Hierarchical Structure in Attribute-Based Inner-Product Functional Encryption

Hirotomo Shinoki
Hitachi, Ltd., Japan
hirotomo.shinoki.sw@hitachi.com

Abstract

Attribute-based inner-product functional encryption (AB-IPFE), introduced by Abdalla et al. (Asiacrypt'20), is a cryptosystem that combines the access-control capability of attribute-based encryption (ABE) with the linear-computation capability of inner-product functional encryption. By introducing a hierarchical structure between these two functionalities, we can decompose the key generation algorithm into two steps. While such a structure has been extensively studied in contexts such as hierarchical IBE and delegatable ABE, it has received little attention in AB-IPFE despite its naturalness. In this paper, we formalize a framework for two-level hierarchies in AB-IPFE and classify existing schemes according to their hierarchizability.

In particular, we focus on adaptively secure schemes that support expressive access control, such as arithmetic programs in the public-index setting and attribute-hiding inner-product predicates in the private-index setting. To the best of our knowledge, the only efficient constructions known to meet these requirements are those obtained from the framework of Abdalla et al. in the private-index setting and from the scheme of Datta and Pal (Asiacrypt'21) in the public-index setting. We propose several new pairing-based constructions that achieve adaptive security.

First, we revisit the approach of Abdalla et al. Based on their framework, we propose three types of constructions that trade off hierarchizability, ciphertext size, and secret-key size. These constructions are based on predicate encodings and therefore support arithmetic span programs or attribute-hiding inner-product predicates.

Second, we revisit the approach of Datta and Pal. Their scheme supports attribute-weighted sums, which differ slightly from our target functionality. Although this already yields an AB-IPFE scheme, the resulting scheme is limited to the non-zero-type key-policy setting. We efficiently adapt their scheme to obtain AB-IPFE schemes that also support zero-type predicates, and we propose a ciphertext-policy variant. All of the resulting constructions can be hierarchized, but their intermediate secret keys are large. We also propose variant schemes with shorter intermediate secret keys, at the cost of somewhat larger final secret keys. These schemes support arithmetic branching programs.

Contents

1 Introduction **3**
- 1.1 Our results . . . 3
- 1.2 Technical Overview . . . 5

2 Preliminaries **7**
- 2.1 Bilinear Groups . . . 8
- 2.2 Functional Encryption . . . 8
- 2.3 Predicate Encoding and Function Encoding . . . 9
- 2.4 Arithmetic Branching Programs and Arithmetic Key Garbling . . . 11
- 2.5 Slotted Inner-Product Functional Encryption . . . 11

3 Attribute-based Inner-Product Functional Encryption with Hierarchical Structure **12**
- 3.1 Syntax and Security . . . 12
- 3.2 Hierarchical Structure in Existing AB-IPFE . . . 13

4 Hierarchizability of Abdalla et al.'s AD-IND secure scheme **15**
- 4.1 Hierarchical Function Encoding . . . 15
- 4.2 Construction of Function Encoding (1) . . . 16
- 4.3 Construction of Function Encoding (2) . . . 18

5 AD-IND Secure Hierarchical Scheme with Compact Ciphertext from Predicate Encodings **19**
- 5.1 Normalized Predicate Encoding . . . 19
- 5.2 Construction . . . 19
- 5.3 Proof of Theorem 3 . . . 21

6 AD-SIM Secure Constructions for ABPs **28**
- 6.1 Construction in Key-Policy Setting . . . 28
- 6.2 Proof of Theorem 4 . . . 30
- 6.3 Construction in Ciphertext-Policy Setting . . . 33
- 6.4 Proof of Theorem 5 . . . 35

7 AD-SIM Secure Constructions for ABPs with Shorter Attribute Keys **39**
- 7.1 Constructions . . . 39
- 7.2 Proof of Theorem 6 . . . 42

1 Introduction

Functional encryption (FE) [BSW11, O'N10] is a cryptosystem in which a user possessing a secret key corresponding to a function f can recover only the partial information $f(x)$ of the message x from the ciphertext. However, functional encryption supporting arbitrary circuits still faces challenges in terms of efficiency. Consequently, active research is underway to develop more efficient schemes by restricting the classes of functions supported. Among these, inner-product functional encryption (IPFE) [ABDP15] is one of the most well-known schemes, specializing in vector inner-product computations. The message takes the form of a d-dimensional vector $\boldsymbol{x}$, and when the ciphertext of $\boldsymbol{x}$ is decrypted using the secret key corresponding to a d-dimensional vector $\boldsymbol{y}$, the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is obtained. In addition, encryption schemes with access control such as identity-based encryption (IBE) and attribute-based encryption (ABE), which have been studied for longer, can also be regarded as a type of FE with restricted functionality.

Attribute-based IPFE. In 2020, Abdalla et al. [ACGU20] proposed attribute-based inner-product functional encryption (AB-IPFE) that supports functions that return inner product values only when the attributes satisfy specific conditions. The message consists of a pair of an attribute $\tilde{x}$ and a vector $\boldsymbol{x}$, and the secret key is associated with a pair of an attribute $\tilde{y}$ and a vector $\boldsymbol{y}$. If $\tilde{x}$ and $\tilde{y}$ satisfy a specific relation R, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is obtained; otherwise, decryption fails. AB-IPFE is a scheme that combines the capabilities of ABE and IPFE, but it cannot be constructed by simply combining them. The constructions proposed by Abdalla et al. [ACGU20] are based on either pairings or lattices. The pairing-based schemes can be broadly categorized into two types. The first is a transformation from predicate encoding [CGW15, Wee14], which achieves selective simulation (SEL-SIM) security. The second is a transformation from functional encoding [ACGU20], which achieves adaptive indistinguishability (AD-IND). The lattice-based schemes support only identity-based access control. One is SEL-IND secure in the standard model, and the other is AD-IND secure in the random oracle model.

Following this work, Lai et al. [LLW21] proposed new lattice-based AB-IPFE schemes. By introducing a new lattice sampling technique, they constructed an AD-IND secure identity-based IPFE (IB-IPFE) and a SEL-IND secure collusion-bounded AB-IPFE, both of which are based on the LWE assumption in the standard model. Nguyen et al. [NPP22] proposed a new AD-IND secure key-policy AB-IPFE scheme and its extension to the multi-client setting by utilizing dual pairing vector spaces. None of these schemes achieves both adaptive security and simulation security. Edamura and Takayasu [ET25] proposed AD-SIM secure IB-IPFE constructions. In addition, many extensions have been proposed, such as constructions in the multi-client/multi-input setting [ATY23, DMPT25, NPP22, NPP25], the unbounded setting [CT26, DDM+23], the decentralized setting [DP23, NPS25], and the registered setting [DPY24, ZZZ+25].

Independently, Abdalla et al. [AGW20] proposed FE for attribute-weighted sums (AWS-FE), which is closely related to AB-IPFE. In AWS-FE, the message consists of the public part $\boldsymbol{x}$ and the private d-dimensional vector $\boldsymbol{z}$. The secret key is associated with a function f that outputs a d-dimensional vector. Decryption by sk_f returns the value of the inner product $\langle f(\boldsymbol{x}), \boldsymbol{z} \rangle$. Their scheme was proved to be SEL-SIM secure. Later, Datta and Pal [DP21] proposed an AD-SIM secure AWS-FE scheme.

Hierarchical FE. The hierarchical structure refers to the property that secret keys with weaker authority can be derived from secret keys with stronger authority. It is used to distribute the burden of key generation. Starting with the proposal of Hierarchical IBE (HIBE) in 2002 [GS02, HL02], the hierarchical structure (i.e. delegatability) of encryption schemes has been studied for a long time. It has been generalized to delegatable ABE [BGG+14, GVW15, GPSW06, LW11] and hierarchical FE [ABG+13, BCG+17]. In delegatable ABE, the secret key $\mathsf{sk}_{f \wedge g}$ can be derived from sk_f for any pair of policies f, g. Similarly, in hierarchical FE, the secret key $\mathsf{sk}_{g \circ f}$ can be derived from sk_f for any pair of functions f, g.

1.1 Our results

In this paper, we formalize the hierarchical structures of AB-IPFE. The key generation algorithm in AB-IPFE takes two inputs: an attribute $\tilde{y}$ and a vector $\boldsymbol{y}$. Therefore, we can consider two types of two-level hierarchical structures. Specifically, we introduce an attribute key $\mathsf{sk}_{\tilde{y}}$ that allows one

Table 1: Adaptively Secure AB-IPFE Schemes Supporting Arithmetic Programs

	Attribute	Security	CT	SK	$\mathrm{SK}_{\mathrm{Att}}$	$\mathrm{SK}_{\mathrm{Vec}}$
[ACGU20] + Sect. 4.2	PE	IND	$O(\lvert\mathsf{ct}\rvert_{\mathcal{P}} + d)$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}})$	–	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}})$
[ACGU20] + Sect. 4.3	PE	IND	$O(\lvert\mathsf{ct}\rvert_{\mathcal{P}} \cdot d)$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}})$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}} \cdot d)$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}})$
Sect. 5	PE	IND	$O(\lvert\mathsf{ct}\rvert_{\mathcal{P}} + d)$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}} \cdot d)$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}} \cdot d)$	$O(\lvert\mathsf{sk}\rvert_{\mathcal{P}} \cdot d)$
Sect. 6.1	ABP (KP)	SIM	$O(n + d)$	$O(mn + d)$	$O(d(mn + d))$	$O(d)$
Sect. 6.3	ABP (CP)	SIM	$O(mn + d)$	$O(n + d)$	$O(d(n + d))$	$O(d)$
Sect. 7.1	ABP (KP)	SIM	$O(n + d)$	$O(dmn)$	$O(dmn)$	$O(d)$
Sect. 7.1	ABP (CP)	SIM	$O(mn + d)$	$O(dn)$	$O(dn)$	$O(d)$

"PE" denotes predicate encodings, and "ABP" denotes arithmetic branching programs. "CT", "SK", "$\mathrm{SK}_{\mathrm{Att}}$", and "$\mathrm{SK}_{\mathrm{Vec}}$" denote the numbers of group elements in ciphertexts, secret keys, attribute keys, and vector keys respectively. d is the dimension of the message vector. $\lvert\mathsf{ct}\rvert_{\mathcal{P}}$ and $\lvert\mathsf{sk}\rvert_{\mathcal{P}}$ are parameters determined by the underlying predicate encoding. m and n are parameters used in ABP.

to derive $\mathsf{sk}_{\tilde{y},\boldsymbol{y}}$ for any $\boldsymbol{y}$. Conversely, we can introduce a vector key $\mathsf{sk}_{\boldsymbol{y}}$ that allows us to derive $\mathsf{sk}_{\tilde{y},\boldsymbol{y}}$ for any $\tilde{y}$. We investigate whether the two types of structures can be added to the existing (collusion-unbounded) AB-IPFE schemes. This paper does not address the hierarchical structure within attributes or within vectors. Recently, Zhu et al. [ZZZ+25] have introduced similar two-level structures in the context of registered AB-IPFE. However, their efficient constructions are secure in the registration-bounded setting, which means that they yield only collusion-bounded AB-IPFE. Thus, their constructions do not cover our target setting.

Technically, we mainly focus on the adaptively secure schemes supporting arithmetic programs or attribute-hiding inner-product predicates, where there are trade-offs between hierarchical structure and efficiency. Table 1 summarizes our new AB-IPFE constructions. Note that the schemes in Section 4.2, 6.1, and 6.3 are of interest even without introducing hierarchical structures.

To the best of our knowledge, Abdalla et al.'s pairing-based construction [ACGU20] (which we call ACGU20ad) is the only adaptively secure scheme from standard assumptions that directly supports ciphertext-policy access control or attribute-hiding inner-product predicate. However, this scheme is based on function encodings, which have not been well studied. In fact, known constructions are limited to the ones proposed by Abdalla et al., which support identity-based predicates, inner-product predicates, attribute-hiding inner-product predicates, or ciphertext-policy monotone span programs (MSPs). Unfortunately, the security analysis of the encoding for ciphertext-policy MSPs has a flaw: admissible queries are not taken into account.

1. In Section 4.2, we propose a generic transformation from predicate encodings to function encodings. Predicate encoding supports various types of access control, including ciphertext-policy arithmetic span program (ASP), key-policy ASP, and full attribute-hiding inner-product predicate. Combining this transformation with the ACGU20ad framework yields an AB-IPFE scheme with ciphertexts of size $O(\lvert\mathsf{ct}\rvert_{\mathcal{P}} + d)$ and secret keys of size $O(\lvert\mathsf{sk}\rvert_{\mathcal{P}})$. In terms of hierarchizability, the resulting AB-IPFE supports only the vector-on-top hierarchical structure.
2. In Section 4.3, we propose another transformation from predicate encodings to function encodings. Combining this transformation with the ACGU20ad framework yields an AB-IPFE scheme that supports both types of hierarchical structures, at the cost of larger ciphertexts. Namely, it has ciphertexts of size $O(\lvert\mathsf{ct}\rvert_{\mathcal{P}} \cdot d)$ and secret keys of size $O(\lvert\mathsf{sk}\rvert_{\mathcal{P}})$.

These two constructions exhibit a trade-off between compact ciphertexts and support for the attribute-on-top hierarchy. At the cost of larger secret-key size, our third AB-IPFE construction achieves both of them by using a modification of the ACGU20ad framework.

3. In Section 5, we propose a transformation from predicate encodings to AD-IND secure AB-IPFE schemes that support both types of hierarchical structures. The resulting scheme has ciphertexts of size $O(\lvert\mathsf{ct}\rvert_{\mathcal{P}} + d)$ and secret keys of size $O(\lvert\mathsf{sk}\rvert_{\mathcal{P}} \cdot d)$.

Another approach is based on arithmetic branching programs (ABPs). An adaptively secure ABE scheme with ABP-based access control was introduced by Lin and Luo [LL20], and its core technique was used in the AWS-FE scheme of Datta and Pal [DP21]. This AWS-FE scheme can be used to obtain AB-IPFE, but its access-control capability is limited. In fact, it supports only non-zero-type predicates, for which decryption is allowed if $f(\tilde{x}) \neq 0$. This type of predicate can be captured by an AWS-FE scheme capable of computing $\langle f(\tilde{x}) \cdot \boldsymbol{y}, \boldsymbol{x} \rangle$, since we consider the case where $\tilde{x}$ is public. However, zero-type predicates, for which decryption succeeds if $f(\tilde{x}) = 0$, cannot be embedded into AWS-FE in such an efficient way. On the other hand, Lin and Luo's ABE scheme supports both zero-type and non-zero-type predicates. In Section 6.1, by combining these techniques, we construct an AD-SIM-secure AB-IPFE scheme that supports both types of predicates in the key-policy setting. Although Lin and Luo's ABE scheme is a key-policy scheme, in Section 6.3, we also construct a ciphertext-policy AB-IPFE scheme using a similar technique.

Our ABP-based constructions are hierarchizable and have compact ciphertexts and secret keys; however, the attribute keys can be large. In fact, their size is quadratic in d. When d is much larger than the other parameters, this may be undesirable. In Section 7.1, we propose variant schemes whose attribute-key size is linear in d, at the cost of larger secret keys.

1.2 Technical Overview

Constructions based on predicate encodings. We start with a function encoding, which is a 1-key 1-ciphertext secure secret-key FE. As an example, we describe the function encoding for attribute-hiding inner-product predicate IPFE, proposed by Abdalla et al. Let ℓ be the length of attributes and d be the length of vectors. The master secret key is represented as $(\boldsymbol{u}, \mathbf{V}, \boldsymbol{w}) \xleftarrow{\$} \mathbb{Z}_p^d \times \mathbb{Z}_p^{d \times \ell} \times \mathbb{Z}_p^d$. The ciphertext and secret key are given by

$$\mathsf{Enc}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = (\boldsymbol{u}\tilde{\boldsymbol{x}}^\top + \mathbf{V}, \boldsymbol{w} + \boldsymbol{x}), \quad \mathsf{KeyGen}(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = \boldsymbol{y}^\top \mathbf{V} \tilde{\boldsymbol{y}} + \boldsymbol{y}^\top \boldsymbol{w}.$$

Note that $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$ are public. When $\langle \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \rangle = 0$, we can compute $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ as

$$(\boldsymbol{w} + \boldsymbol{x})^\top \boldsymbol{y} - (\boldsymbol{y}^\top \mathbf{V} \tilde{\boldsymbol{y}} + \boldsymbol{y}^\top \boldsymbol{w}) + \boldsymbol{y}^\top (\boldsymbol{u}\tilde{\boldsymbol{x}}^\top + \mathbf{V}) \tilde{\boldsymbol{y}} = \boldsymbol{x}^\top \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{u} \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}}.$$

We observe that this construction leaves room for improvement. Concretely, we use $u \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^\ell$ instead of $\boldsymbol{u}, \mathbf{V}$. In the ciphertext, $\boldsymbol{u}\tilde{\boldsymbol{x}}^\top + \mathbf{V}$ is replaced with $u\tilde{\boldsymbol{x}}^\top + \boldsymbol{v}^\top$. In the secret key, $\boldsymbol{y}^\top \mathbf{V}$ and $\boldsymbol{y}^\top \boldsymbol{u}$ are replaced with $\boldsymbol{v}^\top$ and u respectively. The resulting scheme is described as follows:

$$\mathsf{Enc}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = (u\tilde{\boldsymbol{x}}^\top + \boldsymbol{v}^\top, \boldsymbol{w} + \boldsymbol{x}), \quad \mathsf{KeyGen}(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = \boldsymbol{v}^\top \tilde{\boldsymbol{y}} + \boldsymbol{y}^\top \boldsymbol{w}.$$

This modification significantly reduces the size of the ciphertext from $(d + d\ell)|\,\mathbb{Z}_p\,|$ to $(d + \ell)|\,\mathbb{Z}_p\,|$. Furthermore, the modified scheme satisfies the required security notion, and can be considered superior to the original scheme.

However, once we take into account the hierarchical structures introduced in this paper, the modification has a drawback. Consider adding the attribute-on-top hierarchy to the function encoding. In other words, we want to construct a function $\mathsf{KeyGen}_{\mathsf{Att}}$ such that $\mathsf{KeyGen}(\tilde{\boldsymbol{y}}, \boldsymbol{y})$ is computable from $\mathsf{KeyGen}_{\mathsf{Att}}(\tilde{\boldsymbol{y}})$ without using the master secret key. In the original scheme, this can be done simply by setting $\mathsf{KeyGen}_{\mathsf{Att}}(\tilde{\boldsymbol{y}}) = \mathbf{V}\tilde{\boldsymbol{y}} + \boldsymbol{w}$. By contrast, in the modified scheme, it appears difficult to embed such a simple structure. In fact, setting $\mathsf{KeyGen}_{\mathsf{Att}}(\tilde{\boldsymbol{y}}) = (\boldsymbol{v}^\top \tilde{\boldsymbol{y}}, \boldsymbol{w})$ would break security, since $\boldsymbol{w}$ would be revealed even when $\langle \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \rangle \neq 0$. Thus, these schemes exhibit a trade-off between ciphertext compactness and support for the attribute-on-top hierarchy. This trade-off naturally carries over to the full-fledged public-key AB-IPFE scheme. In Section 4, we generalize the above observation and show that these two types of schemes can be generically constructed from predicate encodings.

It remains to construct a scheme that has compact ciphertexts and supports the attribute-on-top hierarchical structure. To this end, we introduce component-wise encodings. Let $\rho_1, \ldots, \rho_d$ be random values in $\mathbb{Z}_p$ whose sum is 0. For each $j \in [d]$, we consider the following encodings:

$$\mathsf{Enc}(\tilde{\boldsymbol{x}}, \boldsymbol{x}) = (u\tilde{\boldsymbol{x}}^\top + \boldsymbol{v}^\top, \boldsymbol{w} + \boldsymbol{x}), \quad \mathsf{KeyGen}_j(\tilde{\boldsymbol{y}}, \boldsymbol{y}, \rho_j) = \boldsymbol{y}[j] \cdot (\boldsymbol{v}^\top \tilde{\boldsymbol{y}} + \boldsymbol{w}[j]) - \rho_j.$$

If $\langle \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \rangle = 0$, using $\mathsf{Enc}(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ and $\mathsf{KeyGen}_j(\tilde{\boldsymbol{y}}, \boldsymbol{y}, \rho_j)$, we can compute

$$\boldsymbol{y}[j] \cdot (\boldsymbol{w} + \boldsymbol{x})[j] - (\boldsymbol{y}[j](\boldsymbol{v}^\top \tilde{\boldsymbol{y}} + \boldsymbol{w}[j]) - \rho_j) + \boldsymbol{y}[j] \cdot (u\tilde{\boldsymbol{x}}^\top + \boldsymbol{v}^\top)\tilde{\boldsymbol{y}} = \boldsymbol{x}[j]\boldsymbol{y}[j] + \rho_j.$$

By summing these values over all $j \in [d]$, we can recover the value $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Furthermore, the hierarchical structure can be added by setting $\mathsf{KeyGen}_{\mathsf{Att},j}(\tilde{\boldsymbol{y}}) = \boldsymbol{v}^\top \tilde{\boldsymbol{y}} + \boldsymbol{w}[j]$. However, this decryption procedure does not carry over straightforwardly to the full-fledged public-key AB-IPFE. In AB-IPFE schemes obtained from the ACGU20ad framework, decryption computes $[(s\boldsymbol{a})^\top \mathbf{W}_0(r\boldsymbol{b})]_T, [\langle \boldsymbol{x}, \boldsymbol{y} \rangle \cdot (s\boldsymbol{a})^\top \mathbf{W}_0(r\boldsymbol{b})]_T$ where $r \xleftarrow{\$} \mathbb{Z}_p$ is the randomness used in encryption, $s \xleftarrow{\$} \mathbb{Z}_p$ is the randomness used in key generation, and the remaining parameters $\boldsymbol{a} \in \mathbb{Z}_p^2, \mathbf{W}_0 \in \mathbb{Z}_p^{2\times 3}, \boldsymbol{b} \in \mathbb{Z}_p^3$ are part of the master secret key. $[(s\boldsymbol{a})^\top \mathbf{W}_0]_1$ appears in the ciphertext, and $[r\boldsymbol{b}]_2$ appears in the secret key. It is assumed that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ lies in a set of polynomial size, which enables one to recover $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ by brute-forcing the discrete logarithm. When component-wise encoding is applied, we can compute $[(s\boldsymbol{a})^\top \mathbf{W}_0(r_j\boldsymbol{b})]_T$ and $\left[\boldsymbol{x}[j]\boldsymbol{y}[j](s\boldsymbol{a})^\top \mathbf{W}_0(r_j\boldsymbol{b}) + (s\boldsymbol{a})^\top \boldsymbol{\rho}_j\right]_T$ where $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_d \in \mathbb{Z}_p^2$ are random vectors such that their sum is $\mathbf{0}$, and $r_1, \ldots, r_d \in \mathbb{Z}_p$ are independent random values. Since the base for logarithm $[(s\boldsymbol{a})^\top \mathbf{W}_0(r_j\boldsymbol{b})]_T$ depends on j, we cannot efficiently compute $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$.

To overcome this challenge, we make several modifications to the ACGU20ad framework. We first give an informal overview of the idea. Using the dual-system encryption technique, the security of ACGU20ad can be reduced to that of a secret-key variant. In this variant, $(s\boldsymbol{a})^\top \mathbf{W}_0$ in the ciphertext is replaced by a secret vector $\boldsymbol{w}^\top$. Let $(\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3)$ and $(\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \boldsymbol{b}_3^*)$ be dual orthogonal bases with $\boldsymbol{b}_1^* = \boldsymbol{b}$. The challenge ciphertext is decomposed into its $\boldsymbol{b}_i$-components for $i \in [3]$. The goal is to change the challenge ciphertext from $C_0 = \mathsf{Enc}(\tilde{\boldsymbol{x}}_0, \boldsymbol{x}_0)$ to $C_1 = \mathsf{Enc}(\tilde{\boldsymbol{x}}_1, \boldsymbol{x}_1)$. First, the $\boldsymbol{b}_3$-component is switched to the corresponding component of C_1. This change can be performed because $[r\boldsymbol{b}_1^*]_2$ in the secret key is orthogonal to the $\boldsymbol{b}_3$-component. Using the DDH assumption and the security of the function encoding, $[r\boldsymbol{b}_1^*]_2$ in the secret key is switched to $[r\boldsymbol{b}_2^*]_2$ and eventually to $[r\boldsymbol{b}_3^*]_2$. Then, by orthogonality, the $\boldsymbol{b}_1$- and $\boldsymbol{b}_2$-components can be switched to the corresponding components of C_1.

