Adaptive-Secure Identity-Based Inner-Product Functional Encryption and its Leakage-Resilience

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. There are lots of applications of inner-product functional encryption (IPFE). In this paper, we consider two important extensions of it. One is to enhance IPFE with access control such that only users with a pre-defined identity are allowed to compute the inner product, referred as identity-based inner-product functional encryption (IBIPFE). We formalize the definition of IBIPFE, and propose the first adaptive-secure IBIPFE scheme from Decisional Bilinear Diffie-Hellman (DBDH) assumption. In an IBIPFE scheme, the ciphertext is related to a vector $\boldsymbol{x}$ and a new parameter, identity ID. Each secret key is also related to a vector $\boldsymbol{y}$ and an identity ID$'$. The decryption algorithm will output the inner-product value $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ only if ID = ID$'$.
The other extension is to make IBIPFE leakage resilient. We consider the bounded-retrieval model (BRM) in which an adversary can learn at most l bits information from each secret key. Here, l is the leakage bound determined by some external parameters, and it can be set arbitrarily large. After giving the security definition of leakage-resilient IBIPFE, we extend our IBIPFE scheme into a leakage-resilient IBIPFE scheme in the BRM by hash proof system (HPS).

Keywords: Identity-based access control · Inner-product functional encryption · Bounded-retrieval model · Hash proof system

1 Introduction

Functional encryption (FE) [11, 42] is a cryptographic primitive that addresses the "all-or-nothing" issue of traditional *Public key encryption* (PKE). FE allows users with secret keys to learn specific functional values of the encrypted data. Roughly speaking, in an FE scheme, given a ciphertext of x, a user who holds secret key sk_f for function f can only learn $f(x)$ and nothing else. Further, FE is the most general form of encryption. *Identity-based encryption* (IBE) [10, 29, 44, 46], *Attribute-based encryption* (ABE) [32, 47] and *Predicate encryption* (PE) [45] are considered as special cases of FE.

FE schemes for general functionalities (such as general circuits, Turing machines) have been proposed in many works [7,28,30,31,48]. But they have to rely on impractical and not well studied assumptions such as indistinguishability obfuscation (IO) or multilinear maps. Attacks were found for some constructions

of IO and multilinear maps [8, 14, 15, 17]. It is not clear if these FE schemes, based on IO and multilinear maps, are secure or not.

From 2005, researchers started to focus on giving FE schemes more restricted functionalities with security guaranteed by simple and well studied assumptions [1–3, 9, 51]. [1] first proposed FE schemes for *inner-product* (IPFE), which were proved selective-secure under DDH and LWE assumptions. [3] improved its work to achieve adaptive-secure under DDH, LWE and Composite residuosity hardness assumptions. In IPFE schemes, given an encryption of vector $\boldsymbol{x}$ and a secret key $\mathrm{sk}_{\boldsymbol{y}}$ based on a vector $\boldsymbol{y}$, the decryption algorithm will output the inner-product value $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$.

IPFE has numerous applications, such as computing the weighted mean. It is important to make it more powerful and secure. First, we consider adding access control to it. Similar to IBE, we can allow users specify an identity ID in their ciphertext $\mathrm{ct}_{(\mathrm{ID},\boldsymbol{x})}$. Each secret key $\mathrm{sk}_{(\mathrm{ID}',\boldsymbol{y})}$ is also related to an identity ID'. The decryption algorithm with input $\mathrm{ct}_{(\mathrm{ID},\boldsymbol{x})}$ and $\mathrm{sk}_{(\mathrm{ID}',\boldsymbol{y})}$ will output the inner-product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ only if $\mathrm{ID} = \mathrm{ID}'$. We call it *identity-based inner-product functional encryption* (IBIPFE). There is only one work [23] that considers such identity-based access control of FE. They proposed an unbounded inner-product functional encryption scheme with identity-based access control as a byproduct without giving the formal descriptions of the definitions. And the scheme is only proven to be selective-secure in the random oracle model.

1.1 Our contributions

As the first contribution of our work, in Section 3, we formalize the IND-security definition[1] of IBIPFE, and give the first adaptive-secure IBIPFE scheme under the DBDH assumption. As another benefit, our security proof is simpler than the security proof in the selective case [23]. We use the fact that the master secret key is known to the reduction at any time, then it can handle secret key queries without knowing the challenge messages in advance. Actually, IBIPFE can be viewed as a variation of functional encryption as follows:

$$F((\mathrm{ID}, \boldsymbol{x}), (\mathrm{ID}', \boldsymbol{y})) = \begin{cases} \langle \boldsymbol{x}, \boldsymbol{y} \rangle, & \text{If} \quad \mathrm{ID} = \mathrm{ID}' \\ \perp, & \text{otherwise} \end{cases}$$

Both the plaintext and the key consist of 2 parts: an identity and a vector. So the IND-security states that the adversary who can query the secret keys for a set of $(\mathrm{ID}, \boldsymbol{y})$ cannot distinguish which of the challenge messages $(\mathrm{ID}^*, \boldsymbol{x}_0)$ or $(\mathrm{ID}^*, \boldsymbol{x}_1)$ was encrypted under the condition that: For all pairs $(\mathrm{ID}, \boldsymbol{y})$ have been queried, it must hold that $F((\mathrm{ID}^*, \boldsymbol{x}_0), (\mathrm{ID}, \boldsymbol{y})) = F((\mathrm{ID}^*, \boldsymbol{x}_1), (\mathrm{ID}, \boldsymbol{y}))$.

[1] Unlike traditional PKE, the simulation-based security is not always achievable for FE [42]. So Indistinguishability-based security (IND-security) is widely used in FE research. Generally speaking, IND-security states that the adversary who has the secret keys for functions $\{f_i\}_{i \in [\eta]}$ cannot distinguish which of the challenge messages x_0 or x_1 was encrypted under the condition that for all $i \in [\eta]$, $f_i(x_0) = f_i(x_1)$.

The main difference between selective-IND-security and adaptive-IND-security is: in the adaptive-IND-security game, the adversary is given the master public key at the beginning and chooses the challenge messages after the first round of secret key queries. While in the selective-IND-security game, the adversary has to decide the challenge messages before the generation of master public key and master secret key.

In Sections 4 and 5, we further enhance the security of IPFE by allowing an adversary to learn some information about the secret keys, i.e. *leakage-resilience*. We extend our IBIPFE scheme under the bounded-retrieval model (BRM) [18, 24], one of the most widely used models in *leakage-resilient cryptography*. It states that our scheme can be proven secure even if an adversary can obtain l bits from a secret key sk, where l is the leakage bound and is decided by external factors. So in the security definition of leakage-resilient IBIPFE, in addition to *secret key query*, the adversary can make *leakage query*, in which the adversary chooses a pair $(\mathrm{ID}, \boldsymbol{y})$ and a leakage function f^*, and the challenger will reply with $f^*(\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})})$ as long as the bit-length of output is at most l. The adversary can know all secret keys in the form $\mathrm{sk}_{(\mathrm{ID},\cdot)}$ by making *secret key query* when $\mathrm{ID} \neq \mathrm{ID}^*$ (ID^* is the challenge identity). Thus, the key issue in the *leakage query* is the queries on the challenge identity ID^*. Also, the BRM requires that all efficiency parameters other than the secret key size (such as public key size, encryption time, decryption time and even master secret key size) only depend on the security parameter, and not the leakage bound l.

Our leakage-resilient IBIPFE scheme and its security proof build on *hash proof system* (HPS). [5, 39] showed how to use an 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 allows a sender that knows the public key, to securely agree on randomness k with a receiver possesses the secret key, by sending an encapsulation ciphertext. A KEM consists of 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) It 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 valid ciphertext into invalid ciphertext in the first step, we can argue the leakage using information-theoretic analysis.

However, existing HPS schemes such as IB-HPS [5] cannot be applied to our cases. When we build an encryption scheme from an HPS scheme, we usually use the encapsulated key as a mask to hide the plaintext in the encryption algorithm, and recover the plaintext from the ciphertext by running the decap-

sulation algorithm to get the encapsulated key. When we apply it to IBIPFE, if the decapsulation algorithm of the underlying HPS still outputs the encapsulated key directly, then the decryption of IBIPFE will reveal the plaintext vector, other than only an inner-product value. (Recall that IBIPFE requires that the decryption result only reveals an inner-product value and nothing else.) In order to guarantee the security of resulting IBIPFE scheme, modifications are necessary. We first develop the notion *Identity-based inner-product hash proof system*(IBIP-HPS), which can yield an IBIPFE scheme. Different from other HPS schemes, in an IBIP-HPS scheme, the decapsulation algorithm with input a ciphertext $\mathrm{ct}_{\mathrm{ID}}$ and a secret key $\mathrm{sk}_{(\mathrm{ID}',\boldsymbol{y})}$ will only output an inner-product value of $\boldsymbol{y}$ and the encapsulated key $\boldsymbol{k}$ when $\mathrm{ID} = \mathrm{ID}'$, and nothing else. Now, we can get a secure IBIPFE from IBIP-HPS very easily, by simply using the encapsulated key as a one-time pad to encrypt a plaintext vector.

Next, we briefly introduce a key property of IBIP-HPS: *Leakage-smoothness*. Leakage-smoothness states that the distribution of encapsulated key derived from an invalid ciphertext and a set of secret keys is almost uniform over the key space, even if the adversary can learn at most l' bits information from the secret keys, where l' is a pre-determined leakage bound. Then, we move our focus from the leakage-resilience of IBIPFE to the leakage-smoothness of IBIP-HPS by proving the following theorem:

Theorem 1 (informal). *Given a leakage-smooth IBIP-HPS with leakage bound l', which satisfies the efficiency requirements of the BRM, we can obtain a leakage-resilient IBIPFE with leakage bound $l = \frac{l'}{n}$ in the BRM, where n is the length of vectors.*

Now our goal is to design a l'-leakage-smooth IBIP-HPS, which meets the efficiency requirements of the BRM. To make it simple, we would like to design an IBIP-HPS scheme from simple assumptions, regardless the requirements of the leakage-smoothness and efficiency. We build an IBIP-HPS Π_1 from our IBIPFE scheme Π. The main challenge is that, the key generation algorithm in Π is deterministic, while in IBIP-HPS, the secret key should be generated randomly. We first choose a random number z and define a new vector $\boldsymbol{y}^*$ by concatenating $\boldsymbol{y}$ and z. After that, we get the secret key by running the key generation algorithm of Π with input the new vector $\boldsymbol{y}^*$. Thus, the secret key is related to the random number z. Then, we study the 0-universality of the decapsulation algorithm, where the 0-universality ensures that it is unlikely that any two distinct secret keys for the same pair $(\mathrm{ID}, \boldsymbol{y})$ will decapsulate a ciphertext to the same value. The formal definition of 0-universality is given in Definition 10. Now, we show that we can convert Π_1 into a l'-leakage-smooth IBIP-HPS that allows for arbitrarily large leakage-bounds l'. We prove a theorem here:

Theorem 2 (informal). *Given an 0-universal IBIP-HPS Π_1, we can get a l'-leakage-smooth IBIP-HPS Π_2 for arbitrarily large leakage bound l'. And Π_2 meets the efficiency requirements of the BRM.*

To handle arbitrarily large leakage bound l', [5] used a leakage amplification method, which can be viewed as *parallel-repetition* with small public key size.

However, it 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 directly from the 0-universality by leftover-hashing lemma [40]. Thus the only thing they need to do is to amplify the leakage bound while meeting the efficiency requirements of the BRM. However, in IBIP-HPS, the output of the decapsulation is only an inner-product value between the encapsulated key and the vector in the secret key. In order to determine an encapsulated key, we need at least n secret keys for n linear independent vectors. Thus, we cannot find the relation between leakage-smoothness and universality very easily, which is one of the most challenging part in our work.

Although the leakage amplification 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 t) 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 which 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 obtain 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 t, which is independent of the leakage bound l'.

In summary, we do the followings:
- Formalize the definitions of IBIPFE and give an adaptive-IND-secure IBIPFE scheme Π from DBDH assumption.
- Propose the definition of IBIP-HPS and desired properties. Then, we give an IBIP-HPS construction Π_1 based on our adaptive-secure IBIPFE scheme Π.
- Construct a leakage-smooth IBIP-HPS Π_2 from Π_1 for arbitrarily large leakage bound l', while Π_2 still satisfies the efficiency requirements of the BRM.
- Build a leakage-resilient IBIPFE scheme Π_3 in the BRM from Π_2.

