Fully Secure Spatial Encryption under Simple Assumptions with Constant-Size Ciphertexts

Jie Chen Hoon Wei Lim San Ling Huaxiong Wang

Nanyang Technological University, Singapore
s080001@e.ntu.edu.sg;{hoonwei,lingsan,hxwang}@ntu.edu.sg

December 2, 2011

Abstract

In this paper, we propose two new spatial encryption (SE) schemes based on existing inner product encryption (IPE) schemes. Both of our SE schemes are fully secure under simple assumptions and in prime order bilinear groups. Moreover, one of our SE schemes has constant-size ciphertexts. Since SE implies hierarchical identity-based encryption (HIBE), we also obtain a fully secure HIBE scheme with constant-size ciphertexts under simple assumptions. Our second SE scheme is attribute-hiding (or anonymous). It has sizes of public parameters, secret keys and ciphertexts that are quadratically smaller than the currently known SE scheme with similar properties. As a side result, we show that negated SE is equivalent to non-zero IPE. This is somewhat interesting since the latter is known to be a special case of the former.

Keywords: Functional Encryption, Spatial Encryption, Inner Product Encryption.

1 Introduction

The notion of spatial encryption (SE), introduced by Boneh and Hamburg [4], is a generalized concept of identity-based encryption (IBE) [3, 6] that finds a broad range of applications. An SE scheme is defined generically in that plaintexts/ciphertexts are associated with some "policies" and secret (decryption) keys are associated with some "roles". This way, one can embed various cryptosystems, such as hierarchical IBE (HIBE), broadcast IBE, and inclusive/co-inclusive IBE into SE. Consequently, SE can be turned into many useful systems, such as broadcast HIBE, forward-secure IBE, identity-based ring signatures, encryption supporting flexible and sparse products, and encryption providing zero-handshake transport layer security. See [4, 11] for more details of the applications of SE.

1.1 Problem Statement

Informally, in n-dimensional SE, a secret key is associated with an affine space in $\mathbb{Z}_q^n$ for some integer n and some prime q, while a ciphertext is associated with a vector in $\mathbb{Z}_q^n$. A ciphertext can be decrypted by a secret key if and only if the vector for the ciphertext is an element of the affine space for the secret key. This can be represented as a function $\mathcal{F}(\mathcal{S}, (\vec{x}, m)) = m$ in the functional encryption sense [5] for an affine space $\mathcal{S}$, a vector $\vec{x}$ and a message m. Furthermore, one can use a secret key for $\mathcal{S}$ to delegate a key for a relevant subspace $\mathcal{S}'$ of $\mathcal{S}$.

The first SE scheme, proposed by Boneh and Hamburg [4], has short ciphertexts—a very attractive and important property. This is so since any application of the Boneh-Hamburg SE scheme as mentioned above will inherit the same property. However, the Boneh-Hamburg SE scheme is proven to be only selective secure and under complex assumptions. Subsequently, Zhou and Cao [24] proposed a selective secure SE scheme under simple assumptions. However, their scheme no longer produces short ciphertexts but linear in the dimension n. Chen et al. [8] then gave a method of generically constructing SE from hierarchical inner product encryption (HIPE) [17, 13, 18, 20]. As expected, any SE scheme derived using their method preserves properties of the original HIPE scheme. Such a transformation technique is useful, for example, in obtaining attribute-hiding SE schemes (since most existing HIPE scheme are attribute-hiding [17, 13, 20]). Unfortunately, current HIPE schemes do not have short or even constant-size ciphertexts. Further, there are other SE constructions, such as [11, 16], that are fully secure but proven either under more complex assumptions or composite order bilinear groups [9] (which are less efficient than prime order bilinear groups). Summarizing the prior art of SE, to our knowledge, there is currently no fully secure SE scheme under simple assumptions *and* in prime order bilinear groups *with* short or constant-size ciphertexts. The main goal of this paper is, therefore, to construct such a scheme.

We note that, on the other hand, Attrapadung and Libert [1] showed that it is possible to construct an inner product encryption (IPE) scheme with the aforementioned desired properties. IPE, introduced by Katz, Sahai and Waters [12], is another instance of functional encryption where both the secret key and ciphertext are associated with vectors in $\mathbb{Z}_q^n$ for some integer n and prime q. A ciphertext for a vector $\vec{x}$ can be decrypted by a secret key $\mathsf{sk}_{\vec{v}}$ for a vector $\vec{v}$ if and only if the inner product $\vec{x} \cdot \vec{v} = 0$, i.e., $\mathcal{F}(\vec{v}, (\vec{x}, m)) = m$. We note that SE and IPE are somehow related in the sense that it is not difficult to convert SE to IPE. We can achieve this by simply running the key generation algorithm of the SE scheme for the space $\mathcal{S}^{\perp}(\vec{v})$ and setting the resulting key to be a secret key for the vector $\vec{v}$, where $\mathcal{S}^{\perp}(\vec{v})$ is the orthogonal space of $\vec{v}$ (i.e., the Euclidean space spanned by $\vec{x}$ such that

$\vec{x} \cdot \vec{v} = 0$). However, it seems that there does not exist any known generic construction of SE from IPE. This is partly because IPE has no key delegation mechanism. Without key delegation, it is even not clear how one can derive HIBE from IPE (it is, in principle, easier to derive HIBE from IPE in comparison to SE from IPE). One may naturally consider extend IPE to the hierarchical setting, namely HIPE, by adding a delegation mechanism. However, even if the IPE scheme has constant-size ciphertexts, the corresponding HIPE scheme may have ciphertexts with sizes dependent on the depth of the hierarchy. Moreover, constructing SE from HIPE is quadratically expensive as shown in [8].

1.2 Our Contributions

We take an approach along the line of extending IPE to an appropriate "intermediate" form such that constructing SE introduces only minimal cost while retaining all desired properties. Instead of extending IPE to HIPE, we extend IPE to a variant of IPE we call *multi-predicate IPE*, which preserves the sizes of public parameters and ciphertexts of the IPE scheme from which it is derived, while the sizes of secret keys increase by about a factor of $\mathcal{O}(n)$. We then construct an SE scheme from the multi-predicate IPE scheme under linear reduction. Hence, our approach of constructing SE preserves the sizes of public parameters and ciphertexts of the original IPE.

We investigate two IPE schemes of [1, 20] and propose two fully secure SE schemes under simple assumptions and in prime order bilinear groups:

- Our first scheme achieves constant-size ciphertexts. It is based on the Attrapadung-Libert IPE scheme [1]. The corresponding multi-predicate IPE scheme is presented in Section 3.1.

  Particularly, we obtain a non-anonymous HIBE (presented in Section 5.1) with all the mentioned desired properties. Note that there exist a few HIBE schemes [22, 23, 20] that are known to be fully secure under simple assumptions, but none of them has constant-size ciphertexts.

- Our second scheme is based on the Okamoto-Takashima IPE scheme [20]. The corresponding multi-predicate IPE scheme is shown in Section 3.2. It is proven secure under the adaptively attribute-hiding model [12, 20]. Although an SE scheme with similar properties could be derived from the techniques of [8], our construction quadratically reduces the sizes of public parameters, secret keys, and ciphertexts.

We focus on the IPE schemes of [1, 20] because they are more efficient and possess more desired properties than the others.

As explained, given an IPE scheme, one may extend it to HIPE or our proposal of multi-predicate IPE (mIPE), which in turn, is converted to an SE scheme:

$$\text{IPE} \longrightarrow \text{HIPE or mIPE} \longrightarrow \text{SE}$$

Our results show that SE derived from multi-predicate IPE is more compact and efficient than those from HIPE in terms of the sizes of public parameters PP, secret keys sk, ciphertexts c in the number of group elements, and the number # of pairing computation during decryption. These are summarized in Table 1. Here, we assume all IPE, multi-predicate IPE, SE schemes are in $\mathbb{Z}_q^n$ while all HIPE schemes have hierarchy $\vec{\mu} := (n, d; \mu_1, \ldots, \mu_d)$ that is

defined in Appendix C.1. Also, for comparison sake, we extend the Attrapadung-Libert IPE scheme to HIPE, which is presented in Appendix C.2. We note that the sizes of ciphertexts are not constant for the SE scheme derived from this HIPE scheme.

Table 1: Comparisons of SE schemes derived from HIPE and those from mIPE.

	size of PP	size of sk	size of c	# pairings
Attrapadung-Libert IPE [1]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	10	9
HIPE (Appendix C.2)	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(d)$	$\mathcal{O}(d)$
$\downarrow$				
SE	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
mIPE	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	10	$\mathcal{O}(n)$
$\downarrow$				
SE	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	10	$\mathcal{O}(n)$
Okamoto-Takashima IPE [20]	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
HIPE [20]	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
$\downarrow$				
SE	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
mIPE	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
$\downarrow$				
SE	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table 2 gives a summary of the comparisons of existing SE schemes against our schemes. Here, we assume all SE schemes are n-dimensional. We compare their efficiency as well as their properties. The schemes of 'OT* [18]' and 'OT* [20]' are the SE schemes derived from the Okamoto-Takashima HIPE schemes of [18] and [20], respectively, using the transformation techniques of [8].

Table 2: Comparisons between existing and our SE schemes.

Source	Non-Anonymous SE				Anonymous SE	
	BH [4]	ZC [24]	OT* [18]	Our 1^{st} scheme	OT* [20]	Our 2^{nd} scheme
size of PP	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$
size of sk	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$
size of c	3	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	10	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
# pairings	2	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
fully secure	No	No	Yes	Yes	Yes	Yes
anonymous	No	No	No	No	Yes	Yes
constant c	Yes	No	No	Yes	No	No
prime order	Yes	Yes	Yes	Yes	Yes	Yes
simple assumptions	No	Yes	Yes	Yes	Yes	Yes

As a further contribution, we show generic construction of negated SE, i.e., $\mathcal{F}(\mathcal{S}, (\vec{x}, m)) = m$ iff $\vec{x} \notin \mathcal{S}$, from non-zero IPE, i.e., $\mathcal{F}(\vec{v}, (\vec{x}, m)) = m$ iff $\vec{x} \cdot \vec{v} \neq 0$. As before, our transformation preserves the sizes of public parameters and ciphertexts. Interestingly, although non-zero

IPE is known as a special case of negated SE, we prove that negated SE and non-zero IPE are in fact equivalent.

1.3 Our Techniques

We obtain our SE schemes from IPE schemes through some multi-predicate IPE schemes. Informally, in our multi-predicate IPE schemes, a secret key is associated with a set of vectors $\Gamma = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ instead of a single vector in the IPE setting (hence the moniker "multi-predicate"), while a ciphertext is still associated with a single vector $\vec{x}$. Hence, we have $\mathcal{F}(\Gamma, (\vec{x}, m)) = m$ if and only if the inner product $\vec{x} \cdot \vec{v}_i = 0$ for all $i \in [\ell]$.[1] Moreover, we allow key delegation, that is, a secret key for a set of vectors Γ can create a key for a set of vectors Γ' where $\Gamma \subset \Gamma'$. Our definition of multi-predicate IPE is very close to that of the standard IPE. The key difference is that the former delegates secret keys by adding in more vectors while the latter does not. We summarize our multi-predicate IPE schemes, which are fully secure under simple assumptions and in prime order bilinear groups, as follows:

- The first multi-predicate IPE scheme is based on the Attrapadung-Libert IPE scheme, which has constant-size ciphertexts. In our construction, the setup, encryption algorithms are unchanged from the IPE scheme, which make the multi-predicate IPE preserving the size of public parameters and ciphertexts. We employ a secret-sharing technique, which is similar with the technique used in the HIBE schemes of [23, 15]. More precisely, a secret key for $\Gamma = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ is associated with an ℓ-out-of-ℓ secret sharing. When the secret key delegates a new key for $\Gamma' = \{\vec{v}_1, \ldots, \vec{v}_{\ell'}\}$, it changes the secret sharing to an ℓ'-out-of-ℓ' by introducing fresh randomness. Our construction looks rather similar to a combination of the Attrapadung-Libert IPE scheme and the Waters HIBE scheme of [23], except that our delegation mechanism is not hierarchical.

  There are new challenges in our security reduction. This is because of not only our multi-predicate IPE that allows key delegation as opposed to IPE without delegation mechanism, but also the adversary has less query restriction compared to that of HIBE.

  As with the Waters HIBE scheme, our multi-predicate IPE scheme also has the "tag lineage" problem, namely, if we run the delegation algorithm to generate a new secret key, the new key will inherit the previous key's tag values and there is no method for rerandomizing them. However, it is not adequate to adopt the proof techniques for the Waters HIBE scheme to our multi-predicate IPE setting. We introduce a new strategy based on tree structure, which helps us to overcome the tag lineage problem. We discuss this in Section 3.1.

- The second multi-predicate IPE scheme is based on the Okamoto and Takashima IPE scheme [20], which is secure under adaptively attribute-hiding model. In our construction, the setup and encryption algorithms are also unchanged from the IPE scheme. We use a different way to generate a secret key since the structure of this IPE construction is different from the Attrapadung-Libert IPE scheme. To generate a secret key for a vector set $\Gamma = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$, we first mask each vector with a random value and then sum all elements of the masked vectors, namely, we compute a secret key for $\sum_{i=1}^{\ell} \sigma_i \vec{v}_i$, where $\sigma_1, \ldots, \sigma_\ell \xleftarrow{\$} \mathbb{F}_q$. Note that such a construction method for IPE with general delegation has also been mentioned in the full version of [13]. The security can be proved

[1] In this paper, we use $[\ell]$ to denote the set $\{1, \ldots, \ell\}$, and $[\ell_1, \ell_2]$ to denote the set $\{\ell_1, \ell_1 + 1, \ldots, \ell_2\}$.

exactly in the same way as the Okamoto-Takashima HIPE scheme due to such minor modification.

We then adopt techniques, which are similar to HIPE-to-SE transformation of [8], to convert the above multi-predicate IPE schemes into SE schemes. The crux of the transformation from a multi-predicate scheme to an SE scheme is to convert vector inclusion by space relation to inner product relation. This involves two major steps: (i) construct an n-dimensional SE scheme that works in a subset of affine spaces—Euclidean spaces; (ii) use encoding techniques to derive an n-dimensional SE scheme in affine spaces from an $(n+1)$-dimensional SE scheme in Euclidean spaces. We provide further details in Section 4.1.

1.4 Outline of the Paper

We organize the rest of the paper as follows. In Section 2, we provide the definitions and security models of multi-predicate IPE and SE. Moreover, we give the necessary preliminaries for our constructions. In Section 3, we give two concrete constructions of multi-predicate IPE schemes. In Section 4, we then present generic construction of SE from multi-predicate IPE. In Section 5, we discuss how to how to construct a fully secure HIBE scheme under simple assumptions with constant-size ciphertexts and how to derive negated SE from non-zero IPE. We conclude our work in Section 6.

2 Background

In what follows, we borrow the definition and the game-based security model for functional encryption (FE) from [5] which are adequate to define all encryption systems in this paper.

2.1 Functional Encryption

As in [5], we first describe a functionality $\mathcal{F}$ of the syntactic definition of FE. The functionality $\mathcal{F}$ describes the functions of a plaintext that can be learned from the ciphertext:

Definition 1. *A functionality $\mathcal{F}$ defined over $(\mathcal{K}, \mathcal{X})$ is a function $\mathcal{F} : \mathcal{K} \times \mathcal{X} \to \{0,1\}^*$ described as a (deterministic) Turing Machine. The set $\mathcal{K}$ is called the key space and the set $\mathcal{X}$ is called the plaintext space. We require that the key space $\mathcal{K}$ contain a special key called the empty key denoted ϵ.*

An FE scheme for the functionality $\mathcal{F}$ enables one to evaluate $\mathcal{F}(k, x)$ given the encryption of x and a secret key sk_k for k. The algorithm for evaluation $\mathcal{F}(k, x)$ using sk_k is called *decrypt*. More precisely, an FE scheme is defined as follows:

Definition 2. *A functional encryption scheme (FE) for a functionality $\mathcal{F}$ defined over $(\mathcal{K}, \mathcal{X})$ is a tuple of four probabilistic polynomial-time (PPT) algorithms (*Setup*,* KeyGen*,* Enc*,* Dec*) and an additional, but optional PPT algorithm* KeyDel *satisfying the following correctness*

condition for all $k \in \mathcal{K}$ and $x \in \mathcal{X}$:

$(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{Setup}(\lambda)$	(*generate a public and master secret key pair*)
$\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{PP}, \mathsf{MK}, k)$	(*generate a secret key for k*)
$c \leftarrow \mathsf{Enc}(\mathsf{PP}, x)$	(*encrypt plaintext x*)
$y \leftarrow \mathsf{Dec}(\mathsf{PP}, \mathsf{sk}_k, c)$	(*use sk_k to compute $\mathcal{F}(k, x)$ from c*)
$\mathsf{sk}_{k'} \leftarrow \mathsf{KeyDel}(\mathsf{PP}, k, \mathsf{sk}_k, k')$	(*generate a secret key for k' that satisfies a partial order relation denoted as $k' \preceq k$*)

then we require that $y = \mathcal{F}(k, x)$ with probability 1.