A first attempt is to fix $r = 1$, but this invalidates the security proof. Indeed, the randomness of r is essential when switching $[r\boldsymbol{b}_1^*]_2$ in the secret key to $[r\boldsymbol{b}_2^*]_2$ and, later, $[r\boldsymbol{b}_2^*]_2$ to $[r\boldsymbol{b}_3^*]_2$ using the DDH assumption. To reintroduce randomness and make the switching argument work again, we consider using a 4-dimensional vector space with dual bases $(\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_4)$ and $(\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \boldsymbol{b}_3^*, \boldsymbol{b}_4^*)$. We also replace $[\boldsymbol{b}_1^*]_2$ in the secret key with $[\boldsymbol{b}_1^* + r\boldsymbol{b}_4^*]_2$. To make $\boldsymbol{w}^\top(\boldsymbol{b}_1^* + r\boldsymbol{b}_4^*)$ independent of r, $\boldsymbol{w}$ must lie in $\mathsf{span}(\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3)$. Fortunately, setting $\boldsymbol{w} = \boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3$ enables the key-switching steps from $[\boldsymbol{b}_1^* + r\boldsymbol{b}_4^*]_2$ to $[\boldsymbol{b}_2^* + r\boldsymbol{b}_4^*]_2$ and from $[\boldsymbol{b}_2^* + r\boldsymbol{b}_4^*]_2$ to $[\boldsymbol{b}_3^* + r\boldsymbol{b}_4^*]_2$ in the public-index setting.

However, a new problem arises in the private-index setting: the $\boldsymbol{b}_4$-component of the challenge ciphertext cannot be changed to the corresponding component of C_1. Thus, we use a 6-dimensional vector space rather than a 4-dimensional one. The additional vectors $\boldsymbol{b}_5^*$ and $\boldsymbol{b}_6^*$ make it possible to apply the switching technique also to $\mathsf{span}(\boldsymbol{b}_4, \boldsymbol{b}_5, \boldsymbol{b}_6)$. As a result, $[\boldsymbol{b}_1^* + r\boldsymbol{b}_4^*]_2$ is switched to $[\boldsymbol{b}_3^* + r\boldsymbol{b}_6^*]_2$ at the end of the proof.

In Section 5, we present this scheme in a generalized form as a generic construction from predicate encodings. Moreover, applying this technique to the original ACGU20ad framework makes it possible to directly compute $[\langle \boldsymbol{x}, \boldsymbol{y} \rangle]_T$ even when the range of $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is unbounded. This may be of independent interest, since Abdalla et al.'s conversion from AB-IPFE to multi-input AB-IPFE [ACGU20] requires the underlying scheme to have this property.

Constructions based on ABP. Since the KP-ABE scheme of Lin and Luo [LL20] supports both zero-type and non-zero-type predicates, we extend their scheme to AB-IPFE. For simplicity, we review the structure of their KP-ABE scheme under the SXDH assumption. The key generation algorithm and the encryption algorithm sample $\mu \xleftarrow{\$} \mathbb{Z}_p$ and $s \xleftarrow{\$} \mathbb{Z}_p$, respectively, and $[\mu \cdot s]_T$ is used to mask the message. Given μ and the input policy, the key generation algorithm computes coefficient vectors $(\mathbf{L}_1, \ldots, \mathbf{L}_m)$ for a linear garbling $\langle \mathbf{L}_1, (1, \tilde{\boldsymbol{x}}) \rangle, \ldots, \langle \mathbf{L}_m, (1, \tilde{\boldsymbol{x}}) \rangle$ such that μ can be recovered if and only if $\tilde{\boldsymbol{x}}$ satisfies the policy.

Let $\mathcal{E}$ be a slotted IPFE scheme; see Section 2.5 for the definition. Then the ciphertext and secret key contain, respectively, the following terms:

$$\mathcal{E}.\mathsf{Enc}([s(1, \tilde{\boldsymbol{x}})]_1 : [\mathbf{0}]_1), \quad \{\mathcal{E}.\mathsf{KeyGen}([\mathbf{L}_r]_2 : [\mathbf{0}]_2)\}_{r\in[m]}.$$

The vectors before and after ":" correspond to the public and private slots, respectively. It follows that $[\mu \cdot s]_T$ can be recovered if $\tilde{\boldsymbol{x}}$ satisfies the policy.

In the AB-IPFE setting, μ and s can be used to mask $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ in a straightforward way. Specifically, we add the following terms to the ciphertext and secret key, respectively:

$$\mathcal{E}'.\mathsf{Enc}([\boldsymbol{x}]_1, [s]_1 : [\boldsymbol{0}]_1), \quad \mathcal{E}'.\mathsf{KeyGen}([\boldsymbol{y}]_2, [\mu]_2 : [\boldsymbol{0}]_2).$$

Here, $\mathcal{E}'$ denotes an independent instance of $\mathcal{E}$ run with a different key pair. We can use a technique similar to that of [DP21] to prove AD-SIM security. Moreover, it is straightforward to see that the attribute-on-top hierarchy can be added using the homomorphic property. The attribute key consists of secret keys associated with the unit vectors $\boldsymbol{e}_j$ for $j \in [d]$, together with additional keys for rerandomization. As a result, the attribute-key size is quadratic in d.

To reduce the size to $O(d)$, we can use component-wise key generation. The ciphertext and secret key contain, respectively, the following terms:

$$\{\mathcal{E}_j.\mathsf{Enc}([\boldsymbol{x}[j]]_1, [s]_1 : [\boldsymbol{0}]_1)\}_{j\in[d]}, \quad \{\mathcal{E}_j.\mathsf{KeyGen}([\boldsymbol{y}[j]]_2, [\boldsymbol{y}[j]\mu_j + \rho_j]_2 : [\boldsymbol{0}]_2)\}_{j\in[d]},$$

where $\rho_1 + \cdots + \rho_d = 0$, and each $\mathcal{E}_j$ is run with its own independent key pair. As in the scheme of Section 5, the attribute-on-top hierarchy can be added by setting the attribute key to

$$\{\mathcal{E}_j.\mathsf{KeyGen}([1]_2, [\mu_j]_2 : [\boldsymbol{0}]_2)\}_{j\in[d]}.$$

In this way, the attribute-key size can be reduced to linear in d.

Although these schemes are limited to the key-policy setting, a ciphertext-policy scheme can be constructed using analogous configurations:

Ciphertext:	$\mathcal{E}'.\mathsf{Enc}([\boldsymbol{x}]_1, [\mu]_1 : [\boldsymbol{0}]_1)$,	$\{\mathcal{E}.\mathsf{Enc}([\mathbf{L}_r]_1 : [\boldsymbol{0}]_1)\}_{r\in[m]}$,
Secret key:	$\mathcal{E}'.\mathsf{KeyGen}([\boldsymbol{y}]_2, [s]_2 : [\boldsymbol{0}]_2)$,	$\mathcal{E}.\mathsf{KeyGen}([s(1, \tilde{\boldsymbol{y}})]_2 : [\boldsymbol{0}]_2)$.

The proof strategy is similar to that of [LL20], but the private slot must be extended. We describe the core part of the security proof that is specific to the ciphertext-policy setting. Let $\boldsymbol{r}$ denote the randomness used in the garbling, with μ as one of its components. By the linearity of the garbling, we can decompose $\langle \mathbf{L}_r, (1, \tilde{\boldsymbol{y}}) \rangle$ as $\langle \mathsf{Decomp}(\mathbf{L}_r), \boldsymbol{r} \otimes (1, \tilde{\boldsymbol{y}}) \rangle$. The relevant parts of the ciphertext and of the response to the q-th query can be represented as follows:

$\mathcal{E}'.\mathsf{Enc}([\boldsymbol{x}]_1, [0]_1 : [1]_1, -)$,	$\{\mathcal{E}.\mathsf{Enc}([\boldsymbol{0}]_1 : [\mathsf{Decomp}(\mathbf{L}_r)]_1, -)\}_{r\in[m]}$,
$\mathcal{E}'.\mathsf{KeyGen}([\boldsymbol{y}_q]_2, [s_q]_2 : [s_q\mu]_2, -)$,	$\mathcal{E}.\mathsf{KeyGen}([s_q(1, \tilde{\boldsymbol{y}}_q)]_2 : [s_q\boldsymbol{r} \otimes (1, \tilde{\boldsymbol{y}}_q)]_2, -)$.

Here, "$-$" means that the remaining part of the private slot is $[\boldsymbol{0}]_1$ or $[\boldsymbol{0}]_2$. By the SXDH assumption, $[s_q\mu]_2$ and $[s_q\boldsymbol{r}]_2$ can be replaced with fresh random values $[\bar{\mu}_q]_2$ and $[\bar{\boldsymbol{r}}_q]_2$, respectively, where $\bar{\mu}_q$ is a component of $\bar{\boldsymbol{r}}_q$. For any fixed q^*, by the function-hiding property of $\mathcal{E}$, we can change the second part as follows:

Ciphertext:	$\{\mathcal{E}.\mathsf{Enc}([\boldsymbol{0}]_1 : [\mathsf{Decomp}(\mathbf{L}_r)]_1, [\bar{\mathbf{L}}_r]_1, -)\}_{r\in[m]}$,
q^*-th query:	$\mathcal{E}.\mathsf{KeyGen}([s_{q^*}(1, \tilde{\boldsymbol{y}}_{q^*})]_2 : [\boldsymbol{0}]_2, [(1, \tilde{\boldsymbol{y}}_{q^*})]_2, -)$.

Here, $\bar{\mathbf{L}}_r$ is generated using $\bar{\boldsymbol{r}}_{q^*}$ as the randomness. Lin and Luo [LL20] showed that if the q^*-th query is inadmissible, then $\bar{\mu}_{q^*}$ in the input to $\mathcal{E}.\mathsf{KeyGen}$ can be replaced with a fresh value μ'_{q^*} that is independent of $\bar{\boldsymbol{r}}_{q^*}$. In this case, $\langle \boldsymbol{x}, \boldsymbol{y}_{q^*} \rangle$ is masked by μ'_{q^*}. The rest of the proof is similar to that for the key-policy setting. Although this construction is simple, it applies the MDDH_k assumption to all entries of the vector $\bar{\boldsymbol{r}}_q$ at once, and hence the length of the private slot becomes much larger than the key-policy scheme. In our construction, we reduce this length by applying the MDDH_k assumption one entry at a time in the security proof.

2 Preliminaries

"Probabilistic Polynomial-Time" is abbreviated to "PPT". For a finite set S, $x \xleftarrow{\$} S$ means sampling x uniformly at random from S, and $\mathrm{Unif}(S)$ denotes the uniform distribution over S. We

say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible and write $f(n) = \mathsf{negl}(n)$ if for any positive integer k there exists an integer n_k such that $|f(n)| < n^{-k}$ for any $n > n_k$. We say that a probability $p(n)$ is overwhelming if $1 - p(n)$ is negligible. Let $\mathbb{Z}_p$ denote the quotient ring $\mathbb{Z}/p\mathbb{Z}$. We use the notations $(A|B)$ and $(A;B)$ for horizontal and vertical concatenation of matrices respectively. For a matrix $\mathbf{X} = (\boldsymbol{x}_1|\cdots|\boldsymbol{x}_n)$, $\mathsf{vec}(\mathbf{X})$ denotes the vector $(\boldsymbol{x}_1;\cdots;\boldsymbol{x}_n)$. Let $\boldsymbol{e}_j$ denote a unit vector whose j-th coefficient is 1.

2.1 Bilinear Groups

The bilinear group generator $\mathcal{G}$ is a PPT algorithm that outputs $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$. Here, p is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order p, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map. For matrices $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_p^{m\times n}$, we write $[\mathbf{A}]_1 := (g_1^{a_{i,j}}) \in \mathbb{G}_1^{m\times n}$, $[\mathbf{A}]_2 = (g_2^{a_{i,j}}) \in \mathbb{G}_2^{m\times n}$, and $[\mathbf{A}]_T = (e(g_1, g_2)^{a_{i,j}}) \in \mathbb{G}_T^{m\times n}$. Also, for matrices $\mathbf{A}$ and $\mathbf{B}$, we write $e([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T$.

In relation to bilinear groups, we use dual pairing vector spaces (DPVS) [OT09,OT10]. Let Dual denote the algorithm such that $\mathsf{Dual}(\mathbb{Z}_p^n)$ returns a random orthogonal basis $(\mathbb{B} = \{\boldsymbol{b}_i\}_{i\in[n]}, \mathbb{B}^* = \{\boldsymbol{b}_i^*\}_{i\in[n]})$ of $\mathbb{Z}_p^n$. In other words, Dual randomly samples the vectors such that $e([\boldsymbol{b}_i]_1, [\boldsymbol{b}_i^*]_2) = [1]_T$ and $e([\boldsymbol{b}_i]_1, [\boldsymbol{b}_j^*]_2) = [0]_T$ if $i \neq j$.

Definition 1 (Matrix Diffie-Hellman Assumption [EHK+17])**.** We say that the $\mathcal{U}_k$-Matrix Decisional Diffie-Hellman (MDDH_k) assumption on $\mathbb{G}_i$ $(i = 1, 2)$ holds if $(\mathbb{G}, [\mathbf{A}]_i, [\mathbf{Ar}]_i)$ is computationally indistinguishable from $(\mathbb{G}, [\mathbf{A}]_i, [\boldsymbol{u}]_i)$ where $\ell = k + 1$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times k}$, $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$, and $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^\ell$. It is known that this assumption in the setting $\ell > k + 1$ is implied by the MDDH_k assumption. We just say that the MDDH_k assumption holds if it holds for both $\mathbb{G}_1$ and $\mathbb{G}_2$.

2.2 Functional Encryption

Functional encryption (FE) [BSW11] consists of the following four PPT algorithms. Let $\mathcal{F}$ denote a set of functions that are defined on the message space.

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter λ as input, it outputs a master public key mpk and a master secret key msk.
- $\mathsf{Enc}(\mathsf{mpk}, x)$: Given a master public key mpk and a message x as inputs, it outputs a ciphertext C.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$: Given a master secret key msk and a function $f \in \mathcal{F}$, it outputs a secret key sk_f.
- $\mathsf{Dec}(\mathsf{sk}_f, C)$: Given a secret key sk_f and a ciphertext C as inputs, it outputs a value.

KeyGen and Dec also takes mpk as input, but we omit it from description for simplicity. This scheme is called correct if the minimum value of

$$\Pr[(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); C \leftarrow \mathsf{Enc}(\mathsf{mpk}, x); \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) : \mathsf{Dec}(\mathsf{sk}_f, C) = f(x)]$$

with respect to x, f is overwhelming in λ. Let $\mathsf{part}(x)$ denote the information in x that is allowed to be revealed from the ciphertext.

Attribute-Based Inner-Product Functional Encryption (AB-IPFE) [ACGU20] is a special case of FE. Each message consists of an attribute $\tilde{x} \in \tilde{\mathcal{X}}$ and a vector $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{Z}^d$. Each function is associated with an attribute $\tilde{y} \in \tilde{\mathcal{Y}}$ and a vector $\boldsymbol{y} \in \mathcal{Y} \subset \mathbb{Z}^d$. Let $R : \tilde{\mathcal{X}} \times \tilde{\mathcal{Y}} \to \{0, 1\}$ be the relation for access control. An AB-IPFE scheme is correct if decryption outputs $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ whenever $R(\tilde{x}, \tilde{y}) = 1$. While it is essential that the ciphertext hide $\boldsymbol{x}$, we can consider both schemes that hide $\tilde{x}$, called private-index schemes, and those that do not, called public-index schemes. In this paper, we write $\mathsf{part}(\tilde{x})$ instead of $\mathsf{part}(\tilde{x}, \boldsymbol{x})$. Throughout this paper, we set $\mathcal{X} = \mathcal{Y} = [0, B]^d$, where B is polynomially bounded. This is because all known pairing-based (AB-)IPFE schemes require one to recover $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ from $g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}$ for some $g \in \mathbb{G}_T$ by brute-forcing the discrete logarithm.

Two types of security notions are standard for FE schemes: indistinguishability (IND) and simulation-based security (SIM). We review the adaptive version of IND and SIM security notions.

Definition 2 (AD-IND security)**.** We define the following game:

$\underline{\mathsf{Exp}^{\mathsf{G}}_{\mathcal{A}}(1^\lambda)}$:
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(x_0^*, x_1^*, \mathsf{st}) \leftarrow \mathcal{A}_{\mathsf{find}}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$
$b \xleftarrow{\$} \{0,1\}$
$C^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_b^*)$
$b' \leftarrow \mathcal{A}_{\mathsf{guess}}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{st}, C^*)$
Output 1 if $b = b'$, otherwise output 0

The restriction is that any input f to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ satisfies $f(x_0^*) = f(x_1^*)$. We define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{G}}_{\mathcal{A}}(\lambda) := \left| 2\Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{G}}_{\mathcal{A}}(1^\lambda)] - 1 \right|.$$

An FE scheme is said to achieve Adaptive Indistinguishability (AD-IND) if the advantage is negligible for any PPT algorithm $\mathcal{A}$.

Definition 3 (AD-SIM security)**.** $(\mathsf{Setup}^*, \mathsf{Enc}^*, \mathsf{KeyGen}_0^*, \mathsf{KeyGen}_1^*)$ be PPT algorithms. We define the following two games:

$\underline{\mathsf{Exp}^{\mathsf{Real}}_{\mathcal{A}}(1^\lambda)}$:
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(x^*, \mathsf{st}) \leftarrow \mathcal{A}_{\mathsf{find}}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$
$C^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*)$
Output $\mathcal{A}_{\mathsf{guess}}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{st}, C^*)$

$\underline{\mathsf{Exp}^{\mathsf{Ideal}}_{\mathcal{A}}(1^\lambda)}$:
$(\mathsf{mpk}^*, \mathsf{msk}^*) \leftarrow \mathsf{Setup}^*(1^\lambda)$
$(x^*, \mathsf{st}) \leftarrow \mathcal{A}_{\mathsf{find}}^{\mathsf{KeyGen}_0^*(\mathsf{msk}^*,\cdot)}(\mathsf{mpk}^*)$
$C^* \leftarrow \mathsf{Enc}^*(\mathsf{mpk}^*, \mathsf{msk}^*, \mathsf{part}(x^*), \mathcal{V})$
Output $\mathcal{A}_{\mathsf{guess}}^{\mathsf{KeyGen}_1^*(\mathsf{msk}^*,\mathsf{part}(x^*),\cdot,\cdot)}(\mathsf{mpk}^*, \mathsf{st}, C^*)$

The restriction is as follows:

- Let f_i be the i-th query to KeyGen_0^* and sk_{f_i} be its response. $\mathcal{V}$ denotes the set of all $(f_i, f_i(x^*), \mathsf{sk}_{f_i})$.
- When $\mathcal{A}_{\mathsf{guess}}$ makes a query on f, KeyGen_1^* gets f and $f(x^*)$ as input.

An FE scheme is said to achieve Adaptive Simulation-based (AD-SIM) security if there exists a simulator $(\mathsf{Setup}^*, \mathsf{Enc}^*, \mathsf{KeyGen}_0^*, \mathsf{KeyGen}_1^*)$ such that the advantage

$$\mathsf{Adv}^{\mathsf{AD\text{-}SIM}}_{\mathcal{A}}(\lambda) := \left| \Pr[\mathsf{Exp}^{\mathsf{Real}}_{\mathcal{A}}(1^\lambda) = 0] - \Pr[\mathsf{Exp}^{\mathsf{Ideal}}_{\mathcal{A}}(1^\lambda) = 0] \right|$$

is negligible for any PPT algorithm $\mathcal{A}$.

2.3 Predicate Encoding and Function Encoding

Predicate encodings [CGW15, Wee14] are encodings of attributes that have been used for constructions of ABE. To construct adaptively secure AB-IPFE, Abdalla et al. [ACGU20] introduced function encodings, a variant of predicate encodings.

Definition 4 (Predicate Encoding [CGW15, Wee14])**.** Let $R_p : \widetilde{\mathcal{X}}_p \times \widetilde{\mathcal{Y}}_p \to \{0,1\}$ be a family of relations, and let $\mathsf{part}_p : \widetilde{\mathcal{X}}_p \to \mathcal{S}_p$ be a family of information leakage functions, which are parameterized by a prime number p. A corresponding $(n, |\mathsf{ct}|, |\mathsf{sk}|)$-predicate encoding consists of the following functions.

$$\mathbf{C}_p : \widetilde{\mathcal{X}}_p \to \mathbb{Z}_p^{n \times |\mathsf{ct}|}, \quad \mathbf{K}_p : \widetilde{\mathcal{Y}}_p \to \mathbb{Z}_p^{(n+1) \times |\mathsf{sk}|}, \quad \boldsymbol{d}_p : \mathcal{S}_p \times \widetilde{\mathcal{Y}}_p \to \mathbb{Z}_p^{|\mathsf{ct}|+|\mathsf{sk}|}$$

We assume that, when p is generated by $\mathcal{G}(1^\lambda)$, each function is computable in polynomial time in the security parameter. In what follows, we omit the subscript p from the notations. For simplicity of description, we set

$$\mathbf{L}(\tilde{x}, \tilde{y}) = \left(\begin{array}{c|c} \mathbf{0}^\top & \\ \hline \mathbf{C}(\tilde{x}) & \mathbf{K}(\tilde{y}) \end{array} \right).$$

Correctness We say that the predicate encoding is correct if

$$\mathbf{L}(\tilde{x},\tilde{y})\boldsymbol{d}(\mathsf{part}(\tilde{x}),\tilde{y}) = (1,0,\ldots,0)^\top \in \mathbb{Z}_p^{n+1}$$

holds for any $\tilde{x},\tilde{y}$ such that $R(\tilde{x},\tilde{y}) = 1$.

Security We say that the predicate encoding is secure if the following conditions hold:

1. For any $\tilde{x},\tilde{y}$ such that $R(\tilde{x},\tilde{y}) = 0$, $(\alpha,\boldsymbol{v}^\top)\mathbf{L}(\tilde{x},\tilde{y})$ and $(0,\boldsymbol{v}^\top)\mathbf{L}(\tilde{x},\tilde{y})$ are identically distributed, where $\alpha \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^n$.
2. For any $\tilde{x}_0,\tilde{x}_1,\tilde{y}$ such that $\mathsf{part}(\tilde{x}_0) = \mathsf{part}(\tilde{x}_1)$ and $R(\tilde{x}_0,\tilde{y}) = R(\tilde{x}_1,\tilde{y})$, the distribution of $(0,\boldsymbol{v}^\top)\mathbf{L}(\tilde{x}_b,\tilde{y})$, where $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^n$, does not depend on $b \in \{0,1\}$.

The first condition is commonly used in the public-index setting, and the second condition is needed in the private-index setting. In addition, we introduce the following mild condition for supporting the vector-on-top hierarchical structure, which is meaningful only in the private-index setting.

3. For $i \in [n]$, let $\delta_i := 0$ if the $(i+1)$-th row of $\mathbf{K}$ is identically the zero vector, and otherwise $\delta_i := 1$. In this paper, we write $\mathbf{I}_\mathbf{K} := \mathsf{diag}(\delta_1,\ldots,\delta_n)$. For any $\tilde{x}_0,\tilde{x}_1$ such that $\mathsf{part}(\tilde{x}_0) = \mathsf{part}(\tilde{x}_1)$ and $R(\tilde{x}_0,\tilde{y}) = R(\tilde{x}_1,\tilde{y})$ for any $\tilde{y}$, the distribution of $\{\boldsymbol{v}^\top\mathbf{C}(\tilde{x}_b), \boldsymbol{v}^\top\mathbf{I}_\mathbf{K}\}$ where $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^n$ does not depend on $b \in \{0,1\}$.