1.2 Related works: Leakage-resilient cryptography

Due to the advancement of side channel attacks [33–36], traditional cryptographic model, where an adversary can know nothing about the secret values, becomes insufficient. *Leakage-resilient cryptography* was proposed to formalize the security guarantees when the adversary can obtain some information of the secret values. Lots of leakage models were proposed to measure what and how much information of secrets the adversary can learn.

[38] introduced the first leakage model: *only computation leaks information.* In this model, a function of only the bits accessed is leaked when the crypto-

graphic system is called each time. Many cryptographic schemes were proposed under this model, such as stream ciphers [26, 43] and signature schemes [27]. However, a famous type of side-channel attacks, cold-boot attack [33] was proposed and was not captured by this model, where all memory contents can leak information, regardless of whether it is accessed.

Relative-leakage model was proposed for these attacks. In this model, an adversary can learn a proportion of secret values. And also, there are many schemes were proposed under this model, such as PKE schemes [4, 39], IBE scheme [16]. After that, *Bounded-retrieval model* was introduced by [18, 24]. In this model, the amount of information can be leaked is bounded by a leakage bound l, where l is decided by an external parameter. And it requires that the efficiency of the system, except the length of secret key, should be independent of the leakage bound l. Many schemes [5, 13, 25, 41, 52] were proposed under this model. Then, *Auxiliary inputs model* was proposed by [21]. In this scheme, an adversary can learn an auxiliary input $h(s)$ of secret values s subject to the condition that it is computationally hard to find s from $h(s)$. Many works [19,49] proposed different kinds of cryptographic systems under this model.

As another line of work, *Continual leakage model* was introduced by [12, 20]. This model considers the setting that there is a notion of time periods and secret values will be updated at the end of each time period. Here, an adversary can only learn a bounded amount of information in each time period, but it can learn an unbounded amount of information in all time periods. [37, 49, 50] proposed many cryptographic systems under this model.

2 Preliminaries

Notations. Let $[n]$ denote the 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 as the uniform distribution over S. Similarly, for an integer $v \in \mathbb{N}$, let U_v be the uniform distribution over $\{0,1\}^v$.

2.1 Functional Encryption (FE)

We give the definition of FE and its indistinguishable security. Following [11], we define functional encryption scheme for functionality $\mathcal{F}$.

Definition 1 (FE scheme). *A functional encryption scheme for functionality $\mathcal{F}$ consists of 4 PPT algorithms:* $(\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encrypt}, \mathrm{Decrypt})$. *The algorithms have the following syntax.*

- $\mathrm{Setup}(1^\lambda)$*: 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.*
- $\mathrm{KeyGen}(\mathrm{msk}, k)$*: It uses the master secret key* msk *and key $k \in \mathcal{K}$ to sample a secret key* sk_k*.*
- $\mathrm{Encrypt}(x)$*: It uses the master public key* mpk *and a message $x \in \mathcal{X}$ to generate a ciphertext* ct_x*.*

– $\mathrm{Decrypt}(\mathrm{sk}_k, \mathrm{ct}_x)$*: It takes a ciphertext* ct_x *and a secret key* sk_k *as input and outputs* $\mathcal{F}(k, x)$

We require that a FE scheme satisfies the following properties:
Correctness. For any $(\mathrm{mpk}, \mathrm{msk})$ generated by $\mathrm{Setup}(1^\lambda)$, any $k \in \mathcal{K}$ and $x \in \mathcal{X}$, we have:

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

Indistinguishable security. We define the *indistinguishable security game*, parameterized by a security parameter λ as the 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 $Adv^{\text{FE-IND}}_{\text{FE},\mathcal{A}}(\lambda) := |\mathbf{Pr}[\mathcal{A}\ wins] - \frac{1}{2}|$.

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* $Adv^{FE\text{-}IND}_{FE,\mathcal{A}}(\lambda) = \mathrm{negl}(\lambda)$.

Inner-product functionality. In this paper, we are interested in the *inner-product functionality* over the field $\mathbb{Z}_p$ defined in [1]. It is a family of functionalities with vector space $\mathcal{V}$ consisting of vectors in $\mathbb{Z}_p$ of length n: for any $\boldsymbol{y}, \boldsymbol{x} \in \mathcal{V}$, the functionality $\mathcal{F}(\boldsymbol{y}, \boldsymbol{x}) = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$.

2.2 Bilinear groups

Our construction relies on the widely-used technique: bilinear map.

Definition 3 (Bilinear map). *Let* $\mathbb{G}_1$, $\mathbb{G}_2$ *and* $\mathbb{G}_T$ *be cyclic groups of order* p*. Define function* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*. Then* e *is an efficient computable (non-degenerate) bilinear map if it is:*

1. *Bilinear:* $\forall a, b \in \mathbb{Z}_p, (x_1, x_2) \in \mathbb{G}_1 \times \mathbb{G}_2, e(x_1^a, x_2^b) = e(x_1, x_2)^{ab}$.
2. *Non-degenerate: e does not map all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to the identity in $\mathbb{G}_T$.*
3. *Efficiently computable: There's an efficient algorithm to compute any of the function value of e.*

Let g_i be the generator of $\mathbb{G}_i$, for each $i \in \{1, 2, T\}$. Given an efficient computable (non-degenerate) bilinear map e, by definition we have $e(g_1, g_2) = g_T$.

We will use a generator which on input 1^λ, it will efficiently return $(\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, e)$ satisfying the above description.

[10] introduce an assumption for the case when $\mathbb{G}_1 = \mathbb{G}_2$, and it has been adapted to asymmetric setting in [23]. We use the latter one and give the definition.

Definition 4 (Decisional Bilinear Diffie-Hellman Assumption). *The Decisional Bilinear Diffie-Hellman Assumption(DBDH) Assumption in the asymmetric case is, given $(\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, e)$ returned by our generator, no PPT adversary can distinguish between the two following distributions with non-negligible advantage: $\left(g_1^a, g_1^b, g_2^a, g_2^c, g_T^q\right), \left(g_1^a, g_1^b, g_2^a, g_2^c, g_T^{abc}\right)$, where a, b, c, q are sampled from $\mathbb{Z}_p$.*

2.3 Entropy, extractors and hashing

We introduce the definition of *min-entropy* which measures the worst-case predictability of a random variable. Further, for a randomized function f, let $f(x; r)$ be the unique output of f for input x, with random coins r. For simplicity, we will write $f(x)$ to denote a random variable for the output of $f(x; r)$ over the random coins r.

Definition 5. *The* min-entropy *of a random variable X is* $\mathbf{H}_\infty(X) := -\log(\max_x \mathbf{Pr}[X = x])$.

A generalized version from [22] is called the average conditional min-entropy*:*

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

where Z is another random variable.

Lemma 1 (Lemma 2.2 in [22]). *Let X, Y, Z be random variables where Z takes on values in a set of size at most 2^l. Then $\tilde{\mathbf{H}}_\infty(X|(Y, Z)) \geq \tilde{\mathbf{H}}_\infty((X, Y)|Z) - l \geq \tilde{\mathbf{H}}_\infty(X|Z) - l$. In particular, $\tilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - l$.*

Statistical distance and Extractors. For two random variables X, Y, we can define the *statistical distance* between them as $\mathbf{SD}(X, Y) := \frac{1}{2}\sum_x |\mathbf{Pr}[X = x] - \mathbf{Pr}[Y = x]|$. If $\mathbf{SD}(X, Y) \leq \epsilon$ then we write $X \approx_\epsilon Y$. Further, if $\mathbf{SD}(X, Y)$ is negligible, we write $X \approx Y$. An extractor [40] can be used to extract uniform randomness out of a weakly-random variable which is only assumed to have sufficient min-entropy.

Definition 6. *An efficient randomized function* $\mathrm{Ext} : \{0,1\}^a \to \{0,1\}^v$ *is a* (β, ϵ)-extractor *if for all* X, Z *such that* X *is distributed over* $\{0,1\}^a$ *and* $\tilde{\mathbf{H}}_\infty(X|Z) \geq \beta$*, we get* $(Z, R, \mathrm{Ext}(X;R)) \approx_\epsilon (Z, R, U_v)$ *where* R *is a random variable for the coins of* Ext.

Definition 7 (ρ-Universal Hashing). *A family* $\mathcal{H}$*, consisting of (deterministic) functions* $h : \{0,1\}^a \to \{0,1\}^v$*, is* ρ-universal hash family *if for any* $x_1 \neq x_2 \in \{0,1\}^a$*, we have* $\mathbf{Pr}_{h \leftarrow \mathcal{H}}[h(x_1) = h(x_2)] \leq \rho$.

[40] states that universal hash functions are good extractors in the following *leftover-hash lemma*.

Lemma 2 (Leftover-Hash Lemma [40]). *Assume that the family* $\mathcal{H}$ *of functions* $h : \{0,1\}^a \to \{0,1\}^v$ *is* ρ*-universal hash family. Then the random extractor* $\mathrm{Ext}(x;h) = h(x)$*, where* h *is uniform over* $\mathcal{H}$*, is an* (β, ϵ)*-extractor as long as:*

$$\beta \geq v + 2\log(1/\epsilon) - 1, \rho \leq 2^{-v}(1+\epsilon^2) \ .$$

Also, we say that e *is efficiently computable if given the description of* e *we are able to obtain a polynomial time algorithm for computing* e.

3 Adaptive-secure IBIPFE scheme

3.1 Definitions

Identity-based inner-product functional encryption. Firstly, we give the definition of IBIPFE as follows: An IBIPFE scheme consists of 4 PPT algorithms just like IBE and IPFE: (Setup, KeyGen, Encrypt, Decrypt). The algorithms have the following syntax. [2]

- $\mathrm{Setup}(1^\lambda, 1^n)$: It takes the security parameter λ and n as input, and produces the *master public key* mpk and the *master secret key* msk. The following algorithms implicitly include mpk as input.
- $\mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y})$: It uses the master secret key msk, an identity $\mathrm{ID} \in \mathcal{ID}$ and a vector $\boldsymbol{y} \in \mathcal{V}$ with length n to sample a secret key $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})}$.
- $\mathrm{Encrypt}(\mathrm{ID}, \boldsymbol{x})$: This is the encryption algorithm. It uses $\mathrm{ID} \in \mathcal{ID}$ and $\boldsymbol{x} \in \mathcal{V}$ to output a ciphertext $\mathrm{ct}_{(\mathrm{ID},\boldsymbol{x})}$.
- $\mathrm{Decrypt}(\mathrm{ct}_{(\mathrm{ID},\boldsymbol{x})}, \mathrm{sk}_{(\mathrm{ID}',\boldsymbol{y})})$: This is the decryption algorithm(deterministic). It takes a ciphertext and a secret key as input and outputs the inner product: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ if $\mathrm{ID} = \mathrm{ID}'$.

Here, we define a function $F : (\mathcal{ID}, \mathcal{V}) \times (\mathcal{ID}, \mathcal{V}) \to \mathcal{IP}$, where

$$F((\mathrm{ID}, \boldsymbol{x}), (\mathrm{ID}', \boldsymbol{y})) = \begin{cases} \langle \boldsymbol{x}, \boldsymbol{y} \rangle, & \text{If} \quad \mathrm{ID} = \mathrm{ID}' \\ \perp, & \text{otherwise} \end{cases}$$

[2] Here, let $\mathcal{ID}$ be the identity space, $\mathcal{V}$ be the vector space, and $\mathcal{IP}$ be the inner-product value space.

Correctness. Given $\mathrm{msk}, \mathrm{mpk}$ from $\mathrm{Setup}(1^\lambda, 1^n)$, any $\mathrm{ID} \in \mathcal{ID}$, and $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}$, we have:

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

Indistinguishable security. We define the *indistinguishable security game*, parametrized by a security parameter λ, a parameter of vector length n, as the game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ in Table 2.