The empty key ϵ: The special key ϵ in $\mathcal{K}$ captures all the information about the plaintext that intentionally leaks from the ciphertext. The secret key for ϵ is empty and also denoted by ϵ. Thus, anyone can run $\mathsf{Dec}(\mathsf{PP}, \epsilon, c)$ on a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{PP}, x)$ and obtain all the information about x that intentionally leaks from c. Take IBE for example, $\mathcal{F}(\epsilon, (ID, m))$ outputs only $|m|$ (the length of message m) in the attribute-hiding setting while it outputs $|m|$ and the identity ID in the payload-hiding setting. Henceforth, we assume that every FE scheme contains the empty key ϵ in the key space $\mathcal{K}$ and we will not explicitly mention it.

We now define the security model for FE. For the plaintext pair $(x_{(0)}, x_{(1)})$ of an adversary's choice, we need the following requirement to make the experiment non-trivial:

$$\mathcal{F}(k, x_{(0)}) = \mathcal{F}(k, x_{(1)}) \text{ for all } k \text{ for which the adversary has } \mathsf{sk}_k. \tag{1}$$

Then we define a security game for an FE scheme as follows:

Definition 3. *For $\beta = 0, 1$ define an experiment β for an adversary $\mathcal{A}$ as follows:*

- Setup*: It runs $(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{Setup}(\lambda)$ and gives PP to $\mathcal{A}$.*
- Query*: $\mathcal{A}$ adaptively makes repeated key queries of one of three types:*
  - Generate: *$\mathcal{A}$ submits a key generation query $k \in \mathcal{K}$. The challenger generates a secret key sk_k for k, but does not give it to $\mathcal{A}$. It instead adds the key to the set S and gives the adversary a reference to it.*
  - Delegate: *$\mathcal{A}$ specifies a key sk_k in the set S for $k \in \mathcal{K}$, then it submits a key delegation query for $k' \in \mathcal{K}$, where $k' \preceq k$. The challenger runs the $\mathsf{KeyDel}(\mathsf{PP}, k, \mathsf{sk}_k, k')$ algorithm to get a new secret key $\mathsf{sk}_{k'}$ and adds this to the set S.*
  - Reveal: *$\mathcal{A}$ specifies an element of the set S for a secret key sk_k. The challenger removes the item from the set S and gives $\mathcal{A}$ the secret key. We note at this point there is no need for the challenger to allow more delegate queries on the key since $\mathcal{A}$ can run them itself.*
- Challenge*: $\mathcal{A}$ submits two plaintexts $x_{(0)}$, $x_{(1)} \in \mathcal{X}$ satisfying requirement (1) and in return, it receives $\mathsf{Enc}(\mathsf{PP}, x_{(\beta)})$.*
- Guess*: $\mathcal{A}$ continues to issue key queries as before subject to requirement (1) and eventually outputs a bit in $\{0, 1\}$.*

For $\beta = 0, 1$ let W_β be the event that the adversary outputs 1 in Experiment β and define

$$\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}}(\lambda) := |Pr[W_0] - Pr[W_1]|.$$

Definition 4. *An FE scheme is secure if for all PPT adversaries $\mathcal{A}$ the function $\mathsf{Adv}^{FE}_{\mathcal{A}}(\lambda)$ is negligible.*

2.2 Multi-Predicate Inner Product and Spatial Encryption

In both multi-predicate IPE and SE, a plaintext $x \in \mathcal{X}$ is itself a pair $(\mathsf{ind}, m) \in \mathcal{I} \times \mathcal{M}$ where ind is called an index and m is called the payload message.

In the multi-predicate IPE setting, a functionality $\mathcal{F}$ is defined over a key space and an index space using sets of vectors. In an n-dimensional multi-predicate IPE scheme, the key space $\mathcal{K}$ corresponds to all sets of vectors in the form $\Gamma = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ where $\vec{v}_i \in \mathbb{Z}_q^n$ for all $i \in [\ell]$ (w.l.o.g, we require that Γ is linearly independent and $\ell < n$) and the index space $\mathcal{I}$ corresponds to all vectors $\vec{x}$ in $\mathbb{Z}_q^n$. Hence we have

$$\mathcal{F}(\Gamma, (\vec{x}, m)) := \begin{cases} m & \text{if } \vec{x} \cdot \vec{v}_i = 0 \text{ for all } i \in [\ell] \\ \perp & \text{otherwise.} \end{cases}$$

Moreover, if $\Gamma' = \{\vec{v}'_1, \ldots, \vec{v}'_{\ell'}\}$, then $\Gamma' \preceq \Gamma$ iff $\Gamma \subset \Gamma'$. Note that it is IPE when all Γ are restricted to be only one vector.

In the SE setting, a functionality $\mathcal{F}$ is defined over a key space and an index space using sets of spaces and vectors, respectively. In an n-dimensional SE scheme, the key space $\mathcal{K}$ corresponds to all affine spaces $\mathcal{S}$ in $\mathbb{Z}_q^n$ and the index space $\mathcal{I}$ corresponds to all vectors $\vec{x}$ in $\mathbb{Z}_q^n$. Hence

$$\mathcal{F}(\mathcal{S}, (\vec{x}, m)) := \begin{cases} m & \text{if } \vec{x} \in \mathcal{S} \\ \perp & \text{otherwise.} \end{cases}$$

Moreover, $\mathcal{S}' \preceq \mathcal{S}$ iff $\mathcal{S}'$ is a subspace of $\mathcal{S}$.

Let $x_{(0)} = (\mathsf{ind}_{(0)}, m_{(0)}), x_{(1)} = (\mathsf{ind}_{(1)}, m_{(1)}) \in \mathcal{X}$ be the adversary's choice of plaintext pair. The security game for both multi-predicate IPE and SE can then be defined using Definition 3 with the following variations:

- If the adversary outputs the challenge indices $\mathsf{ind}_{(0)}, \mathsf{ind}_{(1)}$ before the Setup phase, the security game is under the *selective security* model. Otherwise it is under the *full security* model.
- If the adversary outputs the challenge indices such that $\mathsf{ind}_{(0)} = \mathsf{ind}_{(1)}$, the security game is under the *payload-hiding* security model, that is $\mathcal{F}(\epsilon, (\mathsf{ind}, m)) = (\mathsf{ind}, |m|)$. Otherwise it is under the *attribute-hiding* security model, that is $\mathcal{F}(\epsilon, (\mathsf{ind}, m)) = |m|$.
- In the attribute-hiding model, if the adversary is allowed key queries k_i in which $\mathcal{F}(k_i, (\mathsf{ind}_{(0)}, m_{(0)})) = m_{(0)} = m_{(1)} = \mathcal{F}(k_i, (\mathsf{ind}_{(1)}, m_{(1)}))$, the security game is considered *adaptively or fully* attribute-hiding [12, 20]. Otherwise, if $\mathcal{F}(k_i, (\mathsf{ind}_{(0)}, m_{(0)}))$ (resp. $\mathcal{F}(k_i, (\mathsf{ind}_{(1)}, m_{(1)}))$) does not reveal $m_{(0)}$ (resp. $m_{(1)}$) for all key queries k_i, the security game is considered *weakly* attribute-hiding.

2.3 Notation

In the remainder of the paper, if not explicitly specified, we assume that all vectors are row vectors in $\mathbb{Z}_q^n$ for some integer n and prime q and spaces are Euclidean spaces spanned by row vectors. Table 3 summarizes some notation used in the remainder of the paper.

Table 3: Notation.

$t \xleftarrow{R} T$	t is chosen at random from a set T according to its distribution
$t \xleftarrow{U} T$	t is chosen uniformly at random from a set T
$\mathcal{S}(\vec{v}_1, \ldots, \vec{v}_r)$	the space spanned by $\{\vec{v}_1, \ldots, \vec{v}_r\}$
$\mathcal{S}^{\perp}(\vec{v}_1, \ldots, \vec{v}_r)$	the orthogonal space of $\mathcal{S}(\vec{v}_1, \ldots, \vec{v}_r)$, i.e., the space spanned by all $\vec{x}$ where $\vec{x} \cdot \vec{v}_i = 0$ for $i \in [r]$
$\mathcal{B}(\mathcal{S})$	a basis of $\mathcal{S}$
$\mathcal{B}^{\perp}(\mathcal{S})$	a basis of $\mathcal{S}^{\perp}$
$\mathsf{dim}(\mathcal{S})$	the dimension of $\mathcal{S}$
M_i	the i-th row of matrix M
$\mathsf{det}(\mathrm{M})$	the determinant of matrix M
$\mathcal{S}(M, \vec{y})$	the affine space $\{\vec{z}M + \vec{y} : \vec{z} \in \mathbb{Z}_q^r\}$, where $M \in \mathbb{Z}_q^{r \times n}$

Note also that we use a subscript to indicate the set type of a scheme. For example, we use $\mathcal{K}_{SE}$ (resp. $\mathcal{I}_{SE}$) to denote the key space $\mathcal{K}$ (resp. index space $\mathcal{I}$) with regards to SE.

2.4 Concepts from Linear Algebra

We require the following lemmata for our generic constructions. The proofs are simple and can be obtained from [7], for example.

Lemma 1. *Given a space $\mathcal{S}$, then $\mathsf{dim}(\mathcal{S}) + \mathsf{dim}(\mathcal{S}^{\perp}) = n$ and $(\mathcal{S}^{\perp})^{\perp} = \mathcal{S}$.*

Lemma 2. *Given a space $\mathcal{S}$, there exists a polynomial-time algorithm* $\mathsf{BasisGen}$ *taking as input $\mathcal{S}$ and outputting a basis $\mathcal{B}^{\perp}$ of $\mathcal{S}^{\perp}$.*

Lemma 3. *Given a space $\mathcal{S}$, a subspace $\mathcal{S}'$ and a basis $\mathcal{B}^{\perp}$ of $\mathcal{S}^{\perp}$, there exists a polynomial-time algorithm* $\mathsf{BasisDel}$ *taking as input $\mathcal{S}, \mathcal{S}', \mathcal{B}^{\perp}$ and outputting a basis $\mathcal{B}'^{\perp}$ of $\mathcal{S}'^{\perp}$, which contains all the vectors of $\mathcal{B}^{\perp}$.*

2.5 Dual Pairing Vector Spaces by Direct Product of Symmetric Pairing Groups

Definition 5. *"Symmetric bilinear pairing groups" $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ are a tuple of a prime q, cyclic (multiplicative) groups $\mathbb{G}$ and $\mathbb{G}_T$ of order q, $g \neq 1 \in \mathbb{G}$, and a polynomial-time computable nondegenerate bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ i.e., $e(g^s, g^t) = e(g, g)^{st}$ and $e(g, g) \neq 1$. Let $\mathcal{G}_{\mathsf{bpg}}$ be an algorithm that takes as input λ and outputs a description of bilinear pairing groups $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ with security parameter λ.*

In this paper, we concentrate on the symmetric version of dual pairing vector spaces [17, 13] constructed using symmetric bilinear pairing groups given in Definition 5.

Definition 6. *"Dual pairing vector spaces (DPVS)"* $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ *by a direct product of symmetric pairing groups* $q, \mathbb{G}, \mathbb{G}_T, g, e$ *are a tuple of prime* q*,* n*-dimensional vector space*

$$\mathbb{V} := \overbrace{\mathbb{G} \times \cdots \times \mathbb{G}}^{n}$$

over $\mathbb{Z}_q$*, cyclic group* $\mathbb{G}_T$ *of order* q*, canonical basis* $\mathbb{A} := (\mathbf{a}_1, \ldots, \mathbf{a}_n)$ *of* $\mathbb{V}$*, where*

$$\mathbf{a}_i := (\overbrace{1, \ldots, 1}^{i-1}, g, \overbrace{1, \ldots, 1}^{n-i})$$

and pairing $e : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$*.*

The pairing is defined by $e(\mathbf{x}, \mathbf{y}) := \prod_{i=1}^{n} e(g_i, h_i) \in \mathbb{G}_T$ *where* $\mathbf{x} := (g_1, \ldots, g_n) \in \mathbb{V}$ *and* $\mathbf{y} := (h_1, \ldots, h_n) \in \mathbb{V}$*. This is nondegenerate bilinear, i.e.* $e(s\mathbf{x}, t\mathbf{y}) = e(\mathbf{x}, \mathbf{y})^{st}$ *and if* $e(\mathbf{x}, \mathbf{y}) = 1$ *for all* $\mathbf{y} \in \mathbb{V}$*, then* $\mathbf{x} = \mathbf{0}$*. For all* i *and* j*,* $e(\mathbf{a}_i, \mathbf{a}_j) = g_T^{\delta_{i,j}}$ *where* $\delta_{i,j} = 1$ *if* $i = j$*, and* 0 *otherwise, and* $g_T := e(g, g) \neq 1 \in \mathbb{G}_T$*.*

DPVS also has linear transformations $\phi_{i,j}$ *on* $\mathbb{V}$ *s.t.* $\phi_{i,j}(\mathbf{a}_i) = \mathbf{a}_i$*,* $\phi_{i,j}(\mathbf{a}_k) = \mathbf{0}$ *if* $k \neq j$*, which can be easily achieved by*

$$\phi_{i,j} := (\overbrace{1, \ldots, 1}^{i-1}, g_j, \overbrace{1, \ldots, 1}^{n-i})$$

where $\mathbf{x} := (g_1, \ldots, g_n)$*. We call* $\phi_{i,j}$ *"distortion maps". DPVS generation algorithm* $\mathcal{G}_{\mathsf{dpvs}}$ *takes input* λ *(*$\lambda \in \mathbb{N}$*) and* $n \in \mathbb{N}$*, and outputs a description of* $\mathsf{param}'_{\mathbb{V}} := (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ *with security parameter* λ *and* n*-dimensional* $\mathbb{V}$*. It can be constructed by using* $\mathcal{G}_{\mathsf{bpg}}$*.*

Based on the above definitions, we now describe a random dual orthonormal bases generator $\mathcal{G}_{\mathsf{ob}}$ that is used as a subroutine in our proposed multi-predicate IPE schemes:

$$\begin{aligned}
&\mathcal{G}_{\mathsf{ob}}(\lambda, n) : \mathsf{param}'_{\mathbb{V}} := (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \xleftarrow{U} \mathcal{G}_{\mathsf{dpvs}}(\lambda, n),\ \psi \xleftarrow{U} \mathbb{Z}_q^* \\
&\quad X := (\chi_{i,j}) \xleftarrow{U} GL(n, \mathbb{Z}_q),\ (\vartheta_{i,j}) := (X^{\top})^{-1},\ g_T := e(g, g)^{\psi},\ \mathsf{param}_{\mathbb{V}} := (\mathsf{param}'_{\mathbb{V}}, g_T) \\
&\quad \mathbf{b}_i := \sum_{j=1}^{n} \chi_{i,j} \mathbf{a}_j,\ \mathbb{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_n),\ \mathbf{b}_i^* := \sum_{j=1}^{n} \vartheta_{i,j} \mathbf{a}_j,\ \mathbb{B}^* := (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*), \\
&\quad \text{return } (\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^*).
\end{aligned}$$

2.6 Complexity Assumptions

Definition 7 (DLIN: Decisional Linear Assumption.)**.** *The DLIN problem is to guess* $\beta \in \{0, 1\}$*, given* $(\mathit{param}_{\mathbb{G}}, g, f, v, g^{\theta_0}, f^{\theta_1}, T) \xleftarrow{U} \mathcal{G}_{\beta}^{\mathsf{DLIN}}(\lambda)$*, where*

$$\begin{aligned}
&\mathcal{G}_{\mathsf{DLIN}}(\lambda) : \mathit{param}_{\mathbb{G}} := (q, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{R} \mathcal{G}_{\mathsf{bpg}}(\lambda), \\
&\quad \theta_0, \theta_1 \xleftarrow{U} \mathbb{Z}_q,\ f, \nu \xleftarrow{U} \mathbb{G} \\
&\quad Y_0 := \nu^{\theta_0 + \theta_1},\ Y_1 \xleftarrow{U} \mathbb{G} \\
&\quad \beta \xleftarrow{U} \{0, 1\},\ T := Y_{\beta} \\
&\quad \textit{return } (\mathit{param}_{\mathbb{G}}, g, f, \nu, g^{\theta_0}, f^{\theta_1}, T).
\end{aligned}$$

For $\beta = 0, 1$ let W_β be the event that the adversary outputs 1 and define

$$\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{A}}(\lambda) := |Pr[W_0] - Pr[W_1]|.$$

The DLIN assumption is: For any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{DLIN}}$ is negligible in λ.

Definition 8 (DBDH: Decisional Bilinear Diffie-Hellman Assumption.)**.** *The DBDH problem is to guess $\beta \in \{0,1\}$, given $(\mathsf{param}_{\mathbb{G}}, g, g^{\theta_0}, g^{\theta_1}, g^{\theta_2}, T) \xleftarrow{U} \mathcal{G}^{\mathsf{DBDH}}_{\beta}(\lambda)$, where*

$$\begin{aligned}
&\mathcal{G}_{\mathsf{DBDH}}(\lambda): \mathsf{param}_{\mathbb{G}} := (q, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{R} \mathcal{G}_{\mathsf{bpg}}(\lambda),\\
&\quad \theta_0, \theta_1, \theta_2 \xleftarrow{U} \mathbb{Z}_q, Y_0 := g^{\theta_0\theta_1\theta_2}, Y_1 \xleftarrow{U} \mathbb{G}\\
&\quad \beta \xleftarrow{U} \{0,1\}, T := Y_\beta\\
&\quad \textit{return } (\mathsf{param}_{\mathbb{G}}, g, g^{\theta_0}, g^{\theta_1}, g^{\theta_2}, T).
\end{aligned}$$

For $\beta = 0, 1$ let W_β be the event that the adversary outputs 1 and define

$$\mathsf{Adv}^{\mathsf{DBDH}}_{\mathcal{A}}(\lambda) := |Pr[W_0] - Pr[W_1]|.$$

The DBDH assumption is: For any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{DBDH}}$ is negligible in λ.

