang, L., Chen, Y., Zhang, J., He, M., Yiu, S.M.: From quadratic functions to polynomials: Generic functional encryption from standard assumptions. In: International Conference on Codes, Cryptology, and Information Security. pp. 142–167. Springer (2019)

A Proof of Theorem 3

Correctness and Valid/Invalid Ciphertext Indistinguishiability. For any $\boldsymbol{z}, \boldsymbol{y}$ with length $n+1$ and n respectively, and for any correctly generated $\mathrm{mpk}, \mathrm{msk}, \mathrm{sk}_{\boldsymbol{y}}$ from the above algorithms, if $(C, D, \{E_i\}_{i=1}^{n+1}, \boldsymbol{z})$ is generated by $\mathrm{Encap}(\boldsymbol{z})$, then correctness is proved by calculating $\log_g(E_{\boldsymbol{y}})$:

$$E_{\boldsymbol{y}} = \frac{\prod_{i=1}^{n+1} E_i^{y_i^* z_i}}{C^{\mathrm{sk}_{\boldsymbol{y}}(1)} D^{\mathrm{sk}_{\boldsymbol{y}}(2)}} = \frac{\prod_{i=1}^{n+1} g^{x_i^* y_i^*} g^{r s_i y_i^*} h^{r t_i y_i^*}}{g^{r\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle} h^{r\langle \boldsymbol{t}, \boldsymbol{y}^* \rangle}} = \prod_{i=1}^{n+1} g^{x_i^* y_i^*} = \prod_{i=1}^{n} g^{x_i y_i} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \ .$$

For the valid/invalid ciphertext indistinguishability, we show how to use an adversary $\mathcal{A}$, which can distinguish valid and invalid cipihertexts, to construct an adversary $\mathcal{B}$, which can distinguish whether $c = ab$ or c is randomly chosen from $\mathbb{Z}_p$. $\mathcal{B}$ receives a DDH tuple (g, g^a, g^b, g^c), then it sets $C = g^a$, $h_i = g^b$ and $E_i = g^{\frac{x_i^*}{z_i}} g^{\frac{c}{z_i}}$, where i is randomly chosen from $[n]$, and sends mpk and the challenge ciphertext to $\mathcal{A}$. If $\mathcal{A}$ outputs it is a valid ciphertext, then $\mathcal{B}$ outputs $c = ab$. Otherwise, $\mathcal{B}$ outputs that c is randomly chosen from $\mathbb{Z}_p$.

0-Universality of Π_1. We show that the decapsulation function of Π_1 is a 0-universal hash family. Fix any $(\mathrm{mpk}, \mathrm{msk})$ produced by $\mathrm{Setup}(1^\lambda, 1^n)$, a set of linear independent vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ and $\boldsymbol{z}$, let $\mathrm{ct} = (C, D, \{E_i\}_{i=1}^{n+1}, \boldsymbol{z}) \leftarrow \mathrm{Encap}^*(\boldsymbol{z})$. From our construction of Encap^* we have $C = g^r, D = h^r, E_i = g^{\frac{x_i^*}{z_i}} h_i^{\frac{r'}{z_i}}$, where r, r' are uniformly sampled from $\mathbb{Z}_p$ with $r \neq r'$. Then, for any secret key $\mathrm{sk}_{\boldsymbol{y}} = (\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle, \langle \boldsymbol{t}, \boldsymbol{y}^* \rangle, u)$, it's a random variable generated from $\mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y})$ with $\boldsymbol{y} \in \{\boldsymbol{y}_i\}_{i=1}^n$. Then we can obtain (Assume $h = g^w$):