All existing predicate encodings satisfy the third condition. Among the existing predicate encodings [CGW15, Wee14], there is only one construction that is shown to be attribute-hiding, which supports inner-product predicates:

$$\mathbf{C}(\tilde{\boldsymbol{x}}) = (\tilde{\boldsymbol{x}}^\top;\mathbf{I}_\ell), \quad \mathbf{K}(\tilde{\boldsymbol{y}}) = (1;0;\tilde{\boldsymbol{y}}), \quad \boldsymbol{d}(\tilde{\boldsymbol{y}}) = (-\tilde{\boldsymbol{y}};1).$$

We show that this construction satisfies the third security notion in Definition 4. When $u \xleftarrow{\$} \mathbb{Z}_p$ and $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^\ell$, we have $(u,\boldsymbol{w}^\top)\mathbf{C}(\tilde{\boldsymbol{x}}_b) = u\tilde{\boldsymbol{x}}_b + \boldsymbol{w}$. Thus, it suffices to show that the distribution of $(u\tilde{\boldsymbol{x}}_b + \boldsymbol{w}, \boldsymbol{w})$ does not depend on b. Since we assume that $\langle\tilde{\boldsymbol{x}}_0,\tilde{\boldsymbol{y}}\rangle = 0$ holds iff $\langle\tilde{\boldsymbol{x}}_1,\tilde{\boldsymbol{y}}\rangle = 0$, there exists $a \in \mathbb{Z}_p\setminus\{0\}$ such that $\tilde{\boldsymbol{x}}_0 = a\tilde{\boldsymbol{x}}_1$. By setting $u' := au$, we have $u\tilde{\boldsymbol{x}}_0 = u'\tilde{\boldsymbol{x}}_1$.

Definition 5 (Function Encoding [ACGU20])**.** Let $\mathcal{F}_p$ be a family of functions from $\mathcal{X}_p$ to $\mathbb{Z}_p$, and let $\mathsf{part}_p : \widetilde{\mathcal{X}}_p \to \mathcal{S}_p$ be a family of information leakage functions, which are parameterized by a prime number p. The corresponding $(n,|\mathsf{ct}|,|\mathsf{sk}|)$-function encoding consists of the following functions:

$$\mathbf{C}_p : \mathcal{X}_p \to \mathbb{Z}_p^{(n+1)\times|\mathsf{ct}|}, \quad \mathbf{K}_p : \mathcal{F}_p \to \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}|}, \quad \boldsymbol{d}_p : \mathcal{S}_p \times \mathcal{F}_p \to \mathbb{Z}_p^{|\mathsf{ct}|+|\mathsf{sk}|}.$$

Correctness We say that the function encoding is correct if, for any $x \in \mathcal{X}$ and $f \in \mathcal{F}$,

$$(\mathbf{C}(x)|\mathbf{K}(f))\boldsymbol{d}(\mathsf{part}(x),f) = (f(x),0,\ldots,0)^\top \in \mathbb{Z}_p^{n+1}.$$

Security We say that the function encoding is secure if, for $x_0,x_1 \in \mathcal{X}$ and $f \in \mathcal{F}$ such that $f(x_0) = f(x_1)$ and $\mathsf{part}(x_0) = \mathsf{part}(x_1)$, the distribution of $(1,\boldsymbol{z}^\top)(\mathbf{C}(x_b)|\mathbf{K}(f))$, where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$, does not depend on $b \in \{0,1\}$.

We assume that the first row of $\mathbf{K}(f)$ is identically the zero vector. This is not assumed in the original definition, but all the existing constructions are in this form.

Remark 1. This security definition is slightly stronger than the definition of Abdalla et al., which requires that the distribution of $(z_0,\boldsymbol{z}^\top)(\mathbf{C}(x_b)|\mathbf{K}(f))$ where $z_0 \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$ is independent of b. However, this stronger property is implicitly assumed in the security proof for Abdalla et al.'s scheme.

2.4 Arithmetic Branching Programs and Arithmetic Key Garbling

Arithmetic branching programs (ABPs) are a computational model that captures arithmetic formulae and Boolean branching programs. To construct an ABE scheme with ABP-based access control, Lin and Luo [LL20] introduced the piecewise secure arithmetic key garbling scheme (AKGS). Their construction is based on partial garbling by Ishai and Wee [IW14]. We omit the definition of piecewise security since its use in our security proofs simply follows Lin and Luo's proof.

Definition 6 (Arithmetic Branching Program)**.** An arithmetic branching program $f = (V, E, s, t, w)$ over $\mathbb{Z}_p^n$ is a tuple consisting of a directed acyclic graph (V, E), its two distinguished vertices (s, t), and a weight function $w : E \to (\mathbb{Z}_p^n \to \mathbb{Z}_p)$ that returns affine functions. Assume that s has no incoming edges and that t has no outgoing edges. Let P be the set of all paths from s to t. The program f computes a function defined as follows:

$$f(\boldsymbol{x}) = \sum_{e_1 \cdots e_i \in P} \prod_{j=1}^{i} w(e_j)(\boldsymbol{x}).$$

Definition 7 (Arithmetic Key Garbling Scheme)**.** An arithmetic key garbling scheme (AKGS) for a function class $\mathcal{F} = \{f : \mathbb{Z}_p^n \to \mathbb{Z}_p\}$ computable by ABPs consists of the following two algorithms.

- $\mathsf{Garble}(f, \alpha, \beta; \boldsymbol{r})$: Given a function $f \in \mathcal{F}$ and secret values $\alpha, \beta \in \mathbb{Z}_p$, it outputs coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ of affine functions $L_1, \ldots, L_m : \mathbb{Z}_p^n \to \mathbb{Z}_p$. For each $r \in [m]$, we write

  $$L_r(\boldsymbol{x}) = \mathbf{L}_r[\mathsf{const}] + \sum_{i \in [n]} \mathbf{L}_r[\mathsf{coef}_i]\boldsymbol{x}[i].$$

  The randomness is represented as $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^{m-1}$. The coefficient vectors are required to be linear in $(\alpha, \beta, \boldsymbol{r})$.

- $\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_m)$: Given a function $f \in \mathcal{F}$, $\boldsymbol{x} \in \mathbb{Z}_p^n$, and labels $\ell_1, \ldots, \ell_m$, it outputs a value $\gamma \in \mathbb{Z}_p$. γ is required to be linear in $(\ell_1, \ldots, \ell_m)$. When $\ell_i = L_i(\boldsymbol{x})$ holds for any $i \in [m]$, γ is equal to $\alpha f(\boldsymbol{x}) + \beta$.

In the concrete construction [LL20], there exists a constant matrix $\mathbf{M}_{f,r} \in \mathbb{Z}_p^{n \times (m-1)}$ such that $\mathbf{L}_r = \mathbf{M}_{f,r}\boldsymbol{r} + \boldsymbol{c}_{\alpha,\beta,r}$ where $\boldsymbol{c}_{\alpha,\beta,1}[\mathsf{const}] = \beta$, $\boldsymbol{c}_{\alpha,\beta,m}[\mathsf{const}] = \alpha$, and the other components are 0.

2.5 Slotted Inner-Product Functional Encryption

Slotted Inner-Product Functional Encryption (sIPFE) [Lin17,LV16] is a hybrid of public-key IPFE and function-hiding secret-key IPFE. The existing sIPFE schemes are secure under the MDDH_k assumption, and have a homomorphic property.

Definition 8 (Slotted IPFE)**.** Slotted IPFE consists of the following five PPT algorithms.

- $\mathsf{Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$: Given a security parameter λ, a public slot $\mathfrak{s}_{\mathsf{pub}}$, and a private slot $\mathfrak{s}_{\mathsf{priv}}$, it outputs a master public key mpk and a master secret key msk.
- $\mathsf{Enc}(\mathsf{msk}, [\boldsymbol{u}]_1)$: Given a master secret key msk and $[\boldsymbol{u}]_1 \in \mathbb{G}_1^{\mathfrak{s}_{\mathsf{pub}} \cup \mathfrak{s}_{\mathsf{priv}}}$, it outputs a ciphertext C.
- $\mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{v}]_2)$: Given a master secret key msk and $[\boldsymbol{v}]_2 \in \mathbb{G}_2^{\mathfrak{s}_{\mathsf{pub}} \cup \mathfrak{s}_{\mathsf{priv}}}$, it outputs a secret key K.
- $\mathsf{Dec}(K, C)$: Given a secret key K and a ciphertext C, it outputs $[w]_T$.
- $\mathsf{SlotEnc}(\mathsf{mpk}, [\boldsymbol{u}]_1)$: Given a master public key and $[\boldsymbol{u}]_1 \in \mathbb{G}_1^{\mathfrak{s}_{\mathsf{pub}}}$, it outputs a ciphertext C.

Correctness We say that the sIPFE is correct if the minimum value of

$$\Pr[(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}}); C \leftarrow \mathsf{Enc}(\mathsf{msk}, [\boldsymbol{u}]_1); \\ K \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{v}]_2) : \mathsf{Dec}(K, C) = [\langle \boldsymbol{u}, \boldsymbol{v} \rangle]_T]$$

with respect to $[\boldsymbol{u}]_1, [\boldsymbol{v}]_2$ is overwhelming in λ.

Slot-mode Correctness We say that the sIPFE satisfies slot-mode correctness if for any $[\boldsymbol{u}]_1 \in \mathbb{G}_1^{\mathfrak{s}_{\mathsf{pub}}}$, and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$, the distributions of $\mathsf{Enc}(\mathsf{msk}, [\boldsymbol{u} \mid 0^{\mathfrak{s}_{\mathsf{priv}}}]_1)$ and $\mathsf{SlotEnc}(\mathsf{mpk}, [\boldsymbol{u}]_1)$ are identical.

Security For each $b \in \{0, 1\}$, we define the following game.

$\underline{\mathsf{Exp}^{\mathsf{G}_b}_{\mathcal{A}}(1^\lambda)}$:
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$
Output $\mathcal{A}^{\mathcal{O}^b_{\mathsf{Enc}}(\mathsf{msk},\cdot,\cdot),\mathcal{O}^b_{\mathsf{KeyGen}}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk})$

Here, $\mathcal{O}^b_{\mathsf{Enc}}(\mathsf{msk}, \cdot, \cdot)$ outputs $C \leftarrow \mathsf{Enc}(\mathsf{msk}, [\boldsymbol{u}_b]_1)$ given $([\boldsymbol{u}_0]_1, [\boldsymbol{u}_1]_1)$ as input. $\mathcal{O}^b_{\mathsf{KeyGen}}(\mathsf{msk}, \cdot, \cdot)$ outputs $K \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{v}_b]_2)$ given $([\boldsymbol{v}_0]_2, [\boldsymbol{v}_1]_2)$ as input.
The restrictions are as follows:

- For every input $(\boldsymbol{u}_0, \boldsymbol{u}_1)$ to $\mathcal{O}^b_{\mathsf{Enc}}$ and every input $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ to $\mathcal{O}^b_{\mathsf{KeyGen}}$, we require $\langle \boldsymbol{u}_0, \boldsymbol{v}_0 \rangle = \langle \boldsymbol{u}_1, \boldsymbol{v}_1 \rangle$.
- For every input $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ to $\mathcal{O}^b_{\mathsf{KeyGen}}$, we require that the projections to $\mathfrak{s}_{\mathsf{pub}}$ are equal: $\boldsymbol{v}_0|_{\mathfrak{s}_{\mathsf{pub}}} = \boldsymbol{v}_1|_{\mathfrak{s}_{\mathsf{pub}}}$.

An sIPFE scheme is said to achieve function-hiding security if the advantage

$$\mathsf{Adv}^{\mathsf{FH}}_{\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{G}_0}_{\mathcal{A}}(1^\lambda) = 0] - \Pr[\mathsf{Exp}^{\mathsf{G}_1}_{\mathcal{A}}(1^\lambda) = 0]|$$

is negligible for any PPT adversary $\mathcal{A}$.

Homomorphic Property There exists an efficient algorithm Add such that, for any $\boldsymbol{v}_1, \boldsymbol{v}_2$, if $\mathsf{sk}_1 \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{v}_1]_2)$ and $\mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{v}_2]_2)$, then $\mathsf{Add}(\mathsf{mpk}, \mathsf{sk}_1, \mathsf{sk}_2)$ has the same distribution as $\mathsf{KeyGen}(\mathsf{msk}, [\boldsymbol{v}_1 + \boldsymbol{v}_2]_2)$. We simply write the output as $\mathsf{sk}_1 \boxplus \mathsf{sk}_2$. Furthermore, multiplication of sk by $c \in \mathbb{Z}_p$ is denoted by $c \boxdot \mathsf{sk}$.

3 Attribute-based Inner-Product Functional Encryption with Hierarchical Structure

In this section, we formalize a framework for AB-IPFE with hierarchical key-generation algorithms. We also examine whether existing schemes can be made hierarchical.

3.1 Syntax and Security

Depending on whether attributes or vectors are placed at the top level, two types of hierarchical structures can be considered. In hierarchical AB-IPFE, the key generation algorithm KeyGen is replaced with the following algorithms.

- Attribute-on-top hierarchy
  - $\mathsf{KeyGen}_{\mathsf{Att}}(\mathsf{msk}, \tilde{y})$: Given a master secret key msk and an attribute $\tilde{y}$ as input, it outputs an attribute secret key $\mathsf{sk}_{\tilde{y}}$.
  - $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}(\mathsf{sk}_{\tilde{y}}, \boldsymbol{y})$: Given a secret key $\mathsf{sk}_{\tilde{y}}$ and a vector $\boldsymbol{y}$ as input, it outputs a secret key $\mathsf{sk}_{\tilde{y},\boldsymbol{y}}$.
- Vector-on-top hierarchy
  - $\mathsf{KeyGen}_{\mathsf{Vec}}(\mathsf{msk}, \boldsymbol{y})$: Given a master secret key msk and a vector $\boldsymbol{y}$ as input, it outputs a vector secret key $\mathsf{sk}_{\boldsymbol{y}}$.
  - $\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}(\mathsf{sk}_{\boldsymbol{y}}, \tilde{y})$: Given a secret key $\mathsf{sk}_{\boldsymbol{y}}$ and an attribute $\tilde{y}$ as input, it outputs a secret key $\mathsf{sk}_{\tilde{y},\boldsymbol{y}}$.

Since we start from (non-hierarchical) AB-IPFE and decomposing the key generation algorithm, we assume that hierarchical AB-IPFE satisfies delegation invariance [LJ20], which means that the

Table 2: Hierarchizability of AB-IPFE schemes

	Att-on-top	Vec-on-top	Access Control	Security	Assumption
ACGU20sel [ACGU20]	✓	✓	Predicate Encoding	SEL-SIM	SXDH
ACGU20ad [ACGU20]	$*$	✓	Function Encoding	AD-IND	SXDH
[NPP22]	✓	✓	Key-policy LSSS	AD-IND	SXDH
[ET25]	✓	–	ID-based	AD-SIM	DBDH
[ACGU20, BS25, LLW21]	✓	–	ID-based	AD/SEL-IND	(R)LWE

output distribution of Derive does not depend on the intermediate secret key, but is determined solely by msk, $\tilde{y}$, and $\boldsymbol{y}$.

In this case, the security definition for hierarchical AB-IPFE can be obtained by treating it as an FE scheme. Specifically, the key generation oracle is augmented with the ability to output $\mathsf{sk}_{\tilde{y}}$ given $\tilde{y}$ as input, or to output $\mathsf{sk}_{\boldsymbol{y}}$ given $\boldsymbol{y}$ as input, in addition to outputting $\mathsf{sk}_{\tilde{y},\boldsymbol{y}}$ given $(\tilde{y}, \boldsymbol{y})$ as input. $\mathsf{sk}_{\tilde{y}}$ is treated as a secret key that can fully decrypt ciphertexts satisfying relation R, while $\mathsf{sk}_{\boldsymbol{y}}$ is treated as a secret key that can compute inner product values for ciphertexts of any attribute.

3.2 Hierarchical Structure in Existing AB-IPFE

Table 2 summarizes the hierarchizability of existing AB-IPFE schemes. Both hierarchical structures can be added to ACGU20sel and NPP22. ET25 and the lattice-based schemes can be augmented with the attribute-on-top hierarchy, but adding the vector-on-top hierarchy to them appears difficult. Constructing an LWE-based AB-IPFE scheme with the vector-on-top hierarchy remains an open problem.

The hierarchizability of ACGU20ad depends on the underlying function encoding. We discuss this point in the next section.

Example 1 (ACGU20sel)**.** ACGU20sel [ACGU20] is based on predicate encodings. Let $\mathbf{K}_{\mathcal{P}}$ denote the key generation function of the underlying predicate encoding $\mathcal{P}$. Setup and KeyGen of ACGU20sel are as follows.

- $\mathsf{Setup}(1^\lambda)$:
  1. Generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$.
  2. Samples $a, b \xleftarrow{\$} \mathbb{Z}_p$ and sets $\boldsymbol{a} := (1, a)^\top, \boldsymbol{b} := (1, b)^\top$.
  3. Samples $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{2\times d}, \mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{2\times 2} (i \in [n])$ and sets $\boldsymbol{w}_i := \mathbf{W}_i \boldsymbol{b}$.
  4. Outputs $\mathsf{mpk} := ([\boldsymbol{a}]_1, [\boldsymbol{b}]_2, [\mathbf{U}^\top \boldsymbol{a}]_1, \{[\mathbf{W}_i^\top \boldsymbol{a}]_1, [\boldsymbol{w}_i]_2\}_{i\in[n]})$ and $\mathsf{msk} := \mathbf{U}$.
- $\mathsf{KeyGen}(\mathsf{msk}, \tilde{y}, \boldsymbol{y})$:
  1. Samples $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_1]_2 := [r\boldsymbol{b}]_2$.
  2. Computes $[\mathbf{K}_2]_2 := [(\mathbf{U}\boldsymbol{y} | r\boldsymbol{w}_1 | \cdots | r\boldsymbol{w}_n)\mathbf{K}_{\mathcal{P}}(\tilde{y})]_2$.
  3. Outputs $(\tilde{y}, \boldsymbol{y}, [\boldsymbol{k}_1]_2, [\mathbf{K}_2]_2)$.

Attribute-on-top hierarchy can be introduced by utilizing the homomorphic property of KeyGen.

- $\mathsf{KeyGen}_{\mathsf{Att}}(\mathsf{msk}, \tilde{y})$:
  1. Runs $(\tilde{y}, \boldsymbol{e}_j, [\boldsymbol{k}_{1,j}]_2, [\mathbf{K}_{2,j}]_2) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \tilde{y}, \boldsymbol{e}_j)$ for each $j \in [d]$.
  2. Outputs $\mathsf{sk}_{\tilde{y}} := (\tilde{y}, \{[\boldsymbol{k}_{1,j}]_2, [\mathbf{K}_{2,j}]_2\}_{j\in[d]})$.
- $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}(\mathsf{sk}_{\tilde{y}}, \boldsymbol{y} = (y_1, \ldots, y_d)^\top)$:
  1. Computes $([\boldsymbol{k}_{1,0}]_2, [\mathbf{K}_{2,0}]_2)$ distributed as $\mathsf{KeyGen}(\mathsf{msk}, \tilde{y}, \mathbf{0})$ without using msk.

2. Computes $[\boldsymbol{k}_1]_2 := [\boldsymbol{k}_{1,0} + \sum_{j=1}^{d} y_j \boldsymbol{k}_{1,j}]_2$ and $[\mathbf{K}_2]_2 := \left[\mathbf{K}_{2,0} + \sum_{j=1}^{d} y_j \mathbf{K}_{2,j}\right]_2$.
3. Outputs $(\tilde{y}, \boldsymbol{y}, [\boldsymbol{k}_1]_2, [\mathbf{K}_2]_2)$.

The distribution of secret keys generated by $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}$ is the same as that by KeyGen.

Additionally, the hierarchical structure satisfies SEL-SIM security. In fact, $\mathsf{sk}_{\tilde{y}}$ consists of $\mathsf{sk}_{\tilde{y},\boldsymbol{e}_j}$ for each j. Making a query on $\tilde{y}$ corresponds to making queries on $(\tilde{y}, \boldsymbol{e}_j)$ for every j, which is captured by the SEL-SIM game of the original ACGU20sel scheme.

Vector-on-top hierarchy can be introduced by setting $\mathsf{sk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathbf{U}\boldsymbol{y})$. $\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}$ computes the secret key by using $\mathbf{U}\boldsymbol{y}$. Security can be proved by following a process nearly identical to the original proof for ACGU20sel.

Example 2 (NPP22)**.** NPP22 [NPP22] supports linear secret sharing scheme (LSSS) in the key-policy setting. A SEL-IND secure scheme and an AD-IND secure scheme are proposed. Since their difference lies only on the dimension of DPVS, we only consider the AD-IND secure scheme. To introduce the hierarchical structures, boxed parts are made public whereas they are included in the master secret key in the original construction. This modification does not affect the security proof. See the original paper for the definitions of $\Lambda_{a_0}(\mathbb{A})$ and $\mathsf{List}-\mathsf{Att}(\mathbb{A})$.

- $\mathsf{Setup}(1^\lambda)$:
  1. Generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$.
  2. Generates $(\mathbf{F} = \{\mathbf{f}_i\}_{i\in[d+7]}, \mathbf{F}^* = \{\mathbf{f}_i^*\}_{i\in[d+7]}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^{d+7})$ and $(\mathbf{H} = \{\mathbf{h}_i\}_{i\in[d+3]}, \mathbf{H}^* = \{\mathbf{h}_i^*\}_{i\in[d+3]}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^{d+3})$.
  3. Samples $\mu \xleftarrow{\$} \mathbb{Z}_p$ and $\boldsymbol{s}, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^d$.
  4. Outputs $\mathsf{mpk} := ([\mathbf{h}_1 + \mu\mathbf{h}_2]_1, [\mathbf{h}_3]_1, \{[\mathbf{f}_i]_1\}_{i\in[3]}, \boxed{\{[\mathbf{f}_i^*]_2\}_{i\in[3]}, \{[\mathbf{h}_i^*]_2\}_{i\in[3]}}, [\boldsymbol{s} + \mu\boldsymbol{u}]_1)$ and $\mathsf{msk} := (\boldsymbol{s}, \boldsymbol{u})$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbb{A}, \boldsymbol{y})$:
  1. Samples $a_0 \xleftarrow{\$} \mathbb{Z}_p$ and runs $(a_j)_{j\in\mathsf{List}-\mathsf{Att}(\mathbb{A})} \leftarrow \Lambda_{a_0}(\mathbb{A})$.
  2. Samples $\pi_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_j^*]_2 := [\pi_j(j \cdot \boldsymbol{f}_1^* + \boldsymbol{f}_2^*) + a_j \boldsymbol{f}_3^*]_2$ for each j.
  3. Computes $[\boldsymbol{k}_{\mathsf{ipfe}}^*]_2 := [(\boldsymbol{s}^\top\boldsymbol{y}) \cdot \boldsymbol{h}_1^* + (\boldsymbol{u}^\top\boldsymbol{y}) \cdot \boldsymbol{h}_2^* + a_0\boldsymbol{h}_3^*]_2$.
  4. Outputs $(\mathbb{A}, \boldsymbol{y}, \{[\boldsymbol{k}_j^*]_2\}_{j\in\mathsf{List}-\mathsf{Att}(\mathbb{A})}, [\boldsymbol{k}_{\mathsf{ipfe}}^*]_2)$.

The hierarchical structures are similar to those of ACGU20sel. Attribute-on-top hierarchy can be added by using the homomorphic property. Also, vector-on-top hierarchy can be added by setting $(\boldsymbol{s}^\top\boldsymbol{y}, \boldsymbol{u}^\top\boldsymbol{y})$ as the vector key. Note that rerandomization can be performed without msk since $\boldsymbol{f}_1^*$, $\boldsymbol{f}_2^*$, $\boldsymbol{f}_3^*$, and $\boldsymbol{h}_3^*$ are made public.

Example 3 (ET25)**.** ET25 [ET25] utilizes a symmetric bilinear group, a pseudorandom function f from the identity space to $\mathbb{Z}_p^d$, and a hash function H from the identity space to $\mathbb{G}^d$. Let $[\![x]\!]$ denote x^a for $x \in \mathbb{Z}_p$.

- $\mathsf{Setup}(1^\lambda)$:
  1. Generates $(p, \mathbb{G}, \mathbb{G}_T, g, e)$.
  2. Samples $a, b \xleftarrow{\$} \mathbb{Z}_p$.
  3. Samples a key k of the pseudorandom function f.
  4. Outputs $\mathsf{mpk} := [\![b]\!]$ and $\mathsf{msk} := (a, k)$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{y} = (y_1, \ldots, y_d)^\top)$:
  1. Sets $(h_{\mathsf{id},1}, \ldots, h_{\mathsf{id},d}) := H(\mathsf{id})$.
  2. Computes $\kappa := \langle f_k(\mathsf{id}), \boldsymbol{y}\rangle$.