3 Fully Secure Multi-Predicate IPE under Simple Assumptions

3.1 Achieving Constant-Size Ciphertexts

Our first construction of multi-predicate IPE scheme is from the IPE scheme of [1]. The latter is a variant of Waters' dual system IBE scheme [23], which is fully secure under the DLIN and DBDH assumptions with constant-size ciphertexts. We describe our multi-predicate IPE scheme as follows:

- $\mathsf{Setup}_{mIPE}(\lambda, n)$: It picks $\mathsf{param}_{\mathbb{G}} := (q, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{R} \mathcal{G}_{\mathsf{bpg}}(\lambda)$. It then picks $\alpha, a_0, a_1, b \xleftarrow{U} \mathbb{Z}_q$ and $w, w_0, w_1, h_0, h_1, \ldots, h_n \xleftarrow{U} \mathbb{G}$. The public parameters consist of

$$\mathsf{PP} = \left(\begin{array}{l} g,\ w,\ w_0,\ w_1,\ h_0,\ h_1,\ \ldots,\ h_n,\\ A_0 = g^{a_0},\ A_1 = g^{a_1}\ B = g^{b}, B_0 = g^{ba_0},\ B_1 = g^{ba_1},\\ \tau_0 = w \cdot w_0^{a_0},\ \tau_1 = w \cdot w_1^{a_1},\ T_0 = \tau_0^b,\ T_1 = \tau_1^b, Z = e(g,g)^{\alpha a_0 b} \end{array} \right).$$

  The master key is defined to be $\mathsf{MK} = (g^\alpha, g^{\alpha a_0})$.

- $\mathsf{KeyGen}_{mIPE}(\mathsf{PP}, \mathsf{MK}, \Gamma)$: It parses Γ to be $\{\vec{v}_1, \ldots, \vec{v}_\ell\}$. It further parses each $\vec{v}_i \in \Gamma$ as $(v_{i,1}, \ldots, v_{i,n})$ and returns $\perp$ if $v_{i,n} = 0$. It then runs the following steps:

  - Pick $r_0, \ldots, r_\ell, z_0, z_1 \xleftarrow{U} \mathbb{Z}_q$, set $r = r_0 + \cdots + r_\ell$.
  - For $i \in [\ell]$, pick $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1} \xleftarrow{U} \mathbb{Z}_q$.

– Generate $\mathsf{sk}_\Gamma = (D_1, \ldots, D_7, \{K_{i,0}, K_{i,1}, \ldots, K_{i,n-1}, \mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1}\}_{i\in[\ell]})$ by computing

$$D_1 = g^{\alpha a_0} \cdot w^r, \quad D_2 = g^{-\alpha} \cdot w_0^r \cdot g^{z_0}, \quad D_3 = B^{-z_0}, \quad D_4 = w_1^r \cdot g^{z_1},$$
$$D_5 = B^{-z_1}, \quad D_6 = B^{r_0}, \quad D_7 = g^{r_1+\cdots+r_\ell},$$

for $i \in [\ell]$ we set :

$$K_{i,0} = g^{r_i}, \quad K_{i,j} = (h_n^{-\frac{v_{i,j}}{v_{i,n}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r_i} \quad \text{for } j \in [n-1].$$

- $\mathsf{Enc}_{mIPE}(\mathsf{PP}, \vec{x}, m)$: To encrypt $m \in \mathbb{G}_T$ under $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, it picks $s_0, s_1, t, \mathsf{tagc} \xleftarrow{U} \mathbb{Z}_q$ and computes $c = (C_1, \ldots, C_7, C, E_0, E_1, \mathsf{tagc})$ where

$$C_1 = B^{s_0+s_1}, \quad C_2 = B_0^{s_0}, \quad C_3 = A_0^{s_0}, \quad C_4 = B_1^{s_1},$$
$$C_5 = A_1^{s_1}, \quad C_6 = \tau_0^{s_0} \cdot \tau_1^{s_1}, \quad C_7 = T_0^{s_0} \cdot T_1^{s_1} \cdot h_0^{-t},$$
$$C = m \cdot Z^{s_1}, \quad E_0 = g^t, \quad E_1 = (h_0^{\mathsf{tagc}} \cdot \prod_{i=1}^{n} h_i^{x_i})^t.$$

- $\mathsf{Dec}_{mIPE}(\mathsf{PP}, \mathsf{sk}_\Gamma, c)$: It recovers the message in the following steps:
  - Compute $\mathsf{tagk}_i = \mathsf{tagk}_{i,1}x_1 + \cdots + \mathsf{tagk}_{i,n-1}x_{n-1}$ for $i \in [\ell]$.
  - Compute $W = \prod_{j=1}^{5} e(C_j, D_j) \cdot (\prod_{j=6}^{7} e(C_j, D_j))^{-1} = e(g,g)^{\alpha a_0 b s_1} \cdot e(g, h_0)^{(r_1+\cdots+r_\ell)t}$.
  - Compute $W_i = (\frac{e((K_{i,1})^{x_1} \ldots (K_{i,n-1})^{x_{n-1}}, E_0)}{e(E_1, K_{i,0})})^{\frac{1}{\mathsf{tagk}_i - \mathsf{tagc}}} = e(g, h_0)^{r_i t}$.
  - Recover the message as $m = C/Z^{s_1} = C/e(g,g)^{\alpha a_0 b s_1} \leftarrow C \cdot W_2 \cdots W_\ell \cdot W^{-1}$.

- $\mathsf{Del}_{mIPE}(\mathsf{PP}, \Gamma, \mathsf{sk}_\Gamma, \Gamma')$: It parses Γ as $\{\vec{v}_1, \ldots, \vec{v}_\ell\}$ and Γ' as $\{\vec{v}_1, \ldots, \vec{v}_\ell, \vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}\}$. For all $i \in [\ell']$ it parses each $\vec{v}_i$ as $(v_{i,1}, \ldots, v_{i,n})$, and returns $\perp$ if $v_{i,n} = 0$. It then runs the following steps:
  - Parse sk_Γ to be

$$D_1, \ldots, D_7, \{K_{i,0}, K_{i,1}, \ldots, K_{n-1}, \mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1}\}_{i=1,\ldots,\ell}.$$

  - Pick $r'_0, \ldots, r'_{\ell'}, z'_0, z'_1 \xleftarrow{U} \mathbb{Z}_q$ and set $r' = r'_0 + \cdots + r'_{\ell'}$.
  - For $i \in [\ell+1, \ell']$, pick $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1} \xleftarrow{U} \mathbb{Z}_q$.
  - Generate $\mathsf{sk}_{\Gamma'} = (D'_1, \ldots, D'_7, \{K'_{i,0}, K'_{i,1}, \ldots, K'_{i,n-1}, \mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1}\}_{i\in[\ell']})$ by computing

$$D'_1 = D_1 \cdot w^{r'}, \quad D'_2 = D_2 \cdot w_0^{r'} \cdot g^{z'_0}, \quad D'_3 = D_3 \cdot B^{-z'_0}, \quad D'_4 = D_4 \cdot w_1^{r'} \cdot g^{z'_1},$$
$$D'_5 = D_5 \cdot B^{-z'_1}, \quad D'_6 = D_6 \cdot B^{r'_0}, \quad D'_7 = D_7 \cdot g^{r'_1+\ldots+r'_{\ell'}},$$

for $i \in [\ell]$ we set :

$$K'_{i,0} = K_{i,0} \cdot g^{r'_i}, \quad K'_{i,j} = K_{i,j} \cdot (h_n^{-\frac{v_{i,j}}{v_{i,n}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r'_i} \quad \text{for } j \in [n-1],$$

for $i \in [\ell+1, \ell']$ we set :

$$K'_{i,0} = g^{r'_i}, \quad K'_{i,j} = (h_n^{-\frac{v_{i,j}}{v_{i,n}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r'_i} \quad \text{for } j \in [n-1].$$

Theorem 1. *The multi-predicate IPE scheme is fully secure under the DLIN and DBDH assumptions.*

Proof. Our proof uses the dual system methodology similar to that in [23, 1], which involves ciphertexts and secret keys that can be normal or semi-functional. The semi-functional ciphertext and key generation algorithms (which are not used in a real system) can be described as follows:

Semi-Functional Ciphertexts: The algorithm first runs the encryption algorithm to generate a normal ciphertext $(C'_1, \ldots, C'_7, C', E'_0, E'_1, \mathsf{tagc}')$. It then chooses $\chi \xleftarrow{U} \mathbb{Z}_q$ before replacing (C'_4, C'_5, C'_6, C'_7) by

$$C_4 = C'_4 \cdot g^{ba_1\chi}, C_5 = C'_5 \cdot g^{a_1\chi}, C_6 = C'_6 \cdot w_1^{a_1\chi}, C_7 = C'_7 \cdot w_1^{a_1 b\chi},$$

respectively.

Semi-Functional Secret Keys: The algorithm first runs the encryption algorithm to generate a normal private key $D'_1, \ldots, D'_7, \{K'_{i,0}, K'_{i,1}, \ldots, K'_{i,n-1}, \mathsf{tagk}'_{i,1}, \ldots, \mathsf{tagk}'_{i,n-1}\}_{i\in[\ell]}$. The semi-functional key is obtained by choosing $\gamma \xleftarrow{U} \mathbb{Z}_q$ and replacing (D'_1, D'_2, D'_4) by

$$D_1 = D'_1 \cdot g^{-a_0 a_1 \gamma}, D_2 = D'_2 \cdot g^{a_1\gamma}, D_4 = D'_4 \cdot g^{a_0\gamma},$$

respectively.

Let us assume that adversary $\mathcal{A}$ makes at most q_R key reveal queries and q_A key generation & delegation queries. Our proof can then proceed in a sequence of games defined as follows:

- Game_{Real} is the actual multi-predicate IPE security game.
- Game_0 is identical to Game_{Real} except that the challenge ciphertext is semi-functional.
- Game_k (for $1 \le k \le q_R$) is identical to Game_0 except that the first k key reveal queries are answered by returning a semi-functional key.
- Game_{Final} is as Game_{q_R} but the challenge ciphertext is semi-functional encryption of a random element of $\mathbb{G}_T$ instead of the actual plaintext.

In Appendix B, we prove a set of lemmata that argue the indistinguishability between any two consecutive games under the DLIN and DBDH assumptions The game ends in Game_{Final}, where any adversary's advantage must be negligibly close to 0. □

The indistinguishability of Game_{Real} and Game_0 (see Lemma 4) as well as that of Game_{q_R} and Game_{Final} (Lemma 6) can be proved in the similar way as with [23]. It turns out that the main challenge is to show indistinguishability between Game_{k-1} and Game_k (Lemma 5). This is mainly because multi-predicate IPE allows key delegation as opposed to IPE without delegation mechanism and the adversary's queries have less restriction than that of HIBE.

In the HIBE scheme of [23], the challenger embeded some 2-equation trapdoors into the k-th key reveal query's identity $(id_1, \ldots, id_\ell)$ and the challenge identity $(id^*_1, \ldots, id^*_{\ell^*})$. More precisely, it plans to embed a 2-equation trapdoor into id_ℓ and id^*_ℓ, this is because the key query and challenge identity always have the restriction that $id^*_\ell \neq id_\ell$ if $\ell^* \ge \ell$. To do this, the challenger guesses a key generation or delegation query as the k-th key reveal query. This strategy overcomes the "tag lineage" problem because the challenger embeds the trapdoor

before the tag values are "locked", resulting the challenger's guess to be correct with probability $1/q'_A$, where q'_A is the number of key generation & delegation queries. Subsequently, Attrapadung and Libert [1] used an n-equation trapdoor as opposed to a 2-equation trapdoor for their IPE scheme to deal with the difficulties arising from the richer structure of IPE and the aggregation of ciphertexts into a constant number of elements. To embed the n-equation trapdoor, it requires that the adversary's k-th key reveal query on $\vec{v}$ has the restriction of $\vec{x}^* \cdot \vec{v} \neq 0$ for a challenge vector $\vec{x}^*$, which is clearly necessary under payload-hiding model.

In our multi-predicate IPE scheme, the adversary's query is less restricted than the previous cases in the sense that, for a challenge vector $\vec{x}^*$ and a key reveal query $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$, we only require some $\vec{v}_i$ such that $\vec{x}^* \cdot \vec{v}_i \neq 0$. Thus, our goal is to embed an n-equation trapdoor into the challenge vector $\vec{x}^*$ and some vector $\vec{v}_i$ of the k-th reveal query $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$, where $\vec{x}^* \cdot \vec{v}_i \neq 0$ holds. As expected, our construction also faces the "tag lineage" problem. However, adopting Waters's strategy described above does not work here since we may get $\vec{x}^* \cdot \vec{v}_\ell = 0$. Moreover, we cannot simply guess further a vector in the key reveal query that will satisfy the restriction. This is because the k-th reveal query may be a key delegation query from a previous key for $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ delegated to $\Gamma' := \{\vec{v}_1, \ldots, \vec{v}_{\ell'}\}$ while the tag values for vectors in Γ have already been "locked". We introduce a new strategy based on tree structure. The challenger arranges all queries and their vectors (each vector is associated with an universal counter value) in a tree. It guesses a value k^* for all vectors in the tree such that the k^*-th vector will appear in the k-th key reveal query and the corresponding vector $\vec{v}$ satisfies the condition of $\vec{x} \cdot \vec{v} \neq 0$. This allows us to embed the n-equation trapdoor to the correct vector of the key reveal query.

3.2 Improving Efficiency under Adaptively Attribute-Hiding Model

Our second construction of multi-predicate IPE scheme makes use of the IPE scheme of [20] as a building block. The latter is fully secure and adaptively attribute-hiding under the DLIN assumption but without constant-size ciphertexts. The main idea of our construction is that to generate a secret key, we mask each vector in Γ with a random value form $\mathbb{Z}_q$ and then compute the sum of all the masked vectors. We include a delegation mechanism in a similar way to the HIPE scheme of [13, 20]. We use $\vec{0}$ to denote the n-dimensional zero vector and $\mathsf{span}\langle \mathbf{b}_1, \ldots, \mathbf{b}_n \rangle$ to denote all linear combination of $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ (i.e., $\sigma_1 \mathbf{b}_1 + \ldots + \sigma_n \mathbf{b}_n$ where $\sigma_1, \ldots, \sigma_n \in \mathbb{Z}_q$). Our multi-predicate IPE scheme is as follows:

- $\mathsf{Setup}_{mIPE}(\lambda, n)$: It performs the following steps:
  - Pick $(\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathcal{G}_{\mathsf{ob}}(\lambda, 4n+2)$.
  - Set $\widehat{\mathbb{B}} := (\mathbf{b}_0, \ldots, \mathbf{b}_n, \mathbf{b}_{4n+1})$, $\widehat{\mathbb{B}}^* := (\mathbf{b}^*_0, \ldots, \mathbf{b}^*_n)$, $\widetilde{\mathbb{B}}^* := (\mathbf{b}^*_{3n+1}, \ldots, \mathbf{b}^*_{4n})$.
  - Return $\mathsf{PP} := (\mathsf{param}_{\mathbb{V}}, \widehat{\mathbb{B}}, \widetilde{\mathbb{B}}^*)$ and $\mathsf{MK} := \widehat{\mathbb{B}}^*$.
- $\mathsf{KeyGen}_{mIPE}(\mathsf{PP}, \mathsf{MK}, \Gamma)$: It sets Γ to be $\{\vec{v}_1, \ldots, \vec{v}_\ell\}$ and does the following steps:
  - Pick $\sigma_{\mathsf{dec},i}, \sigma_{\mathsf{ran},1,i}, \ldots, \sigma_{\mathsf{ran},\ell,i}, \sigma_{\mathsf{del},1,i}, \ldots, \sigma_{\mathsf{del},n,i}, \psi \xleftarrow{\$} \mathbb{F}_q$ for $i \in [\ell]$.
  - Pick $\vec{\eta}_{\mathsf{dec}}, \vec{\eta}_{\mathsf{ran},1}, \ldots, \vec{\eta}_{\mathsf{ran},\ell}, \vec{\eta}_{\mathsf{del},1}, \ldots, \vec{\eta}_{\mathsf{del},n} \xleftarrow{\$} \mathbb{F}_q^n$.