Table 2: IBIPFE-IND(λ, n)

Setup: The challenger computes $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{Setup}(1^\lambda, 1^n)$ and sends mpk to the adversary.
Query 1: The adversary $\mathcal{A}$ adaptively queries $\mathcal{C}$ with $(\mathrm{ID}, \boldsymbol{y}) \in \mathcal{ID} \times \mathcal{V}$. And the challenger $\mathcal{C}$ responds with $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})}$.
Challenge: The adversary $\mathcal{A}$ chooses an challenge identity $\mathrm{ID}^* \in \mathcal{ID}$ and two messages $\boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathcal{V}$ subject to the condition that for all $(\mathrm{ID}, \boldsymbol{y})$ $\mathcal{A}$ has queried in Query 1, it must hold that $F((\mathrm{ID}^*, \boldsymbol{x}_0), (\mathrm{ID}, \boldsymbol{y})) = F((\mathrm{ID}^*, \boldsymbol{x}_1), (\mathrm{ID}, \boldsymbol{y}))$
Query 2: The adversary $\mathcal{A}$ adaptively queries $\mathcal{C}$ with $(ID, \boldsymbol{y})$ as long as $F((\mathrm{ID}^*, \boldsymbol{x}_0), (\mathrm{ID}, \boldsymbol{y})) = F((\mathrm{ID}^*, \boldsymbol{x}_1), (\mathrm{ID}, \boldsymbol{y}))$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and wins if $b' = b$.

Note: $F((\mathrm{ID}^*, \boldsymbol{x}_0), (\mathrm{ID}, \boldsymbol{y})) = F((\mathrm{ID}^*, \boldsymbol{x}_1), (\mathrm{ID}, \boldsymbol{y}))$ holds if and only if (1) $\mathrm{ID} \neq \mathrm{ID}^*$, or (2) $\mathrm{ID} = \mathrm{ID}^*$ and $\langle \boldsymbol{x}_0, \boldsymbol{y}\rangle = \langle \boldsymbol{x}_1, \boldsymbol{y}\rangle$.

Definition 8 (Adaptive-IND-secure IBIPFE). *An IBIPFE scheme is adaptive-IND-secure, if (1) it satisfies the* correctness*, and (2) the advantage of any admissible PPT adversary $\mathcal{A}$ in the indistinguishable security game is:* $Adv^{IBIPFE\text{-}IND}_{IBIPFE,\mathcal{A}}(\lambda, n) = \mathrm{negl}(\lambda)$.

3.2 Adaptive-IND-secure IBIPFE scheme

Now, we present our adaptive-IND-secure IBIPFE scheme Π. Here we set $\mathcal{IP} = \mathbb{Z}_p$, $\mathcal{V} = \mathbb{Z}_p^n$ and $\mathcal{ID} = \{0,1\}^{\log p - \log n}$. Also, the set $[n]$ can be seen as the set of all non-zero $(0,1)$-strings with length $\log n$ (i.e. $\{0,1\}^{\log n} \setminus \mathbf{0}^{\log n}$). For any $i \in [n]$ and $\mathrm{ID} \in \mathcal{ID}$, let i^{01} be the binary representation of i (we will use this notion below), then $\mathrm{ID}||i^{01}$ is in fact a non-zero $(0,1)$-string with length $\log p$ (i.e. $\{0,1\}^{\log p} \setminus \mathbf{0}^{\log p}$).

- $\mathrm{Setup}(1^\lambda, 1^n)$: Pick a bilinear group $\mathcal{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, p)$ of prime order p, where g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and we have $g_T = e(g_1, g_2)$. Choose random numbers $w, v, s, t \leftarrow_R \mathbb{Z}_p^*$ Now we can pick a random one-to-one function $h : \{0,1\}^{\log p} \setminus \mathbf{0}^{\log p} \rightarrow \mathbb{Z}_p^*$. And set $h_1 = g_1^w$, $h_2 =$

$g_2^v, k_1 = g_1^s \cdot h_1^t, f(\cdot) = v \cdot h(\cdot)$. [3] Output: $\text{mpk} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, h_2, k_1, f)$, $\text{msk} = (s, t, h)$.

- Encrypt$(\text{mpk}, \text{ID}, \boldsymbol{x})$: Pick a random number $r \leftarrow_R \mathbb{Z}_p$ and set $C = g_1^r$, $D = e(h_1, h_2)^r$, and for $i \in [n]$, $E_i = g_T^{x_i} \cdot e(k_1, g_2^{f(\text{ID}||i^{01})})^r$. Output: $\text{ct}_{(\text{ID},\boldsymbol{x})} = (C, D, E_1, ..., E_n)$.
- KeyGen$(\text{msk}, \text{ID}, \boldsymbol{y})$: Set $d_1 = (\prod_{i=1}^n g_2^{f(\text{ID}||i^{01})y_i})^{-s}$ and $d_2 = -t\big(\sum_{i=1}^n h(\text{ID}||i^{01})y_i\big)$. Output: $\text{sk}_{(\text{ID},\boldsymbol{y})} = (d_1, d_2)$.
- Decrypt$(\text{ct}_{(\text{ID},\boldsymbol{x})}, \text{sk}_{(\text{ID}',\boldsymbol{t})})$: Compute: $\text{IP} = e(C, d_1) \cdot D^{d_2} \cdot (\prod_{i=1}^n E_i^{y_i})$. Then, compute and output the discrete logarithm $\log_{g_T} \text{IP}$.

For simplicity we use $\text{ID}||i$ to represent $\text{ID}||i^{01}$ in the following of this paper.

Correctness. When $\text{ID} = \text{ID}'$, we have

$$
\begin{aligned}
\text{IP} &= e(C, d_1) \cdot D^{d_2} \cdot (\prod_{i=1}^n E_i^{y_i}) \\
&= e\Big(g_1^r, (\prod_{i=1}^n g_2^{f(\text{ID}'||i)y_i})^{-s}\Big) \cdot e(h_1, h_2)^{r(-t\sum_{i=1}^n h(\text{ID}'||i)y_i)} \cdot (\prod_{i=1}^n E_i^{y_i}) \\
&= g_T^{-rv(s+wt)\sum_{i=1}^n h(\text{ID}'||i)y_i} \prod_{i=1}^n \Big(g_T^{x_i y_i} g_T^{rv(s+wt)\sum_{i=1}^n h(\text{ID}||i)y_i}\Big) \\
&= g_T^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g_T^{rv(s+wt)\sum_{i=1}^n (h(\text{ID}||i) - h(\text{ID}'||i))y_i} = g_T^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \ .
\end{aligned}
$$

Therefore Decrypt$(\text{ct}_{(\text{ID},\boldsymbol{x})}, \text{sk}_{(\text{ID},\boldsymbol{y})})$ outputs $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$.

Similar to many IPFE works based on DDH assumption and its variants, the decryption algorithm of Π requires to compute the discrete logarithm. We can use some methods to reduce the cost of it (see the analysis in [3]). The security analysis follows similar arguments to [3] in that at some steps, the challenge ciphertext is generated using msk instead of mpk. It will perfectly hide which challenge message is encrypted as long as msk retains a sufficient amount of entropy from the adversary's view. We state the security in the following theorem and the proof are shown in Supporting material A.

Theorem 3 (adaptive-IND security). *The IBIPFE scheme Π described above is adaptive-IND-secure under the DBDH assumption.*

[3] Here we keep function h secret and set function f to be a black box, which means that one can only get the function value of f by making a query to the oracle, instead of computing it directly.

4 Identity-based Inner-product hash proof system(IBIP-HPS)

4.1 Definitions

To construct a leakage-resilient IBIPFE scheme, we introduce the notion, IBIP-HPS, and the required properties:
An *Identity-based Inner-product hash proof system* (IBIP-HPS) consists of 5 PPT algorithms: $(\text{Setup}, \text{KeyGen}, \text{Encap}, \text{Encap}^*, \text{Decap})$. The algorithms have the following syntax. ($\mathcal{ID}$ is the identity space, $\mathcal{V}$ is the vector space.)

- $\text{Setup}(1^\lambda, 1^n)$: It takes the security parameter λ and n as input, and produces the *master public key* mpk and the *master secret key* msk. The following algorithms implicitly include mpk as input.
- $\text{KeyGen}(\text{msk}, \text{ID}, \boldsymbol{y})$: It uses the master secret key msk, an identity $\text{ID} \in \mathcal{ID}$ and a vector $\boldsymbol{y} \in \mathcal{V}$ with length n to sample a secret key $\text{sk}_{(\text{ID},\boldsymbol{y})}$.
- $\text{Encap}(\text{ID})$: This is the *valid* encapsulation algorithm. It uses $\text{ID} \in \mathcal{ID}$ to output a valid ciphertext ct_{ID} and a encapsulated key $\boldsymbol{k} \in \mathcal{V}$.
- $\text{Encap}^*(\text{ID})$: This is the *invalid* encapsulation algorithm. It uses $\text{ID} \in \mathcal{ID}$ to output only an invalid ciphertext ct_{ID}.
- $\text{Decap}(\text{ct}_{\text{ID}}, \text{sk}_{(\text{ID}',\boldsymbol{y})})$: This is the decapsulation algorithm(deterministic). It takes a ciphertext as input and outputs a functional value of the encapsulated key and $\boldsymbol{y}$: $\langle \boldsymbol{k}, \boldsymbol{y} \rangle$ if $\text{ID} = \text{ID}'$.

Correctness. Given msk, mpk from $\text{Setup}(1^\lambda, 1^n)$, any $\text{ID} \in \mathcal{ID}$, and $\boldsymbol{y} \in \mathcal{V}$, we have:

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

Valid/Invalid Ciphertext Indistinguishability. The valid ciphertexts generated by Encap and the invalid ciphertexts generated by Encap^* should be computationally indistinguishable, even if an adversary can obtain one secret key per pair for at most n linear independent vectors with the challenge identity ID^*. For an adversary $\mathcal{A}$, we define the following experiment for an IBIP-HPS Π in Table 3.

Definition 9. *A PPT adversary $\mathcal{A}$ is* admissible *if in the whole experiment, for the challenge identity ID^*, $\mathcal{A}$ can make at most n secret key queries in the form $(\text{ID}^*, \boldsymbol{y}_i), i \in [n]$, where $\boldsymbol{y}_1, ..., \boldsymbol{y}_n$ are linear independent. Then, we say that an IBIP-HPS Π is* adaptively secure *if for any admissible adversary $\mathcal{A}$, the advantage satisfies:* $\text{Adv}^{V/I\text{-}IND}_{\Pi,\mathcal{A}}(\lambda, n) := \left|\mathbf{Pr}[\mathcal{A} \ wins] - \frac{1}{2}\right| = \text{negl}(\lambda)$.

We will explain why such restriction on key queries is needed after show the following property: Smoothness and leakage-smoothness. Intuitively, this property is to ensure that there are many possible results for the decapsulation algorithm with input an invalid ciphertext.

Table 3: V/I-IND(λ, n)

Setup: The challenger $\mathcal{C}$ computes $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$, and gives mpk to the adversary $\mathcal{A}$,
Query 1: The adversary $\mathcal{A}$ adaptively queries $\mathcal{C}$ with $(\text{ID}, \boldsymbol{y}) \in \mathcal{ID} \times \mathcal{V}$, and $\mathcal{C}$ responds with $\text{sk}_{(\text{ID},\boldsymbol{y})}$.
Challenge: The adversary $\mathcal{A}$ chooses an arbitrary $\text{ID}^* \in \mathcal{ID}$, and the challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$
If $b = 0$, then $\mathcal{C}$ computes $(\text{ct}_{\text{ID}^*}, \boldsymbol{k}) \leftarrow \text{Encap}(\text{ID}^*)$. If $b = 1$, then $\mathcal{C}$ computes $\text{ct}_{\text{ID}^*} \leftarrow \text{Encap}^*(\text{ID}^*)$.
The challenger $\mathcal{C}$ gives ct_{ID^*} to the adversary $\mathcal{A}$.
Query 2: The adversary $\mathcal{A}$ adaptively queries $\mathcal{C}$ with $(\text{ID}, \boldsymbol{y}) \in \mathcal{ID} \times \mathcal{V}$, and $\mathcal{C}$ responds with $\text{sk}_{(ID,\boldsymbol{y})}$.
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, and $\mathcal{A}$ wins the game if $b' = b$.

Note: In Query 1,2 the challenger computes $\text{sk}_{(\text{ID},\boldsymbol{y})} \leftarrow \text{KeyGen}(\text{msk}, \text{ID}, \boldsymbol{y})$ the first time that the pair $(\text{ID}, \boldsymbol{y})$ is queried and responds to all future queries on the same pair $(\text{ID}, \boldsymbol{y})$ with the same $\text{sk}_{(\text{ID},\boldsymbol{y})}$.

Smoothness and leakage-smoothness. Define a matrix $Y := [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]$ consisting of n linear independent vectors. We say that an IBIP-HPS Π is *smooth* if, for any fixed values of mpk, msk from $\text{Setup}(1^\lambda, 1^n)$, any fixed Y and $\text{ID} \in \mathcal{ID}$, we have

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

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

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

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

where $\text{ct}, \boldsymbol{k}', \boldsymbol{k}$ and each $\text{sk}_{(\text{ID},\boldsymbol{y}_i)}$ are sampled as above.