3. Computes $\sigma := [\![-ab\kappa]\!] \cdot \prod_{i\in[d]} (h_{\mathsf{id},i})^{ay_i}$.
4. Outputs $(\boldsymbol{y}, \sigma, \kappa)$.

This scheme has the homomorphic property, which makes it possible to add attribute-on-top hierarchy. In fact, given $(\boldsymbol{y}_1, \sigma_1, \kappa_1)$ and $(\boldsymbol{y}_2, \sigma_2, \kappa_2)$, we can compute the secret key $(\boldsymbol{y}_1+\boldsymbol{y}_2, \sigma_1\sigma_2, \kappa_1+\kappa_2)$ associated with $\boldsymbol{y}_1 + \boldsymbol{y}_2$. On the other hand, the term $f_k(\mathsf{id})$ becomes an obstacle to introducing vector-on-top hierarchy.

Example 4 (Lattice-based Schemes)**.** In lattice-based IB-IPFE schemes [ACGU20, BS25, LLW21], KeyGen computes a matrix $\mathbf{R}_{\mathsf{id}}$ deterministically, and outputs $(\boldsymbol{y}, \mathbf{R}_{\mathsf{id}}\boldsymbol{y})$. By utilizing the homomorphic property of $\mathbf{R}_{\mathsf{id}}\boldsymbol{y}$, attribute-on-top hierarchy can be easily added. On the other hand, introducing vector-on-top hierarchy seems difficult. Preimage sampling algorithms are used to compute $\mathbf{R}_{\mathsf{id}}$, but this procedure needs to be deterministic with respect to id. For this purpose, $f_k(\mathsf{id})$ where f_k denotes a pseudorandom function is used as the randomness. Similarly to the case of ET25, this part becomes an obstacle.

Edamura and Takayasu [ET25] proposed a method for transforming an IND-secure $(d+1)$-dimensional IB-IPFE scheme modulo p into a SIM-secure $(d+1)$-dimensional IB-IPFE scheme, which can be applied to these lattice-based schemes. This transformation preserves the attribute-on-top hierarchical structure. Let IBIPFE denote the scheme before the transformation. Then Setup and KeyGen of the transformed scheme are described as follows:

- $\mathsf{Setup}(1^\lambda)$:
  1. Generates $(p, \mathbb{G}, \mathbb{G}_T, g, e)$.
  2. Runs $(\mathsf{mpk}', \mathsf{msk}') \leftarrow \mathsf{IBIPFE.Setup}(1^\lambda)$.
  3. Samples a key k of a pseudorandom function f from the identity space to $\mathbb{Z}_p^d$.
  4. Outputs $\mathsf{mpk} := \mathsf{mpk}'$ and $\mathsf{msk} := (\mathsf{msk}', k)$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{y})$:
  1. Sets $\hat{\boldsymbol{y}} := (\boldsymbol{y}; \langle f_k(\mathsf{id}), \boldsymbol{y}\rangle)$.
  2. Runs $\mathsf{sk}_{\mathsf{id},\hat{\boldsymbol{y}}} \leftarrow \mathsf{IBIPFE.KeyGen}(\mathsf{msk}', \mathsf{id}, \hat{\boldsymbol{y}})$.
  3. Outputs $(\hat{\boldsymbol{y}}, \mathsf{sk}_{\mathsf{id},\hat{\boldsymbol{y}}})$.

The attribute key of the transformed scheme is $(\mathsf{sk}_{\mathsf{id}}, f_k(\mathsf{id}))$ where $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBIPFE.KeyGen}_{\mathsf{Att}}(\mathsf{msk}', \mathsf{id})$. By $\mathsf{IBIPFE.Derive}_{\mathsf{Att}\to\mathsf{Vec}}$, the secret key for any $\boldsymbol{y}$ can be computed from the attribute key. Also, it is easy to verify from the original proof that the attribute key does not compromise security.

4 Hierarchizability of Abdalla et al.'s AD-IND secure scheme

In this section, we investigate the hierarchizability of the ACGU20ad scheme. We show that it can be reduced to some properties of the underlying function encoding. Then we propose two generic constructions of function encodings from predicate encodings.

4.1 Hierarchical Function Encoding

We review the construction of ACGU20ad. Let $\mathbf{K}$ be the key generation function of the underlying function encoding. Setup and KeyGen of ACGU20ad are as follows.

- $\mathsf{Setup}(1^\lambda)$:
  1. Generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$.
  2. Samples $a \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_p^3$ and sets $\boldsymbol{a} := (1, a)^\top$.
  3. Samples $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{2\times 3} (i = 0, 1, \ldots, n)$ and sets $\boldsymbol{\alpha}_i := \mathbf{W}_i^\top \boldsymbol{a}$, $\boldsymbol{\beta}_i := \mathbf{W}_i \boldsymbol{b}$.
  4. Outputs $\mathsf{mpk} := ([\boldsymbol{a}]_1, \{[\boldsymbol{\alpha}_i]_1\}_i)$ and $\mathsf{msk} := ([\boldsymbol{b}]_2, \{[\boldsymbol{\beta}_i]_2\}_i)$.

- $\mathsf{KeyGen}(\mathsf{msk}, \tilde{y}, \boldsymbol{y})$:

  1. Samples $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_1]_2 := [r\boldsymbol{b}]_2$.
  2. Computes $[\mathbf{K}_2]_2 := [r(\boldsymbol{\beta}_0|\cdots|\boldsymbol{\beta}_n)\mathbf{K}(\tilde{y}, \boldsymbol{y})]_2$ and $[\boldsymbol{k}_3]_2 := [r\boldsymbol{\beta}_0]_2$.
  3. Outputs $(\tilde{y}, \boldsymbol{y}, [\boldsymbol{k}_1]_2, [\mathbf{K}_2]_2, [\boldsymbol{k}_3]_2)$.

ACGU20ad can be augmented with the hierarchical structures by making the underlying function encoding hierarchical. We formalize the hierarchical function encoding.

Definition 9. Let $\mathcal{F} = \{f_{\tilde{y},\boldsymbol{y}} : \widetilde{\mathcal{X}} \times \mathcal{X} \to \mathbb{Z}_p\}_{(\tilde{y},\boldsymbol{y})\in\widetilde{\mathcal{Y}}\times\mathcal{Y}}$ be a family of functions and $\mathsf{part} : \widetilde{\mathcal{X}} \to \mathcal{S}$ be a leakage function. A hierarchical function encoding for $(\mathcal{F}, \mathsf{part})$ is a function encoding with one of the following additional functions, which decompose $\mathbf{K}$ into a product of two matrices. $|\mathsf{sk}_{\mathsf{Att}}|, |\mathsf{sk}_{\mathsf{Vec}}| \in \mathbb{N}$ denote additional parameters.

Attribute-on-top hierarchy

$\mathbf{K}_{\mathsf{Att}} : \widetilde{\mathcal{Y}} \to \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}_{\mathsf{Att}}|}$, $\mathbf{T}_{\mathsf{Vec}} : \mathcal{Y} \to \mathbb{Z}_p^{|\mathsf{sk}_{\mathsf{Att}}|\times|\mathsf{sk}|}$ s.t. $\mathbf{K}(\tilde{y}, \boldsymbol{y}) = \mathbf{K}_{\mathsf{Att}}(\tilde{y})\mathbf{T}_{\mathsf{Vec}}(\boldsymbol{y})$.

Vector-on-top hierarchy

$\mathbf{K}_{\mathsf{Vec}} : \mathcal{Y} \to \mathbb{Z}_p^{(n+1)\times|\mathsf{sk}_{\mathsf{Vec}}|}$,$\mathbf{T}_{\mathsf{Att}} : \widetilde{\mathcal{Y}} \to \mathbb{Z}_p^{|\mathsf{sk}_{\mathsf{Vec}}|\times|\mathsf{sk}|}$ s.t. $\mathbf{K}(\tilde{y}, \boldsymbol{y}) = \mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y})\mathbf{T}_{\mathsf{Att}}(\tilde{y})$.

When both hierarchical structures are present, assume that $\mathbf{K}_{\mathsf{Att}}(\tilde{y})\mathbf{T}_{\mathsf{Vec}}(\boldsymbol{y}) = \mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y})\mathbf{T}_{\mathsf{Att}}(\tilde{y})$ holds for any $\tilde{y} \in \widetilde{\mathcal{Y}}$ and $\boldsymbol{y} \in \mathcal{Y}$. When $\mathbf{K}$ is defined as $\mathbf{K}_{\mathsf{Att}}\mathbf{T}_{\mathsf{Vec}}$ or $\mathbf{K}_{\mathsf{Vec}}\mathbf{T}_{\mathsf{Att}}$, correctness is the same as that of function encoding.

As the function class expands due to the hierarchical structures, corresponding security notions must also be added. Let $(\tilde{x}_0, \boldsymbol{x}_0), (\tilde{x}_1, \boldsymbol{x}_1)$ satisfy $\mathsf{part}(\tilde{x}_0) = \mathsf{part}(\tilde{x}_1)$, and let $\boldsymbol{z} := (z_1, \ldots, z_n)^\top \xleftarrow{\$} \mathbb{Z}_p^n$.

Attribute Key Security If either "$R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 1$ and $\boldsymbol{x}_0 = \boldsymbol{x}_1$" or "$R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 0$", the distribution of $\{(1, \boldsymbol{z}^\top)(\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)|\mathbf{K}_{\mathsf{Att}}(\tilde{y}))\}$ does not depend on $b \in \{0, 1\}$.

Vector Key Security If $\langle \boldsymbol{x}_0, \boldsymbol{y}\rangle = \langle \boldsymbol{x}_1, \boldsymbol{y}\rangle$ and $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y})$ for any $\tilde{y}$, the distribution of $\{(1, \boldsymbol{z}^\top)(\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)|\mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y}))\}$ does not depend on $b \in \{0, 1\}$.

Using hierarchical function encodings, the ACGU20ad framework can be augmented with the attribute-on-top hierarchy. The vector-on-top hierarchy can be added in the same way.

- $\mathsf{KeyGen}_{\mathsf{Att}}(\mathsf{msk}, \tilde{y})$:

  1. Samples $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}'_1]_2 := [r\boldsymbol{b}]_2$.
  2. Computes $[\mathbf{K}'_2]_2 := [r(\boldsymbol{\beta}_0|\cdots|\boldsymbol{\beta}_n)\mathbf{K}_{\mathsf{Att}}(\tilde{y})]_2$ and $[\boldsymbol{k}'_3]_2 := [r\boldsymbol{\beta}_0]_2$.
  3. Outputs $\mathsf{sk}_{\tilde{y}} := (\tilde{y}, [\boldsymbol{k}'_1]_2, [\mathbf{K}'_2]_2, [\boldsymbol{k}'_3]_2)$.

- $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}(\mathsf{sk}_{\tilde{y}}, \boldsymbol{y})$:

  1. Samples $t \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_1]_2 := [t\boldsymbol{k}'_1]_2$, $[\mathbf{K}_2]_2 := [t\mathbf{K}'_2\mathbf{T}_{\mathsf{Vec}}(\boldsymbol{y})]_2$, and $[\boldsymbol{k}_3]_2 := [t\boldsymbol{k}'_3]_2$.
  2. Outputs $(\tilde{y}, \boldsymbol{y}, [\boldsymbol{k}_1]_2, [\mathbf{K}_2]_2, [\boldsymbol{k}_3]_2)$.

The ACGU20ad framework has been shown to be secure not only for AB-IPFE but also for general FE. Thus, security of ACGU20ad with hierarchical structures can be reduced to the security (including attribute key security and vector key security) of the function encoding.

4.2 Construction of Function Encoding (1)

Our first construction, which transforms a predicate encoding $\mathcal{P}$ into a function encoding $T_1(\mathcal{P})$, is described in Figure 1. $T_1(\mathcal{P})$ supports the vector-on-top hierarchy.

Theorem 1. If $\mathcal{P}$ is correct and secure, the hierarchical function encoding $T_1(\mathcal{P})$ satisfies correctness and security.

Function Encoding:

$$\mathbf{C}(\tilde{x}, \boldsymbol{x}) = \left(\begin{array}{c|c} \boldsymbol{x}^\top & \\ \hline \mathbf{I}_d & \mathbf{O} \\ \hline \mathbf{O} & \mathbf{C}_{\mathcal{P}}(\tilde{x}) \end{array}\right), \quad \mathbf{K}(\tilde{y}, \boldsymbol{y}) = \left(\begin{array}{c} \mathbf{0}^\top \\ \hline \boldsymbol{y}\boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top \\ \hline \mathbf{K}'_{\mathcal{P}}(\tilde{y}) \end{array}\right),$$

$$\boldsymbol{d}(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y}) = (\boldsymbol{y}; -\boldsymbol{d}_{\mathcal{P}}(\mathsf{part}(\tilde{x}), \tilde{y})).$$

Vector-on-top Hierarchical Structure:

$$\mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y}) = \left(\begin{array}{c} \mathbf{0}^\top \\ \hline \begin{array}{c|c} \boldsymbol{y} & \mathbf{O} \end{array} \\ \hline \begin{array}{c|c} \mathbf{0} & \mathbf{I}_{\mathbf{K}_{\mathcal{P}}} \end{array} \end{array}\right), \quad \mathbf{T}_{\mathsf{Att}}(\tilde{y}) = \mathbf{K}_{\mathcal{P}}(\tilde{y}).$$

Figure 1: Construction of function encoding $T_1(\mathcal{P})$

Proof. We first prove correctness. For simplicity, we omit the function inputs when they are clear from context. We have

$$(\mathbf{C}(\tilde{x}, \boldsymbol{x})|\mathbf{K}(\tilde{y}, \boldsymbol{y}))\boldsymbol{d}(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y}; (1 - \boldsymbol{k}_{\mathcal{P}}^\top \boldsymbol{d}_{\mathcal{P},2})\boldsymbol{y}; -(\mathbf{C}_{\mathcal{P}}|\mathbf{K}'_{\mathcal{P}})\boldsymbol{d}_{\mathcal{P}}).$$

When $R(\tilde{x}, \tilde{y}) = 1$, it is equal to $(\boldsymbol{x}^\top \boldsymbol{y}; 0; \cdots; 0)$ by correctness of $\mathcal{P}$.

We next show that $T_1(\mathcal{P})$ is secure. For each $b \in \{0, 1\}$, we write $\mathbf{C}_b := \mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)$, $\mathbf{C}_{\mathcal{P},b} := \mathbf{C}_{\mathcal{P}}(\tilde{x}_b)$. When we sample $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^d, \boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^{n_{\mathcal{P}}}$ and set $\boldsymbol{z} = (\boldsymbol{w}; \boldsymbol{v})$, we have

$$(1, \boldsymbol{z}^\top)(\mathbf{C}_0|\mathbf{K}(\tilde{y}, \boldsymbol{y})) = (\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{w}^\top \boldsymbol{y}\boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top + \boldsymbol{v}^\top \mathbf{K}'_{\mathcal{P}}(\tilde{y})).$$

If $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 0$, by security of $\mathcal{P}$, this distribution is identical to that of

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \alpha \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top + \boldsymbol{v}^\top \mathbf{K}'_{\mathcal{P}}(\tilde{y}))$$

where $\alpha \xleftarrow{\$} \mathbb{Z}_p$. By applying the security of $\mathcal{P}$ again, the distribution is identical to that of

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{v}^\top \mathbf{K}'_{\mathcal{P}}(\tilde{y})).$$

By setting $\boldsymbol{w}' = \boldsymbol{w} + (\boldsymbol{x}_0 - \boldsymbol{x}_1)$, this vector can be written as

$$(\boldsymbol{x}_1^\top + \boldsymbol{w}'^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{v}^\top \mathbf{K}'_{\mathcal{P}}(\tilde{y})).$$

By the security of $\mathcal{P}$, we may replace $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$, and hence the distributions of $(1, \boldsymbol{z}^\top)(\mathbf{C}_0|\mathbf{K}(\tilde{y}, \boldsymbol{y}))$ and $(1, \boldsymbol{z}^\top)(\mathbf{C}_1|\mathbf{K}(\tilde{y}, \boldsymbol{y}))$ are identical. If $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 1$, we have $\langle \boldsymbol{x}_0, \boldsymbol{y} \rangle = \langle \boldsymbol{x}_1, \boldsymbol{y} \rangle$. By setting $\boldsymbol{w}' = \boldsymbol{w} + (\boldsymbol{x}_0 - \boldsymbol{x}_1)$, the vector $(1, \boldsymbol{z}^\top)(\mathbf{C}_0|\mathbf{K}(\tilde{y}, \boldsymbol{y}))$ can be represented as

$$(\boldsymbol{x}_1^\top + \boldsymbol{w}'^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{w}'^\top \boldsymbol{y}\boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top + \boldsymbol{v}^\top \mathbf{K}'_{\mathcal{P}}(\tilde{y})).$$

By replacing $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$, we can see that the distributions of $(1, \boldsymbol{z}^\top)(\mathbf{C}_0|\mathbf{K}(\tilde{y}, \boldsymbol{y}))$ and $(1, \boldsymbol{z}^\top)(\mathbf{C}_1|\mathbf{K}(\tilde{y}, \boldsymbol{y}))$ are identical.

We then show that $T_1(\mathcal{P})$ satisfies vector key security. We have

$$(1, \boldsymbol{z}^\top)(\mathbf{C}_0|\mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y})) = (\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{w}^\top \boldsymbol{y} | \boldsymbol{v}^\top \mathbf{I}_{\mathbf{K}_{\mathcal{P}}}).$$

By setting $\boldsymbol{w}' = \boldsymbol{w} + (\boldsymbol{x}_0 - \boldsymbol{x}_1)$, the vector can be written as

$$(\boldsymbol{x}_1^\top + \boldsymbol{w}'^\top | \boldsymbol{v}^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{w}'^\top \boldsymbol{y} | \boldsymbol{v}^\top \mathbf{I}_{\mathbf{K}_{\mathcal{P}}}).$$

By the security of $\mathcal{P}$, we can replace $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$. □

Function Encoding:

$$\mathbf{C}(\tilde{x}, \boldsymbol{x}) = \left(\begin{array}{c|c} \boldsymbol{x}^\top & \\ \hline \mathbf{I}_d & \mathbf{O} \\ \hline \mathbf{O} & \mathbf{C}_{\mathcal{P}}(\tilde{x}) \otimes \mathbf{I}_d \end{array}\right), \quad \mathbf{K}(\tilde{y}, \boldsymbol{y}) = \left(\begin{array}{c} \mathbf{0}^\top \\ \hline \mathbf{K}_{\mathcal{P}}(\tilde{y}) \otimes \boldsymbol{y} \end{array}\right),$$

$$\boldsymbol{d}(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y}) = (\boldsymbol{y}; -\boldsymbol{d}_{\mathcal{P},1}(\mathsf{part}(\tilde{x}), \tilde{y}) \otimes \boldsymbol{y}; -\boldsymbol{d}_{\mathcal{P},2}(\mathsf{part}(\tilde{x}), \tilde{y})).$$

Attribute-on-top Hierarchical Structure:

$$\mathbf{K}_{\mathsf{Att}}(\tilde{y}) = \left(\begin{array}{c} \mathbf{0}^\top \\ \hline \mathbf{K}_{\mathcal{P}}(\tilde{y}) \otimes \mathbf{I}_d \end{array}\right), \quad \mathbf{T}_{\mathsf{Vec}}(\boldsymbol{y}) = \mathbf{I}_{|\mathsf{sk}|_{\mathcal{P}}} \otimes \boldsymbol{y}.$$

Vector-on-top Hierarchical Structure:

$$\mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y}) = \left(\begin{array}{c|c} & \mathbf{0}^\top \\ \hline \boldsymbol{y} & \mathbf{O} \\ \hline \mathbf{0} & \mathbf{I}_{\mathbf{K}_{\mathcal{P}}} \otimes \boldsymbol{y} \end{array}\right), \quad \mathbf{T}_{\mathsf{Att}}(\tilde{y}) = \mathbf{K}_{\mathcal{P}}(\tilde{y}).$$

Figure 2: Construction of function encoding $T_2(\mathcal{P})$

4.3 Construction of Function Encoding (2)

Our second construction, which transforms a predicate encoding $\mathcal{P}$ into a function encoding $T_2(\mathcal{P})$, is described in Figure 2. Compared to $T_1(\mathcal{P})$, the function encoding $T_2(\mathcal{P})$ has larger $|\mathsf{ct}|$, but it supports both types of hierarchical structures.

Theorem 2. If $\mathcal{P}$ is correct and secure, the hierarchical function encoding $T_2(\mathcal{P})$ satisfies correctness and security.

Proof. Let $\boldsymbol{y} = (y_1, \ldots, y_d)^\top$. We first prove correctness. We have

$$(\mathbf{C}(\tilde{x}, \boldsymbol{x}) | \mathbf{K}(\tilde{y}, \boldsymbol{y}))\boldsymbol{d}(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y}; (1 - \boldsymbol{k}_{\mathcal{P}}^\top \boldsymbol{d}_{\mathcal{P},2})\boldsymbol{y}; -((\mathbf{C}_{\mathcal{P}} | \mathbf{K}'_{\mathcal{P}})\boldsymbol{d}_{\mathcal{P}}) \otimes \boldsymbol{y}).$$

When $R(\tilde{x}, \tilde{y}) = 1$, it is equal to $(\boldsymbol{x}^\top \boldsymbol{y}; 0; \cdots; 0)$ by correctness of $\mathcal{P}$.

We next show that $T_2(\mathcal{P})$ is secure. For each $b \in \{0, 1\}$, we write $\mathbf{C}_b := \mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)$, $\mathbf{C}_{\mathcal{P},b} := \mathbf{C}_{\mathcal{P}}(\tilde{x}_b)$. When we sample $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^d, \mathbf{V} = (\boldsymbol{v}_1^\top; \cdots; \boldsymbol{v}_d^\top) \xleftarrow{\$} \mathbb{Z}_p^{d \times n_{\mathcal{P}}}$ and set $\boldsymbol{z} = (\boldsymbol{w}; \mathsf{vec}(\mathbf{V}))$, we can represent $(1, \boldsymbol{z}^\top)(\mathbf{C}_0 | \mathbf{K}(\tilde{y}, \boldsymbol{y}))$ as

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}_1^\top \mathbf{C}_{\mathcal{P},0} | \cdots | \boldsymbol{v}_d^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{w}^\top \boldsymbol{y} \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top + (y_1 \boldsymbol{v}_1 + \cdots y_d \boldsymbol{v}_d)^\top \mathbf{K}_{\mathcal{P}}(\tilde{y})).$$

The case where $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 1$ can be proved in the same way as Theorem 1. We consider the case where $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 0$. If $\boldsymbol{y} \neq \mathbf{0}$, we can eliminate $\boldsymbol{w}^\top \boldsymbol{y} \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top$ as in the proof of Theorem 1 since $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ are independent. If $\boldsymbol{y} = \mathbf{0}$, we have $\boldsymbol{w}^\top \boldsymbol{y} \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top = \mathbf{0}$. We can replace $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$ by the security of $\mathcal{P}$ and independence of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$. By setting $\boldsymbol{w}' = \boldsymbol{w} + (\boldsymbol{x}_0 - \boldsymbol{x}_1)$, we can see that the desired two distributions are identical.

We show that $T_2(\mathcal{P})$ satisfies attribute key security. $(1, \boldsymbol{z}^\top)(\mathbf{C}_0 | \mathbf{K}_{\mathsf{Att}}(\tilde{y}))$ is equal to

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}_1^\top \mathbf{C}_{\mathcal{P},0} | \cdots | \boldsymbol{v}_d^\top \mathbf{C}_{\mathcal{P},0} | w_1 \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top + \boldsymbol{v}_1^\top \mathbf{K}_{\mathcal{P}}(\tilde{y}) | \cdots | w_d \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top + \boldsymbol{v}_d^\top \mathbf{K}_{\mathcal{P}}(\tilde{y})).$$

If $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 0$, by security of $\mathcal{P}$ and independence of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$, this distribution is identical to that of

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}_1^\top \mathbf{C}_{\mathcal{P},0} | \cdots | \boldsymbol{v}_d^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{v}_1^\top \mathbf{K}_{\mathcal{P}}(\tilde{y}) | \cdots | \boldsymbol{v}_d^\top \mathbf{K}_{\mathcal{P}}(\tilde{y})).$$

Thus, as in the proof of Theorem 1, we set $\boldsymbol{w}' = \boldsymbol{w} + (\boldsymbol{x}_0 - \boldsymbol{x}_1)$ and replace $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$. If $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 1$, we have $\boldsymbol{x}_0 = \boldsymbol{x}_1$. Thus, it suffices to replace $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$ using security of $\mathcal{P}$.