– Compute

$$\mathbf{k}^*_{\ell,\mathsf{dec}} := (1,\, \sigma_{\mathsf{dec},1}\vec{v}_1 + \ldots + \sigma_{\mathsf{dec},\ell}\vec{v}_\ell,\, \vec{0},\, \vec{0},\, \vec{\eta}_{\mathsf{dec}},\, 0)_{\mathbb{B}^*},$$
$$\mathbf{k}^*_{\ell,\mathsf{ran},j} := (0,\, \sigma_{\mathsf{ran},j,1}\vec{v}_1 + \ldots + \sigma_{\mathsf{ran},j,\ell}\vec{v}_\ell,\, \vec{0},\, \vec{0},\, \vec{\eta}_{\mathsf{ran},j},\, 0)_{\mathbb{B}^*} \text{ for } j \in [\ell],$$
$$\mathbf{k}^*_{\ell,\mathsf{del},j} := (0,\, \sigma_{\mathsf{del},j,1}\vec{v}_1 + \ldots + \sigma_{\mathsf{del},j,\ell}\vec{v}_\ell + \psi\vec{v}_j, \vec{0},\, \vec{0},\, \vec{\eta}_{\mathsf{del},j},\, 0)_{\mathbb{B}^*} \text{ for } j \in [n].$$

– Return $\mathsf{sk}_\Gamma := (\mathbf{k}^*_{\ell,\mathsf{dec}}, \mathbf{k}^*_{\ell,\mathsf{ran},1}, \ldots, \mathbf{k}^*_{\ell,\mathsf{ran},\ell}, \mathbf{k}^*_{\ell,\mathsf{del},1}, \ldots, \mathbf{k}^*_{\ell,\mathsf{del},n})$.

- $\mathsf{Enc}_{mIPE}(\mathsf{PP}, \vec{x}, m)$: It
  - Picks $\zeta, \delta, \eta \xleftarrow{\$} \mathbb{F}_q$.
  - Computes $c_0 := m \cdot g_T^\zeta$, $\mathbf{c}_1 := (\zeta,\, \delta\vec{x},\, \vec{0},\, \vec{0},\, \vec{0},\, \eta)_{\mathbb{B}}$.
  - Returns $c := (c_0, \mathbf{c}_1)$.
- $\mathsf{Dec}_{muIPE}(\mathsf{PP}, \mathsf{sk}_\Gamma, c)$: It recovers message $m := c_0/e(\mathbf{c}_1, \vec{k}^*_{\ell,\mathsf{dec}})$.
- $\mathsf{Del}_{mIPE}(\mathsf{PP}, \Gamma, \mathsf{sk}_\Gamma, \Gamma')$: It sets Γ to be $\{\vec{v}_1, \ldots, \vec{v}_\ell\}$, Γ' to be $\{\vec{v}_1, \ldots, \vec{v}_\ell, \vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}\}$, and $\vec{v}_i$ to be $(v_{i,1}, \ldots, v_{i,n})$ for $i \in [\ell']$. It then does the following steps:
  - Pick $\alpha_{\mathsf{dec},i}, \alpha_{\mathsf{ran},1,i}, \ldots, \alpha_{\mathsf{ran},\ell',i}, \alpha_{\mathsf{del},1,i}, \ldots, \alpha_{\mathsf{del},n,i}, \psi' \xleftarrow{\$} \mathbb{F}_q$ for $i \in [\ell']$.
  - Pick $\mathbf{r}^*_{\mathsf{dec}}, \mathbf{r}^*_{\mathsf{ran},1}, \ldots, \mathbf{r}^*_{\mathsf{ran},\ell'}, \mathbf{r}^*_{\mathsf{del},1}, \ldots, \mathbf{r}^*_{\mathsf{del},n} \xleftarrow{\$} \mathsf{span}\langle \mathbf{b}^*_{3n+1}, \ldots, \mathbf{b}^*_{4n}\rangle$.
  - Compute

$$\mathbf{k}^*_{\ell',\mathsf{dec}} := \mathbf{k}^*_{\ell,\mathsf{dec}} + \sum_{i=1}^{\ell} \alpha_{\mathsf{dec},i}\mathbf{k}^*_{\ell,\mathsf{ran},i} + \sum_{i=\ell+1}^{\ell'} \alpha_{\mathsf{dec},i}(\sum_{t=1}^{n} v_{i,t}\mathbf{k}^*_{\ell,\mathsf{del},t}) + \mathbf{r}^*_{\mathsf{dec}}$$
$$\mathbf{k}^*_{\ell',\mathsf{ran},j} := \sum_{i=1}^{\ell} \alpha_{\mathsf{ran},j,i}\mathbf{k}^*_{\ell,\mathsf{ran},i} + \sum_{i=\ell+1}^{\ell'} \alpha_{\mathsf{ran},j,i}(\sum_{t=1}^{n} v_{i,t}\mathbf{k}^*_{\ell,\mathsf{del},t}) + \mathbf{r}^*_{\mathsf{ran},j} \text{ for } j \in [\ell'],$$
$$\mathbf{k}^*_{\ell',\mathsf{del},j} := \sum_{i=1}^{\ell} \alpha_{\mathsf{del},j,i}\mathbf{k}^*_{\ell,\mathsf{ran},i} + \sum_{i=\ell+1}^{\ell'} \alpha_{\mathsf{del},j,i}(\sum_{t=1}^{n} v_{i,t}\mathbf{k}^*_{\ell,\mathsf{del},t}) + \psi'\mathbf{k}^*_{\ell,\mathsf{del},j} + \mathbf{r}^*_{\mathsf{del},j} \text{ for } j \in [n].$$

  - Return $\mathsf{sk}_{\Gamma'} := (\mathbf{k}^*_{\ell',\mathsf{dec}}, \mathbf{k}^*_{\ell',\mathsf{ran},1} \ldots, \mathbf{k}^*_{\ell',\mathsf{ran},\ell'}, \mathbf{k}^*_{\ell',\mathsf{del},1}, \ldots, \mathbf{k}^*_{\ell',\mathsf{del},n})$.

Theorem 2. *The multi-predicate IPE scheme is fully secure under the DLIN assumption.*

The security proof of the theorem is essentially similar to that for the IPE scheme of [20]. Hence, we do not discuss any further here.

4 Generic Construction of SE from Multi-Predicate IPE

In this section, we describe how to construct an n-dimensional SE scheme from a multi-predicate IPE scheme that works in Euclidean spaces. One can adopt similar techniques of [8] to derive an n-dimensional SE scheme in affine spaces from an $(n+1)$-dimensional SE scheme in Euclidean spaces. For completeness, we describe such techniques in Appendix A.

4.1 Intuition

In a multi-predicate IPE scheme, a secret key associated with a set of vectors $\{\vec{v}_1, \ldots, \vec{v}_\ell\} \in \mathcal{K}_{mIPE}$ can be viewed as being associated with the space $\mathcal{S}^\perp(\vec{v}_1, \ldots, \vec{v}_\ell)$. In other words, we can interpret the relation between a ciphertext and a secret key as $\mathcal{F}(\mathcal{S}^\perp(\vec{v}_1, \ldots, \vec{v}_\ell), (\vec{x}, m)) = m$ in the SE setting. From this observation and the property that $(\mathcal{S}^\perp)^\perp = \mathcal{S}$ from Lemma 1, instead of generating a secret key for an $(n-\ell)$-dimensional space $\mathcal{S} \in \mathcal{K}_{SE}$, we generate a secret key for a basis $\mathcal{B}^\perp(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_\ell\} \in \mathcal{K}_{mIPE}$ of $\mathcal{S}^\perp$ by using the KeyGen algorithm of the multi-predicate IPE scheme. Here $\mathcal{B}^\perp(\mathcal{S})$ can be generated by running the $\mathsf{BasisGen}$ algorithm as described in Lemma 2.

We note that when more linearly independent vectors are added into the vector set, the dimension of the orthogonal space gets smaller. Namely $\mathcal{S}^\perp(\vec{v}_1, \ldots, \vec{v}_\ell, \vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'})$ is a subspace of $\mathcal{S}^\perp(\vec{v}_1, \ldots, \vec{v}_\ell)$. This property is crucial for key delegation. Given a space $\mathcal{S}$, a subspace $\mathcal{S}'$ of $\mathcal{S}$, and a "fixed" basis $\mathcal{B}^\perp = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ of $\mathcal{S}^\perp$, we can derive a basis in the form $\mathcal{B}'^\perp = \{\vec{v}_1, \ldots, \vec{v}_\ell, \vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}\}$ of $\mathcal{S}'^\perp$ by running the $\mathsf{BasisDel}$ algorithm as described in Lemma 3.

4.2 Construction

We now describe the construction of an SE scheme from a multi-predicate IPE scheme using the above idea. To construct an n-dimensional SE scheme, we require a multi-predicate IPE scheme with the same dimension. Given a multi-predicate IPE scheme with five algorithms: Setup_{mIPE}, KeyGen_{mIPE}, Enc_{mIPE}, Dec_{mIPE}, and Del_{mIPE}, we construct an SE scheme with the corresponding five algorithms: Setup_{SE}, KeyGen_{SE}, Enc_{SE}, Dec_{SE}, and Del_{SE}, as follows:

- $\mathsf{Setup}_{SE}(\lambda, n)$: It runs $\mathsf{Setup}_{mIPE}(\lambda, n)$ and outputs public parameters PP and a master key MK.
- $\mathsf{KeyGen}_{SE}(\mathsf{PP}, \mathsf{MK}, \mathcal{S})$: It generates a secret key for an $(n-\ell)$-dimensional space $\mathcal{S}$. It first runs $\mathsf{BasisGen}(\mathcal{S})$ and outputs $\mathcal{B}^\perp(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$. It then runs $\mathsf{KeyGen}_{mIPE}(\mathsf{PP}, \mathsf{MK}, \mathcal{B}^\perp(\mathcal{S}))$ and outputs a secret key $\mathsf{sk}_\mathcal{S}$ with $\mathcal{B}^\perp(\mathcal{S})$.
- $\mathsf{Enc}_{SE}(\mathsf{PP}, \vec{x}, m)$: It runs $\mathsf{Enc}_{mIPE}(\mathsf{PP}, (\vec{x}, m))$ and outputs a ciphertext c.
- $\mathsf{Dec}_{SE}(\mathsf{PP}, \mathsf{sk}_\mathcal{S}, c)$: It runs $\mathsf{Dec}_{mIPE}(\mathsf{PP}, \mathsf{sk}_\mathcal{S}, c)$ and outputs a message m.
- $\mathsf{Del}_{SE}(\mathsf{PP}, \mathcal{S}, \mathsf{sk}_\mathcal{S}, \mathcal{S}')$: It delegates a secret key to an $(n-\ell')$-dimensional subspace $\mathcal{S}'$ of $\mathcal{S}$, where $\mathcal{B}^\perp(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$. It first runs $\mathsf{BasisDel}(\mathcal{S}, \mathcal{S}', \mathcal{B}^\perp(\mathcal{S}))$ and outputs $\mathcal{B}^\perp(\mathcal{S}') = \{\vec{v}_1, \ldots, \vec{v}_\ell, \vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}\}$. It then runs $\mathsf{Del}_{mIPE}(\mathsf{PP}, \mathcal{B}^\perp(\mathcal{S}), \mathsf{sk}_\mathcal{S}, \mathcal{B}^\perp(\mathcal{S}'))$ and outputs a secret key $\mathsf{sk}_{\mathcal{S}'}$ with $\mathcal{B}^\perp(\mathcal{S}')$.

We now show that the resulting SE scheme works correctly and is indeed secure.

Theorem 3. *The SE scheme constructed from the multi-predicate IPE scheme works correctly.*

Proof. From our construction, the only algorithm that needs to be considered is delegation. Given a partial order pair $\mathcal{S}' \preceq \mathcal{S}$ of $\mathcal{K}_{SE}$, we transform it into a pair $(\mathcal{B}^\perp(\mathcal{S}'), \mathcal{B}^\perp(\mathcal{S}))$ using the $\mathsf{BasisDel}$ algorithm such that $\mathcal{B}^\perp(\mathcal{S}') \preceq \mathcal{B}^\perp(\mathcal{S})$ in the multi-predicate IPE setting. Thus, the key delegation algorithm works as required.

Given a plaintext $(\vec{x}, m) \in \mathcal{X}_{SE}$ and an $(n-\ell)$-dimensional space $\mathcal{S} \in \mathcal{K}_{SE}$, we transform them into $(\vec{x}, m) \in \mathcal{X}_{mIPE}$ and $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_r\} \in \mathcal{K}_{mIPE}$ respectively. Then we have

$$\begin{aligned}\mathcal{F}(\mathcal{S}, (\vec{x}, m)) = m &\Leftrightarrow \vec{x} \in \mathcal{S} \\ &\Leftrightarrow \vec{x} \cdot \vec{v}_i = 0 \text{ for all } \vec{v}_i \in \mathcal{B}^{\perp}(\mathcal{S}) \\ &\Leftrightarrow \mathcal{F}(\mathcal{B}^{\perp}(\mathcal{S}), (\vec{x}, m)) = m.\end{aligned}$$

This implies that the resulting SE scheme inherits the decryptability from the original multi-predicate IPE scheme, i.e., $\mathcal{F}(\mathcal{S}, (\vec{x}, m)) = m$ iff $\vec{x} \in \mathcal{S}$ as defined in Section 2.2. □

Theorem 4. *For any adversary $\mathcal{A}$ against the SE scheme in the same security model for the multi-predicate IPE scheme, there is an adversary $\mathcal{D}$ against the multi-predicate IPE scheme, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{SE}(\lambda) \leq \mathsf{Adv}_{\mathcal{D}}^{mIPE}(\lambda).$$

Moreover, the SE scheme preserves properties from the multi-predicate IPE scheme.

Proof. Given any adversary $\mathcal{A}$ against the SE scheme in the same security model for the multi-predicate IPE scheme (which is fully/selective secure, attribute/payload-hiding), we simulate an adversary $\mathcal{D}$ with advantage $\mathsf{Adv}_{\mathcal{A}}^{SE}(\lambda)$ against the multi-predicate IPE scheme as follows:

- Setup: It runs a real game $\mathcal{RG}$ of multi-predicate IPE and forwards PP to adversary $\mathcal{A}$.
- Query: It answers $\mathcal{A}$'s queries by querying $\mathcal{RG}$'s key generation and delegation oracles.
- Challenge: It forwards $\mathcal{A}$'s challenge to $\mathcal{RG}$ and then returns $\mathcal{RG}$'s output to $\mathcal{A}$.
- Guess: It answers $\mathcal{A}$'s queries as Query phase and $\mathcal{D}$ forwards $\mathcal{A}$'s guess to $\mathcal{RG}$.

In the above security game, we can efficiently transform the elements in $\mathcal{K}_{SE}$ and $\mathcal{X}_{SE}$, as required by the multi-predicate IPE setting. From Theorem 3 and its proof, all the oracles of the SE scheme can be simulated correctly. Moreover, the plaintexts $x_{(0)} = (\vec{x}_{(0)}, m_{(0)})$, $x_{(1)} = (\vec{x}_{(1)}, m_{(1)}) \in \mathcal{X}_{SE}$, any space $\mathcal{S} \in \mathcal{K}_{SE}$ of $\mathcal{A}$'s choices and those in the multi-predicate IPE setting satisfy requirement (1) (and other restrictions of the security model) simultaneously. Thus, the simulation is perfect and we conclude that $\mathsf{Adv}_{\mathcal{D}}^{mIPE}(\lambda)$ is at least $\mathsf{Adv}_{\mathcal{A}}^{SE}(\lambda)$.

It is clear that properties such as full/selective security (under simple assumptions) and attribute/payload-hiding are preserved in the transformation since the model of security game we simulate for the SE scheme is identical to that for the original multi-predicate IPE scheme. Moreover, should the original multi-predicate IPE scheme work in prime order bilinear groups and have constant ciphertexts (and other properties), the derived SE scheme would also inherit such properties since the SE scheme can be viewed as a "restricted" form or an embedding of the multi-predicate IPE scheme. □

Remark: We note that our first multi-predicate IPE scheme described in Section 3.1 requires that $v_n \neq 0$ for any vector $(v_1, \ldots, v_n)$. Thus, when we derive an SE scheme using the above generic techniques, we can add one more dimension as a "dummy scalar". That is, each vector $\vec{v} = (v_1, \ldots, v_n)$ in $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ is replaced by a vector $(v_1, \ldots, v_n, 1)$, and $\vec{x} = (x_1, \ldots, x_n)$ is replaced by a vector $(x_1, \ldots, x_n, 0)$ for key generation and encryption, respectively.

5 Discussion

5.1 Fully Secure HIBE Scheme under Simple Assumptions with Constant Ciphertexts

Building on our results in the previous sections, we now show how one can construct a fully secure HIBE scheme under simple DLIN and DBDH assumptions with constant-size ciphertexts. Our HIBE scheme can be regarded as an improved version the Waters HIBE scheme of [23] by having constant-size ciphertexts not achievable by the latter..

In the HIBE setting, a functionality $\mathcal{F}$ is defined over a key space and an index space using sets of hierarchical identities. In an HIBE scheme with maximum depth n, the key space $\mathcal{K}$ (resp. index space $\mathcal{I}$) corresponds to all hierarchical identities in the form $(id_1, \ldots, id_\ell)$ (resp. $(id'_1, \ldots, id'_{\ell'})$), where $\ell \leq n$. Here

$$\mathcal{F}((id_1, \ldots, id_\ell), ((id'_1, \ldots, id'_{\ell'}), m)) := \begin{cases} m & \text{if } \ell \leq \ell' \text{ and } id'_i = id_i \text{ for all } i \in [\ell] \\ \perp & \text{otherwise.} \end{cases}$$

Moreover, $(id'_1, \ldots, id'_{\ell'}) \preceq (id_1, \ldots, id_\ell)$ iff $\ell \leq \ell'$ and $id'_i = id_i$ for all $i \in [\ell]$.