Definition 10 (0-universal IBIP-HPS). *For any fixed* mpk, msk *returned by* $\text{Setup}(1^\lambda)$*, any fixed* $\text{ID} \in \mathcal{ID}$*, and any fixed vector* $\boldsymbol{y}$*, we let* SK *be the support of all possible output of* $\text{KeyGen}(\text{msk}, \text{ID}, \boldsymbol{y})$*. Then we say that an IBIP-HPS is* ***0-universal*** *if, for any fixed distinct values* $\text{sk}_{\text{ID}} \neq \text{sk}'_{\text{ID}}$*, we have*

$$\mathbf{Pr}_{c \leftarrow \text{Encap}^*(\text{ID})}[\text{Decap}(c, \text{sk}_{\text{ID}}) = \text{Decap}(c, \text{sk}'_{\text{ID}})] = 0 \ .$$

We also say that its decapsulation algorithm is 0-universal.

The restriction on key queries with the challenge identity in this game is due to the fact that, like all other HPS schemes, in order to achieve the smoothness/leakage-smoothness, the key generation algorithm should be a randomized algorithm. It means that we will get different output from KeyGen with the same input in different running. As a result of it, the output of the decapsulation algorithm

with input an **invalid** ciphertext and a secret key is also random, which depends on the random numbers used to generate the secret key. However, no matter what random numbers are used in the secret key, the output of the decapsulation algorithm with input a **valid** ciphertext and the secret key is always the real inner-product value when the identities in ciphertext and secret key are the same. If we do not have any restriction on the key queries with the challenge identity, the adversary can distinguish between valid and invalid ciphertexts easily, by making use of the above properties. For example, the adversary makes 2 queries: $(\mathrm{ID}^*, \boldsymbol{y}_1 = (1, 0, ..., 0))$ and $(\mathrm{ID}^*, \boldsymbol{y}_2 = (2, 0, ..., 0))$. If the ciphertext $\mathrm{ct}_{\mathrm{ID}^*}$ is a valid ciphertext, then we have $\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}^*}, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_2)}) - \mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}^*}, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_1)}) = 2k_1 - k_1 = \mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}^*}, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_1)})$. While if it is an invalid ciphertext, then $\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}^*}, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_2)}) - \mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}^*}, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_1)}) \neq \mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}^*}, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_1)})$ in general.

Instead of aiming at building an IBIP-HPS scheme which satisfies correctness, valid/invalid ciphertext indistinguishability and leakage-smoothness at the same time, we first consider how to construct an IBIP-HPS scheme which only satisfies orrectness and valid/invalid ciphertext indistinguishability. We show that we can get an IBIP-HPS scheme Π_1 from our IBIPFE scheme Π easily, by adding random numbers into key generation algorithm.

We construct an IBIP-HPS Π_1 from our adaptive-IND-secure IBIPFE scheme Π.

- $\mathrm{Setup}(1^\lambda, 1^n)$: It runs $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \Pi.\mathrm{Setup}(1^\lambda, 1^{n+1})$, and outputs mpk, msk.
- $\mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y})$: It samples a random number $z \leftarrow_R \mathbb{Z}_p$ and sets $\boldsymbol{y}^* = \boldsymbol{y}||z$. Then it runs $\mathrm{sk}^0_{(\mathrm{ID},\boldsymbol{y}^*)} \leftarrow \Pi, \mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}^*)$ and outputs $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})} := (\mathrm{sk}^0_{(\mathrm{ID},\boldsymbol{y}^*)}, z)$.
- $\mathrm{Encap}(\mathrm{ID})$: It samples $\boldsymbol{x} \leftarrow_R \mathcal{V}$ and sets $\boldsymbol{x}^* = \boldsymbol{x}||0$. It chooses a random number $r \leftarrow_R \mathbb{Z}_p$ and computes $\mathrm{ct}^0_{(\mathrm{ID},\boldsymbol{x}^*)} = \Pi.\mathrm{Encrypt}(\mathrm{mpk}, \mathrm{ID}, \boldsymbol{x}^*)$ with randomness r. It outputs $(\mathrm{ct}_{\mathrm{ID}} = \mathrm{ct}^0_{(\mathrm{ID},\boldsymbol{x}^*)}, \boldsymbol{k} = \boldsymbol{x})$.
- $\mathrm{Encap}^*(\mathrm{ID})$: It samples $\boldsymbol{x} \leftarrow_R \mathcal{V}$ and sets $\boldsymbol{x}^* = \boldsymbol{x}||0$. It chooses a random number $r, r' \leftarrow_R \mathbb{Z}_p$ and $r \neq r'$. It sets $C = g_1^r$, $D = e(h_1, h_2)^{r'}$ and for $i \in [n+1]$, $E_i = g_T^{x_i^*} \cdot e(k_1, g_2^{f(\mathrm{ID}||i)})^r$, where f is obtained from mpk. It outputs $\mathrm{ct}_{\mathrm{ID}} = (C, D, E_1, ..., E_{n+1})$.
- $\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID}',\boldsymbol{y})})$: It outputs $\Pi.\mathrm{Decrypt}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID}',\boldsymbol{y})})$.

Both the correctness and valid/invalid ciphertext indistinguishability of IBIP-HPS Π_1 can be easily proved from the correctness and the adaptive-IND security of the underlying IBFE scheme Π. We have the following theorem and the proof are shown in Supporting material B.

Theorem 4. *Under DBDH assumption, the above IBIP-HPS construction Π_1 satisfies the correctness requirement and valid/invalid ciphertext indistinguishability.*

We also show that the decapsulation function of Π_1 is a **0-universal**, which could help us to construct a leakage-smooth IBIP-HPS in the next section.

0-Universality of the decapsulation function. For any fixed mpk, msk produced by $\Pi_1.\text{Setup}(1^\lambda, 1^n)$, a set of linear independent vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ and identity ID, let ct_{ID} be some output of $\Pi_1.\text{Encap}^*(\text{ID})$, so that $\text{ct}_{\text{ID}} = (C, D, E_1, ..., E_{n+1})$ for $C = g_1^r$, $D = e(h_1, h_2)^{r'}$, and for $i \in [n+1]$, $E_i = g_T^{x_i^*} e(k_1, g_2^{f(\text{ID}||i)})^r$ where $r \neq r'$. Then for any secret key $\text{sk}_{(\text{ID},\boldsymbol{y})} = (d_1, d_2, z)$, which is generated by $\Pi_1.\text{KeyGen}(\text{msk}, \text{ID}, \boldsymbol{y})$ with $\boldsymbol{y} \in \{\boldsymbol{y}_i\}_{i=1}^n$. Then we can get:

$$
\begin{aligned}
&\Pi_1.\text{Decap}(\text{ct}_{\text{ID}}, \text{sk}_{(\text{ID},\boldsymbol{y})}) = \Pi.\text{Decrypt}(\text{ct}_{\text{ID}}, \text{sk}_{(\text{ID},\boldsymbol{y})}) \\
&= \log_{g_T}\left(e(C, d_1) \cdot D^{d_2} \cdot (\prod_{i=1}^{n+1} E_i^{y_i^*})\right) \\
&= \log_{g_T}\left(e(g_1^r, (\prod_{i=1}^{n+1} g_2^{f(\text{ID}||i)y_i^*})^{-s}) \cdot e(h_1, h_2)^{r'(-t\sum_{i=1}^{n+1} h(\text{ID}||i)y_i^*)} \cdot (\prod_{i=1}^{n+1} E_i^{y_i^*})\right) \\
&= \log_{g_T}\left(g_T^{-v(sr+twr')(\sum_{i=1}^{n+1} h(\text{ID}||i)y_i^*)} \cdot g_T^{\sum_{i=1}^{n+1} x_i^* y_i^*} \cdot g_T^{rv(s+wt)(\sum_{i=1}^{n+1} h(\text{ID}||i)y_i^*))}\right) \\
&= \log_{g_T}\left(g_T^{\langle \boldsymbol{x}, \boldsymbol{y}\rangle} \cdot g_T^{wvt(r-r')(\sum_{i=1}^{n+1} h(\text{ID}||i)y_i^*))}\right) \\
&= \langle \boldsymbol{k}, \boldsymbol{y}\rangle + wvt(r-r')\sum_{i=1}^{n} h(\text{ID}||i)y_i + wvt(r-r')h(\text{ID}||(n+1))z\ .
\end{aligned}
\tag{1}
$$

From $r \neq r'$, we can get that if $\Pi_1.\text{Decap}(\text{ct}_{\text{ID}}, \text{sk}_{(\text{ID},\boldsymbol{y})}) = \Pi_1.\text{Decap}(\text{ct}_{\text{ID}}, \text{sk}'_{(\text{ID},\boldsymbol{y})})$ and $\text{sk}_{(\text{ID},\boldsymbol{y})}, \text{sk}'_{(\text{ID},\boldsymbol{y})}$ are outputs of $\Pi_1.\text{KeyGen}(\text{msk}, \text{ID}, \boldsymbol{y})$, then $\text{sk}_{(\text{ID},\boldsymbol{y})} = \text{sk}'_{(\text{ID},\boldsymbol{y})}$. This implies 0-universality of the decapsulation function.

4.2 leakage-smooth IBIP-HPS

In this section, we show how to construct an l'-leakage-smooth IBIP-HPS, for arbitrarily large values of l', meeting the efficiency requirements of the BRM. We start with the IBIP-HPS scheme $\Pi_1 = (\Pi_1.\text{Setup}, \Pi_1.\text{KeyGen}, \Pi_1.\text{Encap}, \Pi_1.\text{Encap}^*, \Pi_1.\text{Decap})$ and compile it into a new IBIP-HPS scheme $\Pi_2 = (\Pi_2.\text{Setup}, \Pi_2.\text{KeyGen}, \Pi_2.\text{Encap}, \Pi_2.\text{Encap}^*, \Pi_2.\text{Decap})$.

Before showing our construction, we introduce our main idea of leakage amplification. In order to tolerate arbitrarily large leakage l', we introduce a key-size parameter m and map each secret key in Π_2 into m secret keys generated by Π_1. And m will depend on the desired leakage parameter l'. Recall that the encapsulated key $\boldsymbol{k}$ in the definition of leakage-smoothness is recovered by n secret keys for n linear independent vectors. In the proof of leakage-smoothness, we need to use the same random numbers to generate the n secret keys for each $\boldsymbol{y}_i$. Thus, the key generation algorithm will take n linear independent vectors (an invertible matrix Y) as input.

In order to meet the efficiency requirements, the encapsulation and decapsulation algorithms should be independent of m. So they will target only a small subset of t-out-of-m of the secret keys. The encapsulation algorithm will choose t

random indices from $\{1, ..., m\}$ to generate t ciphertexts. And the decapsulation algorithm will read only the corresponding t secret keys. Finally, since the key generation algorithm will output n secret keys for n vectors and one identity ID', the encapsulation algorithm will also run n times to get n ciphertexts. These n ciphertexts share one identity ID and one encapsulated key $\boldsymbol{k}$. If $\mathrm{ID} = \mathrm{ID}'$, then the i-th ciphertext can be decapsulated by the i-th secret key.

IBIP-HPS construction. Let $\Pi_1 = (\Pi_1.\mathrm{Setup}, \Pi_1.\mathrm{KeyGen}, \Pi_1.\mathrm{Encap}, \Pi_1.\mathrm{Encap}^*, \Pi_1.\mathrm{Decap})$ be an IBIP-HPS with identity space $\mathcal{ID}_1$, vector space $\mathcal{V}$ and secret key randomness space $\mathcal{R}$.