Finally, we show that $T_2(\mathcal{P})$ satisfies vector key security. $(1, \boldsymbol{z}^\top)(\mathbf{C}_0 | \mathbf{K}_{\mathsf{Vec}}(\boldsymbol{y}))$ can be represented as

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}_1^\top \mathbf{C}_{\mathcal{P},0} | \cdots | \boldsymbol{v}_d^\top \mathbf{C}_{\mathcal{P},0} | \boldsymbol{w}^\top \boldsymbol{y} | (y_1 \boldsymbol{v}_1 + \cdots y_d \boldsymbol{v}_d)^\top \mathbf{I}_{\mathbf{K}_{\mathcal{P}}}).$$

By security of $\mathcal{P}$ and independence of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$, we can replace $\mathbf{C}_{\mathcal{P},0}$ with $\mathbf{C}_{\mathcal{P},1}$. By setting $\boldsymbol{w}' = \boldsymbol{w} + (\boldsymbol{x}_0 - \boldsymbol{x}_1)$, we can see that the distributions are identical. □

5 AD-IND Secure Hierarchical Scheme with Compact Ciphertext from Predicate Encodings

Although our function encoding $T_2(\mathcal{P})$ supports both types of hierarchical structures and has compact secret keys, its ciphertext size is roughly d times larger than that of $T_1(\mathcal{P})$. In this section, we propose a new variant of the ACGU20ad framework. It achieves AD-IND security, supports both types of hierarchical structures, and has compact ciphertexts, at the cost of a large secret-key size.

5.1 Normalized Predicate Encoding

First, we introduce the normalized predicate encoding and show that any predicate encoding can be converted to the normalized form.

Definition 10. We say that a predicate encoding $\mathcal{P} = (\mathbf{C}_{\mathcal{P}}, \mathbf{K}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{P}})$ is normalized if $\boldsymbol{k}_{\mathcal{P}}^\top = (0, \ldots, 0, 1)$ and the last component of $\boldsymbol{d}_{\mathcal{P}}$ is 1.

Lemma 1. Any $(n, |\mathsf{ct}|, |\mathsf{sk}|)$-predicate encoding can be transformed to a normalized $(n+1, |\mathsf{ct}|, |\mathsf{sk}|+1)$-predicate encoding.

Proof. Let $\mathcal{P}$ be any predicate encoding. We define a normalized encoding $\mathcal{P}'$ as follows:

$$\mathbf{C}_{\mathcal{P}'}(\tilde{x}) = \left(\begin{array}{c} \mathbf{0}^\top \\ \hline \mathbf{C}_{\mathcal{P}}(\tilde{x}) \end{array}\right), \mathbf{K}_{\mathcal{P}'}(\tilde{y}) = \left(\begin{array}{c|c} \mathbf{0}^\top & 1 \\ \hline \boldsymbol{k}_{\mathcal{P}}(\tilde{y})^\top & -1 \\ \hline \mathbf{K}'_{\mathcal{P}}(\tilde{y}) & \mathbf{0} \end{array}\right), \boldsymbol{d}_{\mathcal{P}'}(\mathsf{part}(\tilde{x}), \tilde{y}) = (\boldsymbol{d}_{\mathcal{P}}(\mathsf{part}(\tilde{x}), \tilde{y}); 1).$$

It is easy to see that $\mathcal{P}'$ is correct if $\mathcal{P}$ is correct. We prove that $\mathcal{P}'$ is secure. It suffices to prove the first condition since the other conditions are not affected by the transformation. We have

$$(\alpha, \beta, \boldsymbol{v}^\top)(\mathbf{C}_{\mathcal{P}'}(\tilde{x}) | \mathbf{K}_{\mathcal{P}'}(\tilde{y})) = (\boldsymbol{v}^\top \mathbf{C}_{\mathcal{P}}(\tilde{x}) | (\beta, \boldsymbol{v}^\top) \mathbf{K}_{\mathcal{P}}(\tilde{y}) | \alpha - \beta).$$

If $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^n$, this distribution is identical to that of $(\boldsymbol{v}^\top \mathbf{C}_{\mathcal{P}}(\tilde{x}) | (0, \boldsymbol{v}^\top) \mathbf{K}_{\mathcal{P}}(\tilde{y}) | \alpha - \beta)$ by security of $\mathcal{P}$. Thus, when $\beta \xleftarrow{\$} \mathbb{Z}_p$, this distribution is independent of α. □

5.2 Construction

Let $\mathcal{P} = (\mathbf{C}_{\mathcal{P}}, \mathbf{K}_{\mathcal{P}}, \boldsymbol{d}_{\mathcal{P}})$ be a normalized predicate encoding. Based on the construction of $T_1(\mathcal{P})$, we define the functions $\mathbf{C}, \mathbf{K}_j, \boldsymbol{d}_j, \mathbf{K}_{\mathsf{Att},j}, \mathbf{K}_{\mathsf{Vec},j} (j \in [d])$ as follows. Note that $\mathbf{C}$ is the same as in T_1.

$$\mathbf{C}(\tilde{x}, \boldsymbol{x}) = \left(\begin{array}{c|c} \boldsymbol{x}^\top & \\ \hline \mathbf{I}_d & \mathbf{O} \\ \hline \mathbf{O} & \mathbf{C}_{\mathcal{P}}(\tilde{x}) \end{array}\right), \quad \mathbf{K}_j(\tilde{y}, \boldsymbol{y}) = \left(\begin{array}{c|c} \mathbf{0}^\top & 1 \\ \hline \mathbf{O} & \boldsymbol{y}[j]\boldsymbol{e}_j \\ \hline \multicolumn{2}{c}{\boldsymbol{y}[j]\mathbf{K}'_{\mathcal{P}}(\tilde{y})} \end{array}\right),$$

$$\boldsymbol{d}_j(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y}) = (\boldsymbol{y}[j]\boldsymbol{e}_j; -\boldsymbol{y}[j]\boldsymbol{d}_{\mathcal{P},1}(\mathsf{part}(\tilde{x}), \tilde{y}); -\boldsymbol{d}_{\mathcal{P},2}(\mathsf{part}(\tilde{x}), \tilde{y})),$$

$$\mathbf{K}_{\mathsf{Att},j}(\tilde{y}) = \left(\begin{array}{c|c} \mathbf{0}^\top & \\ \hline \mathbf{O} & \boldsymbol{e}_j \\ \hline \mathbf{K}'_{\mathcal{P}}(\tilde{y}) & \end{array}\right), \quad \mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y}) = \left(\begin{array}{c|c} 1 & \mathbf{0}^\top \\ \hline \boldsymbol{y}[j]\boldsymbol{e}_j & \mathbf{O} \\ \hline \mathbf{0} & \boldsymbol{y}[j]\mathbf{I}_{\mathbf{K}_{\mathcal{P}}} \end{array}\right).$$

Using these functions, we describe the new AB-IPFE scheme below.

$\underline{\mathsf{Setup}(1^\lambda)}$:

1. Generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$.
2. Samples $a \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_p^3$ and sets $\boldsymbol{a} := (1, a)^\top$.
3. Generates $(\mathbb{B} = \{\boldsymbol{b}_i\}_{i\in[6]}, \mathbb{B}^* = \{\boldsymbol{b}_i^*\}_{i\in[6]}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^6)$.
4. Samples $\mathbf{W}_i \xleftarrow{\$} \mathbb{Z}_p^{2\times 6} (i = 1, \ldots, n)$ and sets $\boldsymbol{\alpha}_i := \mathbf{W}_i^\top \boldsymbol{a}$.
5. Outputs $\mathsf{mpk} := ([\boldsymbol{a}]_1, \{[\boldsymbol{\alpha}_i]_1\}_i, [\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3]_1)$ and $\mathsf{msk} := ([\boldsymbol{b}_1^*]_2, [\boldsymbol{b}_4^*]_2, \{\mathbf{W}_i\}_i)$.

$\underline{\mathsf{Enc}(\mathsf{mpk}, \tilde{x}, \boldsymbol{x})}$:

1. Samples $s \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{c}_1]_1 := [s\boldsymbol{a}]_1$.
2. Computes $[\mathbf{C}_2]_1 := \left[\left(\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3 \middle| s\boldsymbol{\alpha}_1 \middle| \cdots \middle| s\boldsymbol{\alpha}_n\right)\mathbf{C}(\tilde{x}, \boldsymbol{x})\right]_1$.
3. Outputs $(\mathsf{part}(\tilde{x}), [\boldsymbol{c}_1]_1, [\mathbf{C}_2]_1)$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \tilde{y}, \boldsymbol{y})}$:

1. Samples $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{d-1} \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $\boldsymbol{\rho}_d := -\boldsymbol{\rho}_1 - \cdots - \boldsymbol{\rho}_{d-1}$.
2. Samples $r_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{1,j}]_2 := [\boldsymbol{b}_1^* + r_j\boldsymbol{b}_4^*]_2$ for each $j \in [d]$.
3. Computes $[\mathbf{K}_{2,j}]_2 := \left[\left(\boldsymbol{\rho}_j \middle| \mathbf{W}_1\boldsymbol{k}_{1,j} \middle| \cdots \middle| \mathbf{W}_n\boldsymbol{k}_{1,j}\right)\mathbf{K}_j(\tilde{y}, \boldsymbol{y})\right]_2$.
4. Outputs $(\tilde{y}, \boldsymbol{y}, \{[\boldsymbol{k}_{1,j}]_2, [\mathbf{K}_{2,j}]_2\}_{j\in[d]})$.

$\underline{\mathsf{KeyGen}_{\mathsf{Att}}(\mathsf{msk}, \tilde{y})}$:

1. Samples $r_j, r'_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{\mathsf{Att},1,j}]_2 := [\boldsymbol{b}_1^* + r_j\boldsymbol{b}_4^*]_2, [\boldsymbol{k}'_{\mathsf{Att},1,j}]_2 := [r'_j\boldsymbol{b}_4^*]_2$ for each j.
2. Computes $[\mathbf{K}_{\mathsf{Att},2,j}]_2 := \left[\left(\mathbf{0} \middle| \mathbf{W}_1\boldsymbol{k}_{\mathsf{Att},1,j} \middle| \cdots \middle| \mathbf{W}_n\boldsymbol{k}_{\mathsf{Att},1,j}\right)\mathbf{K}_{\mathsf{Att},j}(\tilde{y})\right]_2$.
3. Computes $[\mathbf{K}'_{\mathsf{Att},2,j}]_2 := \left[\left(\mathbf{0} \middle| \mathbf{W}_1\boldsymbol{k}'_{\mathsf{Att},1,j} \middle| \cdots \middle| \mathbf{W}_n\boldsymbol{k}'_{\mathsf{Att},1,j}\right)\mathbf{K}_{\mathsf{Att},j}(\tilde{y})\right]_2$.
4. Outputs $(\tilde{y}, \{[\boldsymbol{k}_{\mathsf{Att},1,j}]_2, [\boldsymbol{k}'_{\mathsf{Att},1,j}]_2, [\mathbf{K}_{\mathsf{Att},2,j}]_2, [\mathbf{K}'_{\mathsf{Att},2,j}]_2\}_{j\in[d]})$.

$\underline{\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}(\mathsf{sk}_{\tilde{y}}, \boldsymbol{y})}$:

1. Samples $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{d-1} \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $\boldsymbol{\rho}_d := -\boldsymbol{\rho}_1 - \cdots - \boldsymbol{\rho}_{d-1}$.

2. Samples $t_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{1,j}]_2 := [\boldsymbol{k}_{\mathsf{Att},1,j} + t_j \boldsymbol{k}'_{\mathsf{Att},1,j}]_2$.
3. Computes $[\mathbf{K}_{2,j}]_2 := [\boldsymbol{y}[j](\mathbf{K}_{\mathsf{Att},2,j} + t_j \mathbf{K}'_{\mathsf{Att},2,j}) + (\mathbf{O}|\boldsymbol{\rho}_j)]_2$.
4. Outputs $(\tilde{y}, \boldsymbol{y}, \{[\boldsymbol{k}_{1,j}]_2, [\mathbf{K}_{2,j}]_2\}_{j\in[d]})$.

$\underline{\mathsf{KeyGen}_{\mathsf{Vec}}(\mathsf{msk}, \boldsymbol{y})}$:

1. Samples $\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_{d-1} \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $\boldsymbol{\sigma}_d := -\boldsymbol{\sigma}_1 - \cdots - \boldsymbol{\sigma}_{d-1}$.
2. Samples $\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_{d-1} \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $\boldsymbol{\sigma}'_d := -\boldsymbol{\sigma}'_1 - \cdots - \boldsymbol{\sigma}'_{d-1}$.
3. Samples $r_j, r'_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{\mathsf{Vec},1,j}]_2 := [\boldsymbol{b}^*_1 + r_j \boldsymbol{b}^*_4]_2, [\boldsymbol{k}'_{\mathsf{Vec},1,j}]_2 := [r'_j \boldsymbol{b}^*_4]_2$ for each j.
4. Computes $[\mathbf{K}_{\mathsf{Vec},2,j}]_2 := \left[\left(\boldsymbol{\sigma}_j \middle| \mathbf{W}_1 \boldsymbol{k}_{\mathsf{Vec},1,j} \middle| \cdots \middle| \mathbf{W}_n \boldsymbol{k}_{\mathsf{Vec},1,j}\right) \mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y})\right]_2$.
5. Computes $[\mathbf{K}'_{\mathsf{Vec},2,j}]_2 := \left[\left(\boldsymbol{\sigma}'_j \middle| \mathbf{W}_1 \boldsymbol{k}'_{\mathsf{Vec},1,j} \middle| \cdots \middle| \mathbf{W}_n \boldsymbol{k}'_{\mathsf{Vec},1,j}\right) \mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y})\right]_2$.
6. Outputs $(\boldsymbol{y}, \{[\boldsymbol{k}_{\mathsf{Vec},1,j}]_2, [\boldsymbol{k}'_{\mathsf{Vec},1,j}]_2, [\mathbf{K}_{\mathsf{Vec},2,j}]_2, [\mathbf{K}'_{\mathsf{Vec},2,j}]_2\}_{j\in[d]})$.

$\underline{\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}(\mathsf{sk}_{\boldsymbol{y}}, \tilde{y})}$:

1. Samples $\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_{d-1} \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $\boldsymbol{\tau}_d := -\boldsymbol{\tau}_1 - \cdots - \boldsymbol{\tau}_{d-1}$.
2. Samples $t_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{1,j}]_2 := [\boldsymbol{k}_{\mathsf{Vec},1,j} + t_j \boldsymbol{k}'_{\mathsf{Vec},1,j}]_2$.
3. Computes $[\mathbf{K}_{2,j}]_2 := [(\mathbf{K}_{\mathsf{Vec},2,j} + t_j \mathbf{K}'_{\mathsf{Vec},2,j})\mathbf{K}_{\mathcal{P}}(\tilde{y}) + (\mathbf{O}|\boldsymbol{\tau}_j)]_2$.
4. Outputs $(\tilde{y}, \boldsymbol{y}, \{[\boldsymbol{k}_{1,j}]_2, [\mathbf{K}_{2,j}]_2\}_{j\in[d]})$.

$\underline{\mathsf{Dec}(\mathsf{sk}_{\tilde{y},\boldsymbol{y}} = (\tilde{y}, \boldsymbol{y}, \{[\boldsymbol{k}_{1,j}]_2, [\mathbf{K}_{2,j}]_2\}_{j\in[d]}), C = (\mathsf{part}(\tilde{x}), [\boldsymbol{c}_1]_1, [\mathbf{C}_2]_1))}$:

1. Computes $[\boldsymbol{v}_{1,j}]_T := e([\mathbf{C}_2]_1^\top, [\boldsymbol{k}_{1,j}]_2)$ and $[\boldsymbol{v}_{2,j}^\top]_T := e([\boldsymbol{c}_1]_1^\top, [\mathbf{K}_{2,j}]_2)$.
2. Computes $[\gamma_j]_T := [(\boldsymbol{v}_{1,j}; \boldsymbol{v}_{2,j})^\top \boldsymbol{d}_j(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y})]_T$.
3. Computes $\gamma_1 + \cdots + \gamma_d$ from $[\gamma_1 + \cdots + \gamma_d]_T$ by brute-forcing the discrete logarithm.

Correctness

Let $R(\tilde{x}, \tilde{y}) = 1$. We have $\boldsymbol{v}_{1,j}^\top = (1, s\boldsymbol{a}^\top \mathbf{W}_1 \boldsymbol{k}_{1,j}, \dots, s\boldsymbol{a}^\top \mathbf{W}_n \boldsymbol{k}_{1,j}) \mathbf{C}(\tilde{x}, \boldsymbol{x})$ and $\boldsymbol{v}_{2,j}^\top = (\boldsymbol{c}_1^\top \boldsymbol{\rho}_j, s\boldsymbol{a}^\top \mathbf{W}_1 \boldsymbol{k}_{1,j}, \dots, s\boldsymbol{a}^\top \mathbf{W}_n \boldsymbol{k}_{1,j}) \mathbf{K}_j(\tilde{y}, \boldsymbol{y})$. By correctness and normalized structure of the predicate encoding, it follows that

$$(\boldsymbol{v}_{1,j}; \boldsymbol{v}_{2,j})^\top \boldsymbol{d}_j(\mathsf{part}(\tilde{x}), \tilde{y}, \boldsymbol{y}) = x_j y_j + \boldsymbol{c}_1^\top \boldsymbol{\rho}_j.$$

Then, we have $\gamma_1 + \cdots + \gamma_d = \boldsymbol{x}^\top \boldsymbol{y}$. Delegation invariance follows from straightforward matrix computations.

We give the security proof in the next section.

Theorem 3. If $\mathcal{P}$ is a secure predicate encoding and the SXDH assumption holds, the resulting hierarchical AB-IPFE scheme is AD-IND secure.

5.3 Proof of Theorem 3

We show that the encoding in Section 5.2 satisfies some useful properties. It will serve as a substitute for function encoding security in the security proof of AB-IPFE. In the case that the key is admissible, we treat keys for all $j \in [d]$ at once. On the other hand, when the key is not admissible, these keys are treated separately.

Lemma 2. Assume that $\mathcal{P}$ is a secure normalized predicate encoding with the leakage function part. In addition, let $\tilde{x}_0, \boldsymbol{x}_0, \tilde{x}_1, \boldsymbol{x}_1, \tilde{y}, \boldsymbol{y}$ satisfy $\mathsf{part}(\tilde{x}_0) = \mathsf{part}(\tilde{x}_1)$, $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 1$, and $\langle \boldsymbol{x}_0, \boldsymbol{y} \rangle = \langle \boldsymbol{x}_1, \boldsymbol{y} \rangle$. Then the following properties hold for any $\beta = 0, 1$:

(1) The distribution of $\big((\beta, \boldsymbol{z}^\top)\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b), \{(\rho_j, \boldsymbol{z}^\top)\mathbf{K}_j(\tilde{y}, \boldsymbol{y})\}_{j\in[d]}\big)$ where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n, \rho_1, \ldots, \rho_{d-1} \xleftarrow{\$} \mathbb{Z}_p, \rho_d := -\rho_1 - \cdots - \rho_{d-1}$ does not depend on $b \in \{0,1\}$.

(2) If $\boldsymbol{x}_0 = \boldsymbol{x}_1$, the distribution of $\big((\beta, \boldsymbol{z}^\top)\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b), \{(0, \boldsymbol{z}^\top)\mathbf{K}_{\mathsf{Att},j}(\tilde{y})\}_{j\in[d]}\big)$ where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$ does not depend on $b \in \{0,1\}$.

(3) The distribution of $\big((\beta, \boldsymbol{z}^\top)\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b), \{(\rho_j, \boldsymbol{z}^\top)\mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y})\}_{j\in[d]}\big)$ where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n, \rho_1, \ldots, \rho_{d-1} \xleftarrow{\$} \mathbb{Z}_p, \rho_d := -\rho_1 - \cdots - \rho_{d-1}$ does not depend on $b \in \{0,1\}$.

Proof. Let $\boldsymbol{y} = (y_1, \ldots, y_d)^\top$. It is enough to show in the case $\beta = 1$. In fact, the case of $\beta = 0$ follows by sampling two instances of $\beta = 1$ and computing the difference between them. When we write $\boldsymbol{z} = (\boldsymbol{w}; \boldsymbol{v})$ where $\boldsymbol{w} = (w_1, \ldots, w_d)^\top \xleftarrow{\$} \mathbb{Z}_p^d, \boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^{n_\mathcal{P}}$, it holds that $(1, \boldsymbol{z}^\top)\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0) = (\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}^\top \mathbf{C}_\mathcal{P}(\tilde{x}_0))$.

(1) We have $(\rho_j, \boldsymbol{z}^\top)\mathbf{K}_j(\tilde{y}, \boldsymbol{y}) = (0, \ldots, 0, \rho_j + y_j w_j) + y_j \boldsymbol{v}^\top \mathbf{K}'_\mathcal{P}(\tilde{y})$.
When we set $\boldsymbol{w}' = \boldsymbol{w} + \boldsymbol{x}_0 - \boldsymbol{x}_1$ and $\rho'_j = \rho_j - y_j(\boldsymbol{x}_0 - \boldsymbol{x}_1)^\top \boldsymbol{e}_j$, it holds that

$$(\boldsymbol{x}_0^\top + \boldsymbol{w}^\top | \boldsymbol{v}^\top \mathbf{C}_\mathcal{P}(\tilde{x}_0)) = (\boldsymbol{x}_1^\top + \boldsymbol{w}'^\top | \boldsymbol{v}^\top \mathbf{C}_\mathcal{P}(\tilde{x}_0)), (\rho_j, \boldsymbol{w}^\top, \boldsymbol{v}^\top)\mathbf{K}_j(\tilde{y}, \boldsymbol{y}) = (\rho'_j, \boldsymbol{w}'^\top, \boldsymbol{v}^\top)\mathbf{K}_j(\tilde{y}, \boldsymbol{y}).$$

Note that $\rho'_1 + \cdots + \rho'_d = 0$. By security of the predicate encoding, the distribution of $(\boldsymbol{v}^\top \mathbf{C}_\mathcal{P}(\tilde{x}_0), \boldsymbol{v}^\top \mathbf{K}'_\mathcal{P}(\tilde{y}))$ is the same as that of $(\boldsymbol{v}^\top \mathbf{C}_\mathcal{P}(\tilde{x}_1), \boldsymbol{v}^\top \mathbf{K}'_\mathcal{P}(\tilde{y}))$.

(2) It easily follows from security of the predicate encoding.

(3) Since $(\rho_j, \boldsymbol{z}^\top)\mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y}) = (\rho_j + y_j w_j, 0, \ldots, 0) + y_j \boldsymbol{v}^\top \mathbf{I}_{\mathbf{K}_\mathcal{P}}$, it follows in the same way as (1).

□

Lemma 3. Assume that $\mathcal{P}$ is a secure normalized predicate encoding with the leakage function part. In addition, let $\tilde{x}_0, \boldsymbol{x}_0, \tilde{x}_1, \boldsymbol{x}_1, \tilde{y}, \boldsymbol{y}$ satisfy $\mathsf{part}(\tilde{x}_0) = \mathsf{part}(\tilde{x}_1)$ and $R(\tilde{x}_0, \tilde{y}) = R(\tilde{x}_1, \tilde{y}) = 0$. Then the following properties hold for any $\beta = 0, 1$:

(1) For any $j \in [d]$ and $\rho \in \mathbb{Z}_p$, the distribution of $\big((\beta, \boldsymbol{z}^\top)\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b), (\rho, \boldsymbol{z}^\top)\mathbf{K}_j(\tilde{y}, \boldsymbol{y})\big)$ where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$ does not depend on $b \in \{0,1\}$.

(2) For any $j \in [d]$, the distribution of $\big((\beta, \boldsymbol{z}^\top)\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b), (0, \boldsymbol{z}^\top)\mathbf{K}_{\mathsf{Att},j}(\tilde{y})\big)$ where $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$ does not depend on $b \in \{0,1\}$.