Generally, to embed an HIBE scheme with maximal depth n, we require an n-dimensional SE in affine spaces. We assume that identities are elements in $\mathbb{Z}_q^*$, and hence one will use a collision resistant hash function to hash identities of arbitrary length to $\mathbb{Z}_q^*$. For $(id_1, \ldots, id_\ell)$ associated with a secret key, we encode it as the affine space which begins with $(id_1, \ldots, id_\ell)$, namely

$$\{(id_1, \ldots, id_\ell, x_{\ell+1}, \ldots, x_n) : x_i \in \mathbb{Z}_q\}.$$

For $(id_1, \ldots, id_\ell)$ associated with a ciphertext, we encode it as the vector

$$(id_1, \ldots, id_\ell, 0, \ldots, 0).$$

Our HIBE scheme is derived from our SE scheme, which in turn, is obtained from the multi-predicate IPE in Section 3.1. To construct an n-dimensional SE scheme, we require an $(n+1)$-dimensional multi-predicate IPE scheme. Here, we do not need an additional dimension as a dummy scalar as mentioned before, since we always have $v_{n+1} \neq 0$ for any vector $\vec{v} = (v_1, \ldots, v_n, v_{n+1})$ in our construction when running the key generation or delegation algorithm.

- $\mathsf{Setup}_{HIBE}(\lambda, n)$: It chooses bilinear groups $(q, \mathbb{G}, \mathbb{G}_T, g', e)$ of prime order $q \leq 2^\lambda$. It then picks $g, w, w_0, w_1, h_0, h_1, \ldots, h_n \xleftarrow{U} \mathbb{G}$ and $\alpha, a_0, a_1, b \xleftarrow{U} \mathbb{Z}_q$. The public parameters consist of

$$\mathsf{PP} = \left(\begin{array}{l} g,\ w,\ w_0,\ w_1,\ h_0,\ h_1,\ \ldots,\ h_n, h_{n+1} \\ A_0 = g^{a_0},\ A_1 = g^{a_1}\ B = g^b, B_0 = g^{b \cdot a_0},\ B_1 = g^{b \cdot a_1}, \\ \tau_0 = w \cdot w_0^{a_0},\ \tau_1 = w \cdot w_1^{a_1},\ T_0 = \tau_0^b,\ T_1 = \tau_1^b, Z = e(g,g)^{\alpha \cdot a_0 \cdot b} \end{array} \right).$$

  The master key is defined to be $\mathsf{MK} = (g^\alpha, g^{\alpha a_0})$.

- $\mathsf{KeyGen}_{HIBE}(\mathsf{PP}, \mathsf{MK}, (id_1, \ldots, id_\ell))$: It picks $r_0, \ldots, r_\ell, z_0, z_1 \xleftarrow{U} \mathbb{Z}_q$, sets $r = \sum_{i=0}^{\ell} r_i$. For $i \in [\ell]$, it picks $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n} \xleftarrow{U} \mathbb{Z}_q$ It then generates $\mathsf{sk}_{(id_1,\ldots,id_\ell)} =$

$(D_1, \ldots, D_7, \{K_{i,0}, K_{i,1}, \ldots, K_{id_i,n}, \mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n}\}_{i\in[\ell]})$ by computing

$$D_1 = g^{\alpha a_0} \cdot w^r, \quad D_2 = g^{-\alpha} \cdot w_0^r \cdot g^{z_0}, \quad D_3 = B^{-z_0}, \quad D_4 = w_1^r \cdot g^{z_1},$$
$$D_5 = B^{-z_1}, \quad D_6 = B^{r_0}, \quad D_7 = g^{r_1+\ldots+r_\ell},$$
for $i \in [\ell]$:
$$K_{i,0} = g^{r_i}, \quad K_{i,i} = (h_{n+1}^{\frac{1}{id_i}} \cdot h_i \cdot h_0^{\mathsf{tagk}_{i,i}})^{r_i}, \quad K_{i,j} = (h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r_i} \text{ for } j \in [n]\backslash\{i\}.$$

- $\mathsf{Enc}_{HIBE}(\mathsf{PP}, (id_1, \ldots, id_\ell), m)$: To encrypt $m \in \mathbb{G}_T$, it picks $s_0, s_1, t, \mathsf{tagc} \xleftarrow{U} \mathbb{Z}_q$ and computes $c = (C_1, \ldots, C_7, E_0, E_1, E_2, \mathsf{tagc})$ where
$$C_1 = B^{s_0+s_1}, \quad C_2 = B_0^{s_0}, \quad C_3 = A_0^{s_0}, \quad C_4 = B_1^{s_1},$$
$$C_5 = A_1^{s_1}, \quad C_6 = \tau_0^{s_0} \cdot \tau_1^{s_1}, \quad C_7 = T_0^{s_0} \cdot T_1^{s_1} \cdot h_0^{-t},$$
$$C = m \cdot Z^{s_1}, \quad E_0 = g^t, \quad E_1 = (h_0^{\mathsf{tagc}} \cdot h_{n+1} \cdot \prod_{i=1}^{n} h_i^{id_i})^t.$$
- $\mathsf{Dec}_{HIBE}(\mathsf{PP}, \mathsf{sk}_{(id_1,\ldots,id_\ell)}, c)$: It recovers a message by performing the following steps:
  - Compute $\mathsf{tagk}_i = \mathsf{tagk}_{i,1} id_1 + \cdots + \mathsf{tagk}_{i,\ell} id_\ell$ for $i \in [\ell]$.
  - Compute $W = \prod_{j=1}^{5} e(C_j, D_j) \cdot (\prod_{j=6}^{7} e(C_j, D_j))^{-1} = e(g,g)^{\alpha a_0 b s_1} \cdot e(g, h_0)^{(r_1+\cdots+r_\ell)t}$.
  - Compute $W_i = (\frac{e((K_{i,1})^{id_1} \ldots (K_{i,\ell})^{id_\ell}, E_0)}{e(E_1, K_{i,0})})^{\frac{1}{\mathsf{tagk}_i - \mathsf{tagc}}} = e(g, h_0)^{r_i t}$.
  - Recover the message as $m = C/Z^{s_1} = C/e(g,g)^{\alpha a_0 b s_1} \leftarrow C \cdot W_1 \cdots W_\ell \cdot W^{-1}$.
- $\mathsf{Del}_{HIBE}(\mathsf{PP}, (id_1, \ldots, id_\ell), \mathsf{sk}_\Gamma, (id_1, \ldots, id_{\ell'}))$: To delegate a key, it runs the following steps:
  - Set $\mathsf{sk}_{(id_1,\ldots,id_\ell)}$ to be
$$D_1, \ldots, D_7, \{K_{i,0}, K_{i,1}, \ldots, K_{i,n}, \mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n}\}_{i\in[\ell]}.$$
  - Pick $r'_0, \ldots, r'_\ell, r'_{\ell+1}, \ldots, r'_{\ell'}, z'_0, z'_1 \xleftarrow{U} \mathbb{Z}_q$ and set $r' = r'_0 + \ldots + r'_{\ell'}$.
  - For $i \in [\ell+1, \ell']$, pick $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n} \xleftarrow{U} \mathbb{Z}_q$.
  - Generate $\mathsf{sk}_{\Gamma'} = (D'_1, \ldots, D'_7, \{K'_{i,0}, K'_{i,1}, \ldots, K'_{i,n}, \mathsf{tagk}'_{i,1}, \ldots, \mathsf{tagk}'_{i,n}\}_{i\in[\ell']})$ by computing
$$D'_1 = D_1 \cdot w^{r'}, \quad D'_2 = D_2 \cdot w_0^{r'} \cdot g^{z'_0}, \quad D'_3 = D_3 \cdot B^{-z'_0},$$
$$D'_4 = D_4 \cdot w_1^{r'} \cdot g^{z'_1}, \quad D'_5 = D_5 \cdot B^{-z'_1}, \quad D'_6 = D_6 \cdot B^{r'_0},$$
$$D'_7 = D_7 \cdot g^{r'_1+\ldots+r'_{\ell'}},$$
for $i \in [\ell]$ we set :
$$K'_{i,0} = K_{id_i,0} \cdot g^{r'_i}, \quad K'_{i,i} = K_{i,j} \cdot (h_{n+1}^{\frac{1}{id_i}} \cdot h_i \cdot h_0^{\mathsf{tagk}_{id_i,j}})^{r'_i},$$
$$K'_{i,j} = K_{i,j} \cdot (h_j \cdot h_0^{\mathsf{tagk}_{id_i,j}})^{r'_i} \text{ for } j \in [n]\backslash\{i\},$$
for $i \in [\ell+1, \ell']$ we set :
$$K'_{i,0} = g^{r'_i}, \quad K'_{i,i} = (h_{n+1}^{\frac{1}{id_i}} \cdot h_i \cdot h_0^{\mathsf{tagk}_{i,i}})^{r'_i}, \quad K'_{i,j} = (h_i \cdot h_0^{\mathsf{tagk}_{i,j}})^{r'_i} \text{ for } j \in [n]\backslash\{i\}.$$

The security of our HIBE scheme can be obtained from the Embedding Lemma of [4].

Theorem 5. *The HIBE scheme is fully secure under the DLIN and DBDH assumptions.*

A summary of comparisons of existing HIBE schemes against ours is presented in Table 4, where n denotes the maximal depth of HIBE.

Table 4: Comparisons between existing HIBE schemes and ours.

	BBG[2]	W[22]	W[23]	LW[14]	Ours
size of PP	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
size of sk	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$
size of c	3	$\mathcal{O}(n)$	$\mathcal{O}(n)$	3	10
# pairings	2	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
fully secure	No	Yes	Yes	Yes	Yes
anonymous	No	No	No	No	No
constant c	Yes	No	No	Yes	Yes
prime order	Yes	Yes	Yes	No	Yes
simple assumptions	No	Yes	Yes	No	Yes

We note that the HIBE scheme of [14] has short ciphertexts, but it is under composite order bilinear group, which is less efficient.

5.2 Generic Relation between SE, HIPE and Multi-Predicate IPE

We have seen that multi-predicate IPE implies SE under some linear reduction. Although SE implies HIPE, we can also show how to construct HIPE from multi-predicate IPE directly. In fact, interestingly, multi-predicate IPE can be constructed from SE or HIPE under linear or quadratic reduction, respectively. Thus, we have the following relation:

$$\text{SE} \underset{\text{Linear}}{\overset{\text{Linear}}{\rightleftarrows}} \text{mIPE} \underset{\text{Quadratic}}{\overset{\text{Linear}}{\rightleftarrows}} \text{HIPE}$$

These results are shown in Appendix D. Particularly, we obtain a fully secure HIPE scheme under simple assumptions in prime order group with constant size of ciphertexts from the multi-predicate IPE scheme of Section 3.1. We note that if we directly extend the IPE scheme of [1] to HIPE, the size of ciphertexts may be dependent on the depth of the hierarchy.

5.3 Generic Construction of Negated SE from Non-zero IPE

In addition to the generic construction of SE from multi-predicate IPE, we describe a generic construction of a negated SE scheme that requires only a non-zero IPE scheme. This may be of independent interest.

Given a non-zero IPE scheme, denoted as $nIPE$, with four algorithms: Setup_{nIPE}, KeyGen_{nIPE}, Enc_{nIPE}, and Dec_{nIPE}, we construct a negated SE scheme, denoted as nSE with the corresponding four algorithms: Setup_{nSE}, KeyGen_{nSE}, Enc_{nSE}, and Dec_{nSE}, as follows:

- $\mathsf{Setup}_{nSE}(\lambda, n)$: It runs $\mathsf{Setup}_{nIPE}(\lambda, n)$ and outputs public parameters PP and a master key MK.
- $\mathsf{KeyGen}_{nSE}(\mathsf{PP}, \mathsf{MK}, \mathcal{S})$: It generates a secret key for an $(n-\ell)$-dimensional space $\mathcal{S}$. It first runs $\mathsf{BasisGen}(\mathcal{S})$ and outputs $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$. It then runs $\mathsf{sk}_{\vec{v}_i} = \mathsf{KeyGen}_{nIPE}(\mathsf{PP}, \mathsf{MK}, \vec{v}_i)$ for all $i \in [\ell]$ and outputs a secret key $\mathsf{sk}_{\mathcal{S}} := \{\mathsf{sk}_{\vec{v}_1}, \ldots, \mathsf{sk}_{\vec{v}_\ell}\}$.
- $\mathsf{Enc}_{nSE}(\mathsf{PP}, \vec{x}, m)$: It runs $\mathsf{Enc}_{nIPE}(\mathsf{PP}, \vec{x}, m)$ and outputs a ciphertext c.
- $\mathsf{Dec}_{nSE}(\mathsf{PP}, \mathsf{sk}_{\mathcal{S}}, c)$: For c associated with some $\vec{x}$, it selects some i such that $\vec{x} \cdot \vec{v}_i \neq 0$. It runs $\mathsf{Dec}_{nIPE}(\mathsf{PP}, \mathsf{sk}_{\vec{v}_i}, c)$ and outputs a message m.

By applying our nIPE-to-nSE transformation techniques, we can obtain a fully secure negated SE scheme under DLIN assumption from the scheme of [18] which has about a factor $O(n)$ shorter size of public parameters and ciphertexts than the negated SE scheme presented in [8].

Table 5: Comparisons between existing negated SE schemes and ours.

	AL[1]	CLLW[8]	This Paper
size of PP	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
size of sk	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
size of c	4	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
# pairings	3	$\mathcal{O}(n)$	$\mathcal{O}(n)$
fully secure	No	Yes	Yes
anonymous	No	No	No
constant c	Yes	No	No
prime order	Yes	Yes	Yes
simple assumptions	No	Yes	Yes

6 Open Problem

Existing and our SE schemes are attribute-hiding or have short/constant ciphertexts, but not both. Constructing a fully secure SE, multi-predicate IPE, or HIPE scheme that is attribute-hiding and has short/constant ciphertexts in prime order bilinear groups is still an open problem, particularly one that works under simple assumptions.

References

[1] N. Attrapadung, B. Libert, Functional encryption for inner product achieving constant-size ciphertexts with adaptive security or support for negation. In: P.Q. Nguyen, D. Pointcheval (Eds.) PKC 2010, LNCS 6056, pp. 384-402. Springer, 2010.

[2] D. Boneh, X. Boyen, E. Goh, Hierarchical identity based encryption with constant size ciphertext. In: R. Cramer (Ed.) EUROCRYPT 2005, LNCS 3493, pp. 440-456. Springer, 2005.

[3] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing. In: J. Kilian (Ed.) CRYPTO 2001, LNCS 2139, pp. 213-229. Springer, 2001.

[4] D. Boneh, M. Hamburg, Generalized identity based and broadcast encryption schemes. In: J. Pieprzyk (Ed.) ASIACRYPT 2008, LNCS 5350, pp. 455-470. Springer, 2008.

[5] D. Boneh, A. Sahai, B. Waters, Functional encryption: definitions and challenges. In: Y. Ishai (Ed.) TCC 2011, LNCS 6597, pp. 253-273, Springer, 2011.

[6] C. Cocks, An identity based encryption scheme based on quadratic residues. In: B. Honary (Ed.) Cryptography and Coding 2001, LNCS 2260, pp. 360-363. Springer, 2001.

[7] H. Cohen, A course in computational algebraic number theory. Springer, 1996.

[8] J. Chen, H. W. Lim, S. Ling, H. Wang, The relation and transformation between hierarchical inner product encryption and spatial encryption. Cryptology ePrint Archive, Report 2011/455, http://eprint.iacr.org/2011/455/.

[9] D. M. Freeman, Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: H. Gilbert (Ed.) EUROCRYPT 2010, LNCS 6110, pp. 44-61. Springer, 2010.

[10] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for finegrained access control of encrypted data. In: CCS 2006, pp. 89-98. ACM, 2006.

[11] M. Hamburg, Spatial encryption. Cryptology ePrint Archive, Report 2010/389, http://eprint.iacr.org/2011/389

[12] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: N. P. Smart (Ed.) EUROCRYPT 2008, LNCS 4965, pp. 146-162. Springer, 2008.

[13] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption attribute-based encryption and (hierarchical) inner product encryption. In: H. Gilbert (Ed.) EUROCRYPT 2010, LNCS 6110, pp. 62-91. Springer, 2010.

[14] A. Lewko, B. Waters, New techniques for dual system encryption and fully secure IBE and HIBE with short ciphertexts. In: D. Micciancio (Ed.) TCC 2010, LNCS 5978, pp. 455-479. Springer, 2010.

[15] A. Lewko, B. Waters, Unbounded HIBE and attribute-based encryption. Cryptology ePrint Archive, Report 2011/049, http://eprint.iacr.org/2011/049

[16] D. Moriyama, H. Doi, A fully secure spatial encryption scheme. In: IEICE Transactions 94-A(1), pp. 28-35. 2011.

[17] T. Okamoto, K. Takashima, Hierarchical predicate encryption for inner-products. In: M. Matsui (Ed.) ASIACRYPT 2009, LNCS 5912, pp. 214-231. Springer, 2009.

[18] T. Okamoto, K. Takashima, Fully secure functional encryption with general relations from the decisional linear assumption. In: T. Rabin (Ed.) CRYPTO 2010, LNCS 6223, pp. 191-208. Springer, 2010.

[19] T. Okamoto, K. Takashima, Efficient attribute-based signatures for non-monotone predicates in the standard model.In: D. Catalano et al. (Eds.) PKC 2011, LNCS 6571, pp. 35-52. Springer, 2011.