$$\begin{aligned}
\mathrm{Decap}(\mathrm{ct}, \mathrm{sk}_{\boldsymbol{y}}) &= \log_g \left(\frac{\prod_{i=1}^{n+1} E_i^{y_i^* z_i}}{C^{\mathrm{sk}_{\boldsymbol{y}}(1)} D^{\mathrm{sk}_{\boldsymbol{y}}(2)}} \right) = \log_g \left(\frac{\prod_{i=1}^{n+1} g^{x_i^* y_i^*} g^{r' s_i y_i^*} h^{r' t_i y_i^*}}{g^{r\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle} h^{r\langle \boldsymbol{t}, \boldsymbol{y}^* \rangle}} \right) \\
&= \log_g \left(\frac{g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} g^{r'\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle} h^{r'\langle \boldsymbol{t}, \boldsymbol{y}^* \rangle}}{g^{\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle} h^{\langle \boldsymbol{t}, \boldsymbol{y}^* \rangle}} \right) = \log_g \left(g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} g^{(r'-r)\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle} h^{(r'-r)\langle \boldsymbol{t}, \boldsymbol{y}^* \rangle} \right) \qquad (1) \\
&= \log_g \left(g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle + (r'-r)(\langle \boldsymbol{s}, \boldsymbol{y}^* \rangle + w\langle \boldsymbol{t}, \boldsymbol{y}^* \rangle)} \right) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle + (r'-r)\langle \boldsymbol{s} + w\boldsymbol{t}, \boldsymbol{y} || u \rangle
\end{aligned}$$

Note that if $\mathrm{sk}_{\boldsymbol{y}}$ is fixed, the randomness of $\mathrm{Decap}(\mathrm{ct}, \mathrm{sk}_{\boldsymbol{y}})$ is only from ct, i.e. from $r' - r$, which is uniformly random over $\mathbb{Z}_p \setminus \{0\}$. Further, we can define a hash function family $\mathcal{H} = \{\mathrm{Decap}(\mathrm{ct}, \cdot) | \mathrm{ct} \leftarrow \mathrm{Encap}^*(\boldsymbol{z})\}$. To obtain universality of $\mathcal{H}$, we need to show that given $\mathrm{msk}, \mathrm{mpk}$ and $\boldsymbol{y}$, for any fixed $\mathrm{sk}_{\boldsymbol{y}}, \mathrm{sk}'_{\boldsymbol{y}}$ both generated from $\mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y})$, with $\mathrm{sk}_{\boldsymbol{y}} \neq \mathrm{sk}'_{\boldsymbol{y}}$, the following probability is tiny: $\mathbf{Pr}_{\mathrm{ct} \leftarrow \mathrm{Encap}^*(\boldsymbol{z})}[\mathrm{Decap}(\mathrm{ct}, \mathrm{sk}_{\boldsymbol{y}}) = \mathrm{Decap}(\mathrm{ct}, \mathrm{sk}'_{\boldsymbol{y}})]$.

In fact we can prove that this probability is 0. Let u' be the associated u in $\mathrm{sk}'_{\boldsymbol{y}}$. Note that by our construction of KeyGen, $\mathrm{sk}_{\boldsymbol{y}} \neq \mathrm{sk}'_{\boldsymbol{y}}$ implies $u \neq u'$. By our construction of Setup, the $(n+1)$-th entry of $\boldsymbol{s} + w\boldsymbol{t} \neq 0$. Then, for any $\mathrm{sk}_{\boldsymbol{y}} \neq \mathrm{sk}'_{\boldsymbol{y}}$, $\langle \boldsymbol{s} + w\boldsymbol{t}, \boldsymbol{y} || u \rangle \neq \langle \boldsymbol{s} + w\boldsymbol{t}, \boldsymbol{y} || u' \rangle$. By $r \neq r'$, we know that $\mathbf{Pr}_{\mathrm{ct} \leftarrow \mathrm{Encap}^*(\boldsymbol{z})}[\mathrm{Decap}(\mathrm{ct}, \mathrm{sk}_{\boldsymbol{y}}) = \mathrm{Decap}(\mathrm{ct}, \mathrm{sk}'_{\boldsymbol{y}})] = 0$. We conclude that $\mathcal{H}$ is a 0-universal hash family.