For simplicity, we use $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})} = \Pi_1.\mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, z')$ to denote that $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})}$ is generated by computing $\Pi_1.\mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y})$ with random value z' (setting $z = z'$ in $\Pi_1.\mathrm{KeyGen}$), and use $(\mathrm{ct}_{\mathrm{ID}}, \boldsymbol{k}) = \Pi_1.\mathrm{Encap}(\mathrm{ID}, \boldsymbol{x}')$ to denote that $(\mathrm{ct}_{\mathrm{ID}}, \boldsymbol{k})$ is generated by computing $\Pi_1.\mathrm{Encap}(\mathrm{ID})$ with vector $\boldsymbol{x}'$ (setting $\boldsymbol{x} = \boldsymbol{x}'$ in $\Pi_1.\mathrm{Encap}$).
Let $H : \mathcal{ID}_2 \times [m] \times [n] \to \mathcal{ID}_1$ be a one-to-one function for some set $\mathcal{ID}_2$.
Define $\Pi_2 = (\Pi_2.\mathrm{Setup}, \Pi_2.\mathrm{KeyGen}, \Pi_2.\mathrm{Encap}, \Pi_2.\mathrm{Encap}^*, \Pi_2.\mathrm{Decap})$ with identity space $\mathcal{ID}_2$ as follows:

- $\mathrm{Setup}(1^\lambda, 1^n)$: This is the same as $\Pi_1.\mathrm{Setup}$.
- $\mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, Y)$: Prase $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n]$. Sample $z_1, ..., z_m \leftarrow_R \mathcal{R}$. For $i \in [n], j \in [m]$, $\mathrm{sk}_{i,j}$ is generated by computing $\Pi_1.\mathrm{KeyGen}(\mathrm{msk}, H(\mathrm{ID}, j, i), \boldsymbol{y}_i, z_j)$. Set $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_i)} = (\mathrm{sk}_{i,1}, ..., \mathrm{sk}_{i,m})$ for $i \in [n]$. It outputs
$$\mathrm{sk}_{(\mathrm{ID},Y)} := (\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_1)}, ..., \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_n)}) \ .$$
- $\mathrm{Encap}(\mathrm{ID})$: First it samples a vector $\boldsymbol{k} = (k_1, ..., k_n) \in \mathcal{V}$. This algorithm will run the following steps for n times. In step τ:
(1) Choose t random indices $\boldsymbol{\alpha}[\tau] = (\alpha_1[\tau], ..., \alpha_t[\tau]) \leftarrow [m]^t$, and $\boldsymbol{\beta}[\tau] = (\beta_1[\tau], ..., \beta_t[\tau]) \leftarrow \mathbb{Z}_p^t$.
(2) For $j \in [t]$, sample $\boldsymbol{k}_j^*[\tau] = (k_{j1}^*[\tau], ..., k_{jn}^*[\tau]) \leftarrow \mathcal{V}$, s.t. $\sum_{j=1}^t \beta_j[\tau] k_{ji}^*[\tau] = k_i, i \in [n]$.
(3) For $j \in [t]$, compute $(\mathrm{ct}_j[\tau], \boldsymbol{k}_j^*[\tau]) \leftarrow \Pi_1.\mathrm{Encap}(H(\mathrm{ID}, \alpha_j[\tau], \tau), \boldsymbol{k}_j^*[\tau])$.
It outputs
$$\mathrm{ct}_{\mathrm{ID}} := (\{\mathrm{ct}_1[\tau], ..., \mathrm{ct}_t[\tau], \boldsymbol{\alpha}[\tau], \boldsymbol{\beta}[\tau]\}_{\tau=1}^n), \boldsymbol{k} \ .$$
For simplicity, we just omit the subscription ID in each term of $\mathrm{ct}_{\mathrm{ID}}$, when we only consider one specified ID.
- $\mathrm{Encap}^*(\mathrm{ID})$: First it samples a vector $\boldsymbol{k} = (k_1, ..., k_n) \in \mathcal{V}$. This algorithm will run the following steps for n times. In step τ:
(1) Choose t random indices $\boldsymbol{\alpha}[\tau] = (\alpha_1[\tau], ..., \alpha_t[\tau]) \leftarrow [m]^t$, and $\boldsymbol{\beta}[\tau] = (\beta_1[\tau], ..., \beta_t[\tau]) \leftarrow \mathbb{Z}_p^t$.
(2) For $j \in [t]$, sample $\boldsymbol{k}_j^*[\tau] = (k_{j1}^*[\tau], ..., k_{jn}^*[\tau]) \leftarrow \mathcal{V}$, s.t. $\sum_{j=1}^t \beta_j[\tau] k_{ji}^*[\tau] = k_i, i \in [n]$.
(3) For $j \in [t]$, compute $\mathrm{ct}_j[\tau] \leftarrow \Pi_1.\mathrm{Encap}^*(H(\mathrm{ID}, \alpha_j[\tau], \tau), \boldsymbol{k}_j^*[\tau])$.
It outputs
$$\mathrm{ct}_{\mathrm{ID}} := (\{\mathrm{ct}_1[\tau], ..., \mathrm{ct}_t[\tau], \boldsymbol{\alpha}[\tau], \boldsymbol{\beta}[\tau]\}_{\tau=1}^n) \ .$$

– $\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_\tau)})$: For any $\tau \in [n]$, prase $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_\tau)} = (\mathrm{sk}_{\tau,1}, ..., \mathrm{sk}_{\tau,m})$. For $j \in [t]$, compute $\gamma_j[\tau] := \Pi_1.\mathrm{Decap}(\mathrm{ct}_j[\tau], \mathrm{sk}_{\tau,\alpha_j[\tau]})$. Then, it outputs $\sum_{j=1}^{t}(\beta_j[\tau]\gamma_j[\tau])$.

The correctness and valid/invalid ciphertext indistinguishability of Π_2 can be easily extended from such properties of Π_1, and they are shown in Supporting material C.1.

Efficiency. We will show that Π_2 meets the efficiency requirements of the BRM by giving a lower bound of t, which is independent of the leakage bound l' in the proof of the leakage-smoothness.

leakage-smoothness. Firstly, in Supporting material C.2, we show that the decapsulation algorithm of Π_2 is not 0-universal, so we cannot directly obtain the leakage-smoothness by the *leftover-hash lemma*(Lemma 2).

We try to relax the condition in the definition of universality. We use a similar idea as in [5], where we only insist that $\mathrm{sk}_{(\mathrm{ID},Y)}$ and $\mathrm{sk}'_{(\mathrm{ID},Y)}$ are different enough so that they are unlikely to result in the same $\boldsymbol{k}$. More precisely, we state the following theorem, and will prove it in Supporting material C.3.

Theorem 5. *For any $\varepsilon > 0$, there exists some setting of $\eta = O(\log p)$, so that 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 IBIPFE in the BRM

Here, we define the security for an IBIPFE scheme, which is resistant to key leakage attacks in the Bounded-retrieval model (BRM). Then we show how to construct such an IBIPFE scheme from an leakage-smooth IBIP-HPS we presented in the above section. 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, 39], 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

Recall that in our leakage-smooth IBIP-HPS scheme, the encapsulation algorithm will output n ciphertexts sharing the same encapsulated key $\boldsymbol{k}$. The key generation algorithm will also output n secret keys for n vectors, and the i-th ciphertext can be decapsulated by the i-th secret key. But the input of IBIPFE's key generation algorithm contains only one vector, instead of n vectors, so it cannot output n secret keys in one round. And actually we only need one secret key to get the decryption result $\langle \boldsymbol{k}, \boldsymbol{y} \rangle$ since all n ciphertexts share the same encapsulated key $\boldsymbol{k}$. Therefore, in our leakage-resilient IBIPFE scheme, we will allow the user to choose an index $\tau \in [n]$ in the key generation algorithm to indicate which ciphertext it wants to decrypt with this secret key. It means that

the syntax of Setup, Encrypt, Decrypt is the same as that in IBIPFE (Section 3.1), and KeyGen has the following syntax:
$\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, \tau)$: The key generation algorithm generates the secret key $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)}$, which can decrypt the τ-th ciphertext output by $\mathrm{Encrypt}(\mathrm{ID}, \boldsymbol{x})$.

Indistinguishable security with key 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 4. The advantage of an adversary $\mathcal{A}$ in the security game is $Adv^{\text{IBIPFE-IND}}_{\text{IBIPFE},\mathcal{A}}(\lambda, l) := |\mathbf{Pr}[\mathcal{A}\ wins] - \frac{1}{2}|$.

Table 4: IBIPFE-IND(λ, n)

Setup: The challenger $\mathcal{C}$ computes $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{Setup}(1^\lambda, 1^n)$ and sends mpk to the adversary $\mathcal{A}$. The challenger keeps a list $\mathcal{R}_{\mathrm{ID}}$ to store the random numbers which is sampled in *leakage-query* stage.
Query 1: The adversary $\mathcal{A}$ can adaptively ask the challenger for:
Secret key query: On input an identity ID, a vector $\boldsymbol{y}$ and an index τ, the challenger replies with $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, \tau)$.
Leakage query: On input an identity ID, a vector $\boldsymbol{y}$ and a PPT function f^*, if there is no tuple $(\mathrm{ID}, \boldsymbol{y}, \cdot)$ has been queried to *leakge query* before, the challenger first check whether ID is in the list $\mathcal{R}_{\mathrm{ID}}$:

- If it is not, then the challenger first runs $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},1)} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, 1)$, then store the tuple $(\mathrm{ID}, r, 1)$ in the list $\mathcal{R}_{\mathrm{ID}}$, where r denotes the random numbers sampled by $\mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, 1)$.
- If ID is in $\mathcal{R}_{\mathrm{ID}}$, then the challenger reads and deletes the tuple (ID, r, τ) from $\mathcal{R}_{\mathrm{ID}}$ and generates $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau+1)}$ with the randomness r, and puts $(\mathrm{ID}, r, \tau+1)$ into $\mathcal{R}_{\mathrm{ID}}$.

Then the challenger replies with $f^*(\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\cdot)})$ if $\sum_{f \in \{f'\}_{(\mathrm{ID},\boldsymbol{y})} \cup \{f^*\}} |f(\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\cdot)})| \leq l$, where $\{f'\}_{(\mathrm{ID},\boldsymbol{y})}$ denotes the set of leakage functions that the adversary have queried on the secret key $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\cdot)}$, and $|f(\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\cdot)})|$ is the bit-length of the function value $f(\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\cdot)})$.
Challenge: The adversary $\mathcal{A}$ chooses an challenge identity $\mathrm{ID}^* \in \mathcal{ID}$ and two messages $\boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathcal{V}$ subject to the conditions that (1) ID^* never appeared in any *secret key query* (2) There are at most n different vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ appeared in *leakage query* in the form $(\mathrm{ID}^*, \boldsymbol{y}_i, f^*)$ and these n vectors should be linear independent. The challenger chooses $b \in \{0, 1\}$ uniformly at random and returns $\mathrm{ct} \leftarrow \mathrm{Encrypt}(\mathrm{mpk}, \mathrm{ID}^*, \boldsymbol{x}_b)$ to the adversary.
Query 2: The adversary $\mathcal{A}$ adaptively makes *secret key query* with $(\mathrm{ID}, \boldsymbol{y})$ as long as $\mathrm{ID} \neq \mathrm{ID}^*$
Output: The adversary $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and wins if $b' = b$.

Notes: a) For *secret key queries* in Query 1 and 2, the challenger computes $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, \tau)$ only in the first time that the tuple $(\mathrm{ID}, \boldsymbol{y}, \tau)$ is queried and responds to all future queries on the same tuple $(\mathrm{ID}, \boldsymbol{y}, \tau)$ with the same $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)}$.
b) For *leakage queries* in Query 1, the challenger computes $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)} \leftarrow \mathrm{KeyGen}(\mathrm{msk}, \mathrm{ID}, \boldsymbol{y}, \tau)$ only in the first time that the tuple $(\mathrm{ID}, \boldsymbol{y}, *)$ is queried and responds to all future queries on the tuples $(\mathrm{ID}, \boldsymbol{y}, *)$ with the same secret key.

We now give some explanation about the definition. The restrictions of the definition come from the definitions and proofs of properties of IBIP-HPS. Recall that there are only 3 items in the definition of leakage-smoothness: $\mathrm{ct}, f(\{\mathrm{sk}_{\mathrm{ID},\boldsymbol{y}_i}\}_{i=1}^n), \boldsymbol{k}$. The secret keys $\{\mathrm{sk}_{\mathrm{ID},\boldsymbol{y}_i}\}_{i=1}^n$ used to compute $\boldsymbol{k}$ do not appear in the equation directly. In order to rely our security of Π_3 on the leakage smoothness of Π_2, for the secret keys used to compute $\boldsymbol{k}$ (we need n secret keys of n linear independent vectors to determine a $\boldsymbol{k}$), the adversary can only know a functional value $f(\cdot)$, instead of the secret keys. Thus, we only allow the adversary to perform *leakage queries* on the challenge identity ID^* and arbitrary vector $\boldsymbol{y}$, instead of *secret key queries* on ID^* and $\boldsymbol{y}$ subject to the condition that $\langle \boldsymbol{x}_0, \boldsymbol{y} \rangle = \langle \boldsymbol{x}_1, \boldsymbol{y} \rangle$.