[20] T. Okamoto, K. Takashima, Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. Cryptology ePrint Archive, Report 2011/543, http://eprint.iacr.org/2011/543.

[21] E. Shi, B. Waters, Delegating capabilities in predicate encryption systems. In: L. Aceto et al. (Eds.) ICALP 2008, LNCS 5126, pp. 560-578. Springer, 2008.

[22] B. Waters, Efficient identity-based ecnryption without random oracles, In: R. Cramer (Ed.) EUROCRYPT 2005, LNCS 3493, pp. 114-127. Springer, 2005.

[23] B. Waters, Dual system encryption realizing fully secure IBE and HIBE under simple assumptions. In: S. Halevi (Ed.) CRYPTO 2009, LNCS 5677, pp. 619-636. Springer, 2009.

[24] M. X. Zhou, Z. F. Cao, Spatial encryption under simpler assumption. In: J. Pieprzyk, F. Zhang (Eds.) ProvSec 2009, LNCS 5848, pp. 19-31. Springer, 2009.

A Construction of SE in Affine Spaces

Now, we briefly show how to construct an n-dimensional SE scheme in affine spaces from an $(n+1)$-dimensional SE scheme in Euclidean spaces.

Given an affine space $\mathcal{S}(M, \vec{y}) = \{\vec{z}M + \vec{y} : \vec{z} \in \mathbb{Z}_q^r\}$, where $M \in \mathbb{Z}_q^{r \times n}$, we embed it in $\mathcal{S}((M_1, 0), \ldots, (M_r, 0), (\vec{y}, 1)) \in \mathcal{X}_{SE}$. Given a vector $\vec{x} \in \mathbb{Z}_q^n$, we embed it in $(\vec{x}, 1) \in \mathcal{I}_{SE}$. Then it is not difficult to check that $\vec{x} \in \mathcal{S}(M, \vec{y})$ iff $(\vec{x}, 1) \in \mathcal{S}((M_1, 0), \ldots, (M_r, 0), (\vec{y}, 1))$. Moreover, if $\mathcal{S}(M', \vec{y}')$ is a subspace of $\mathcal{S}(M, \vec{y})$, namely there is some matrix $T \in \mathbb{Z}_q^{r' \times r}$ and vector $\vec{z} \in \mathbb{Z}_q^r$ such that $M' = TM$ and $\vec{y}' = \vec{y} + \vec{z}M$, then $\mathcal{S}((M'_1, 0), \ldots, (M'_{r'}, 0), (\vec{y}', 1))$ is a subspace of $\mathcal{S}((M_1, 0), \ldots, (M_r, 0), (\vec{y}, 1))$, and vice versa.

B Lemmas for Theorem 6

We let $\mathsf{Game}_{Real}\mathsf{Adv}_{\mathcal{A}}$ denote an adversary $\mathcal{A}$'s advantage in the real game.

Lemma 4. *Suppose that there exists an adversary $\mathcal{A}$ where $\mathsf{Game}_{Real}\mathsf{Adv}_{\mathcal{A}} - \mathsf{Game}_0\mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the DLIN game.*

Proof. Our algorithm $\mathcal{B}$ receives an instance $(\mathsf{param}_{\mathbb{G}}, g, f, \nu, g^{\theta_0}, f^{\theta_1}, T)$ of the DLIN problem.
Setup It picks $\alpha, b, y_w, y_{w_0}, y_{w_1} \xleftarrow{U} \mathbb{Z}_q$, $h_0, h_1, \ldots, h_n \xleftarrow{U} \mathbb{G}$ and sets

$$g = g,\ w = g^{y_w},\ w_0 = g^{y_{w_0}},\ w_1 = g^{y_{w_1}},\ h_0 = h_0, \ldots,\ h_n = h_n,$$
$$A_0 = g^{a_0} = f,\ A_1 = g^{a_1} = \nu,\ B = g^b,\ B_0 = g^{ba_0} = f^b,\ B_1 = g^{ba_1} = \nu^b,$$
$$\tau_0 = w \cdot w_0^{a_0} = w \cdot f^{y_{w_0}},\ \tau_1 = w \cdot w_1^{a_1} = w \cdot f^{y_{w_1}},\ T_0 = \tau_0^b,\ T_1 = \tau_1^b, Z = e(g,g)^{\alpha a_0 b} = e(g,f)^{\alpha b}.$$

Here, a_0, a_1 are the exponents that the reduction cannot know itself. Finally, $\mathcal{B}$ forwards the public parameters PP to $\mathcal{A}$. We also note that using α it can compute the master key MK for itself

$$g^{\alpha},\ g^{\alpha a_0} = f^{\alpha}.$$

Key Queries Since $\mathcal{B}$ has the actual master secret key MK it simply runs the key generation to generate the keys in both phases. Note that the MK it has only allows for the creation of normal keys.
Challenge $\mathcal{B}$ receives two messages $m_{(0)}, m_{(1)}$ and challenge vector $\vec{x}^*$. It then flips $\beta \xleftarrow{U} \{0,1\}$. It generates the challenge ciphertext in two steps. First, it generates a normal ciphertext using the real algorithm by calling $\mathsf{Enc}(\mathsf{PP}, \vec{x}^*, m_{(\beta)})$, which outputs a ciphertext $c' = (C'_1, \ldots, C'_7, C', E'_0, E'_1, \mathsf{tagc}^*)$. Let s'_0, s'_1, t' be the random exponents used in generating the ciphertext.

Then we modify components of the normal ciphertext as follows and returns ciphertext $c = (C_1, \ldots, C_7, C, E_0, E_1, \mathsf{tagc}^*)$:

$$C_1 = C'_1 \cdot (g^{\theta_0})^b, \quad C_2 = C'_2 \cdot (f^{\theta_1})^{-b}, \quad C_3 = C'_3 \cdot (f^{-\theta_1}), \quad C_4 = C'_4 \cdot T^b,$$
$$C_5 = C'_5 \cdot T, \quad C_6 = C'_6 \cdot (g^{\theta_0})^{-y_w} \cdot (f^{\theta_1})^{-y_{w_0}} \cdot T^{y_{w_1}}, \quad C_7 = C'_7 \cdot ((g^{\theta_0})^{y_w} \cdot (f^{\theta_1})^{-y_{w_0}} \cdot T^{y_{w_1}})^b,$$
$$C = C' \cdot (e(g^{\theta_0}, f) \cdot e(g, f^{\theta_1}))^{\alpha b}, \quad E_0 = E'_0, \quad E_1 = E'_1.$$

If T is a tuple, then this assignment implicitly sets $s_0 = -\theta_1 + s'_0, s_1 = s'_1 + \theta_0 + \theta_1$ and $s = s_0 + s_1 = \theta_0 + s'_0 + s'_1$. If $T = \nu^{\theta_0+\theta_1}$ it will have the same distribution as a normal ciphertext; otherwise, it will be distributed identically to a semi-functional ciphertext. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$. □

Lemma 5. *Suppose that there exists an adversary $\mathcal{A}$ that makes at most q_R queries and $\mathsf{Game}_{k-1}\mathsf{Adv}_{\mathcal{A}} - \mathsf{Game}_k\mathsf{Adv}_{\mathcal{A}} = \epsilon$ for some k where $1 \leq k \leq q_R$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the DLIN game.*

Proof. Our algorithm $\mathcal{B}$ receives an instance $(\mathsf{param}_{\mathbb{G}}, g, f, \nu, g^{\theta_0}, f^{\theta_1}, T)$ of the DLIN problem.
Setup $\mathcal{B}$ picks $\alpha, a_0, a_1, y_{w_0}, y_{w_1}, y_{h_0}, \ldots, y_{h_n} \xleftarrow{U} \mathbb{Z}_q$, $\vec{\zeta} := (\zeta_1, \ldots, \zeta_n) \xleftarrow{U} \mathbb{Z}_q^n$ and sets

$$g = g,\ w = \nu^{-a_0 a_1},\ w_0 = \nu^{a_1} \cdot g^{y_{w_0}},\ w_1 = \nu^{a_0} \cdot g^{y_{w_1}},$$
$$h_0 = f \cdot g^{y_{h_0}},\ h_1 = f^{\zeta_1} \cdot g^{y_{h_1}}, \ldots,\ h_n = f^{\zeta_n} \cdot g^{y_{h_n}},$$
$$A_0 = g^{a_0},\ A_1 = g^{a_1},\ B = g^b = f,\ B_0 = g^{ba_0} = f^{a_0},\ B_1 = g^{ba_1} = f^{a_1},$$
$$\tau_0 = w \cdot w_0^{a_0} = g^{y_{w_0} a_0},\ \tau_1 = w \cdot w_1^{a_1} = g^{y_{w_1} a_1},$$
$$T_0 = \tau_0^b = f^{y_{w_0} a_0},\ T_1 = \tau_1^b = f^{y_{w_1} a_1},\ Z = e(g,g)^{\alpha a_0 b} = e(f,g)^{\alpha a_0}.$$

Next, $\mathcal{B}$ forwards the public parameters PP to $\mathcal{A}$. Note that $\mathcal{B}$ knows MK as g^{α}, $g^{\alpha a_0}$.
Key Queries $\mathcal{B}$ begins by choosing a value k^* uniformly at random between 1 and $(n-1)q_A$ and creates a counter μ. Initially, μ is set to 0. When either a generate or delegate request is made μ is incremented as follows:

- If this is a generate query for a vector set $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$, then μ is incremented by 1 for each vector in Γ one by one.
- Suppose this is a delegate query to delegate from a previous key of a vector set $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ to $\Gamma' := \{\vec{v}_1, \ldots, \vec{v}_{\ell'}\}$. Then μ is incremented by 1 for each vector in $\{\vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}\}$ one by one.

Additionally, $\mathcal{B}$ uses a tree structure S to record information of all key queries. Roughly, the root node is empty, a generate query will be recorded on a new child node of root while a delegate query will be recorded on a new child of the node of previous key.

Key Generation is done the same regardless of whether it is phase 1 or 2. Then, $\mathcal{B}$ answers three types of queries as follows:

- Generate: Assume it is a query for $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$. It generates a new child node of root for Γ. For each vector $\vec{v}_i \in \Gamma$, do

  1. It sets $\mu = \mu + 1$.
  2. If $\mu \neq k^*$, it picks tag values $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1}$. Otherwise $\mu = k^*$, it computes tag values $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1} \xleftarrow{U} \mathbb{Z}_q$ for $\vec{v}_i := (v_1, \ldots, v_n)$ by using n equation trapdoor as

$$\begin{pmatrix} \mathsf{tagk}_{i,1} \\ \mathsf{tagk}_{i,2} \\ \vdots \\ \mathsf{tagk}_{i,n-1} \end{pmatrix} = \begin{pmatrix} \frac{v_1}{v_n} & -1 & & \\ \frac{v_2}{v_n} & & -1 & \\ \vdots & & & \ddots \\ \frac{v_{n-1}}{v_n} & & & -1 \end{pmatrix} \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_n \end{pmatrix}.$$

  3. It associates the tag values $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1}$, μ, and $\vec{v}_i$ to the new node.

- Delegate: Assume it is a query from a previous key of $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ to $\Gamma' := \{\vec{v}_1, \ldots, \vec{v}_{\ell'}\}$. It generates a new child node of Γ for Γ'. For each vector $\vec{v}_i \in \{\vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}\}$, it sets $\mu = \mu + 1$ and generates tag values $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1}$ similar as key generation queries, which are associated to the new node with μ and $\vec{v}_i$.

- Reveal: $\mathcal{B}$ answers the η-th reveal query as follows.

  Case 1: $\eta > k$
  Suppose $\mathcal{A}$ ask for the key of Γ to be revealed. Since $\mathcal{B}$ has the master key MK, it can generate a normal secret key, using the tag values stored in the node of Γ and its ancestors.

  Case 2: $\eta < k$
  Suppose $\mathcal{A}$ ask for the key of Γ to be revealed. It first generates a normal key using MK and the tag values stored in the node of Γ and its ancestors. Then it makes it semi-functional using $g^{a_0 a_1}$ and the semi-functional key generation algorithm described in Theorem 6.

  Case 3: $\eta = k$
  Suppose $\mathcal{A}$ ask for the key of $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ to be revealed. If any of the following events happens then the $\mathcal{B}$ aborts the simulation and guesses whether T is a tuple randomly.

  - If k^* is not stored in node of Γ or its ancestors
  - If $\mathcal{A}$ has already submitted the challenge vector $\vec{x}^*$, we have $\vec{x}^* \cdot \vec{v}_{i^*} = 0$, where $\vec{v}_{i^*}$ has the counter value k^*.

  The algorithm $\mathcal{B}$ first runs the key generation algorithm to generate a normal secret key sk_Γ for Γ with $D'_1, \ldots, D'_7, \{K'_{i,0}, K'_{i,1}, \ldots, K'_{i,n-1}\}_{i \in [\ell]}$ using the tag values stored in the node of Γ and its ancestors. Let $r'_0, \ldots, r'_\ell, z'_0, z'_1$ be the random exponents used.

Assume $\vec{v}_{i^*} := (v_1, \ldots, v_n) \in \Gamma$ has the counter value k^*, it then sets

$$D_1 = D'_1 \cdot T^{-a_0 \cdot a_1}, \quad D_2 = D'_2 \cdot T^{a_1} \cdot (g^{\theta_0})^{y_{w_0}}, \quad D_3 = D'_3 \cdot (f^{\theta_1})^{y_{w_0}},$$
$$D_4 = D'_4 \cdot T^{a_0} \cdot (g^{\theta_0})^{y_{w_1}}, \quad D_5 = D'_5 \cdot (f^{\theta_1})^{y_{w_1}}, \quad D_6 = D'_6 \cdot f^{\theta_1},$$
$$D_7 = D'_7 \cdot (g^{\theta_0}),$$

for $i \in [\ell] \backslash \{i^*\}$ we set :

$$K_{i,0} = K'_{i,0}, \quad K_{i,j} = K'_{i,j} \text{ for } j \in [n-1],$$

for $i = i^*$ we set :

$$K_{i^*,0} = K'_{i^*,0} \cdot (g^{\theta_0}), \quad K_{i^*,j} = K'_{i^*,j} \cdot (g^{\theta_0})^{-y_{h_n}(v_i/v_n) + y_{h_i} + y_{h_0} \mathsf{tagk}_{i^*,j}} \text{ for } j \in [n-1],$$

If $T = \nu^{\theta_0+\theta_1}$, then k-th reveal query results in a normal key under randomness $r_0 = r'_0 + \theta_1$, $r_i = r'_i$ for $i \in [\ell] \backslash \{i^*\}$, $r_{i^*} = r'_{i^*} + \theta_0$, $z_0 = z'_0 - y_{w_0}\theta_1$, and $z_1 = z'_1 - y_{w_1}\theta_1$. Otherwise, if T is a random group element, then we can write $T = \nu^{\theta_0+\theta_1} \cdot g^{\gamma}$ for random $\gamma \in \mathbb{Z}_q$. This forms a semi-functional key where γ is the added randomness to make it semi-functional.

Challenge $\mathcal{B}$ receives two messages $m_{(0)}, m_{(1)}$ and challenge vector $\vec{x}^*$. If $\mu \geq k^*$ and $\vec{x}^* \cdot \vec{v}_{i^*} = 0$ then the $\mathcal{B}$ aborts the simulation and guesses whether T is a tuple randomly, where $\vec{v}_{i^*}$ has the counter value k^*. Otherwise, it picks $\beta \xleftarrow{U} \{0,1\}$ and computes the $\mathsf{tagc}^* = -\langle \vec{x}^*, \vec{\zeta} \rangle$ for which $\mathcal{B}$ will be able to prepare the semi-functional ciphertext. It generates the challenge ciphertext in two steps. First, it generates a normal ciphertext using the real algorithm by calling $\mathsf{Enc}(\mathsf{PP}, \vec{x}^*, m_{(\beta)})$, which outputs a ciphertext $c' = (C'_1, \ldots, C'_7, E'_0, E'_1, E'_2, \mathsf{tagc}^*)$ by using random exponents s'_0, s'_1, t'. Then $\mathcal{B}$ picks $\chi \xleftarrow{U} \mathbb{Z}_q$ and computes

$$C_1 = C_1, \quad C_2 = C'_2 \quad C_3 = C'_3, \quad C_4 = C'_4 \cdot f^{a_1\chi},$$
$$C_5 = C'_5 \cdot g^{a_1\chi}, \quad C_6 = C'_6 \cdot w_1^{a_1\chi}, \quad C_7 = C'_7 \cdot \nu^{-y_{h_0}a_0a_1\chi} \cdot f^{y_{w_1}a_1\chi},$$
$$C = C', \quad E_0 = E'_0 \cdot \nu^{a_0a_1\chi}, \quad E_1 = E'_1 \cdot (\nu^{y_{h_0}\mathsf{tagc}^* + \langle \vec{x}^*, \vec{\delta} \rangle})^{a_0a_1\chi}.$$

We claim that $c = (C_1, \ldots, C_7, E_0, E_1, E_2, \mathsf{tagc}^*)$ is a semi-functional ciphertext with underlying exponents $\chi, s_0 = s'_0, s_1 = s'_1$ and $t = t' + \log_g(\nu)a_0a_1\chi$. One can check it as in Lemma 2 of [1]. Also it can show that $\mathsf{tagc}^*, \mathsf{tagk}_{i^*,1}, \ldots, \mathsf{tagk}_{i^*,n-1}$ are still n-wise independent. But this holds since their relations form a system

$$M \cdot \vec{\zeta} := \begin{pmatrix} -\frac{v_1}{v_n} & 1 & & & \\ -\frac{v_2}{v_n} & & 1 & & \\ \vdots & & & \ddots & \\ -\frac{v_{n-1}}{v_n} & & & & 1 \\ x_1 & x_2 & x_3 & \ldots & x_n \end{pmatrix} \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \\ \zeta_n \end{pmatrix} = - \begin{pmatrix} \mathsf{tagk}_{i^*,1} \\ \mathsf{tagk}_{i^*,2} \\ \vdots \\ \mathsf{tagk}_{i^*,n-1} \\ \mathsf{tagc}^* \end{pmatrix}$$

which has a solution in $\vec{\zeta}$ whenever $\mathsf{det}(M) = (-1)^{n+1}\vec{x}^* \cdot \vec{v}_{i^*}/v_n \neq 0$. Since all the other tags of the k-th reveal key are randomly chosen, all the tags are independent in A's view.