Proof. We use the same notation as in the proof of Lemma 2.

(1) immediately follows from (2) since $(\rho, \boldsymbol{z}^\top)\mathbf{K}_j(\tilde{y}, \boldsymbol{y}) = (0, \boldsymbol{z}^\top)\mathbf{K}_{\mathsf{Att},j}(\tilde{y}) \cdot y_j + (0, \ldots, 0, \rho)$. As Lemma 2, it is enough to show (2) when $\beta = 1$. In this case, we have

$$(0, \boldsymbol{z}^\top)\mathbf{K}_{\mathsf{Att},j}(\tilde{y}) = (0, \ldots, 0, w_j) + \boldsymbol{v}^\top \mathbf{K}'_\mathcal{P}(\tilde{y}).$$

By security of $\mathcal{P}$, this distribution is independent of w_j. Thus, $\boldsymbol{w}$ masks $\boldsymbol{x}_0$ completely. □

We use the following two lemmas for security proof.

Lemma 4. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$, $r \xleftarrow{\$} \mathbb{Z}_p$, and $(\mathbb{D} = \{\boldsymbol{d}_i\}_{i\in[n+3]}, \mathbb{D}^* = \{\boldsymbol{d}_i^*\}_{i\in[n+3]}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^{n+3})$. For a PPT algorithm $\mathcal{A}$, we set

$$D := (\mathsf{span}\{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}, \boldsymbol{d}_1 + \boldsymbol{d}_2, \boldsymbol{d}_3, [\boldsymbol{d}_1^*]_2, [\boldsymbol{d}_2^*]_2, [\boldsymbol{d}_3^*]_2, \boldsymbol{d}_4, \ldots, \boldsymbol{d}_{n+3}, \boldsymbol{d}_4^*, \ldots, \boldsymbol{d}_{n+3}^*),$$
$$\mathsf{Adv}^{\mathsf{P1}}_\mathcal{A}(\lambda) := |\Pr[1 \leftarrow \mathcal{A}(D, [\boldsymbol{d}_1^* + r\boldsymbol{d}_3^*]_2)] - \Pr[1 \leftarrow \mathcal{A}(D, [\boldsymbol{d}_2^* + r\boldsymbol{d}_3^*]_2)]|.$$

Then for any $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ such that

$$\mathsf{Adv}^{\mathsf{P1}}_\mathcal{A}(\lambda) \le 2\mathsf{Adv}^{\mathsf{DDH},\mathbb{G}_2}_\mathcal{B}(\lambda).$$

Proof. By symmetry, it is enough to show that there exists $\mathcal{B}$ such that

$$|\Pr[1 \leftarrow \mathcal{A}(D, [\boldsymbol{d}_1^* + r\boldsymbol{d}_3^*]_2)] - \Pr[1 \leftarrow \mathcal{A}(D, [(1-s)\boldsymbol{d}_1^* + s\boldsymbol{d}_2^* + r\boldsymbol{d}_3^*]_2)]| \le \mathsf{Adv}^{\mathsf{DDH},\mathbb{G}_2}_\mathcal{B}(\lambda).$$

where $s \xleftarrow{\$} \mathbb{Z}_p$. $\mathcal{B}$ gets $[a]_2, [b]_2, [c]_2$ where $c = ab$ or $c = ab + s$ where $s \xleftarrow{\$} \mathbb{Z}_p$. $\mathcal{B}$ generates $(\mathbb{F} = \{\boldsymbol{f}_i\}_{i\in[n+3]}, \mathbb{F}^* = \{\boldsymbol{f}_i^*\}_{i\in[n+3]}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^{n+3})$ and virtually sets

$$\begin{aligned}
&\boldsymbol{d}_1 = \boldsymbol{f}_1 + a\boldsymbol{f}_3, && \boldsymbol{d}_2 = \boldsymbol{f}_2 - a\boldsymbol{f}_3, && \boldsymbol{d}_3 = \boldsymbol{f}_3, && \boldsymbol{d}_k = \boldsymbol{f}_k \ (k \geq 4),\\
&\boldsymbol{d}_1^* = \boldsymbol{f}_1^*, && \boldsymbol{d}_2^* = \boldsymbol{f}_2^*, && \boldsymbol{d}_3^* = \boldsymbol{f}_3^* - a\boldsymbol{f}_1^* + a\boldsymbol{f}_2^*, && \boldsymbol{d}_k^* = \boldsymbol{f}_k^* \ (k \geq 4).
\end{aligned}$$

Also, $\mathcal{B}$ virtually sets $r = b$. Then, $\mathcal{B}$ can compute corresponding D and

$$[b\boldsymbol{f}_3^* + c(\boldsymbol{f}_2^* - \boldsymbol{f}_1^*) + \boldsymbol{f}_1^*]_2 = \begin{cases} [\boldsymbol{d}_1^* + r\boldsymbol{d}_3^*]_2 & \text{if } c = ab \\ [(1-s)\boldsymbol{d}_1^* + s\boldsymbol{d}_2^* + r\boldsymbol{d}_3^*]_2 & \text{if } c = ab + s \end{cases}$$

□

Lemma 5 ([CGW18])**.** Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$, $r \xleftarrow{\$} \mathbb{Z}_p$, and $(\mathbb{D}, \mathbb{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^{n+2})$. For a PPT algorithm $\mathcal{A}$, we set

$$\begin{aligned}
D &:= (\mathsf{span}\{\boldsymbol{d}_1, \boldsymbol{d}_2\}, [\boldsymbol{d}_1^*]_2, [\boldsymbol{d}_2^*]_2, \boldsymbol{d}_3, \ldots, \boldsymbol{d}_{n+2}, \boldsymbol{d}_3^*, \ldots, \boldsymbol{d}_{n+2}^*)\\
\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{A}}(\lambda) &:= |\Pr[1 \leftarrow \mathcal{A}(D, [r\boldsymbol{d}_1^*]_2)] - \Pr[1 \leftarrow \mathcal{A}(D, [r\boldsymbol{d}_2^*]_2)]|
\end{aligned}$$

Then for any $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ such that

$$\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{A}}(\lambda) \leq 2\mathsf{Adv}^{\mathsf{DDH},\mathbb{G}_2}_{\mathcal{B}}(\lambda).$$

Now we show AD-IND security of our scheme. Following the proof technique in the previous works, we firstly introduce the symmetric-key variant $\mathsf{Game}_0^{\mathsf{S}}$ of the original security game.

Setup Phase:

1. It generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$.
2. It generates $(\mathbb{B} = \{\boldsymbol{b}_i\}_{i\in[6]}, \mathbb{B}^* = \{\boldsymbol{b}_i^*\}_{i\in[6]}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^6)$.
3. It samples $w_{i,k} \xleftarrow{\$} \mathbb{Z}_p (i \in [n], k \in [6])$ and sets $\boldsymbol{w}_i := \sum_{k=1}^{6} w_{i,k}\boldsymbol{b}_k$.
4. It makes $[\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3]_1$ public.

Challenge Phase:

1. It samples $b \xleftarrow{\$} \{0, 1\}$.
2. Given $(\tilde{x}_0, \boldsymbol{x}_0)$ and $(\tilde{x}_1, \boldsymbol{x}_1)$, it computes $C^* := (\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3 | \boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n)\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)$.
3. It returns $(\mathsf{part}(\tilde{x}_b), C^*)$.

Key Oracle:

1. It samples $\rho_1, \ldots, \rho_{d-1} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\rho_d := -\rho_1 - \cdots - \rho_{d-1}$.
2. It samples $r_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{1,j}]_2 := [\boldsymbol{b}_1^* + r_j\boldsymbol{b}_4^*]_2$ for each $j \in [d]$.
3. It sets $[\boldsymbol{k}_{2,j}^\top]_2 := [(\rho_j, \boldsymbol{w}_1^\top\boldsymbol{k}_{1,j}, \ldots, \boldsymbol{w}_n^\top\boldsymbol{k}_{1,j})\mathbf{K}_j(\tilde{y}, \boldsymbol{y})]_2$ for each $j \in [d]$.
4. It returns $\{[\boldsymbol{k}_{1,j}]_2, [\boldsymbol{k}_{2,j}^\top]_2\}_{j\in[d]}$.

β-Attribute Key Oracle ($\beta = 0, 1$):

1. It samples $r_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{\mathsf{Att},1,j}]_2 := [\beta\boldsymbol{b}_1^* + r_j\boldsymbol{b}_4^*]_2$.
2. It sets $[\boldsymbol{k}_{\mathsf{Att},2,j}]_2 := [(0|\boldsymbol{w}_1^\top\boldsymbol{k}_{\mathsf{Att},1,j}| \cdots |\boldsymbol{w}_n^\top\boldsymbol{k}_{\mathsf{Att},1,j})\mathbf{K}_{\mathsf{Att},j}(\tilde{y})]_2$.
3. It returns $\{[\boldsymbol{k}_{\mathsf{Att},1,j}]_2, [\boldsymbol{k}_{\mathsf{Att},2,j}^\top]_2\}_{j\in[d]}$.

β-Vector Key Oracle ($\beta = 0, 1$):

1. It samples $\sigma_1, \ldots, \sigma_{d-1} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\sigma_d := -\sigma_1 - \cdots - \sigma_{d-1}$.

2. It samples $r_j \xleftarrow{\$} \mathbb{Z}_p$ and computes $[\boldsymbol{k}_{\mathsf{Vec},1,j}]_2 := [\beta \boldsymbol{b}_1^* + r_j \boldsymbol{b}_4^*]_2$.
3. It sets $[\boldsymbol{k}_{\mathsf{Vec},2,j}^\top]_2 := [(\sigma_j | \boldsymbol{w}_1^\top \boldsymbol{k}_{\mathsf{Vec},1,j} | \cdots | \boldsymbol{w}_n^\top \boldsymbol{k}_{\mathsf{Vec},1,j}) \mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y})]_2$.
4. It returns $\{[\boldsymbol{k}_{\mathsf{Vec},1,j}]_2, [\boldsymbol{k}_{\mathsf{Vec},2,j}^\top]_2\}_{j\in[d]}$.

We define the games $\mathsf{Game}_0, \mathsf{Game}_1, \mathsf{Game}_2$ as follows.

$\underline{\mathsf{Game}_0}$: This is the real AD-IND security game.

$\underline{\mathsf{Game}_1}$: In this game, we change the way that the challenge ciphertext is generated. Instead of setting $[\boldsymbol{c}_1]_1 = [s\boldsymbol{a}]_1$, we sample $[\boldsymbol{c}_1]_1$ uniformly at random from $\mathbb{G}_1^2$. Also, each $s\alpha_i$ is replaced with $\mathbf{W}_i^\top \boldsymbol{c}_1$. Then, there exists a PPT algorithm $\mathcal{B}_1$ such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}(\lambda)\right| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{DDH},\mathbb{G}_1}(\lambda).$$

We show that for any $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}_2$ such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{Game}_0^{\mathsf{S}}}(\lambda) + \mathsf{negl}(\lambda).$$

Let $\{\boldsymbol{w}_i\}_{i\in[n]}$ be the master secret key of $\mathsf{Game}_0^{\mathsf{S}}$. After getting $[\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3]_1$, $\mathcal{B}_2$ samples $a \xleftarrow{\$} \mathbb{Z}_p, \boldsymbol{c}_1 \xleftarrow{\$} \mathbb{Z}_p^2$ and sets $\boldsymbol{a} = (1; a)$. Then, $(\boldsymbol{a}|\boldsymbol{c}_1)$ is invertible with an overwhelming probability. $\mathcal{B}_2$ samples $\tilde{\boldsymbol{w}}_i \xleftarrow{\$} \mathbb{Z}_p^6$ and virtually sets $\mathbf{W}_i^\top = (\tilde{\boldsymbol{w}}_i | \boldsymbol{w}_i)(\boldsymbol{a}|\boldsymbol{c}_1)^{-1}$. $\mathcal{B}_2$ sends $[\boldsymbol{a}]_1$, $[\mathbf{W}_i^\top \boldsymbol{a}]_1 = [\tilde{\boldsymbol{w}}_i]_1$, and $[\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3]_1$ to $\mathcal{A}$ as the master public key. $\mathcal{B}_2$ simulates the response to queries from $\mathcal{A}$ in the following way.

- When $\mathcal{A}$ makes a challenge query, $\mathcal{B}_2$ forwards it to the challenger. Then, $\mathcal{B}_2$ sends the response back to $\mathcal{A}$. Note that $\mathbf{W}_i^\top \boldsymbol{c}_1 = \boldsymbol{w}_i$ holds.
- When $\mathcal{A}$ makes a key query, $\mathcal{B}_2$ forwards it to the challenger. Let $\{[\boldsymbol{k}_{1,j}]_2, [\boldsymbol{k}_{2,j}^\top]_2\}_{j\in[d]}$ be its response. $\mathcal{B}_2$ samples $\tilde{\rho}_1, \ldots, \tilde{\rho}_{d-1} \xleftarrow{\$} \mathbb{Z}_p$, sets $\tilde{\rho}_d = -\tilde{\rho}_1 - \cdots - \tilde{\rho}_{d-1}$, and computes

$$[\tilde{\boldsymbol{k}}_{2,j}^\top]_2 = [(\tilde{\rho}_j, \tilde{\boldsymbol{w}}_1^\top \boldsymbol{k}_{1,j}, \ldots, \tilde{\boldsymbol{w}}_n^\top \boldsymbol{k}_{1,j}) \mathbf{K}_j(\tilde{y}, \boldsymbol{y})]_2,$$
$$[\mathbf{K}_{2,j}]_2 = [(\boldsymbol{a}|\boldsymbol{c}_1)^{-\top} (\tilde{\boldsymbol{k}}_{2,j}^\top; \boldsymbol{k}_{2,j}^\top)]_2.$$

  $\mathcal{B}_2$ sends $[\boldsymbol{k}_{1,j}]_2$ and $[\mathbf{K}_{2,j}]_2$ to $\mathcal{A}$.
- When $\mathcal{A}$ makes an attribute key query, $\mathcal{B}_2$ sends the corresponding 0-attribute 1-attribute key query to the challenger. Let $\{[\boldsymbol{k}_{\mathsf{Att},1,j}]_2, [\boldsymbol{k}_{\mathsf{Att},2,j}^\top]_2\}_{j\in[d]}$ and $\{[\boldsymbol{k}'_{\mathsf{Att},1,j}]_2, [\boldsymbol{k}'^\top_{\mathsf{Att},2,j}]_2\}_{j\in[d]}$ be the response to the 1-attribute query and the 0-attribute query respectively. $\mathcal{B}_2$ computes

$$[\tilde{\boldsymbol{k}}_{\mathsf{Att},2,j}^\top]_2 = [(0, \tilde{\boldsymbol{w}}_1^\top \boldsymbol{k}_{\mathsf{Att},1,j}, \ldots, \tilde{\boldsymbol{w}}_n^\top \boldsymbol{k}_{\mathsf{Att},1,j}) \mathbf{K}_{\mathsf{Att},j}(\tilde{y})]_2,$$
$$[\mathbf{K}_{\mathsf{Att},2,j}]_2 = [(\boldsymbol{a}|\boldsymbol{c}_1)^{-\top} (\tilde{\boldsymbol{k}}_{\mathsf{Att},2,j}^\top; \boldsymbol{k}_{\mathsf{Att},2,j}^\top)]_2.$$

  In the same way, $\mathcal{B}_2$ also computes $[\mathbf{K}'_{\mathsf{Att},2,j}]_2$ by using the response to the 0-attribute query. $\mathcal{B}_2$ sends $\{[\boldsymbol{k}_{\mathsf{Att},1,j}]_2, [\boldsymbol{k}'_{\mathsf{Att},1,j}]_2, [\mathbf{K}_{\mathsf{Att},2,j}]_2, [\mathbf{K}'_{\mathsf{Att},2,j}]_2\}_{j\in[d]}$ to $\mathcal{A}$.
- When $\mathcal{A}$ makes the vector key query, $\mathcal{B}_2$ sends the corresponding 0-vector and 1-vector key query to the challenger. Let $\{[\boldsymbol{k}_{\mathsf{Vec},1,j}]_2, [\boldsymbol{k}_{\mathsf{Vec},2,j}^\top]_2\}_{j\in[d]}$ and $\{[\boldsymbol{k}'_{\mathsf{Vec},1,j}]_2, [\boldsymbol{k}'^\top_{\mathsf{Vec},2,j}]_2\}_{j\in[d]}$ be the response to the 1-vector query and the 0-vector key query respectively. $\mathcal{B}_2$ samples $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_{d-1} \xleftarrow{\$} \mathbb{Z}_p$, sets $\tilde{\sigma}_d = -\tilde{\sigma}_1 - \cdots - \tilde{\sigma}_{d-1}$, and computes

$$[\tilde{\boldsymbol{k}}_{\mathsf{Vec},2,j}^\top]_2 = [(\tilde{\sigma}_j, \tilde{\boldsymbol{w}}_1^\top \boldsymbol{k}_{\mathsf{Vec},1,j}, \ldots, \tilde{\boldsymbol{w}}_n^\top \boldsymbol{k}_{\mathsf{Vec},1,j}) \mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y})]_2,$$
$$[\mathbf{K}_{\mathsf{Vec},2,j}]_2 = [(\boldsymbol{a}|\boldsymbol{c}_1)^{-\top} (\tilde{\boldsymbol{k}}_{\mathsf{Vec},2,j}^\top; \boldsymbol{k}_{\mathsf{Vec},2,j}^\top)]_2.$$

  In the same way, $\mathcal{B}_2$ also computes $[\mathbf{K}'_{\mathsf{Vec},2,j}]_2$ by using the response to the 0-vector query and freshly sampled $\{\tilde{\sigma}'_i\}_{i\in[d-1]}$. $\mathcal{B}_2$ sends $\{[\boldsymbol{k}_{\mathsf{Vec},1,j}]_2, [\boldsymbol{k}'_{\mathsf{Vec},1,j}]_2, [\mathbf{K}_{\mathsf{Vec},2,j}]_2, [\mathbf{K}'_{\mathsf{Vec},2,j}]_2\}_{j\in[d]}$ to $\mathcal{A}$.

Finally, $\mathcal{B}_2$ forwards the guess bit from $\mathcal{A}$ to the challenger. Since $\mathcal{B}_2$ correctly simulates the response to $\mathcal{A}$ in Game_1, the inequality holds.

Now, we evaluate the advantage in $\mathsf{Game}_0^{\mathsf{S}}$. Let Q be an upper bound of the number of queries that the adversary makes. For $S \subset [6]$, let $\boldsymbol{w}_{i,S} := \sum_{k \in S} w_{i,k} \boldsymbol{b}_k$.

$\underline{\mathsf{Game}_0^{\mathsf{S},0}}$: The challenger sets $b = 0$ in the setting of $\mathsf{Game}_0^{\mathsf{S}}$.

$\underline{\mathsf{Game}_1^{\mathsf{S}}}$: In this game, we change the way that the challenge ciphertext is generated:

$$C^* = \left(\boldsymbol{b}_1 + \boldsymbol{b}_2 \middle| \boldsymbol{w}_{1,\{1,2,4,5\}} \middle| \cdots \middle| \boldsymbol{w}_{n,\{1,2,4,5\}}\right)\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0) + \left(\boldsymbol{b}_3 \middle| \boldsymbol{w}_{1,\{3,6\}} \middle| \cdots \middle| \boldsymbol{w}_{n,\{3,6\}}\right)\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1).$$

We note that $w_{1,3}, \ldots, w_{n,3}, w_{1,6}, \ldots, w_{n,6}$ are not used in the key generation steps. By security of the function encoding T_1, the distribution $(1, w_{1,3}, \ldots, w_{n,3})\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)$ or $(0, w_{1,6}, \ldots, w_{n,6})\mathbf{C}(\tilde{x}_b, \boldsymbol{x}_b)$ does not depend on b. Thus, we have $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_0^{\mathsf{S},0}}(1^\lambda) = 0] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_1^{\mathsf{S}}}(1^\lambda) = 0]$.

$\underline{\mathsf{Game}_{1,q}^{\mathsf{S}}}$ ($q \in [Q]$): Let $\mathsf{Game}_{1,0}^{\mathsf{S}}$ denote $\mathsf{Game}_1^{\mathsf{S}}$. In this game, we change the response to the q-th query of $\mathsf{Game}_{1,q-1}^{\mathsf{S}}$. The challenger uses $[\beta \boldsymbol{b}_3^* + r_j \boldsymbol{b}_6^*]_2$ instead of $[\beta \boldsymbol{b}_1^* + r_j \boldsymbol{b}_4^*]_2$ where $\beta = 0$ or $\beta = 1$. Note that $\beta = 1$ when it is a key query.

$\underline{\mathsf{Game}_2^{\mathsf{S}}}$: In this game, we change the way that the challenge ciphertext is generated from $\mathsf{Game}_{1,Q}^{\mathsf{S}}$:

$$C^* = (\boldsymbol{b}_1 + \boldsymbol{b}_2 + \boldsymbol{b}_3 | \boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n)\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1).$$

In the same way as the change from $\mathsf{Game}_0^{\mathsf{S},0}$ to $\mathsf{Game}_1^{\mathsf{S}}$, we have $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,Q}^{\mathsf{S}}}(1^\lambda) = 0] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_2^{\mathsf{S}}}(1^\lambda) = 0]$.

$\underline{\mathsf{Game}_0^{\mathsf{S},1}}$: In this game, the challenger sets $b = 1$ in the setting of $\mathsf{Game}_0^{\mathsf{S}}$. The difference from $\mathsf{Game}_2^{\mathsf{S}}$ is the response to the queries. The challenger uses $[\beta \boldsymbol{b}_1^* + r_j \boldsymbol{b}_4^*]_2$ instead of $[\beta \boldsymbol{b}_3^* + r_j \boldsymbol{b}_6^*]_2$. Since swapping $(\boldsymbol{b}_1, \boldsymbol{b}_1^*, \boldsymbol{b}_4, \boldsymbol{b}_4^*)$ and $(\boldsymbol{b}_3, \boldsymbol{b}_3^*, \boldsymbol{b}_6, \boldsymbol{b}_6^*)$ does not change the game, we have $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_2^{\mathsf{S}}}(1^\lambda) = 0] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_0^{\mathsf{S},1}}(1^\lambda) = 0]$.

Finally, we show that

$$\left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q-1}^{\mathsf{S}}}(1^\lambda) = 0] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q}^{\mathsf{S}}}(1^\lambda) = 0]\right| \leq 8(d+1) \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH},\mathbb{G}_2}(\lambda)$$

for any $q \in [Q]$. Let E_q be an event that the q-th query is admissible (Note that it may not be determined at the time the query is made).

For the cases that E_q happens, we define the following intermediate games.

$\underline{\mathsf{Game}_{1,q,1}^{\mathsf{S}}}$: Let $\mathsf{Game}_{1,0}^{\mathsf{S}}$ denote $\mathsf{Game}_1^{\mathsf{S}}$. In this game, we change the response to the q-th query of $\mathsf{Game}_{1,q-1}^{\mathsf{S}}$ (defined later in the cases that $q > 1$). The challenger uses $[\beta \boldsymbol{b}_2^* + r_j \boldsymbol{b}_4^*]_2$ instead of $[\beta \boldsymbol{b}_1^* + r_j \boldsymbol{b}_4^*]_2$. When it is a attribute key query or a vector key query, it is sufficient to consider the case that $\beta = 1$ since the game is not changed if $\beta = 0$.