From the definition of valid/invalid ciphertext indistinguishability in IBIP-HPS, the adversary can make one key query for the challenge identity ID^* with an $n \times n$ invertible matrix. So we also restrict that there are at most n different linear independent vectors appearing in the *leakage queries* with the form $(\mathrm{ID}^*, \boldsymbol{y}, f^*)$. And such n vectors mentioned above are generated from the same random numbers and are corresponding to the 1-th,...,n-th ciphertexts respectively.

Remark on stateful key authority. Similar with [5], in the query stage of our security game, some secret keys are computed only once and reused subsequently. In reality, this requires that key authority that issues secret keys is stateful, and caches the secret keys that it computes. As the analysis in [5], this requirement can be overcome easily by adding a pseudo-random function to the master secret key.

Definition 11 (Leakage-resilient IBIPFE). *An IBIPFE 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 (defined in Table 4) is $Adv^{IBIPFE\text{-}IND}_{IBIPFE,\mathcal{A}}(\lambda, n, l) = \mathrm{negl}(\lambda)$. We define the leakage ratio of the scheme to be $\mu = \frac{l}{|\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)}|}$, where $|\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)}|$ is the number of bits needed to efficiently store a secret key $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y},\tau)}$.*

Definition 12 (leakage-resilient IBIPFE in the BRM). *We say that an IBIPFE scheme is adaptively leakage-resilient in the bounded-retrieval model (BRM), if the scheme is adaptive-secure, 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* $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathrm{KeyGen}(1^\lambda, 1^n, 1^l)$, $\boldsymbol{x} \in \mathcal{V}, \mathrm{ct}_{\boldsymbol{x}} \leftarrow \mathrm{Encrypt}(\mathrm{mpk}, \boldsymbol{x})$*, the scheme satisfies:*

- *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* $|\mathrm{ct}_{(|\mathrm{ID}|,\boldsymbol{x})}| \leq O(\mathrm{ctsize}(\lambda, |\mathrm{ID}|, |\boldsymbol{x}|))$.
- *Run-time of* $\mathrm{Encrypt}(\mathrm{mpk}, \mathrm{ID}, \boldsymbol{x})$ *is* $\leq O(\mathrm{encTime}(\lambda, |\mathrm{ID}|, |\boldsymbol{x}|))$.
- *Run-time of* $\mathrm{Decrypt}(\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})}, \mathrm{ct}_{(\mathrm{ID},\boldsymbol{x})})$*, and the number of bits of* $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})}$ *accessed, are* $\leq O(\mathrm{encTime}(\lambda, |\mathrm{ID}|, |\boldsymbol{x}|))$.

5.2 Construction of Leakage-resilient IBFE for inner-product

Given an *leakage-smooth* IBIP-HPS scheme $\Pi_2 = (\text{Setup}, \text{KeyGen}, \text{Encap}, \text{Encap}^*, \text{Decap})$ where the vector space is $\mathcal{V}$, we construct a leakage-resilient IBIPFE scheme with the same vector space $\mathcal{V}$. We show our construction in Table 5.

Table 5: Leakage-resilient IBIPFE scheme Π_3.

$\text{Setup}(1^\lambda, 1^n)$: The Setup procedure is the same as Π_2.Setup.
$\text{KeyGen}(\text{msk}, \text{ID}, \boldsymbol{y}, \tau)$: 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 $(\text{sk}_{(\text{ID},\boldsymbol{y}_1)}, ..., \text{sk}_{(\text{ID},\boldsymbol{y}_n)}) \leftarrow \Pi_2.\text{KeyGen}(\text{msk}, \text{ID}, Y)$ and returns $\text{sk}_{(\text{ID},\boldsymbol{y},\tau)} = \text{sk}_{(\text{ID},\boldsymbol{y}_\tau)}$.
$\text{Encrypt}(\text{ID}, \boldsymbol{x})$: It computes $(\text{ct}_{\text{ID}}, \boldsymbol{k}) \leftarrow \Pi_2.\text{Encap}(\text{ID})$. It sets $c_1 = \text{ct}_{\text{ID}}$ and $c_2 = \boldsymbol{k} + \boldsymbol{x}$. And output $\text{ct}_{\text{ID},\boldsymbol{x}} = (c_1, c_2)$.
$\text{Decrypt}(\text{ct}_{(\text{ID},\boldsymbol{x})}, \text{sk}_{(ID,\boldsymbol{y})})$: Parse $\text{ct}_{(ID,\boldsymbol{x})} = (c_1, c_2)$ and output $\boldsymbol{y} \cdot c_2 - \Pi_2.\text{Decap}(\text{ct}_{\text{ID}}, \text{sk}_{(\text{ID},\boldsymbol{y},\tau)})$.

Theorem 6. *If we start with an l'-leakage-smooth IBIP-HPS Π_2, then the construction in Table 5 yields a $l = \frac{l'}{n}$-leakage-resilient IBIPFE.*

We use a series of games argument in our security analysis, which begins with the real security game and ends with a game whose challenge ciphertext is independent of the bit b chosen by the challenger. And these games are indistinguishable from each other. The details of proof of Theorem 6 are shown in Supporting material D.

Theorem 7. *Using the l'-leakage-smooth IBIP-HPS construction Π_2, we can get an l-leakage-resilient IBIPFE scheme in the BRM with vector space $\mathcal{V} = \mathbb{Z}_p^n$ and it satisfies:*
(1) The master public-key size and the master secret-key size are the same as in Π_2, and are independent of l.
(2) The size of the ciphertext and the number of secret-key bits read by decryption are the same as in Π_2, both of which are independent of l.
(3) Encryption time consists of the Encap *time of Π_2 and the time of one vector addition 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.*
(4) The leakage ratio is $\mu = \frac{1-\varepsilon}{3n}$, for sufficiently large values of the leakage-parameter l.

Proof. The first 3 statements are easy to check from the construction. For the leakage ratio, by Theorem 5, we have $l = \frac{l'}{n} \leq \frac{(1-\varepsilon)m \log p - n \log p - 2\lambda}{n}$. From it we can get the lower bound of m is that $m \geq \frac{l' + n \log p + 2\lambda}{(1-\varepsilon) \log p}$. Then the leakage ratio for a given l is defined as:

$$\mu = \frac{l}{|\text{sk}_{(\text{ID},\boldsymbol{y},\tau)}|} = \frac{l}{3m \log p} = \frac{(1-\varepsilon)l}{3nl + 3n \log p + 6\lambda} .$$

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. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Annual international cryptology conference. pp. 213–229. Springer (2001)
11. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011)
12. 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)
13. 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)
14. 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)
15. 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)
16. 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)

17. Coron, J.S., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of ggh15 multilinear maps. In: Annual Cryptology Conference. pp. 607–628. Springer (2016)
18. 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)
19. 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)
20. 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)
21. 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)
22. 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)
23. Dufour-Sans, E., Pointcheval, D.: Unbounded inner-product functional encryption with succinct keys. In: International Conference on Applied Cryptography and Network Security. pp. 426–441. Springer (2019)
24. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Theory of Cryptography Conference. pp. 207–224. Springer (2006)
25. 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)
26. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 2008 49th Annual IEEE Symposium on Foundations of Computer Science. pp. 293–302. IEEE (2008)
27. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Theory of Cryptography Conference. pp. 343–360. Springer (2010)
28. 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)
29. Gentry, C.: Practical identity-based encryption without random oracles. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 445–464. Springer (2006)
30. 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)
31. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Annual Cryptology Conference. pp. 162–179. Springer (2012)
32. 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)
33. 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)
34. 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)

35. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Annual International Cryptology Conference. pp. 388–397. Springer (1999)
36. 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)
37. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Theory of Cryptography Conference. pp. 70–88. Springer (2011)
38. Micali, S., Reyzin, L.: Physically observable cryptography. In: Theory of Cryptography Conference. pp. 278–296. Springer (2004)
39. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM Journal on Computing **41**(4), 772–814 (2012)
40. Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences **52**(1), 43–52 (1996)
41. 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)
42. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive **2010**, 556 (2010)
43. 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)
44. 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)
45. Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: International Colloquium on Automata, Languages, and Programming. pp. 560–578. Springer (2008)
46. 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)
47. 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)
48. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Annual Cryptology Conference. pp. 678–697. Springer (2015)
49. 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)
50. 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)
51. 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)
52. Zhang, L., Wang, X., Chen, Y., Yiu, S.M.: Leakage-resilient inner-product functional encryption in the bounded-retrieval model

Supporting Material

A Adaptive IND-security of IBIPFE Π (Proof of Theorem 3).

Proof. We have shown the correctness of our scheme, so we only need to show that the advantage of any admissible PPT adversary $\mathcal{A}$ in the indistinguishable security game (defined in Table 2) is negligible.

The proof uses a sequence of games which begins with the real indistinguishable security game and ends with a game where the adversary has no advantage at all. For $i \in \{0, 1, 2\}$, we use W_i to denote the event that the adversary wins in Game i.

Game 0: This is the real game. In this game, the adversary $\mathcal{A}$ is given $\text{mpk} \leftarrow \Pi.\text{Setup}(1^n, 1^\lambda)$. And after the Query 1 stage, $\mathcal{A}$ chooses two vectors $\boldsymbol{x}_0, \boldsymbol{x}_1$ and one challenge identity ID^*. Then $\mathcal{A}$ obtains an encryption of $(\text{ID}^*, \boldsymbol{x}_b)$, for some random $b \leftarrow \{0, 1\}$. For any pair $(\text{ID}, \boldsymbol{y})$ submitted to the secret key query, it must be the case that $F((\text{ID}^*, \boldsymbol{x}_0), (\text{ID}, \boldsymbol{y})) = F((\text{ID}^*, \boldsymbol{x}_1), (\text{ID}, \boldsymbol{y}))$.

Game 1: In this game, we modify the generation of the challenge ciphertext $\text{ct}_{(\text{ID}^*, \boldsymbol{x}_b)} = (C, D, E_1, ..., E_n)$ as follows. The challenger $\mathcal{C}$ first computes

$$C = g_1^r, D = e(h_1, h_2)^r,$$

for randomly chosen $r \leftarrow_R \mathbb{Z}_p$. Then $\mathcal{C}$ uses msk to compute

$$E_i = g_T^{x_i} \cdot e(C^s, g_2^{f(\text{ID}^*||i)}) \cdot D^{th(\text{ID}^*||i)}, i \in [n].$$

It is easy to be verified that the challenge ciphertext $\text{ct}_{(\text{ID}^*, \boldsymbol{x}_b)}$ has the same distribution as in Game 0. So we have $\mathbf{Pr}[W_0] = \mathbf{Pr}[W_1]$.

Game 2: In this game, we modify the generation of the challenge ciphertext again. $\mathcal{C}$ first samples two random number $r, r' \leftarrow_R \mathbb{Z}_p$ and sets

$$C = g_1^r, D = e(h_1, h_2)^{r+r'}, E_i = g_T^{x_i} \cdot e(C^s, g_2^{f(\text{ID}^*||i)}) \cdot D^{th(\text{ID}^*||i)}, i \in [n].$$

Now we show that $|\mathbf{Pr}[W_2] - \mathbf{Pr}[W_1]| \leq \text{Adv}_{\mathcal{C}}^{\text{DBDH}}(\lambda)$, which means that if $\mathcal{A}$ can distinguish between Game 1 and Game 2, then we can construct an adversary $\mathcal{B}$ to break the DBDH assumption. $\mathcal{B}$ receives a DBDH tuple $(g_1^a, g_1^b, g_2^a, g_2^c, g_T^q)$, and its task is to distinguish whether $q = abc$ or q is randomly chosen from $\mathbb{Z}_p$. $\mathcal{B}$ sets $C = g_1^a, h_1 = g_1^b, h_2 = g_2^c, D = g_T^q$ and sends mpk and the challenge ciphertext to $\mathcal{A}$. When answering secret key queries, $\mathcal{B}$ computes d_1 by $d_1 = (\prod_{i=1}^n h_2^{h(\text{ID}||i^{01})y_i})^{-s} = (\prod_{i=1}^n g_2^{f(\text{ID}||i^{01})y_i})^{-s}$. If $\mathcal{A}$ outputs that the ciphertext is generated in Game 1 (which means $D = e(h_1, h_2)^r = e(g_1, g_2)^{abc} = g_T^{abc}$), then $\mathcal{B}$ outputs $q = abc$. Otherwise $\mathcal{B}$ outputs: q is randomly chosen from $\mathbb{Z}_p$.