Eventually, $\mathcal{B}$ receives a bit β' from $\mathcal{A}$ and outputs 0 iff $\beta = \beta'$.

The reduction will not abort $1/(n \cdot q_A)$ amount of the time and the abort condition will be independent of the adversary's success. □

Lemma 6. *Suppose that there exists an adversary $\mathcal{A}$ that makes at most q_R queries and $\mathsf{Game}_{q_R}\mathsf{Adv}_{\mathcal{A}} - \mathsf{Game}_{Final}\mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the DBDH game.*

Proof. Our algorithm $\mathcal{B}$ receives an instance $(\mathsf{param}_{\mathbb{G}}, g, g^{\theta_0}, g^{\theta_1}, g^{\theta_2}, T)$ of the DBDH problem. Note that in both of these two games the challenge ciphertexts and all the private keys are semi-functional. Therefore, $\mathcal{B}$ only needs to be able to generate semi-functional private keys.
Setup The algorithm $\mathcal{B}$ picks $a_0, b, y_w, y_{w_0}, y_{w_1}, y_{h_0}, \ldots, y_{h_n} \xleftarrow{\$} \mathbb{Z}_q$ and sets

$$
\begin{aligned}
&g = g,\ w = g^{y_w},\ w_0 = g^{y_{w_0}},\ w_1 = g^{y_{w_1}},\ h_0 = g^{y_{h_0}}, \ldots, h_n = g^{y_{h_n}}\\
&B = g^b,\ A_0 = g^{a_0},\ A_1 = g^{\theta_1},\ B_0 = g^{ba_0},\ B_1 = g^{ba_1} = (g^{\theta_1})^b,\\
&\tau_0 = w \cdot w_0^{a_0},\ \tau_1 = w \cdot (g^{\theta_1})^{y_{w_1}},\\
&T_0 = \tau_0^b,\ T_1 = \tau_1^b, Z = e(g,g)^{\alpha a_0 b} = e(g^{\theta_0}, g^{\theta_1})^{a_0 b}.
\end{aligned}
$$

Next, $\mathcal{B}$ forwards the public parameters PP to $\mathcal{A}$. Note that the master key g^α is not available to $\mathcal{B}$, where $\alpha = \theta_0 \cdot \theta_1$ and $a_1 = \theta_1$.
Key Queries All key generations result in semi-functional keys. When a request for $\Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ is made, the key generation algorithm chooses random $r_0, \ldots, r_\ell, z_0, z_1, \gamma' \xleftarrow{U} \mathbb{Z}_q$ and sets $r = r_0 + \ldots + r_\ell$ and implicitly sets the variable $\gamma = \theta_0 + \gamma'$. It also picks $\mathsf{tagk}_{i,1}, \ldots, \mathsf{tagk}_{i,n-1} \xleftarrow{U} \mathbb{Z}_q$ for $i \in [\ell]$, It then generates the key as:

$$
\begin{aligned}
&D_1 = (g^{\theta_1})^{-a_0\gamma'} \cdot w^r, \quad D_2 = (g^{\theta_1})^{-\gamma'} \cdot w_0^r \cdot g^{z_0}, \quad D_3 = B^{-z_0},\\
&D_4 = (g^{\theta_1})^{a_0} \cdot g^{a_0\gamma'} \cdot w_1^r \cdot g^{z_1}, \quad D_5 = B^{-z_1}, \quad D_6 = B^{r_0}, \quad D_7 = g^{r_1+\cdots+r_\ell},\\
&\text{for } i \in [\ell] \text{ we set :}
\end{aligned}
$$

$$
K_{i,0} = g^{r_i}, \quad K_{i,j} = (h_n^{-\frac{v_{i,j}}{v_{i,n}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r_i} \quad \text{for } j \in [n-1].
$$

Challenge $\mathcal{B}$ receives two messages $m_{(0)}, m_{(1)}$ and challenge vector $\vec{x}^*$. $\mathcal{B}$ will now generate a challenge ciphertext that is a semi-functional ciphertext of either $m_{(\beta)}$ or a random message, depending on T. It first picks $\beta \xleftarrow{U} \{0,1\}$.

$\mathcal{B}$ picks $s_0, t, \mathsf{tagc} \xleftarrow{U} \mathbb{Z}_q$. It will implicitly let $s_1 = \theta_2$. It then picks $\chi' \xleftarrow{U} \mathbb{Z}_q$ and will implicitly set $\chi = -\theta_2 + \chi'$.

$$
\begin{aligned}
&C_1 = g^{bs_0} \cdot (g^{\theta_2})^b, \quad C_2 = g^{ba_0 s_0} \quad C_3 = g^{a_0 s_0}, \quad C_4 = (g^{\theta_1})^{b\chi'},\\
&C_5 = (g^{\theta_1})^{\chi'}, \quad C_6 = \tau_0^{s_0} \cdot (g^{\theta_2})^{y_w} \cdot (g^{\theta_1})^{y_{w_1}\chi'}, \quad C_7 = T_0^{s_0} \cdot (g^{\theta_2})^{by_w} \cdot (g^{\theta_1})^{by_{w_1}\chi'} \cdot h_0^{-t},\\
&C = m_{(\beta)} \cdot T^{ba_0}, \quad E_0 = g^t, \quad E_1 = (h_0^{\mathsf{tagc}} \cdot \prod_{i=1}^{n} h_i^{x_i})^t.
\end{aligned}
$$

If T is a tuple, then we are in Game_{q_R} otherwise, we are in Game_{Final}. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$. □

C Hierarchical Inner Product Encryption

C.1 Definition of HIPE

Here, we give the definition of n-dimensional HIPE scheme with a hierarchy of depth d. Similarly as the definition of multi-predicate IPE and SE, in the HIPE setting, a functionality

$\mathcal{F}$ is defined over a key space and an index space using sets of hierarchical vectors. Let a hierarchy of depth d vector spaces have the form of $\vec{\mu} := (n, d; \mu_1, \ldots, \mu_d)$ where $\mu_0 = 0 < \mu_1 < \mu_2 < \ldots < \mu_d = n$. Let $\Phi_i := \mathbb{Z}_q^{\mu_i - \mu_{i-1}} \backslash \{\vec{0}\}$ for $i \in [d]$ be the sets of vectors. Let $\Phi := \bigcup_{i=1}^{d} (\Phi_1 \times \ldots \times \Phi_d)$, where the union is a disjoint union. The key space $\mathcal{K}$ (resp. index space $\mathcal{I}$) for HIPE then corresponds to all hierarchical vectors $(\vec{v}_1, \ldots, \vec{v}_\ell)$ (resp. $(\vec{x}_1, \ldots, \vec{x}_h)$) of depth at most d in Φ. Here

$$\mathcal{F}((\vec{v}_1, \ldots, \vec{v}_\ell), ((\vec{x}_1, \ldots, \vec{x}_h), m)) := \begin{cases} m & \text{if } \ell \leq h \text{ and } \vec{x}_i \cdot \vec{v}_i = 0 \text{ for all } i \in [\ell] \\ \perp & \text{otherwise.} \end{cases}$$

Moreover, $(\vec{v}'_1, \ldots, \vec{v}'_{\ell'}) \preceq (\vec{v}_1, \ldots, \vec{v}_\ell)$ iff $\ell \leq \ell'$ and $\vec{v}'_i = \vec{v}_i$ for all $i \in [\ell]$. Namely $(\vec{v}_1, \ldots, \vec{v}_\ell, \vec{v}_{\ell+1}, \ldots, \vec{v}_{\ell'}) \preceq (\vec{v}_1, \ldots, \vec{v}_\ell)$.

C.2 Fully Secure HIPE Scheme under Simple Assumptions

- $\mathsf{Setup}_{HIPE}(\lambda, \vec{\mu})$: It picks $\mathsf{param}_{\mathbb{G}} := (q, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{R} \mathcal{G}_{\mathsf{bpg}}(\lambda)$. It then picks $\alpha, a_0, a_1, b \xleftarrow{U} \mathbb{Z}_q$ and $w, w_0, w_1, h_0, h_1, \ldots, h_n \xleftarrow{U} \mathbb{G}$. The public parameters consist of

$$\mathsf{PP} = \left(\begin{array}{l} g,\, w,\, w_0,\, w_1,\, h_0,\, h_1,\, \ldots,\, h_n, \\ A_0 = g^{a_0},\, A_1 = g^{a_1}\, B = g^b, B_0 = g^{ba_0},\, B_1 = g^{ba_1}, \\ \tau_0 = w \cdot w_0^{a_0},\, \tau_1 = w \cdot w_1^{a_1},\, T_0 = \tau_0^b,\, T_1 = \tau_1^b, Z = e(g,g)^{\alpha a_0 b} \end{array} \right).$$

  The master key is defined to be $\mathsf{MK} = (g^\alpha, g^{\alpha a_0})$.

- $\mathsf{KeyGen}_{HIPE}(\mathsf{PP}, \mathsf{MK}, (\vec{v}_1, \ldots, \vec{v}_\ell))$: It parses $\vec{v}_i$ as $(v_{\mu_{i-1}+1}, \ldots, v_{\mu_i})$ and returns $\perp$ if $v_{\mu_i} = 0$ for $i \in [\ell]$. Otherwise, it picks $r_0, r_1, r_\ell, z_0, z_1 \xleftarrow{U} \mathbb{Z}_q$ and sets $r = r_0 + r_1 + \cdots + r_\ell$. It also picks $\mathsf{tagk}_{i,\mu_{i-1}+1}, \ldots, \mathsf{tagk}_{i,\mu_\ell - 1} \xleftarrow{U} \mathbb{Z}_q$ for $i \in [\ell]$. It then generates $\mathsf{sk}_{(\vec{v}_1, \ldots, \vec{v}_\ell)} = (D_1, \ldots, D_7, \{K_{i,0}, K_{\mu_{i-1}+1}, \ldots, K_{\mu_i - 1}, \mathsf{tagk}_{i,\mu_{i-1}+1}, \ldots, \mathsf{tagk}_{i,\mu_i - 1}\}_{i \in [\ell]})$ by computing

$$\begin{aligned} &D_1 = g^{\alpha a_0} \cdot w^r, \quad D_2 = g^{-\alpha} \cdot w_0^r \cdot g^{z_0}, \quad D_3 = B^{-z_0}, \quad D_4 = w_1^r \cdot g^{z_1}, \\ &D_5 = B^{-z_1}, \quad D_6 = B^{r_0}, \quad D_7 = g^{r_1}, \\ &\text{for } i \in [\ell] \text{ we set :} \\ &\qquad K_{i,0} = g^{r_i}, \quad K_j = (h_{\mu_i}^{-\frac{v_j}{v_{\mu_i}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r_i} \text{ for } j \in [\mu_{i-1} + 1, \mu_i - 1]. \end{aligned}$$

- $\mathsf{Enc}_{HIPE}(\mathsf{PP}, (\vec{x}_1, \ldots, \vec{x}_\ell), m)$: To encrypt $m \in \mathbb{G}_T$, it parses $\vec{x}_i = (x_{\mu_{i-1}-1}, \ldots, x_{\mu_i})$ for $i \in [\ell]$. It picks $s_0, s_1, t \xleftarrow{U} \mathbb{Z}_q$, $\mathsf{tagc}_1, \ldots, \mathsf{tagc}_\ell \xleftarrow{U} \mathbb{Z}_q$ and computes $c = (C_1, \ldots, C_7, C, E_0, E_1 \ldots, E_\ell, \mathsf{tagc}_1, \ldots, \mathsf{tagc}_\ell)$ where

$$\begin{aligned} &C_0 = m \cdot Z^{s_1}, \quad C_1 = B^{s_0 + s_1}, \quad C_2 = B_0^{s_0}, \quad C_3 = A_0^{s_0}, \quad C_4 = B_1^{s_1}, \\ &C_5 = A_1^{s_1}, \quad C_6 = \tau_0^{s_0} \cdot \tau_1^{s_1}, \quad C_7 = T_0^{s_0} \cdot T_1^{s_1} \cdot h_0^{-t}, \\ &C = m \cdot Z^{s_1}, \quad E_0 = g^t, \quad E_i = (h_0^{\mathsf{tagc}_i} \cdot \prod_{j=\mu_{i-1}+1}^{\mu_i} h_j^{x_j})^t \text{ for } i \in [\ell]. \end{aligned}$$

- $\mathsf{Dec}_{HIPE}(\mathsf{PP}, \mathsf{sk}_{(\vec{v}_1, \ldots, \vec{v}_\ell)}, c)$: To recover a message, it performs the following steps:
  - Compute $\mathsf{tagk}_i = \mathsf{tagk}_{i,\mu_{i-1}+1} x_{\mu_{i-1}+1} + \cdots + \mathsf{tagk}_{i,\mu_i - 1} x_{\mu_i - 1}$ for $i \in [\ell]$;

– Compute $W = \prod_{j=1}^{5} e(C_j, D_j) \cdot (\prod_{j=6}^{7} e(C_j, D_j))^{-1} = e(g, g)^{\alpha a_0 b s_1} \cdot e(g, h_0)^{(r_1+\cdots+r_\ell)t}$.

– Compute $W_i = (\frac{e((K_{\mu_{i-1}+1})^{x_{\mu_{i-1}+1}} \cdots (K_{\mu_i-1})^{x_{\mu_i-1}}, E_0)}{e(E_i, K_{i,0})})^{\frac{1}{\mathsf{tagk}_i - \mathsf{tagc}_i}} = e(g, h_0)^{r_i t}$ for $i \in [\ell]$.

– Recover the message as $m = C/Z^{s_1} = C/e(g, g)^{\alpha a_0 b s_1} \leftarrow C \cdot W_1 \cdots W_\ell \cdot W^{-1}$.

- $\mathsf{Del}_{HIPE}(\mathsf{PP}, (\vec{v}_1, \ldots, \vec{v}_\ell), \mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}, (\vec{v}_1, \ldots, \vec{v}_{\ell'}))$: It parses $\vec{v}_i$ as $(v_{\mu_{i-1}+1}, \ldots, v_{\mu_i})$ and returns $\perp$ if $v_{\mu_i} = 0$ for $i \in [\ell']$. It also parses $\mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}$ to be

$$D_1, \ldots, D_7, \{K_{i,0}, K_{\mu_{i-1}+1}, \ldots, K_{\mu_i-1}, \mathsf{tagk}_{i,\mu_{i-1}+1}, \ldots, \mathsf{tagk}_{i,\mu_i-1}\}_{i\in[\ell]}.$$

It then runs the following steps:

– Pick $r'_0, \ldots, r'_\ell, r'_{\ell+1}, \ldots, r'_{\ell'}, z'_0, z'_1 \xleftarrow{U} \mathbb{Z}_q$ and set $r' = r'_0 + \cdots + r'_{\ell'}$.

– For $i \in [\ell+1, \ell']$, pick $\mathsf{tagk}_{i,\mu_{i-1}+1}, \ldots, \mathsf{tagk}_{i,\mu_i-1} \xleftarrow{U} \mathbb{Z}_q$.

– Generate $\mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_{\ell'})} = (D'_1, \ldots, D'_7, \{K'_{i,0}, K'_{\mu_{i-1}+1}, \ldots, K'_{\mu_i-1}, \mathsf{tagk}_{i,\mu_{i-1}+1}, \ldots, \mathsf{tagk}_{i,\mu_i-1}\}_{i\in[\ell']})$ by computing

$$D'_1 = D_1 \cdot w^{r'}, \quad D'_2 = D_2 \cdot w_0^{r'} \cdot g^{z'_0}, \quad D'_3 = D_3 \cdot B^{-z'_0},$$
$$D'_4 = D_4 \cdot w_1^{r'} \cdot g^{z'_1}, \quad D'_5 = D_5 \cdot B^{-z'_1}, \quad D'_6 = D_6 \cdot B^{r'_0},$$
$$D'_7 = D_7 \cdot g^{r'_1+\cdots+r'_{\ell'}}.$$

Then for $i \in [\ell]$ we set :

$$K'_{i,0} = K_{i,0} \cdot g^{r'_i}, \quad K'_j = K_j \cdot (h_{\mu_i}^{-\frac{v_j}{v_{\mu_i}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r'_i} \quad \text{for } j \in [\mu_{i-1}+1, \mu_i - 1],$$

for $i \in [\ell+1, \ell']$ we set :

$$K'_{\vec{v}_i,0} = g^{r'_i}, \quad K'_j = (h_{\mu_i}^{-\frac{v_j}{v_{\mu_i}}} \cdot h_j \cdot h_0^{\mathsf{tagk}_{i,j}})^{r'_i} \quad \text{for } j \in [\mu_{i-1}+1, \mu_i - 1].$$

Theorem 6. *The HIPE scheme is fully secure under the DLIN and DBDH assumptions.*

The security proof is essentially similar to that for the multi-predicate IPE scheme in Section 3.1. The only difference is that showing indistinguishability of Game_{k-1} and Game_k requires a $(\mu_i - \mu_{i-1})$ equation trapdoor for each level. We embed these trapdoors into each level of challenge vector and one trapdoor into a guessed vector of the k-th reveal query. Hence, we do not discuss any further here.