Given any $\mathcal{A}$, we can construct a PPT algorithm $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q-1}^{\mathsf{S}}}(1^\lambda) = 0 \wedge E_q] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q,1}^{\mathsf{S}}}(1^\lambda) = 0 \wedge E_q]\right| \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{P1}}(\lambda).$$

Given D of $\mathsf{P1}$ with $n = 3$, $\mathcal{B}$ sets $(\boldsymbol{b}_3, \boldsymbol{b}_4, \boldsymbol{b}_3^*, \boldsymbol{b}_4^*) = (\boldsymbol{d}_4, \boldsymbol{d}_3, \boldsymbol{d}_4^*, \boldsymbol{d}_3^*)$ and $(\boldsymbol{b}_k, \boldsymbol{b}_k^*) = (\boldsymbol{d}_k, \boldsymbol{d}_k^*)$ for other k. $\mathcal{B}$ can simulate the challenge ciphertext by sampling $\boldsymbol{w}_{i,\{1,2,4\}} \xleftarrow{\$} \mathsf{span}\langle \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_4 \rangle$ and $w_{i,3}, w_{i,5}, w_{i,6} \xleftarrow{\$} \mathbb{Z}_p$. By setting $\boldsymbol{w}_i = \boldsymbol{w}_{i,\{1,2,4\}} + w_{i,3}\boldsymbol{b}_3 + w_{i,5}\boldsymbol{b}_5 + w_{i,6}\boldsymbol{b}_6$, $\mathcal{B}$ can also simulate the response to queries. Assume that $\mathcal{B}$ is given $[\boldsymbol{b}_t^* + r\boldsymbol{b}_4^*]_2$ where $t = 1$ or $t = 2$. $\mathcal{B}$ responds to the

Table 3: Games for the case q-th query is admissible

	Challenge Ciphertext						Queries		
	$\boldsymbol{b}_1$	$\boldsymbol{b}_2$	$\boldsymbol{b}_3$	$\boldsymbol{b}_4$	$\boldsymbol{b}_5$	$\boldsymbol{b}_6$	before q-th	q-th	after q-th
$\mathsf{Game}^{\mathsf{S}}_{1,q-1}$	0	0	1	0	0	1	$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_6$	$\beta\boldsymbol{b}^*_1 + r_j\boldsymbol{b}^*_4$	$\beta\boldsymbol{b}^*_1 + r_j\boldsymbol{b}^*_4$
$\mathsf{Game}^{\mathsf{S}}_{1,q,1}$	0	0	1	0	0	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_4$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,2}$	0	1	1	0	0	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_4$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,3}$	0	1	1	0	0	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_5$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,4}$	0	1	1	0	1	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_5$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,5}$	0	1	1	0	1	1		$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_5$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,6}$	0	1	1	0	1	1		$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_6$	
$\mathsf{Game}^{\mathsf{S}}_{1,q}$	0	0	1	0	0	1		$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_6$	

q-th query by using $[\boldsymbol{b}^*_t + r\boldsymbol{b}^*_4 + r_j\boldsymbol{b}^*_4]_2$ where $r_j \xleftarrow{\$} \mathbb{Z}_p$. This game simulates $\mathsf{Game}^{\mathsf{S}}_{1,q-1}$ if $t = 1$ and simulates $\mathsf{Game}^{\mathsf{S}}_{1,q,1}$ if $t = 2$.

$\underline{\mathsf{Game}^{\mathsf{S}}_{1,q,2}}$: In this game, we change the way that the challenge ciphertext is generated:

$$C^* = \left(\boldsymbol{b}_1 \middle| \boldsymbol{w}_{1,\{1,4,5\}} \middle| \cdots \middle| \boldsymbol{w}_{n,\{1,4,5\}}\right)\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0) + \left(\boldsymbol{b}_2 + \boldsymbol{b}_3 \middle| \boldsymbol{w}_{1,\{2,3,6\}} \middle| \cdots \middle| \boldsymbol{w}_{n,\{2,3,6\}}\right)\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1).$$

Namely, $(1, w_{1,2}, \ldots, w_{n,2})\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0)$ is switched to $(1, w_{1,2}, \ldots, w_{n,2})\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1)$. Except the challenge ciphertext, $w_{1,2}, \ldots, w_{n,2}$ appears only in the response to the q-th query.

Depending on whether this is a key query, an attribute key query, or a vector key query, $w_{1,2}, \ldots, w_{n,2}$ appears in the form of $(\rho_j, w_{1,2}, \ldots, w_{n,2})\mathbf{K}_j(\tilde{y}, \boldsymbol{y})$, $(0, w_{1,2}, \ldots, w_{n,2})\mathbf{K}_{\mathsf{Att},j}(\tilde{y})$, or $(\rho_j, w_{1,2}, \ldots, w_{n,2})\mathbf{K}_{\mathsf{Vec},j}(\boldsymbol{y})$. By Lemma 2, we have

$$\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,1}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] = \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,2}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q].$$

$\underline{\mathsf{Game}^{\mathsf{S}}_{1,q,3}}$: In this game, we change the response to the q-th query of $\mathsf{Game}^{\mathsf{S}}_{1,q,2}$. The challenger uses $[\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_5]_2$ instead of $[\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_4]_2$. Given any $\mathcal{A}$, we can construct a PPT algorithm $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,2}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] - \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,3}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q]\right| \leq \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda).$$

Given D of P2 with $n = 4$, $\mathcal{B}$ sets $(\boldsymbol{b}_4, \boldsymbol{b}_5, \boldsymbol{b}^*_4, \boldsymbol{b}^*_5) = (\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}^*_1, \boldsymbol{d}^*_2)$. $\mathcal{B}$ can simulate the challenge ciphertext by sampling $\boldsymbol{w}_{i,\{4,5\}} \xleftarrow{\$} \mathsf{span}\{\boldsymbol{b}_4, \boldsymbol{b}_5\}$ and $w_{i,1}, w_{i,2}, w_{i,3}, w_{i,6} \xleftarrow{\$} \mathbb{Z}_p$. By setting $\boldsymbol{w}_i = \boldsymbol{w}_{i,\{4,5\}} + w_{i,1}\boldsymbol{b}_1 + w_{i,2}\boldsymbol{b}_2 + w_{i,3}\boldsymbol{b}_3 + w_{i,6}\boldsymbol{b}_6$, $\mathcal{B}$ can also simulate the response to queries. Assume that $\mathcal{B}$ is given $[r\boldsymbol{b}^*_t]_2$ where $t = 4$ or $t = 5$. $\mathcal{B}$ responds to the q-th query by using $[\beta\boldsymbol{b}^*_2 + r_j r\boldsymbol{b}^*_t]_2$ where $r_j \xleftarrow{\$} \mathbb{Z}_p$. This game simulates $\mathsf{Game}^{\mathsf{S}}_{1,q,2}$ if $t = 4$ and simulates $\mathsf{Game}^{\mathsf{S}}_{1,q,3}$ if $t = 5$.

$\underline{\mathsf{Game}^{\mathsf{S}}_{1,q,4}}$: In this game, we change the way that the challenge ciphertext is generated:

$$C^* = \left(\boldsymbol{b}_1 \middle| \boldsymbol{w}_{1,\{1,4\}} \middle| \cdots \middle| \boldsymbol{w}_{n,\{1,4\}}\right)\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0) + \left(\boldsymbol{b}_2 + \boldsymbol{b}_3 \middle| \boldsymbol{w}_{1,\{2,3,5,6\}} \middle| \cdots \middle| \boldsymbol{w}_{n,\{2,3,5,6\}}\right)\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1).$$

Namely, $(0, w_{1,5}, \ldots, w_{n,5})\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0)$ is switched to $(0, w_{1,5}, \ldots, w_{n,5})\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1)$. Except the challenge ciphertext, $w_{1,5}, \ldots, w_{n,5}$ appears only in the response to the q-th query. In the same way as the game hopping from $\mathsf{Game}^{\mathsf{S}}_{1,q,1}$ to $\mathsf{Game}^{\mathsf{S}}_{1,q,2}$, by Lemma 2 we have

$$\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,3}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] = \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,4}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q].$$

Table 4: Games for the case q-th query is non-admissible

	Challenge Ciphertext						q-th query		
	$\boldsymbol{b}_1$	$\boldsymbol{b}_2$	$\boldsymbol{b}_3$	$\boldsymbol{b}_4$	$\boldsymbol{b}_5$	$\boldsymbol{b}_6$	$j' < j$	$j' = j$	$j' > j$
$\mathsf{Game}^{\mathsf{S}}_{1,q,j-1}$	0	0	1	0	0	1	$\beta\boldsymbol{b}^*_3 + r_{j'}\boldsymbol{b}^*_6$	$\beta\boldsymbol{b}^*_1 + r_j\boldsymbol{b}^*_4$	$\beta\boldsymbol{b}^*_1 + r_{j'}\boldsymbol{b}^*_4$
$\mathsf{Game}^{\mathsf{S}}_{1,q,j,1}$	0	0	1	0	0	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_4$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,j,2}$	0	1	1	0	0	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_4$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,j,3}$	0	1	1	0	0	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_5$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,j,4}$	0	1	1	0	1	1		$\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_5$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,j,5}$	0	1	1	0	1	1		$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_5$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,j,6}$	0	1	1	0	1	1		$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_6$	
$\mathsf{Game}^{\mathsf{S}}_{1,q,j}$	0	0	1	0	0	1		$\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_6$	

$\underline{\mathsf{Game}^{\mathsf{S}}_{1,q,5}}$: In this game, we change the response to the q-th query of $\mathsf{Game}^{\mathsf{S}}_{1,q,4}$. The challenger uses $[\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_5]_2$ instead of $[\beta\boldsymbol{b}^*_2 + r_j\boldsymbol{b}^*_5]_2$. In the same way as the game hopping from $\mathsf{Game}^{\mathsf{S}}_{1,q-1}$ to $\mathsf{Game}^{\mathsf{S}}_{1,q,1}$, given any $\mathcal{A}$, we can construct a PPT algorithm $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,4}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] - \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,5}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q]\right| \leq \mathsf{Adv}^{\mathsf{P1}}_{\mathcal{B}}(\lambda).$$

$\underline{\mathsf{Game}^{\mathsf{S}}_{1,q,6}}$: In this game, we change the response to the q-th query of $\mathsf{Game}^{\mathsf{S}}_{1,q,5}$. The challenger uses $[\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_6]_2$ instead of $[\beta\boldsymbol{b}^*_3 + r_j\boldsymbol{b}^*_5]_2$. In the same way as the game hopping from $\mathsf{Game}^{\mathsf{S}}_{1,q,2}$ to $\mathsf{Game}^{\mathsf{S}}_{1,q,3}$, given any $\mathcal{A}$, we can construct a PPT algorithm $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,5}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] - \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,6}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q]\right| \leq \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda).$$

$\underline{\mathsf{Game}^{\mathsf{S}}_{1,q}}$: In this game, we change the way that the challenge ciphertext is generated:

$$C^* = \left(\boldsymbol{b}_1 + \boldsymbol{b}_2\left|\boldsymbol{w}_{1,\{1,2,4,5\}}\right|\cdots\left|\boldsymbol{w}_{n,\{1,2,4,5\}}\right.\right)\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0) + \left(\boldsymbol{b}_3\left|\boldsymbol{w}_{1,\{3,6\}}\right|\cdots\left|\boldsymbol{w}_{n,\{3,6\}}\right.\right)\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1).$$

Namely, $(1, w_{1,2}, \ldots, w_{n,2})\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1)$ is switched back to $(1, w_{1,2}, \ldots, w_{n,2})\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0)$. In addition, $(0, w_{1,5}, \ldots, w_{n,5})\mathbf{C}(\tilde{x}_1, \boldsymbol{x}_1)$ is switched back to $(0, w_{1,5}, \ldots, w_{n,5})\mathbf{C}(\tilde{x}_0, \boldsymbol{x}_0)$. We note that $w_{1,2}, \ldots, w_{n,2}$, $w_{1,5}, \ldots, w_{n,5}$ are not used in the key generation steps. In the same way as the game hopping from $\mathsf{Game}^{\mathsf{S}}_{1,q,1}$ to $\mathsf{Game}^{\mathsf{S}}_{1,q,2}$, we have

$$\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,6}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] = \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q].$$

By Lemma 4 and Lemma 5, $\mathsf{Adv}^{\mathsf{P1}}_{\mathcal{B}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda)$ can be bounded above by DDH advantage. To sum up, for any PPT adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ such that:

$$\left|\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q-1}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q] - \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q}}_{\mathcal{A}}(1^\lambda) = 0 \wedge E_q]\right| \leq 8\mathsf{Adv}^{\mathsf{DDH},\mathbb{G}_2}_{\mathcal{B}}(\lambda).$$

In the case of $\neg E_q$, we repeat the same game hopping for each $j \in [d]$, using Lemma 3. Specifically, for each q-th query and each $j \in [d]$, we define $\mathsf{Game}^{\mathsf{S}}_{1,q,j,1}, \ldots, \mathsf{Game}^{\mathsf{S}}_{1,q,j,6}$ as shown

in Table 4. For each q, $\mathsf{Game}^{\mathsf{S}}_{1,q,0}$ coincides with $\mathsf{Game}^{\mathsf{S}}_{1,q-1}$, and $\mathsf{Game}^{\mathsf{S}}_{1,q,d}$ coincides with $\mathsf{Game}^{\mathsf{S}}_{1,q}$. In the same way as the case of E_q, for each $j \in [d]$, there exists a PPT algorithm $\mathcal{B}_j$ such that

$$\left|\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,j-1}}_{\mathcal{A}}(1^\lambda) = 0 \wedge \neg E_q] - \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q,j}}_{\mathcal{A}}(1^\lambda) = 0 \wedge \neg E_q]\right| \leq 8\mathsf{Adv}^{\mathsf{DDH},\mathbb{G}_2}_{\mathcal{B}_j}(\lambda).$$

Then there exists a PPT algorithm $\mathcal{B}$ such that

$$\left|\Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q-1}}_{\mathcal{A}}(1^\lambda) = 0 \wedge \neg E_q] - \Pr[\mathsf{Exp}^{\mathsf{Game}^{\mathsf{S}}_{1,q}}_{\mathcal{A}}(1^\lambda) = 0 \wedge \neg E_q]\right| \leq 8d \cdot \mathsf{Adv}^{\mathsf{DDH},\mathbb{G}_2}_{\mathcal{B}}(\lambda).$$

By adding the left-hand sides of these inequalities, we can conclude the desired result.

6 AD-SIM Secure Constructions for ABPs

In this section, we extend Lin and Luo's key-policy ABE [LL20] to AB-IPFE. We propose a scheme in the key-policy setting and another scheme in the ciphertext-policy setting. Let sIPFE denote a slotted IPFE scheme and $(\mathsf{Garble}, \mathsf{Eval})$ denote an AKGS with parameters m, n.

6.1 Construction in Key-Policy Setting

We propose an AD-SIM secure key-policy AB-IPFE scheme. KeyGen takes a bit flag together with an ABP f. $\mathsf{flag} = 1$ (resp. $\mathsf{flag} = 0$) indicates that decryption succeeds when $f(\tilde{\boldsymbol{x}}) \neq 0$ (resp. $f(\tilde{\boldsymbol{x}}) = 0$).

$\underline{\mathsf{Setup}(1^\lambda)}$:

1. Sets $\mathfrak{s}_{\mathsf{pub}} = \{\mathbf{vec}, \mathbf{pad}\}$, $\tilde{\mathfrak{s}}_{\mathsf{pub}} = \left\{\{\mathsf{const}^t\}_{t\in[k]}, \{\mathsf{coef}^t_i\}_{t\in[k],i\in[n]}\right\}$ where $|\mathbf{vec}| = d, |\mathbf{pad}| = k$. Sets $|\mathfrak{s}_{\mathsf{priv}}| = 2d + 2, |\tilde{\mathfrak{s}}_{\mathsf{priv}}| = n + 3$.
2. Runs $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$.
3. Runs $(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{msk}}) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \tilde{\mathfrak{s}}_{\mathsf{pub}}, \tilde{\mathfrak{s}}_{\mathsf{priv}})$.
4. Outputs $\mathsf{mpk} := (\mathsf{mpk}_0, \widetilde{\mathsf{mpk}})$ and $\mathsf{msk} := (\mathsf{msk}_0, \widetilde{\mathsf{msk}})$.

$\underline{\mathsf{Enc}(\mathsf{mpk}, \tilde{\boldsymbol{x}}, \boldsymbol{x})}$:

1. Samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$.
2. Sets the vectors $\boldsymbol{u}, \tilde{\boldsymbol{u}}$ as follows:

vector	$\mathbf{vec}$	$\mathbf{pad}$	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{u}$	$\boldsymbol{x}$	$\boldsymbol{s}$	$\mathbf{0}$

vector	const^t	coef^t_i	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{u}}$	$\boldsymbol{s}[t]$	$\boldsymbol{s}[t]\tilde{\boldsymbol{x}}[i]$	$\mathbf{0}$

3. Runs $\mathsf{ict} \leftarrow \mathsf{sIPFE.SlotEnc}(\mathsf{mpk}_0, [\boldsymbol{u}]_1)$, $\widetilde{\mathsf{ict}} \leftarrow \mathsf{sIPFE.SlotEnc}(\widetilde{\mathsf{mpk}}, [\tilde{\boldsymbol{u}}]_1)$.
4. Outputs $(\tilde{\boldsymbol{x}}, \mathsf{ict}, \widetilde{\mathsf{ict}})$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, f, \mathsf{flag} \in \{0,1\}, \boldsymbol{y})}$:

1. Samples $\boldsymbol{\mu}, \boldsymbol{\eta} \xleftarrow{\$} \mathbb{Z}_p^k$.
2. Sets $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^k$ as $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathsf{Embed}(\boldsymbol{\mu}, \boldsymbol{\eta}, \mathsf{flag}) := \begin{cases} (\boldsymbol{\mu}, \mathbf{0}) & \text{if } \mathsf{flag} = 1, \\ (\boldsymbol{\eta}, \boldsymbol{\mu}) & \text{if } \mathsf{flag} = 0. \end{cases}$
3. For each $t \in [k]$, runs $(\mathbf{L}^t_1, \ldots, \mathbf{L}^t_m) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}[t], \boldsymbol{\beta}[t]; \boldsymbol{r}_t)$ where $\boldsymbol{r}_t \xleftarrow{\$} \mathbb{Z}_p^{m-1}$.
4. Sets the vectors $\boldsymbol{v}, \tilde{\boldsymbol{v}}_r$ $(r \in [m])$ as follows:

vector	$\mathbf{vec}$	$\mathbf{pad}$	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{v}$	$\boldsymbol{y}$	$\boldsymbol{\mu}$	$\mathbf{0}$

vector	const^t	coef^t_i	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{v}}_r$	$\mathbf{L}^t_r[\mathsf{const}]$	$\mathbf{L}^t_r[\mathsf{coef}_i]$	$\mathbf{0}$

5. Runs $\mathsf{isk} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}]_2)$, $\widetilde{\mathsf{isk}}_r \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_r]_2)$.
6. Outputs $(f, \mathsf{flag}, \mathsf{isk}, \{\widetilde{\mathsf{isk}}_r\}_{r\in[m]})$.

$\underline{\mathsf{Dec}((f, \mathsf{flag}, \mathsf{isk}, \{\widetilde{\mathsf{isk}}_r\}_{r\in[m]}), (\tilde{\boldsymbol{x}}, \mathsf{ict}, \widetilde{\mathsf{ict}}))}$:

1. Computes $[\nu]_T = \mathsf{sIPFE.Dec}(\mathsf{isk}, \mathsf{ict})$.
2. Computes $[\ell_r]_T = \mathsf{sIPFE.Dec}(\widetilde{\mathsf{isk}}_r, \widetilde{\mathsf{ict}})$ for each $r \in [m]$.
3. Computes

$$[\mu']_T := \begin{cases} [f(\tilde{\boldsymbol{x}})^{-1}\mathsf{Eval}(f, \tilde{\boldsymbol{x}}, \ell_1, \dots, \ell_m)]_T & \text{if } \mathsf{flag} = 1, \\ [\mathsf{Eval}(f, \tilde{\boldsymbol{x}}, \ell_1, \dots, \ell_m)]_T & \text{if } \mathsf{flag} = 0. \end{cases}$$

4. Computes $\nu - \mu'$ from $[\nu - \mu']_T$ by brute-forcing the discrete logarithm.

Correctness

By correctness of sIPFE, we have $\nu = \langle \boldsymbol{x}, \boldsymbol{y}\rangle + \langle \boldsymbol{\mu}, \boldsymbol{s}\rangle$. By linearity of Garble, when we set

$$\bar{\alpha} := \langle \boldsymbol{\alpha}, \boldsymbol{s}\rangle, \quad \bar{\beta} := \langle \boldsymbol{\beta}, \boldsymbol{s}\rangle, \quad \bar{\boldsymbol{r}} := \sum_{t\in[k]} \boldsymbol{r}_t \boldsymbol{s}[t], \quad \bar{\mathbf{L}}_r := \sum_{t\in[k]} \mathbf{L}_r^t \boldsymbol{s}[t],$$

we have $(\bar{\mathbf{L}}_1, \dots, \bar{\mathbf{L}}_m) \leftarrow \mathsf{Garble}(f, \bar{\alpha}, \bar{\beta}; \bar{\boldsymbol{r}})$. Since $\ell_r = \bar{\mathbf{L}}_r(\tilde{\boldsymbol{x}})$ and the AKGS is correct, we have $\mu' = \langle \boldsymbol{\mu}, \boldsymbol{s}\rangle$ if "$\mathsf{flag} = 1$ and $f(\tilde{\boldsymbol{x}}) \neq 0$" or "$\mathsf{flag} = 0$ and $f(\tilde{\boldsymbol{x}}) = 0$". Thus, $\nu - \mu'$ is equal to $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$.

The hierarchical structures can be added similarly to the delegatable ABE [LL20]. Setup samples random vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k \xleftarrow{\$} \mathbb{Z}_p^k$. To achieve delegation, the sIPFE secret keys $\mathsf{bk}_{\mathsf{base},t'}$, $\widetilde{\mathsf{bk}}_{\mathsf{base},t'}$, and $\widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}$, associated with the following vectors $\boldsymbol{v}_{\mathsf{base},t'}$, $\tilde{\boldsymbol{v}}_{\mathsf{base},t'}$, and $\tilde{\boldsymbol{v}}_{\mathsf{base},t',i'}$ respectively for $t' \in [k], i' \in [n]$, are added to the master public key:

vector	**vec**	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{v}_{\mathsf{base},t'}$	$\mathbf{0}$	$\boldsymbol{b}_{t'}$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{v}}_{\mathsf{base},t'}$	$\boldsymbol{b}_{t'}[t]$	0	$\mathbf{0}$
$\tilde{\boldsymbol{v}}_{\mathsf{base},t',i'}$	0	$\boldsymbol{b}_{t'}[t] \cdot \mathbf{1}[i = i']$	$\mathbf{0}$

where $\mathbf{1}[i = i']$ denotes 1 if $i = i'$ and 0 otherwise.

The attribute-on-top hierarchy can be added by using the homomorphic property of sIPFE and linearity of the AKGS. $\mathsf{KeyGen}_{\mathsf{Att}}$ outputs the collection of $\left(f, \mathsf{flag}, \mathsf{isk}_j^{\mathsf{Att}}, \left\{\widetilde{\mathsf{isk}}_{r,j}^{\mathsf{Att}}\right\}_{r\in[m]}\right) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \mathsf{flag}, \boldsymbol{e}_j)$ for $j \in [d]$. $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}$ samples $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\eta}} \xleftarrow{\$} \mathbb{Z}_p^k$, sets $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) := \mathsf{Embed}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\eta}}, \mathsf{flag})$, and runs $(\hat{\mathbf{L}}_1^{t'}, \dots, \hat{\mathbf{L}}_m^{t'}) \leftarrow \mathsf{Garble}(f, \hat{\boldsymbol{\alpha}}[t'], \hat{\boldsymbol{\beta}}[t'])$. Then it computes

$$\mathsf{isk} \leftarrow \left(\sum_{j\in[d]} \boldsymbol{y}[j] \boxdot \mathsf{isk}_j^{\mathsf{Att}}\right) \boxplus \left(\sum_{t'\in[k]} \hat{\boldsymbol{\mu}}[t'] \boxdot \mathsf{bk}_{\mathsf{base},t'}\right),$$

$$\widetilde{\mathsf{isk}}_r \leftarrow \left(\sum_{j\in[d]} \boldsymbol{y}[j] \boxdot \widetilde{\mathsf{isk}}_{r,j}^{\mathsf{Att}}\right) \boxplus \left(\sum_{t'\in[k]} \hat{\mathbf{L}}_r^{t'}[\mathsf{const}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \hat{\mathbf{L}}_r^{t'}[\mathsf{coef}_{i'}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right)$$

where $\sum$ denotes the sum calculated using $\boxplus$.