The last thing we have to prove is that the challenge ciphertext in Game 2 perfectly hides $b \in \{0, 1\}$, so that $\mathbf{Pr}[W_2] = \frac{1}{2}$. Firstly, we have that for $i \in [n]$,

$$\begin{aligned} E_i &= g_T^{x_i} \cdot e(C^s, g_2^{f(\text{ID}^*||i)}) \cdot D^{th(\text{ID}^*||i)} \\ &= g_T^{x_i} \cdot g_T^{rsvh(\text{ID}^*||i)} \cdot g_T^{(r+r')wvth(\text{ID}^*||i)} \end{aligned}$$

$$= g_T^{x_i + r'wvth(\mathrm{ID}^*||i)} \cdot g_T^{rv(s+wt)h(\mathrm{ID}^*||i)}.$$

This means that a PPT adversary can only infer $\boldsymbol{z}_b = \boldsymbol{x}_b + r'wvt\boldsymbol{h}_{\mathrm{ID}^*}$, where $\boldsymbol{h}_{\mathrm{ID}^*} = (h(\mathrm{ID}^*||1), ..., h(\mathrm{ID}^*||n))$ from the challenge ciphertext in Game 2.

We define $\boldsymbol{x} = (\boldsymbol{x}_0 - \boldsymbol{x}_1) \mod p$ and deterministically generate a $\mathbb{Z}_p$-basis $Y_t \in \mathbb{Z}_p^{n\times(n-1)}$ of the $(n-1)$-dimensional subspace $\boldsymbol{x}^\perp = \{\boldsymbol{y} \in \mathbb{Z}_p^n | \langle \boldsymbol{x}, \boldsymbol{y}\rangle = 0 \mod p\}$. We define the invertible matrix

$$Y = [Y_t, \boldsymbol{x'}] \in \mathbb{Z}_p^{n\times n},$$

where $\boldsymbol{x'}$ is a vector outside the subspace $\boldsymbol{x}^\perp$ which we also choose in a deterministic way. Since all the columns of matrix Y are deterministically generated from $\boldsymbol{x} \in \mathbb{Z}_p^n$, they are known to $\mathcal{A}$. Thus, it suffices to prove that $Y^T \cdot \boldsymbol{z}_b \in \mathbb{Z}_p^n$ is information-theoretically independent of $b \in \{0,1\}$ to prove that $\boldsymbol{z}_b$ does not leak anything about the value of b either. The first $n-1$ elements of $Y^T \cdot \boldsymbol{z}_b$ are clearly independent of b since we have $Y_t^T \boldsymbol{x}_0 = Y_t^T \boldsymbol{x}_1$ by construction. Now, we are left with $\left[\langle \boldsymbol{x}_b, \boldsymbol{x'}\rangle + r'wvt\langle \boldsymbol{h}_{\mathrm{ID}^*}, \boldsymbol{x'}\rangle\right] \mod p$.

Let $(s_0, t_0, \boldsymbol{h}^0_{\mathrm{ID}^*}) \in \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p^n$ denote an arbitrary tuple of vectors satisfying $k_1 = g_1^{s_0} h_1^{t_0}$, and

$$\mathrm{sk}_{(\mathrm{ID}^*, \boldsymbol{y}_i)} = \left(d_1 := (\prod_{j=1}^{n} g_2^{vh^0(\mathrm{ID}^*||j)y_{ij}})^{-s_0}, d_2 := -t_0\big(\sum_{j=1}^{n} h^0(\mathrm{ID}^*||j)y_{ij}\big)\right),$$

for all secret key queries with $\{(\mathrm{ID}^*, \boldsymbol{y}_i)\}_{i=1}^{n-1}$. Since all secret key queries with challenge identity ID^* involve vectors $\boldsymbol{y}_i$ in $\boldsymbol{x}^\perp$, so the distribution of $\boldsymbol{h}_{\mathrm{ID}^*}$ is $\{\boldsymbol{h}^0_{\mathrm{ID}^*} + \mu\boldsymbol{x} \mod p | \mu \in \mathbb{Z}_p\}$ in the adversary's view. Then the conditional distribution of $r'wvt\langle \boldsymbol{h}_{\mathrm{ID}^*}, \boldsymbol{x'}\rangle \mod p$ is

$$\mathcal{D} = \{r'wvt(\langle \boldsymbol{h}^0_{\mathrm{ID}^*}, \boldsymbol{x'}\rangle + \mu\langle \boldsymbol{x}, \boldsymbol{x'}\rangle) \mod p | \mu \in \mathbb{Z}_p\}.$$

Since $\langle \boldsymbol{x}, \boldsymbol{x'}\rangle \neq 0$ by construction, so $\mathcal{D}$ is the uniform distribution over $\mathbb{Z}_p$. Further, since $r' = 0$ only happens with only negligible probability, the term $r'wvt\langle \boldsymbol{h}_{\mathrm{ID}^*}, \boldsymbol{x'}\rangle \mod p$ perfectly hides $\langle \boldsymbol{x'}, \boldsymbol{x}_b\rangle$ in the inner product $\langle \boldsymbol{x'}, \boldsymbol{z}_b\rangle \mod p$.

B Analysis of IBIP-HPS Π_1 (Proof of Theorem 4)

Proof. **Correctness.** From the correctness of Π, when $\mathrm{ID} = \mathrm{ID}'$, we have

$$\begin{aligned}
\Pi_1.\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID}', \boldsymbol{y})}) &= \Pi.\mathrm{Decrypt}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID}', \boldsymbol{y})}) \\
&= \langle \boldsymbol{x}^*, \boldsymbol{y}^*\rangle \\
&= \langle \boldsymbol{x}||0, \boldsymbol{y}||z\rangle \\
&= \langle \boldsymbol{x}, \boldsymbol{y}\rangle = \langle \boldsymbol{k}, \boldsymbol{y}\rangle \ .
\end{aligned}$$

Valid/Invalid ciphertext indistinguishability. We show how to use an adversary $\mathcal{A}$, which can distinguish valid and invalid ciphertexts, to construct an adversary $\mathcal{B}$, which can distinguish whether $q = abc$ or q is randomly chosen in the DBDH assumption.

$\mathcal{B}$ receives a DBDH tuple $(g_1^a, g_1^b, g_2^a, g_2^c, g_T^q)$, and its task is to distinguish whether $q = abc$ or q is randomly chosen from $\mathbb{Z}_p$. $\mathcal{B}$ sets $C = g_1^a, h_1 = g_1^b, h_2 = g_2^c, D = g_T^q$ and sends mpk and the challenge ciphertext to $\mathcal{A}$. If $\mathcal{A}$ outputs $D = e(h_1, h_2)^r = e(g_1, g_2)^{abc} = g_T^{abc}$ (which means it is a valid ciphertext), then $\mathcal{B}$ outputs $q = abc$. Otherwise $\mathcal{B}$ outputs that q is randomly chosen from $\mathbb{Z}_p$.

C Analysis of leakage-smooth IBIP-HPS Π_2

C.1 Correctness and valid/invalid ciphertext indistinguishability

Correctness. For the correctness, for any $\mathrm{ID} \in \mathcal{ID}_2$, $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n] \in \mathcal{V}$, $\boldsymbol{y} \in \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$ and any correctly generated $\mathrm{mpk}, \mathrm{msk}, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y})}$, if a ciphertext $\mathrm{ct}_{\mathrm{ID}} = (\{\mathrm{ct}_1[\tau], ..., \mathrm{ct}_t[\tau], \boldsymbol{\alpha}[\tau], \boldsymbol{\beta}[\tau]\}_{\tau=1}^n)$ is generated by Π_2.Encap(ID), then we have: For any $\tau \in [n]$:

$$\begin{aligned}
\Pi_2.\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}[\tau], \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_\tau)}) &= \sum_{j=1}^{t} (\beta_j[\tau]\gamma_j[\tau]) \\
&= \sum_{j=1}^{t} \left(\beta_j[\tau] \sum_{i=1}^{n} k_{ji}^*[\tau] y_{\tau,i} \right) \\
&= \sum_{j=1}^{t} \sum_{i=1}^{n} y_{\tau,i} \beta_j[\tau] k_{ji}^*[\tau] \\
&= \sum_{i=1}^{n} y_{\tau,i} \sum_{j=1}^{t} \beta_j[\tau] k_{ji}^*[\tau] \\
&= \sum_{i=1}^{n} y_{\tau,i} k_i = \langle \boldsymbol{y}_\tau, \boldsymbol{k} \rangle \ .
\end{aligned}$$

Valid/Invalid ciphertext indistinguishability. When talking about the valid/invalid ciphertext indistinguishability of Π_2, we edit the definition of admissible adversary in Definition 9. The input of Π_2.KeyGen is an $n \times n$ matrix instead of a vector with length n. So here we allow the adversary to make key query for the challenge identity with one $n \times n$ invertible matrix, instead of at most n linear independent vectors. Thus, the valid/invalid ciphertext indistinguishability of Π_2 can be easily extended from the valid/invalid ciphertext indistinguishability of Π_1.

C.2 Universality of the decapsulation function in Π_2.

Here, we consider the universality of decapsulation function in Π_2. Follow the same routine as Equation (1), for an invalid $\mathrm{ct}_{\mathrm{ID}}$ we can obtain that for each $j \in [t], \tau \in [n]$:

$$\gamma_j[\tau] = \langle \boldsymbol{k}_j^*, \boldsymbol{y}_\tau \rangle + wvt(r_j[\tau] - r'_j[\tau]) \sum_{i=1}^{n} u[H(\mathrm{ID}, \alpha_j[\tau], \tau)||i] y_{\tau,i} + wvt(r_j[\tau] - r'_j[\tau]) u[H(\mathrm{ID}, \alpha_j[\tau], \tau)||(n+1)] z_{\alpha_j[\tau]} \ .$$

Here $r_j[\tau]$ and $r'_j[\tau]$ are from the invalid ciphertext $\mathrm{ct}_j[\tau]$, and $y_{\tau,i}$ is the i-th term of $\boldsymbol{y}_\tau$. Therefore the result of $\Pi_2.\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_\tau)})$ is:

$$\sum_{j=1}^{t} \beta_j \gamma_j[\tau] = \langle \boldsymbol{k}, \boldsymbol{y}_\tau \rangle + \sum_{j=1}^{t} \beta_j[\tau] \left(wvt(r_j[\tau] - r'_j[\tau]) \sum_{i=1}^{n} u[H(\mathrm{ID}, \alpha_j[\tau], \tau)||i] y_{\tau,i} \right) + \sum_{j=1}^{t} \beta_j[\tau] wvt(r_j[\tau] - r'_j[\tau]) u[H(\mathrm{ID}, \alpha_j[\tau], \tau)||(n+1)] z_{\alpha_j[\tau]} \ .$$

Given invertible $Y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]$ we still have:

$$\boldsymbol{k}^T = [\Pi_2.\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_1)}), \ldots, \Pi_2.\mathrm{Decap}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_n)})] Y^{-1} \ .$$

We can take it as a function of $\mathrm{ct}_{\mathrm{ID}}$ and $\mathrm{sk}_{(\mathrm{ID},Y)}$ and use notation $\boldsymbol{k}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID},Y)})$. Unfortunately, we cannot directly show that the hash family $\mathcal{H} := \{\boldsymbol{k}(\mathrm{ct}_{\mathrm{ID}}, \cdot) | \mathrm{ct}_{\mathrm{ID}} \leftarrow \Pi_2.\mathrm{Encap}^*(\mathrm{ID})\}$ is 0-universal.

In fact, note that for each $i \in [n]$, $\mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_i)}$ shares the same random vector $\boldsymbol{z} = (z_1, \ldots, z_m)$. From our construction, $\mathrm{sk}_{(\mathrm{ID},Y)}$ and $\mathrm{sk}'_{(\mathrm{ID},Y)}$ only differ in $\boldsymbol{z}$ and $\boldsymbol{z}'$. For example, if $\boldsymbol{z}$ and $\boldsymbol{z}'$ only differ in one position, say, α^*, then:

$$\boldsymbol{k}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}_{(\mathrm{ID},Y)}) = \boldsymbol{k}(\mathrm{ct}_{\mathrm{ID}}, \mathrm{sk}'_{(\mathrm{ID},Y)}) \ ,$$

for any $\mathrm{ct}_{\mathrm{ID}}$ that never chooses α^* in any of the vector $\boldsymbol{\alpha}[\tau]$. This will happen with high probability.