D Relation between SE, HIPE and Multi-Predicate IPE

D.1 Generic Construction of HIPE from Multi-Predicate IPE

To constructing an HIPE scheme from a multi-predicate IPE scheme, we use the similar partition techniques of the transformation of SE to HIPE [8]. To construct an HIPE scheme with hierarchy $\vec{\mu} := (n, d; \mu_1, \ldots, \mu_d)$, we require a multi-predicate IPE scheme with dimension $n' = n + d$. Given a vector $\vec{v} \in \Phi_i$ and $\hat{b} \in \{0, 1\}$, we use $\vec{V}^{(i,\hat{b})} = (0, \ldots, 0, (\vec{v}, \hat{b}), 0, \ldots, 0)$ to denote an n'-dimensional vector, where $(\vec{v}, \hat{b})$ is embedded in the $(\mu_{i-1} + i)$-th up to the $(\mu_i + i)$-th scalars. Let $\vec{I}^{(i)} = (0, \ldots, 0, 1, 0, \ldots, 0)$ be an n'-dimensional vector, where the

$(\mu_i + i)$-th scalar is 1. The idea of our generic construction is essentially simple. We embed the i-th level of $\vec{\mu}$ of HIPE into the $(\mu_{i-1}+1)$-th up to the μ_i-th scalars of multi-predicate IPE.

Given a multi-predicate IPE scheme with five algorithms: Setup_{mIPE}, KeyGen_{mIPE}, Enc_{mIPE}, Dec_{mIPE}, and Del_{mIPE}, we construct an HIPE scheme with the corresponding five algorithms: Setup_{HIPE}, KeyGen_{HIPE}, Enc_{HIPE}, Dec_{HIPE}, and Del_{HIPE}, as follows:

- $\mathsf{Setup}_{HIPE}(\lambda, \vec{\mu})$ runs $\mathsf{Setup}_{mIPE}(\lambda, n')$ and outputs public parameters PP and a master key MK.
- $\mathsf{KeyGen}_{HIPE}(\mathsf{PP}, \mathsf{MK}, (\vec{v}_1, \ldots, \vec{v}_\ell))$ sets $\Gamma := \{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\}$. It runs $\mathsf{KeyGen}_{mIPE}(\mathsf{PP}, \mathsf{MK}, \Gamma)$ and outputs a secret key $\mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}$ with Γ.
- $\mathsf{Enc}_{HIPE}(\mathsf{PP}, (\vec{x}_1, \ldots, \vec{x}_h), m)$ sets $\vec{X}_i^{(i,0)} = \vec{I}^{(i)}$ for $i \in [h+1, d]$. It runs $\mathsf{Enc}_{mIPE}(\mathsf{PP}, \sum_{i=1}^{d} \vec{X}_i^{(i)}, m)$ and outputs a ciphertext c.
- $\mathsf{Dec}_{HIPE}(\mathsf{PP}, \mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}, c)$ runs $\mathsf{Dec}_{SE}(\mathsf{PP}, \mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}, c)$ and outputs a message m.
- $\mathsf{Del}_{HIPE}(\mathsf{PP}, (\vec{v}_1, \ldots, \vec{v}_\ell), \mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}, (\vec{v}_1, \ldots, \vec{v}_{\ell'}))$ where $\Gamma := \{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\}$ and sets $\Gamma' := \{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_{\ell'}^{(\ell',1)}\}$. It runs $\mathsf{Del}_{SE}(\mathsf{PP}, \Gamma, \mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_\ell)}, \Gamma')$ and outputs a secret key $\mathsf{sk}_{(\vec{v}_1,\ldots,\vec{v}_{\ell'})}$ with Γ'.

We now show that the resulting HIPE scheme works correctly and is indeed secure.

Theorem 7. *The HIPE scheme constructed from the multi-predicate IPE scheme works correctly. For any adversary $\mathcal{A}$ against the HIPE scheme in the same security model for the original multi-predicate IPE scheme, there is an adversary $\mathcal{D}$ against the multi-predicate IPE scheme, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{HIPE}(\lambda) \leq \mathsf{Adv}_{\mathcal{D}}^{mIPE}(\lambda).$$

Moreover, the HIPE scheme preserves properties (such as full/selective security, attribute/payload hiding, and short/constant ciphertexts) from the original multi-predicate IPE scheme.

Proof. From our construction, the only algorithm that needs to be considered is key delegation. Given a partial order pair $(\vec{v}_1, \ldots, \vec{v}_{\ell'}) \preceq (\vec{v}_1, \ldots, \vec{v}_\ell)$ of $\mathcal{K}_{HIPE}$, we transform it into a pair $(\{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_{\ell'}^{(\ell',1)}\}, \{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\})$ such that $\{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_{\ell'}^{(\ell',1)}\} \preceq \{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\}$ in the multi-predicate IPE setting. Thus, the key delegation algorithm also works.

Given a plaintext $((\vec{x}_1, \ldots, \vec{x}_h), m) \in \mathcal{X}_{HIPE}$ and hierarchical vectors $(\vec{v}_1, \ldots, \vec{v}_\ell) \in \mathcal{K}_{HIPE}$, we transform them into $\vec{X} = \sum_{i=1}^{d} \vec{X}_i^{(i,0)} \in \mathcal{X}_{mIPE}$ and $\{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\} \in \mathcal{K}_{mIPE}$ respectively. Then we have

- if $\ell > h$, we always let $\vec{X}_i^{(i,0)} = \vec{I}^{(i)}$ for $i \in [h+1, d]$ in our transformation, then $\vec{X} \cdot \vec{V}_i^{(i,1)} = 1$ for $i \in [h+1, \ell]$, implying that $\vec{X} \notin \mathcal{S}^{\perp}(\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)})$;
- if $\ell \leq h$, then

$$\begin{aligned} F((\vec{v}_1, \ldots, \vec{v}_\ell), ((\vec{x}_1, \ldots, \vec{x}_h), m)) = m &\Leftrightarrow \vec{x}_i \cdot \vec{v}_i = 0 \text{ for } i \in [\ell] \\ &\Leftrightarrow \vec{X} \cdot V_i^{(i,1)} = 0 \text{ for } i \in [\ell] \\ &\Leftrightarrow \vec{X} \in \{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\} \\ &\Leftrightarrow F(\{\vec{V}_1^{(1,1)}, \ldots, \vec{V}_\ell^{(\ell,1)}\}, (\vec{X}, m)) = m. \end{aligned}$$

This shows that the resulting HIPE scheme inherits the decryptability from the original multi-predicate IPE scheme, i.e., we have $\mathcal{F}((\vec{v}_1, \ldots, \vec{v}_\ell), ((\vec{x}_1, \ldots, \vec{x}_h), m)) = m$ iff $\ell \leq h$ and $\vec{x}_i \cdot \vec{v}_i = 0$ for all $i \in [\ell]$ as defined in Appendix **??**.

The remaining part of the theorem could be obtained by using the same argument of the proof in Theorem 4. □

D.2 Generic Construction of Multi-Predicate IPE from SE

To construct an n-dimensional multi-predicate IPE scheme, we require an SE scheme with same dimension. Given an HIPE scheme with five algorithms: Setup_{SE}, KeyGen_{SE}, Enc_{SE}, Dec_{SE}, and Del_{SE}, we construct an SE scheme with the corresponding five algorithms: Setup_{mIPE}, KeyGen_{mIPE}, Enc_{mIPE}, Dec_{mIPE}, and Del_{mIPE}, as follows:

- $\mathsf{Setup}_{mIPE}(\lambda, n)$ runs $\mathsf{Setup}_{HIPE}(\lambda, n)$ and outputs public parameters PP and a master key MK.
- $\mathsf{KeyGen}_{mIPE}(\mathsf{PP}, \mathsf{MK}, \Gamma)$ first runs $\mathsf{BasisGen}(\mathcal{S}(\Gamma))$ and outputs $\mathcal{B}^\perp$. It then runs $\mathsf{KeyGen}_{SE}(\mathsf{PP}, \mathsf{MK}, \mathcal{S}(\mathcal{B}^\perp))$ and outputs a secret key sk_Γ with $\mathcal{S}(\mathcal{B}^\perp)$.
- $\mathsf{Enc}_{mIPE}(\mathsf{PP}, \vec{x}, m)$ runs $\mathsf{Enc}_{HIPE}(\mathsf{PP}, \vec{x}, m)$ and outputs a ciphertext c.
- $\mathsf{Dec}_{mIPE}(\mathsf{PP}, \mathsf{sk}_\Gamma, c)$ runs $\mathsf{Dec}_{HIPE}(\mathsf{PP}, \mathsf{sk}_\Gamma, c)$ and outputs a message m.
- $\mathsf{Del}_{mIPE}(\mathsf{PP}, \Gamma, \mathsf{sk}_\Gamma, \Gamma')$ where the secret key sk_Γ is associated with a space $\mathcal{S}(\mathcal{B}^\perp)$. It first runs $\mathsf{BasisGen}(\mathcal{S}(\Gamma'))$ and outputs $\mathcal{B}'^\perp$. It then runs $\mathsf{Del}_{SE}(\mathsf{PP}, \mathcal{S}(\mathcal{B}^\perp), \mathsf{sk}_\Gamma, \mathcal{S}(\mathcal{B}'^\perp))$ and outputs a secret key $\mathsf{sk}_{\Gamma'}$ with $\mathcal{S}(\mathcal{B}'^\perp)$.

We now show that the resulting multi-predicate IPE scheme works correctly and is indeed secure.

Theorem 8. *The multi-predicate IPE scheme constructed from the SE scheme works correctly. For any adversary $\mathcal{A}$ against the multi-predicate IPE scheme in the same security model for the original SE scheme, there is an adversary $\mathcal{D}$ against the SE scheme, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{Adv}^{mIPE}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{SE}_{\mathcal{D}}(\lambda).$$

Moreover, the multi-predicate IPE scheme preserves properties (such as full/selective security, attribute/payload hiding, and short/constant ciphertexts) from the original SE scheme.

Proof. From our construction, the only algorithm that needs to be considered is delegation. Given a partial order pair $\Gamma' \preceq \Gamma$ of $\mathcal{K}_{mIPE}$, we transform it into a pair $(\mathcal{S}^\perp(\Gamma'), \mathcal{S}^\perp(\Gamma))$ such that $\mathcal{S}^\perp(\Gamma') \preceq \mathcal{S}^\perp(\Gamma)$ in the SE setting. Thus, the key delegation algorithm also works.

Given a plaintext $(\vec{x}, m) \in \mathcal{X}_{mIPE}$ and a set $\Gamma =: \{\vec{v}_1, \ldots, \vec{v}_\ell\} \in \mathcal{K}_{mIPE}$, we transform them into $(\vec{x}, m) \in \mathcal{X}_{SE}$ and $\mathcal{S}^\perp(\Gamma) \in \mathcal{K}_{SE}$ respectively. Then we have

$$\begin{aligned}\mathcal{F}(\Gamma, (\vec{x}, m)) = m &\Leftrightarrow \vec{x} \cdot \vec{v}_i = 0 \text{ for all } \vec{v}_i \in \Gamma \\ &\Leftrightarrow \vec{x} \in \mathcal{S}^\perp(\Gamma) \\ &\Leftrightarrow \mathcal{F}(\mathcal{S}^\perp(\Gamma), (\vec{x}, m)) = m.\end{aligned}$$

This shows that the resulting multi-predicate IPE scheme inherits the decryptability from the original SE scheme, i.e., we have $\mathcal{F}(\Gamma, (\vec{x}, m)) = m$ iff $\vec{x} \cdot \vec{v}_i = 0$ for all $\vec{v}_i \in \Gamma$ as defined in Section 2.2.

The remaining part of the theorem could be obtained by using the same argument of the proof in Theorem 4. □

D.3 Generic Construction of Multi-Predicate IPE from HIPE

To construct a multi-predicate IPE scheme, we require an HIPE scheme with hierarchy $\vec{\mu} := ((n-1)n, n-1; n, 2n, \ldots, (n-1)n)$. Given an HIPE scheme with five algorithms: Setup_{HIPE}, KeyGen_{HIPE}, Enc_{HIPE}, Dec_{HIPE}, and Del_{HIPE}, we construct an SE scheme with the corresponding five algorithms: Setup_{mIPE}, KeyGen_{mIPE}, Enc_{mIPE}, Dec_{mIPE}, and Del_{mIPE}, as follows:

- $\mathsf{Setup}_{mIPE}(\lambda, n)$ runs $\mathsf{Setup}_{HIPE}(\lambda, \vec{\mu})$ and outputs public parameters PP and a master key MK.
- $\mathsf{KeyGen}_{mIPE}(\mathsf{PP}, \mathsf{MK}, \Gamma)$ phases Γ as $\{\vec{v}_1, \ldots, \vec{v}_\ell\}$. It runs $\mathsf{KeyGen}_{HIPE}(\mathsf{PP}, \mathsf{MK}, (\vec{v}_1, \ldots, \vec{v}_\ell))$ and outputs a secret key sk_Γ.
- $\mathsf{Enc}_{mIPE}(\mathsf{PP}, \vec{x}, m)$ runs $\mathsf{Enc}_{HIPE}(\mathsf{PP}, (\vec{x}, \ldots, \vec{x}), m)$ and outputs a ciphertext c.
- $\mathsf{Dec}_{mIPE}(\mathsf{PP}, \mathsf{sk}_\Gamma, c)$ runs $\mathsf{Dec}_{HIPE}(\mathsf{PP}, \mathsf{sk}_\Gamma, c)$ and outputs a message m.
- $\mathsf{Del}_{mIPE}(\mathsf{PP}, \Gamma, \mathsf{sk}_\Gamma, \Gamma')$ where $\Gamma = \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ and phases $\Gamma' = \{\vec{v}_1, \ldots, \vec{v}_{\ell'}\}$. It runs $\mathsf{Del}_{HIPE}(\mathsf{PP}, (\vec{v}_1, \ldots, \vec{v}_\ell), \mathsf{sk}_\mathcal{S}, (\vec{v}_1, \ldots, \vec{v}_{\ell'}))$ and outputs a secret key $\mathsf{sk}_{\Gamma'}$.

We now show that the resulting multi-predicate IPE scheme works correctly and is indeed secure.

Theorem 9. *The multi-predicate IPE scheme constructed from the HIPE scheme works correctly. For any adversary $\mathcal{A}$ against the multi-predicate IPE scheme in the same security model for the original HIPE scheme, there is an adversary $\mathcal{D}$ against the HIPE scheme, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{Adv}^{mIPE}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{HIPE}_{\mathcal{D}}(\lambda).$$

Moreover, the multi-predicate IPE scheme preserves properties (such as full/selective security, attribute/payload hiding, and short/constant ciphertexts) from the original HIPE scheme.

Proof. From our construction, the only algorithm that needs to be considered is key delegation. Given a partial order pair $\Gamma' := \{\vec{v}_1, \ldots, \vec{v}_{\ell'}\} \preceq \Gamma := \{\vec{v}_1, \ldots, \vec{v}_\ell\}$ of $\mathcal{K}_{mIPE}$, we transform it into a pair $((\vec{v}_1, \ldots, \vec{v}_\ell), (\vec{v}_1, \ldots, \vec{v}_{\ell'}))$ such that $\vec{v}_1, \ldots, \vec{v}_\ell \preceq \vec{v}_1, \ldots, \vec{v}_{\ell'}$ in the HIPE setting. Thus, the key delegation algorithm also works.

Given a plaintext $(\vec{x}, m) \in \mathcal{X}_{mIPE}$ and hierarchical vectors $\Gamma = \{\vec{v}_1, \ldots, \vec{v}_\ell\} \in \mathcal{K}_{mIPE}$, we transform them into $(\vec{x}, \ldots, \vec{x}) \in \mathcal{X}_{HIPE}$ and $(\vec{v}_1, \ldots, \vec{v}_\ell) \in \mathcal{K}_{HIPE}$ respectively. Then we have

$$\begin{aligned} F(\Gamma, (\vec{x}, m)) = m &\Leftrightarrow \vec{x} \cdot \vec{v}_i = 0 \text{ for } i \in [\ell] \\ &\Leftrightarrow F((\vec{v}_1, \ldots, \vec{v}_\ell), (\vec{x}, m)) = m. \end{aligned}$$

This shows that the resulting multi-predicate IPE scheme inherits the decryptability from the original HIPE scheme, i.e., we have $\mathcal{F}(\Gamma, (\vec{x}, m)) = m$ iff $\vec{x} \cdot \vec{v}_i = 0$ for all $\vec{v}_i \in \Gamma$ as defined in Section 2.2.

The remaining part of the theorem could be obtained by using the same argument of the proof in Theorem 4. □