B Analysis and Proofs of Leakage-smooth IP-HPS Π_2

B.1 Correctness and Valid/Invalid Ciphertext Indistinguishability of Π_2

The correctness is as follows. In each $dec[\alpha][i] = \mathrm{Decap}_1(\mathrm{ct}_{\boldsymbol{z}}[\alpha][i], \mathrm{sk}_{\boldsymbol{y}}[w_{i\alpha}])$ where the $\mathrm{ct}_{\boldsymbol{z}}$ is valid, we have (for simplicity we omit index i):

$$E_{\boldsymbol{y}} := \frac{\prod_{j=1}^{n+1} E_j^{y_j^*[w_\alpha] z_j}}{C^{\mathrm{sk}_{\boldsymbol{y}}[w_\alpha](1)} D^{\mathrm{sk}_{\boldsymbol{y}}[w_\alpha](2)}} = \frac{\prod_{j=1}^{n+1} g^{k_j^*[\alpha] y_j^*[w_\alpha]} g^{r[\alpha] s_j y_j^*[w_\alpha]} h^{r[\alpha] t_j y_j^*[w_\alpha]}}{g^{r[\alpha]\langle \boldsymbol{s}, \boldsymbol{y}^*[w_\alpha]\rangle} h^{r[\alpha]\langle \boldsymbol{t}, \boldsymbol{y}^*[w_\alpha]\rangle}} = \prod_{j=1}^{n+1} g^{k_j^*[\alpha] y_j^*[w_\alpha]} = g^{\langle \boldsymbol{k}^*[\alpha], \boldsymbol{y}^*[w_\alpha]\rangle} ,$$

$$\log_g(E_{\boldsymbol{y}}) = \langle \boldsymbol{k}^*[\alpha], \boldsymbol{y}^*[w_\alpha]\rangle = \langle H_2(\boldsymbol{k}, w_\alpha)||0, H_1(\boldsymbol{y}, w_\alpha)||u[w_\alpha]\rangle = \langle M_{w_\alpha}^{-1}\boldsymbol{k}[\alpha], M_{w_\alpha}^T \boldsymbol{y}\rangle = \langle \boldsymbol{k}[\alpha], \boldsymbol{y}\rangle ,$$

which is $\langle \boldsymbol{k}_i[\alpha], \boldsymbol{y}\rangle$. Therefore, $\sum_{\alpha=1}^{\eta} \theta_\alpha \times dec[\alpha][i] = \langle \sum_{\alpha=1}^{\eta} \theta_{i\alpha} \boldsymbol{k}_i[\alpha], \boldsymbol{y}\rangle = \langle \boldsymbol{k}, \boldsymbol{y}\rangle$.

When talk about the valid/invalid ciphertext indistinguishability of Π_2, we edit the definition of admissible adversary in Definition 3. The input of Π_2.KeyGen is a $n \times n$ matrix instead of a vector with length n. So here we allow the adversary to make key query for one $n \times n$ invertible matrix, instead of at most n linear independent vectors. And for the challenge ciphertext ct_b and any secret key $\mathrm{sk}_{\boldsymbol{y}_i}$, the adversary can only do $\Pi_2.\mathrm{Decap}(\mathrm{ct}_b, \mathrm{sk}_{\boldsymbol{y}_i})$ in a black-box way. Thus, the valid/invalid ciphertext indistinguishability can be easily extended from the valid/invalid ciphertext indistinguishability of Π_1. This modified indistinguishability is enough for our security proof of leakage-resilient IPFE scheme in Section 5.

B.2 Proof of Theorem 4

Notations. Let Σ be some alphabet. [28] defined a generalized min-entropy called *average conditional min-entropy*, where Z is another random variable:

$$\tilde{\mathbf{H}}_\infty(X|Z) := -\log\left(\mathbf{E}_{z\leftarrow Z}\left[\max_x \mathbf{Pr}\left[X = x|Z = z\right]\right]\right) = -\log\left(\mathbf{E}_{z\leftarrow Z}\left[2^{-\mathbf{H}_\infty(X|Z=z)}\right]\right) .$$

Definition 8 (Approximately Universal Hashing [6]). *A function family $\mathcal{H}$, consisting of functions $h : \Sigma^m \to \Gamma$, is called* (δ, τ)-approximately universal *if for all $x, x' \in \Sigma^m$ with $d_H(x, x') \le \delta m$ we have $\mathbf{Pr}_{h\leftarrow\mathcal{H}}[h(x) = h(x')] \le \tau$, where $d_H(\cdot,\cdot)$ is the Hamming metric.*