C.3 Leakage-smoothness of Π_2 (Proof of Theorem 5)

We first introduce the new notion *approximate universal hashing* and a variant leftover-hash lemma from [6]. Then we prove Theorem 5. Let Σ be some alphabet.

Definition 13 (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 $d_H(x, x') \leq \delta m$ we have $\mathbf{Pr}_{h \leftarrow \mathcal{H}}[h(x) = h(x')] \leq \tau$, where $d_H(\cdot, \cdot)$ is the Hamming metric.*

Theorem 8 (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:*

$$\mathbf{SD}\Big((h, Z, h(X), (h, Z, U_\Gamma))\Big) \leq \frac{1}{2}\sqrt{2^{H_q(\delta)m\log(q)+v-\beta'} + \tau 2^v - 1}$$

where H_q is q-ary Shannon entropy function. In particular, the statistical distance above is at most ϵ as long as:

$$\beta' \geq H_q(\delta)m\log q + v + 2\log\frac{1}{\epsilon} - 1, \text{and } \tau \leq \frac{1}{2^v}(1+\epsilon^2)\ ,$$

where $H_q(x) := x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x)$ 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 $c := \mathrm{ct}_{\mathrm{ID}} = (\{\mathrm{ct}_1[\tau], ..., \mathrm{ct}_t[\tau], \boldsymbol{\alpha}[\tau], \boldsymbol{\beta}[\tau]\}_{\tau=1}^n) \leftarrow \Pi_2.\mathrm{Encap} * (\mathrm{ID})$, $x := \mathrm{sk}_{(\mathrm{ID},Y)}$, and $x_i := \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_i)} := (\mathrm{sk}_{i,1}, \ldots, \mathrm{sk}_{i,m})$ is a sample of secret key for $\boldsymbol{y_i}$ in Π_2. Let $x_{i,j} := \mathrm{sk}_{i,j}$, $j \in [m]$. Then

$$\boldsymbol{k}^T(c,x) = [\Pi_2.\mathrm{Decap}(c,x_1), \ldots, \Pi_2.\mathrm{Decap}(c,x_n)]Y^{-1}\ .$$

$$\Pi_2.\mathrm{Decap}(c,x_i) = \sum_{j=1}^{t}\beta_j[i]\Pi_1.\mathrm{Decap}(\mathrm{ct}_j[i], x_{i,\alpha_j[i]})\ .$$

Let $\mathcal{F}_{j,i} : \mathbb{Z}_p \to \mathbb{Z}_p$ be a hash function family

$$\left\{ f_{\mathrm{ct}_j[i]}(\cdot) := \Pi_1.\mathrm{Decap}(\mathrm{ct}_j[i], \cdot) \middle| \mathrm{ct}_j[i] \leftarrow \Pi_1.\mathrm{Encap}(H(\mathrm{ID}, \alpha_j[i], i), \boldsymbol{k}_j^*[i]) \right\}\ .$$

In Section 3 we already show that the family $\mathcal{F}_{j,i}$ is 0-universal.

Further, set $g_{\boldsymbol{\beta}} : \mathbb{Z}_p^t \to \mathbb{Z}_p$, $g_{\boldsymbol{\beta}}(\boldsymbol{d}) = \langle\boldsymbol{\beta}, \boldsymbol{d}\rangle$. The family $\mathcal{G} := \{g_\beta | \beta \leftarrow \mathbb{Z}_p^\eta\}$, and it's $\frac{1}{p}$-universal.

Now we write $k^T(c,x)$ as a hash function

$$\begin{aligned} h_c(x) = (&g_{\boldsymbol{\beta}[1]}(f_{\mathrm{ct}_1[1]}(x_{1,\alpha_1[1]}), ..., f_{\mathrm{ct}_t[1]}(x_{1,\alpha_t[1]})), ..., \\ &g_{\boldsymbol{\beta}[n]}(f_{\mathrm{ct}_1[n]}(x_{n,\alpha_1[n]}), ..., f_{\mathrm{ct}_t[n]}(x_{n,\alpha_t[n]})))Y^{-1}\ . \end{aligned}$$

Let $\Phi := \{h_c(\cdot) | c \leftarrow \Pi_2.\mathrm{Encap}^*(\mathrm{ID})\}$. 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 $\boldsymbol{z} \in \mathbb{Z}_p^m$. Firstly, we show that the family Φ is approximately universal in the following lemma.

Lemma 3. *Let $\mathcal{F}$ be a family of ρ-universal hash functions and $\mathcal{G}$ be a family of ρ'-universal hash functions, then the above family Φ is (δ,ϕ)-approximately universal for any $\delta > 0$ and $\phi \leq ((1-\delta)^t + \rho')^n$.*

Proof. For any $x, x' \in \mathbb{Z}_p^n$, where $d_H(x, x') \leq \delta m$, we have

$$
\begin{aligned}
&\mathbf{Pr}_{h_c \leftarrow \Phi}[h_c(x) = h_c(x')] \\
&= \prod_{i=1}^{n} \mathbf{Pr}_{h_c \leftarrow \Phi}\left[g_{\boldsymbol{\beta}_i}\left(f_{\mathrm{ct}_1[i]}(x_{i,\alpha_1[i]}), ..., f_{\mathrm{ct}_t[i]}(x_{i,\alpha_t[i]}) \right) = g_{\boldsymbol{\beta}_i}\left(f_{\mathrm{ct}_1[i]}(x'_{i,\alpha_1[i]}), ..., f_{\mathrm{ct}_t[i]}(x'_{i,\alpha_t[i]}) \right) \right] \\
&\leq \prod_{i=1}^{n} \left(\mathbf{Pr}\left[\left(f_{\mathrm{ct}_1[i]}(x_{i,\alpha_1[i]}), ..., f_{\mathrm{ct}_t[i]}(x_{i,\alpha_t[i]}) \right) = \left(f_{\mathrm{ct}_1[i]}(x'_{i,\alpha_1[i]}), ..., f_{\mathrm{ct}_t[i]}(x'_{i,\alpha_t[i]}) \right) \right] + \rho' \right) \\
&\leq \prod_{i=1}^{n} \left(\sum_{j=0}^{t} \mathbf{Pr}\left[d_H\big((x_{i,\alpha_1[i]}, ..., x_{i,\alpha_t[i]}), (x'_{i,\alpha_1[i]}, ..., x'_{i,\alpha_t[i]}) = j \big) \right] \rho^j + \rho' \right) \\
&\leq \prod_{i=1}^{n} (\sum_{j=0}^{t} (C_t^j \delta^j (1-\delta)^{t-j} \rho^j) + \rho') \\
&\leq \left[(1 - \delta(1-\rho))^t + \rho' \right]^n
\end{aligned}
$$

From the constructions of Π_1 and Π_2, we can know that $\rho = 0, \rho' = \frac{1}{p}$, so we can get $\phi \leq ((1-\delta)^t + \frac{1}{p})^n$. From Theorem 8, in order to ensure that $\mathbf{SD}\left((c, f(\mathrm{sk}_{(\mathrm{ID},Y)}), \boldsymbol{k}), (c, f(\mathrm{sk}_{(\mathrm{ID},Y)}), \boldsymbol{k}')\right) \leq 2^{-\lambda}$, we should have $\phi \leq \frac{1}{p^n}(1 + (2^{-\lambda})^2)$. So we get lower bounds of t and $\beta' := \tilde{\mathbf{H}}_\infty(\mathrm{sk}_{(\mathrm{ID},Y)} | f(\mathrm{sk}_{(\mathrm{ID},Y)}))$ are:

$$
t \geq \frac{\log p - 1}{\log \frac{1}{1-\delta}}, and\ \beta' \geq H_p(\delta) m \log p + n \log p + 2\lambda - 1
$$

.

In our case, $\beta' \geq \mathbf{H}_\infty(\mathrm{sk}_{(\mathrm{ID},Y)}) - l' = m \log p - l'$(Lemma 1). For any constants $\varepsilon > 0$, there exists some constant $c \geq 0$, such that for any $n \geq 1, p \geq 2, t \geq c \log p, m \geq 0$, we have that: If $m \log p - l' \geq \varepsilon m \log p + n \log p + 2\lambda$, then $\mathbf{SD}\left((c, f(\mathrm{sk}_{(\mathrm{ID},Y)}), \boldsymbol{k}), (c, f(\mathrm{sk}_{(\mathrm{ID},Y)}), \boldsymbol{k}')\right) \leq 2^{-\lambda}$. It means that Π_2 is an l'-leakage-smooth IBIP-HPS for $l' = (1-\varepsilon) m \log p - n \log p - 2\lambda$.

D Security of leakage-resilient IBIPFE Π_3(Proof of Theorem 6)

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 $\mathrm{ct}_{(\mathrm{ID}^*, \boldsymbol{x}_b)} \leftarrow \mathrm{Encrypt}(\mathrm{ID}^*, \boldsymbol{x}_b)$ which we parse $\mathrm{ct}_{(\mathrm{ID}^*, \boldsymbol{x}_b)} = (c_1, c_2)$, where $c_1 = \mathrm{ct}_{\mathrm{ID}^*}, c_2 = \boldsymbol{k} + \boldsymbol{x}_b$.
- **Game 1**: We modify the challenge stage, so that the challenger uses the secret keys $\mathrm{sk}_{(\mathrm{ID}^*, \boldsymbol{y}_i, i)}, i \in [\eta], \eta \leq n$ generated by the *leakage query* in Query 1, together with some new keys $\mathrm{sk}_{(\mathrm{ID}^*, \boldsymbol{y}_{\eta+1}, \eta+1)}, ..., \mathrm{sk}_{(\mathrm{ID}^*, \boldsymbol{y}_n, n)}$ generated

by running Π_3.KeyGen$(\mathrm{msk}, \mathrm{ID}^*, \boldsymbol{y}_{\eta+j}, \eta+j), j \in [n-\eta]$ with the same random numbers as $\mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_i,i)}, i \in [\eta]$, where $\boldsymbol{y}_{\eta+j}, j \in [n-\eta]$ are randomly chosen subject to the condition that $Y = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n]$ is an $n \times n$ invertible matrix. It computes $(c_1, \boldsymbol{k}_1) \leftarrow \mathrm{Encap}(\mathrm{ID}^*)$, then finds $\boldsymbol{k}_2$ such that $\boldsymbol{k}_2^T = [\mathrm{Decap}(c_1, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_1,1)}), ..., \mathrm{Decap}(c_1, \mathrm{sk}_{(\mathrm{ID},\boldsymbol{y}_n,n)})]Y^{-1}$ 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 $c_1 \leftarrow \Pi_2.\mathrm{Encap}^*(\mathrm{ID}^*)$, then finds $\boldsymbol{k}_2$ such that $\boldsymbol{k}_2^T = [\mathrm{Decap}(c_1, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_1,1)}), ..., \mathrm{Decap}(c_1, \mathrm{sk}_{(\mathrm{ID}^*,\boldsymbol{y}_n,n)})]Y^{-1}$, and computes 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 IBIP-HPS. Although there is no leakage query in the valid/invalid indistinguishability game, it allows the adversary to learn at most n secret keys for the challenge identity ID^*. The total number of secret keys in the form $\mathrm{sk}_{(\mathrm{ID}^*,\cdot,\cdot)}$ involved in the leakage queries and computation of $\boldsymbol{k}_2$ is also n, and all of these keys were generated by the same random number stored in $\mathcal{R}_{\mathrm{ID}}$. Thus, the indistinguishability between Game 1 and Game 2 holds.

– **Game 3**: The challenge ciphertext $\mathrm{ct}_{(\mathrm{ID},\boldsymbol{x}_b)} = (c_1, c_2)$ is computed by: $c_1 \leftarrow \Pi_2.\mathrm{Encap}^*(\mathrm{ID}^*), c_2 = (c_{2,1}, ..., c_{2,n}) \leftarrow U_{\mathcal{V}}$.

  We claim that Game 2 and Game 3 are statistically indistinguishable by the l'-leakage-smoothness of IBIP-HPS. The only things in Game 2 correlated to $\mathrm{sk}_{(\mathrm{ID}^*,\cdot,\cdot)}$ are outputs of leakage queries and $\boldsymbol{k}_2$. There are at most $l' = l \times n$ bits outputted by the leakage queries for the identity ID^*. And according to the l'-leakage-smoothness of IBIP-HPS, the statistical distance between the two games is negligible. Then, $\boldsymbol{k}_2$ is indistinguishable from choosing a completely independent random variable from $U_{\mathcal{V}}$.

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.