Theorem 7 (Approximate Leftover-hash Lemma [6]). *Assume that $\mathcal{H}$ is (δ, τ)-approximately universal. Let $q = |\Sigma|, v = \log|\Gamma|$. Let $\delta \in [\frac{1}{m}, 1 - \frac{1}{q}]$. Let X, Z be arbitrary random variables where X is distributed over Σ^m and let $\beta' := \tilde{\mathbf{H}}_\infty(X|Z)$. Let h be uniformly random over $\mathcal{H}$. Then $\mathbf{SD}\Big((h, Z, h(X)), (h, Z, U_\Gamma)\Big) \le \frac{1}{2}\sqrt{2^{H_q(\delta)m\log(q)+v-\beta'} + \tau 2^v - 1}$, where H_q is q-ary Shannon entropy function.* [3] *In particular, the statistical distance above is at most ϵ as long as $\beta' \ge H_q(\delta)m\log q + v + 2\log\frac{1}{\epsilon} - 1$, and $\tau \le \frac{1}{2^v}(1 + \epsilon^2)$.*

[3] The definition of q-ary Shannon entropy function is $H_q(x) := x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x)$ is the q-ary Shannon entropy function.

Now, we move to prove the l'-leakage-smoothness of Π_2. First fix an invertible Y. For simplicity, we define $x = \mathrm{sk}_Y$, where $x_i = \mathrm{sk}_{\boldsymbol{y}_i} = (\mathrm{sk}_{\boldsymbol{y}_i}[1], ..., \mathrm{sk}_{\boldsymbol{y}_i}[m])$ is a sample of secret key for $\boldsymbol{y_i}$ in Π_2. Then $\boldsymbol{k}^T(c, x) = [\mathrm{Decap}_2(c, x_1), \ldots, \mathrm{Decap}_2(c, x_n)]Y^{-1}$, where $c \leftarrow \mathrm{Encap}^*(\boldsymbol{z})$.

Then, $\mathrm{Decap}_2(c, x_i) = \sum_{\alpha=1}^{\eta} \theta_{i\alpha}\mathrm{Decap}_1(c[\alpha][i], x_i[w_{i\alpha}])$. So we can set $g_{\boldsymbol{\theta}} : \mathbb{Z}_p^\eta \to \mathbb{Z}_p$, $g_{\boldsymbol{\theta}}(\boldsymbol{d}) = \langle \boldsymbol{\theta}, \boldsymbol{d} \rangle$. The family $\mathcal{G} := \{g_\theta | \theta \leftarrow \mathbb{Z}_p^\eta\}$, and it's $\frac{1}{p}$-universal. In Section 3 we already show that the family of $f_{c[\alpha][i]}(\cdot)$ is 0-universal. Now we write $k^T(c, x)$ as a hash function:

$$h_c(x) = (g_{\boldsymbol{\theta}_1}(f_{c[1][1]}(x_1[w_{11}]), ..., f_{c[\eta][1]}(x_1[w_{1\eta}])), ..., g_{\boldsymbol{\theta}_n}(f_{c[1][n]}(x_n[w_{n1}]), ..., f_{c[\eta][n]}(x_n[w_{n\eta}])))Y^{-1}$$

Let $\mathcal{H} := \{h_c() | c \leftarrow \mathrm{Encap}_2^*(\boldsymbol{z})\}$. Note that it's equivalently a family of $\mathbb{Z}_p^m \to \mathbb{Z}_p^n$, for any fixed invertible Y. This is because the random variable x given Y is determined by vector $\boldsymbol{u} \in \mathbb{Z}_p^m$. Firstly, we show that the family $\mathcal{H}$ is approximately universal in the following lemma.

Lemma 1. *Let function families $\mathcal{F}$ be ρ-universal and $\mathcal{G}$ be ρ'-universal, then the above family $\mathcal{H}$ is (δ, τ)-approximately universal for any $\delta > 0$ and $\tau \leq ((1-\delta)^\eta + \rho')^n$.*

Proof. For any $x, x' \in \mathbb{Z}_p^n$, where $d_H(x, x') \geq \delta m$, we calculate $\mathbf{Pr}_{h_c \leftarrow \mathcal{H}}[h_c(x) = h_c(x')]$:

$$\begin{aligned}
&= \prod_{i=1}^{n} \mathbf{Pr}_{h_c \leftarrow \mathcal{H}}\left[g_{\boldsymbol{\theta}_i}\left(f_{c[1][i]}(x_i[w_{i1}]), ..., f_{c[\eta][i]}(x_i[w_{i\eta}])\right) = g_{\boldsymbol{\theta}_i}\left(f_{c[1][i]}(x_i'[w_{i1}]), ..., f_{c[\eta][i]}(x_i'[w_{i\eta}])\right)\right] \\
&\leq \prod_{i=1}^{n}\left(\mathbf{Pr}\left[\left(f_{c[1][i]}(x_1[w_{i1}]), ..., f_{c[\eta][i]}(x_1[w_{i\eta}])\right) = \left(f_{c[1][i]}(x_1[w_{i1}]), ..., f_{c[\eta][i]}(x_1[w_{i\eta}])\right)\right] + \rho'\right) \\
&\leq \prod_{i=1}^{n}\left(\sum_{j=0}^{\eta}\mathbf{Pr}\left[d_H\big((x_i[w_1], ..., x_i[w_\eta]), (x_i'[w_1], ..., x_i'[w_\eta]) = j\big)\rho^j\right] + \rho'\right) \\
&\leq \prod_{i=1}^{n}(\sum_{j=0}^{\eta}(C_t^i\delta^i(1-\delta)^{t-i}\rho^i) + \rho') \leq \left[(1-\delta(1-\rho))^\eta + \rho'\right]^n
\end{aligned}$$

From the constructions of Π_1 and Π_2, we can know that $\rho = 0, \rho' = \frac{1}{p}$, so we can get $\tau \leq ((1-\delta)^t + \frac{1}{p})^n$. From Theorem 7, in order to ensure that $\mathbf{SD}\left((c, f(\mathrm{sk}_Y), \boldsymbol{k}), (c, f(\mathrm{sk}_Y), \boldsymbol{k}')\right) \leq 2^{-\lambda}$, we should have $\tau \leq \frac{1}{p^n}(1 + (2^{-\lambda})^2)$. So we get lower bounds of η and $\beta' := \tilde{\mathbf{H}}_\infty(\mathrm{sk}_Y | f(\mathrm{sk}_Y))$ are:

$$\eta \geq \frac{\log p - 1}{\log \frac{1}{1-\delta}}, \textit{and } \beta' \geq H_p(\delta)m\log p + n\log p + 2\lambda - 1 \ .$$

In our case, $\beta' \geq \mathbf{H}_\infty(\mathrm{sk}_Y) - l' = m\log p - l'$. For any constants $\varepsilon > 0$, there exists some constant $c \geq 0$, such that for any $n \geq 1, p \geq 2, \eta \geq c\log p, m \geq 0$, we have that: If $m\log p - l' \geq \varepsilon m\log p + n\log p + 2\lambda$, then $\mathbf{SD}\left((c, f(\mathrm{sk}_Y), \boldsymbol{k}), (c, f(\mathrm{sk}_Y), \boldsymbol{k}'\right) \leq 2^{-\lambda}$. It means that Π_2 is an l'-leakage-smooth IP-HPS for $l' = (1-\varepsilon)m\log p - n\log p - 2\lambda$.

C Proof of Theorem 5

Proof. The correctness of decryption follows by the correctness of decapsulation in Π_2. We use a series of games to analyze the security:

- **Game 0**: Define Game 0 to be the IND-security game with leakage l. In the challenge stage of Game 0, the challenger computes $\mathrm{ct}_{\boldsymbol{x}_b} \leftarrow \mathrm{Encrypt}(\mathrm{mpk}, \boldsymbol{x}_b)$ which we parse $\mathrm{ct}_{\boldsymbol{x}_b} = (c_1, c_2)$, where $c_1 = \mathrm{ct}_{\boldsymbol{z}}, c_2 = \boldsymbol{k} + \boldsymbol{x}_b$.
- **Game 1**: We modify the challenge stage, so that the challenger uses the secret keys $\{\mathrm{sk}_{\boldsymbol{y}_i}, i\}_{i=1}^{t}, t \leq n$ queried by $\mathcal{A}$ in Query 1, together with some new keys $\mathrm{sk}_{(\boldsymbol{y}_{t+1},t+1)}, ..., \mathrm{sk}_{(\boldsymbol{y}_n,n)}$ generated by running $\Pi_2.\mathrm{KeyGen}(\mathrm{msk}, \boldsymbol{y}_{t+j}, t+j), j \in [n-t]$ with the same random numbers as $\mathrm{sk}_{(\boldsymbol{y}_i,i)}, i \in [t]$, where $\boldsymbol{y}_{t+1}, ..., \boldsymbol{y}_n$ are randomly chosen subject to the condition that $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n]$ is an $n \times n$ invertible matrix. It computes $(c_1, \boldsymbol{k}_1) \leftarrow \mathrm{Encap}(\boldsymbol{z})$, then finds $\boldsymbol{k}_2$ such that $\boldsymbol{k}_2^T = [\mathrm{Decap}(c_1, \mathrm{sk}_{(\boldsymbol{y}_1,1)}), ..., \mathrm{Decap}(c_1, \mathrm{sk}_{(\boldsymbol{y}_n,n)})]Y^{-1}$, and computes $c_2 = \boldsymbol{k}_2 + \boldsymbol{x}_b$.
  The difference between Game 0 and Game 1 is only the use of $\boldsymbol{k}_1$ versus $\boldsymbol{k}_2$. However, by the correctness of Decapsulation, we have $\boldsymbol{k}_1 \neq \boldsymbol{k}_2$ with negligible probability, given that $\boldsymbol{y}_1, ..., \boldsymbol{y}_n$ are linear independent. So Game 0 and Game 1 are statistically indistinguishable.
- **Game 2**: We modify the challenge stage again, so that the challenger uses Encap^* to compute the ciphertext. It computes $c_1 \leftarrow \mathrm{Encap}^*(\boldsymbol{z})$, then finds $\boldsymbol{k}_2$ such that $\boldsymbol{k}_2^T = [\mathrm{Decap}(c_1, \mathrm{sk}_{(\boldsymbol{y}_1,1)}), ..., \mathrm{Decap}(c_1, \mathrm{sk}_{(\boldsymbol{y}_n,n)})]Y^{-1}$, and computes $c_2 = \boldsymbol{k}_2 + \boldsymbol{x}_b$.
  We claim that Game 1 and Game 2 are computationally indistinguishable by the valid/invalid ciphertext indistinguishability of IP-HPS. Although the valid/invalid ciphertext indistinguishability game does not have leakage queries, it allows the adversary to learn at most n secret keys. The total number of leakage queries the adversary have made in Query 1 is at most n, and all secret keys have been queried by the adversary were generated by the same randomness $\mathcal{R}$. Therefore, indistinguishability between Game 1 and Game 2 holds even if the adversary sees all the full secret keys $\mathrm{sk}_{\boldsymbol{y}}$ that the adversary have made leakage queries in Query 1.
- **Game 3**: The challenge ciphertext $\mathrm{ct}_{\boldsymbol{x}_b} = (c_1, c_2)$ is computed by: $c_1 \leftarrow \mathrm{Encap}^*(\boldsymbol{z}), c_2 \leftarrow U_{\mathcal{K}}$.
  We claim that Game 2 and Game 3 are statistically indistinguishable by the l'-leakage-smoothness of IP-HPS. Indeed, for a fixed value of $\mathrm{mpk}, \mathrm{msk}$, and $i \in [n]$, the only things in Game 2 correlated to $\mathrm{sk}_{\boldsymbol{y}_i}$ are the outputs of leakage query with size $l \leq \frac{l'}{n}$ bits. So the outputs of leakage queries of $\{\mathrm{sk}_{\boldsymbol{y}_i}\}_{i=1}^{n}$ are at most l' bits. Recall the definition of *l'-leakage-smoothness*, by making all leakage queries together as a single randomized function $f(\mathcal{Y})$ with $\mathcal{Y} = \{\mathrm{sk}_{\boldsymbol{y}_i}\}_{i=1}^{n}$, $\boldsymbol{k}_2$ is indistinguishable from choosing a completely independent random variable from $U_{\mathcal{K}}$.

Therefore Game 0 and Game 3 are indistinguishable by any PPT adversary. And the advantage of any adversary in Game 3 is 0, since the challenge ciphertext in Game 3 is independent of the bit b.