The vector key is $\mathsf{isk}^{\mathsf{Vec}} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}]_2)$ where $\boldsymbol{v}[\mathbf{vec}] = \boldsymbol{y}, \boldsymbol{v}[\mathbf{pad}] = \mathbf{0}, \boldsymbol{v}[\mathfrak{s}_{\mathsf{priv}}] = \mathbf{0}$. $\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}$ computes

$$\mathsf{isk} \leftarrow \mathsf{isk}^{\mathsf{Vec}} \boxplus \left(\sum_{t'\in[k]} \boldsymbol{\mu}[t'] \boxdot \mathsf{bk}_{\mathsf{base},t'}\right),$$

$$\widetilde{\mathsf{isk}}_r \leftarrow \left(\sum_{t'\in[k]} \mathbf{L}_r^{t'}[\mathsf{const}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \mathbf{L}_r^{t'}[\mathsf{coef}_{i'}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right)$$

where $\boldsymbol{\mu}$ and $\{\mathbf{L}_r^{t'}\}_{r\in[m],t'\in[k]}$ are generated in the same way as in KeyGen.

We give the security proof in the next section.

Theorem 4. Assume that sIPFE is function-hiding, the AKGS is piecewise secure, and the MDDH_k assumption holds. Then the (hierarchical) key-policy AB-IPFE scheme is AD-SIM secure.

6.2 Proof of Theorem 4

For simplicity, we firstly ignore the hierarchical structure. Let $\mathfrak{s}_{\mathsf{priv}} = \{\mathbf{sim}_1, \mathbf{sim}_2, \mathsf{sim}, \mathsf{ext}\}$ where $|\mathbf{sim}_1| = |\mathbf{sim}_2| = d$ and $\tilde{\mathfrak{s}}_{\mathsf{priv}} = \{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \widetilde{\mathbf{sim}}\}$ where $|\widetilde{\mathbf{sim}}| = 2$. The simulator is defined as follows:

- $\mathsf{Setup}^*(1^\lambda)$: This is the same as Setup.
- $\mathsf{KeyGen}_0^*(\mathsf{msk}, f_q, \mathsf{flag}_q \in \{0,1\}, \boldsymbol{y}_q)$:
  1. Samples $\boldsymbol{\mu}_q, \boldsymbol{\eta}_q \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\boldsymbol{\alpha}_q, \boldsymbol{\beta}_q, (\mathbf{L}_{1,q}^t, \dots, \mathbf{L}_{m,q}^t)$ as in KeyGen.
  2. Samples $\bar{\mu}_q, \bar{\eta}_q \xleftarrow{\$} \mathbb{Z}_p$ and sets $(\bar{\alpha}_q, \bar{\beta}_q) := \mathsf{Embed}(\bar{\mu}_q, \bar{\eta}_q, \mathsf{flag}_q)$, $(\bar{\mathbf{L}}_{1,q}, \dots, \bar{\mathbf{L}}_{m,q}) \leftarrow \mathsf{Garble}(f_q, \bar{\alpha}_q, \bar{\beta}_q)$.
  3. Sets the vector $\boldsymbol{v}_q$ as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\boldsymbol{y}_q$	$\mathbf{0}$	0	$\bar{\mu}_q$

  4. Sets the vectors $\{\tilde{\boldsymbol{v}}_{r,q}\}_{r\in[m]}$ as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathbf{sim}}$
$\tilde{\boldsymbol{v}}_{r,q}$	$\mathbf{L}_{r,q}^t[\mathsf{const}]$	$\mathbf{L}_{r,q}^t[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{coef}_i]$	$\mathbf{0}$

  5. Runs $\mathsf{isk}_q \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}_q]_2)$, $\mathsf{isk}_{r,q} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{r,q}]_2)$.
  6. Outputs $(f_q, \mathsf{flag}_q, \mathsf{isk}_q, \{\mathsf{isk}_{r,q}\}_{r\in[m]})$.
- $\mathsf{Enc}^*(\mathsf{msk}, \tilde{\boldsymbol{x}}, \mathcal{V})$:
  1. Samples a vector $\boldsymbol{d}$ such that $\langle \boldsymbol{x}^*, \boldsymbol{y}_q\rangle = \langle \boldsymbol{d}, \boldsymbol{y}_q\rangle$ for any admissible q.
  2. Sets the vector $\tilde{\boldsymbol{u}}$ as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathbf{sim}}$
$\tilde{\boldsymbol{u}}$	0	0	1	$\tilde{\boldsymbol{x}}[i]$	$\mathbf{0}$

  3. Sets the vector $\boldsymbol{u}$ as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\mathbf{0}$	0	1

  4. Runs $\mathsf{ict} \leftarrow \mathsf{sIPFE.Enc}(\mathsf{msk}_0, [\boldsymbol{u}]_1)$, $\widetilde{\mathsf{ict}} \leftarrow \mathsf{sIPFE.Enc}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{u}}]_1)$.
  5. Outputs $(\tilde{\boldsymbol{x}}, \mathsf{ict}, \widetilde{\mathsf{ict}})$.
- $\mathsf{KeyGen}_1^*(\mathsf{msk}, f_q, \mathsf{flag}_q \in \{0,1\}, \boldsymbol{y}_q, z_q \in \{\langle \boldsymbol{x}^*, \boldsymbol{y}_q\rangle, 0\})$:
  1. Samples $\boldsymbol{\mu}_q, \boldsymbol{\eta}_q \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\boldsymbol{\alpha}_q, \boldsymbol{\beta}_q, (\mathbf{L}_{1,q}^t, \dots, \mathbf{L}_{m,q}^t)$ as in KeyGen.
  2. Samples $\bar{\mu}_q, \bar{\eta}_q \xleftarrow{\$} \mathbb{Z}_p$ and sets $(\bar{\alpha}_q, \bar{\beta}_q) := \mathsf{Embed}(\bar{\mu}_q, \bar{\eta}_q, \mathsf{flag}_q)$, $(\bar{\mathbf{L}}_{1,q}, \dots, \bar{\mathbf{L}}_{m,q}) \leftarrow \mathsf{Garble}(f_q, \bar{\alpha}_q, \bar{\beta}_q)$.
  3. Sets the vector $\boldsymbol{v}_q$ as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\mathbf{0}$	$\mathbf{0}$	0	$z_q + \bar{\mu}_q$

4. Sets the vectors $\{\tilde{\boldsymbol{v}}_{r,q}\}_{r\in[m]}$ as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathbf{sim}}$
$\tilde{\boldsymbol{v}}_{r,q}$	$\mathbf{L}_{r,q}^t[\mathsf{const}]$	$\mathbf{L}_{r,q}^t[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{coef}_i]$	$\mathbf{0}$

5. Runs $\mathsf{isk}_q \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}_q]_2)$, $\mathsf{isk}_{r,q} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{r,q}]_2)$.
6. Outputs $(f_q, \mathsf{flag}_q, \mathsf{isk}_q, \{\mathsf{isk}_{r,q}\}_{r\in[m]})$.

We define the intermediate security games as follows. Let Q_{pre} denote an upper bound on the number of pre-challenge queries.

$\underline{\mathsf{Game}_0}$: This is the real AD-SIM security game.

$\underline{\mathsf{Game}_1}$: In this game, Setup samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and appends it to msk. We set

$$\bar{\mu}_q := \langle \boldsymbol{\mu}_q, \boldsymbol{s}\rangle, \quad \bar{\eta}_q := \langle \boldsymbol{\eta}_q, \boldsymbol{s}\rangle, \quad (\bar{\alpha}_q, \bar{\beta}_q) := \mathsf{Embed}(\bar{\mu}_q, \bar{\eta}_q, \mathsf{flag}_q),$$
$$\bar{\boldsymbol{r}}_q := \sum_{t\in[k]} \boldsymbol{r}_{t,q}\boldsymbol{s}[t], \quad (\bar{\mathbf{L}}_{1,q}, \ldots, \bar{\mathbf{L}}_{m,q}) \leftarrow \mathsf{Garble}(f_q, \bar{\alpha}_q, \bar{\beta}_q; \bar{\boldsymbol{r}}_q).$$

Then we change the way that $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ for any query are generated as

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	0	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\mathbf{0}$	$\boldsymbol{y}_q$	0	$\bar{\mu}_q$

where d is sampled in the same way as in Enc^*. Also, $\tilde{\boldsymbol{u}}$ of the challenge ciphertext and $\tilde{\boldsymbol{v}}_{r,q}$ for any query are modified as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathbf{sim}}$
$\tilde{\boldsymbol{u}}$	0	0	1	$\tilde{\boldsymbol{x}}[i]$	$\mathbf{0}$

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathbf{sim}}$
$\tilde{\boldsymbol{v}}_{r,q}$	$\mathbf{L}_{r,q}^t[\mathsf{const}]$	$\mathbf{L}_{r,q}^t[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{coef}_i]$	$\mathbf{0}$

By the function-hiding property of sIPFE, Game_0 and Game_1 are indistinguishable.

$\underline{\mathsf{Game}_2}$: In this game, for any query, we sample $\bar{\mu}_q, \bar{\eta}_q \xleftarrow{\$} \mathbb{Z}_p$, $\bar{\boldsymbol{r}}_q \xleftarrow{\$} \mathbb{Z}_p^{m-1}$. $(\bar{\alpha}_q, \bar{\beta}_q)$ and $(\bar{\mathbf{L}}_{1,q}, \ldots, \bar{\mathbf{L}}_{m,q})$ are computed in the same way as in Game_1 by using randomly sampled $\bar{\mu}_q, \bar{\eta}_q, \bar{\boldsymbol{r}}_q$. By the MDDH_k assumption, Game_1 and Game_2 are indistinguishable. In fact, these games can be efficiently simulated in the same way from $[\boldsymbol{\mu}_q]_2$, $[\boldsymbol{\eta}_q]_2$, $\{[\boldsymbol{r}_{t,q}]_2\}_{t\in[k]}$, $[\bar{\mu}_q]_2$, $[\bar{\eta}_q]_2$, and $[\bar{\boldsymbol{r}}_q]_2$.

$\underline{\mathsf{Game}_{2,q,1}}$ ($q \in [Q_{\mathsf{pre}}]$): Let $\mathsf{Game}_{2,0,3}$ denote Game_2. In this game, from $\mathsf{Game}_{2,q-1,3}$ (defined later in the cases that $q > 1$), we change the way that $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ for the q-th pre-challenge query is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	$\langle \boldsymbol{x}^*, \boldsymbol{y}_q\rangle$	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_{q'}$ $(q' < q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{\mu}_{q'}$	$\boldsymbol{y}_{q'}$	$\mathbf{0}$	$\mathbf{0}$	$\bar{\mu}_{q'}$
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\mathbf{0}$	$\mathbf{0}$	1	$\bar{\mu}_q$
$\boldsymbol{v}_{q'}$ $(q' > q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{\mu}_{q'}$	$\mathbf{0}$	$\boldsymbol{y}_{q'}$	$\mathbf{0}$	$\bar{\mu}_{q'}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{2,q-1,3}$ and $\mathsf{Game}_{2,q,1}$ are indistinguishable.

$\underline{\mathsf{Game}_{2,q,2}}$ ($q \in [Q_{\mathsf{pre}}]$): In this game, from $\mathsf{Game}_{2,q,1}$, we change the way that $\boldsymbol{u}$ of the challenge ciphertext is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	$\langle \boldsymbol{d}, \boldsymbol{y}_q \rangle$	1

Let E_q be an event that the q-th query is admissible. In the case of E_q, we have $\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle = \langle \boldsymbol{d}, \boldsymbol{y}_q \rangle$. Thus, it follows that

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,1}}(1^\lambda) = 0 \wedge E_q] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,2}}(1^\lambda) = 0 \wedge E_q].$$

In the case of $\neg E_q$, by AD-IND security of the underlying 1-ABE [LL20], $\boldsymbol{v}_q[\mathsf{ext}]$ in $\mathsf{Game}_{2,q,1}$ can be set as an independent fresh value $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. $\widetilde{\mathsf{sim}}$ is used in this part.

By the function-hiding property of sIPFE, $\boldsymbol{u}$ and $\boldsymbol{v}_q$ can be changed in the following way:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	$\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle + \mu'_q$	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\mathbf{0}$	$\mathbf{0}$	1	0

Since $\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle + \mu'_q$ is uniformly random and independent of the other parameters, it can be replaced with $\langle \boldsymbol{d}, \boldsymbol{y}_q \rangle + \mu'_q$. By retracing the previous argument in reverse, it follows that

$$\left| \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,1}}(1^\lambda) = 0 \wedge \neg E_q] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,2}}(1^\lambda) = 0 \wedge \neg E_q] \right|$$

is also negligible. Thus, $\mathsf{Game}_{2,q,1}$ and $\mathsf{Game}_{2,q,2}$ are indistinguishable.

$\underline{\mathsf{Game}_{2,q,3}}$ ($q \in [Q_{\mathsf{pre}}]$): In this game, from $\mathsf{Game}_{2,q,2}$, we change the way that $\boldsymbol{v}_q$ for the q-th pre-challenge query is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_{q'}$ $(q' < q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{\mu}_{q'}$	$\boldsymbol{y}_{q'}$	$\mathbf{0}$	0	$\bar{\mu}_{q'}$
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\boldsymbol{y}_q$	$\mathbf{0}$	0	$\bar{\mu}_q$
$\boldsymbol{v}_{q'}$ $(q' > q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{\mu}_{q'}$	$\mathbf{0}$	$\boldsymbol{y}_{q'}$	0	$\bar{\mu}_{q'}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{2,q,2}$ and $\mathsf{Game}_{2,q,3}$ are indistinguishable.

$\underline{\mathsf{Game}_3}$: In this game, from $\mathsf{Game}_{2,Q_{\mathsf{pre}},3}$, we change the way that $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ for each post-challenge query are generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\mathbf{0}$	0	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\mathbf{0}$	$\mathbf{0}$	0	$\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle + \bar{\mu}_q$

By the function-hiding property of sIPFE, $\mathsf{Game}_{2,Q_{\mathsf{pre}},3}$ and Game_3 are indistinguishable.

$\underline{\mathsf{Game}_4}$: This is the ideal AD-SIM security game. In this game, we change the way that $\boldsymbol{v}_q$ for each post-challenge query is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{\mu}_q$	$\mathbf{0}$	$\mathbf{0}$	0	$z_q + \bar{\mu}_q$

$\boldsymbol{v}_q$ is not changed when the q-th query is admissible. When it is not admissible, by AD-IND security of the underlying 1-ABE [LL20], $\bar{\mu}_q$ can be replaced with a fresh value $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. Thus, in both Game_3 and Game_4, $\boldsymbol{v}_q[\mathsf{ext}]$ is uniformly random and independent of the other parameters. This completes the security proof in the non-hierarchical case.

Then we show that hierarchical structure does not compromise security. Adding attribute-on-top hierarchy does not affect security since it utilizes the homomorphic property of the proposed scheme. We consider vector-on-top hierarchy. The oracle corresponding to $\mathsf{KeyGen}_{\mathsf{Vec}}$ can be simulated similarly to KeyGen_0^* and KeyGen_1^*. Concretely, we set

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\mathbf{0}$	$\boldsymbol{y}_q$	$\mathbf{0}$	0	0

for pre-challenge queries and

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	0	$\mathbf{0}$	$\mathbf{0}$	0	z_q

for post-challenge queries. Also, we can easily see that the additional components in the master public key do not affect the security proof. Setup^* additionally samples $\bar{b}_{t'} \xleftarrow{\$} \mathbb{Z}_p$ for each $t' \in [k]$. The simulated $\boldsymbol{v}_{\mathsf{base},t'}, \tilde{\boldsymbol{v}}_{\mathsf{base},t'}, \tilde{\boldsymbol{v}}_{\mathsf{base},t',i'}$ $(t' \in [k], i' \in [n])$ are represented as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_{\mathsf{base},t'}$	$\mathbf{0}$	$\boldsymbol{b}_{t'}$	$\mathbf{0}$	$\mathbf{0}$	0	$\bar{b}_{t'}$

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathbf{sim}}$
$\tilde{\boldsymbol{v}}_{\mathsf{base},t'}$	$\boldsymbol{b}_{t'}[t]$	0	$\bar{b}_{t'}$	0	$\mathbf{0}$
$\tilde{\boldsymbol{v}}_{\mathsf{base},t',i'}$	0	$\boldsymbol{b}_{t'}[t] \cdot \mathbf{1}[i = i']$	0	$\bar{b}_{t'} \cdot \mathbf{1}[i = i']$	$\mathbf{0}$

In Game_1, we can change the vectors to the above form where $\bar{b}_{t'} = \langle \boldsymbol{b}_{t'}, \boldsymbol{s} \rangle$. In Game_2, we can change $\bar{b}_{t'}$ to a random value by the MDDH_k assumption. They do not affect the rest of the security proof since each $\bar{b}_{t'}$ is independent of $\boldsymbol{s}$.

6.3 Construction in Ciphertext-Policy Setting

We propose an AD-SIM secure ciphertext-policy AB-IPFE scheme. Compared with the key-policy scheme, this scheme requires a larger $|\tilde{\mathfrak{s}}_{\mathsf{priv}}|$.

$\underline{\mathsf{Setup}(1^\lambda)}$:

1. Sets $\mathfrak{s}_{\mathsf{pub}} = \{\mathbf{vec}, \mathbf{pad}\}$, $\tilde{\mathfrak{s}}_{\mathsf{pub}} = \left\{\{\mathsf{const}^t\}_{t\in[k]}, \{\mathsf{coef}_i^t\}_{t\in[k],i\in[n]}\right\}$ where $|\mathbf{vec}| = d, |\mathbf{pad}| = k$. Sets $|\mathfrak{s}_{\mathsf{priv}}| = 2d + 2, |\tilde{\mathfrak{s}}_{\mathsf{priv}}| = 2(k+1)(n+1)$.
2. Runs $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$.
3. Runs $(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{msk}}) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \tilde{\mathfrak{s}}_{\mathsf{pub}}, \tilde{\mathfrak{s}}_{\mathsf{priv}})$.
4. Outputs $\mathsf{mpk} := (\mathsf{mpk}_0, \widetilde{\mathsf{mpk}})$ and $\mathsf{msk} := (\mathsf{msk}_0, \widetilde{\mathsf{msk}})$.

$\underline{\mathsf{Enc}(\mathsf{mpk}, f, \mathsf{flag} \in \{0,1\}, \boldsymbol{x})}$:

1. Samples $\boldsymbol{\mu}, \boldsymbol{\eta} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $(\boldsymbol{\alpha}, \boldsymbol{\beta}) := \mathsf{Embed}(\boldsymbol{\mu}, \boldsymbol{\eta}, \mathsf{flag})$.
2. For each $t \in [k]$, runs $(\mathbf{L}_1^t, \ldots, \mathbf{L}_m^t) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}[t], \boldsymbol{\beta}[t]; \boldsymbol{r}_t)$ where $\boldsymbol{r}_t \xleftarrow{\$} \mathbb{Z}_p^{m-1}$.
3. Sets the vectors $\boldsymbol{u}, \{\tilde{\boldsymbol{u}}_r\}_{r\in[m]}$ as follows:

vector	**vec**	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{u}$	$\boldsymbol{x}$	$\boldsymbol{\mu}$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	$\mathbf{0}$

4. Runs $\mathsf{ict} \leftarrow \mathsf{sIPFE.SlotEnc}(\mathsf{mpk}_0, [\boldsymbol{u}]_1)$, $\widetilde{\mathsf{ict}}_r \leftarrow \mathsf{sIPFE.SlotEnc}(\widetilde{\mathsf{mpk}}, [\tilde{\boldsymbol{u}}_r]_1)$.
5. Outputs $(f, \mathsf{flag}, \mathsf{ict}, \{\widetilde{\mathsf{ict}}_r\}_{r\in[m]})$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \tilde{\boldsymbol{y}}, \boldsymbol{y})}$:

1. Samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$.
2. Sets the vectors $\boldsymbol{v}, \tilde{\boldsymbol{v}}$ as follows.

vector	**vec**	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{v}$	$\boldsymbol{y}$	$\boldsymbol{s}$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{v}}$	$\boldsymbol{s}[t]$	$\boldsymbol{s}[t]\tilde{\boldsymbol{y}}[i]$	$\mathbf{0}$

3. Runs $\mathsf{isk} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}]_2)$, $\widetilde{\mathsf{isk}} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}]_2)$.
4. Outputs $(\tilde{\boldsymbol{y}}, \mathsf{isk}, \widetilde{\mathsf{isk}})$.

$\underline{\mathsf{Dec}((\tilde{\boldsymbol{y}}, \mathsf{isk}, \widetilde{\mathsf{isk}}), (f, \mathsf{flag}, \mathsf{ict}, \{\widetilde{\mathsf{ict}}_r\}_{r\in[m]}))}$:

1. Computes $[\nu]_T := \mathsf{sIPFE.Dec}(\mathsf{isk}, \mathsf{ict})$.
2. Computes $[\ell_r]_T := \mathsf{sIPFE.Dec}(\widetilde{\mathsf{isk}}, \widetilde{\mathsf{ict}}_r)$ for each $r \in [m]$.
3. Computes

$$[\mu']_T := \begin{cases} [f(\tilde{\boldsymbol{y}})^{-1}\mathsf{Eval}(f, \tilde{\boldsymbol{y}}, \ell_1, \ldots, \ell_m)]_T & \text{if } \mathsf{flag} = 1, \\ [\mathsf{Eval}(f, \tilde{\boldsymbol{y}}, \ell_1, \ldots, \ell_m)]_T & \text{if } \mathsf{flag} = 0. \end{cases}$$

4. Computes $\nu - \mu'$ from $[\nu - \mu']_T$ by brute-forcing the discrete logarithm.

We can see that correctness holds and the hierarchical structures can be added as in the key-policy construction. To introduce the hierarchical structures, $\{\mathsf{bk}_{\mathsf{base},t'}\}_{t'\in[k]}$, $\{\widetilde{\mathsf{bk}}_{\mathsf{base},t'}\}_{t'\in[k]}$, and $\{\widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\}_{t'\in[k],i'\in[n]}$ are added to the master public key. $\mathsf{KeyGen}_{\mathsf{Att}}$ outputs the collection of $(\tilde{\boldsymbol{y}}, \mathsf{isk}_j^{\mathsf{Att}}, \widetilde{\mathsf{isk}}_j^{\mathsf{Att}}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \tilde{\boldsymbol{y}}, \boldsymbol{e}_j)$ for $j \in [d]$. $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}$ samples $\hat{\boldsymbol{s}} \xleftarrow{\$} \mathbb{Z}_p^k$ and computes

$$\mathsf{isk} \leftarrow \left(\sum_{j\in[d]} \boldsymbol{y}[j] \boxdot \mathsf{isk}_j^{\mathsf{Att}}\right) \boxplus \left(\sum_{t'\in[k]} \hat{\boldsymbol{s}}[t'] \boxdot \mathsf{bk}_{\mathsf{base},t'}\right),$$

$$\widetilde{\mathsf{isk}} \leftarrow \left(\sum_{j\in[d]} \boldsymbol{y}[j] \boxdot \mathsf{isk}_j^{\mathsf{Att}}\right) \boxplus \left(\sum_{t'\in[k]} \hat{\boldsymbol{s}}[t'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \hat{\boldsymbol{s}}[t']\tilde{\boldsymbol{y}}[i'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right).$$

The vector key is $\mathsf{isk}^{\mathsf{Vec}} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}]_2)$ where $\boldsymbol{v}[\mathbf{vec}] = \boldsymbol{y}, \boldsymbol{v}[\mathbf{pad}] = \mathbf{0}, \boldsymbol{v}[\mathfrak{s}_{\mathsf{priv}}] = \mathbf{0}$. $\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}$ samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and computes

$$\mathsf{isk} \leftarrow \mathsf{isk}^{\mathsf{Vec}} \boxplus \left(\sum_{t'\in[k]} \boldsymbol{s}[t'] \boxdot \mathsf{bk}_{\mathsf{base},t'}\right),$$

$$\widetilde{\mathsf{isk}} \leftarrow \left(\sum_{t'\in[k]} \boldsymbol{s}[t'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \boldsymbol{s}[t']\tilde{\boldsymbol{y}}[i'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right).$$

We give the security proof in the next section.

Theorem 5. Assume that sIPFE is function-hiding, the AKGS is piecewise secure, and the MDDH_k assumption holds. Then the ciphertext-policy AB-IPFE scheme is AD-SIM secure.

Remark 2. A concurrent and independent work [DST26] proposed an attribute-based slotted IPFE scheme in the ciphertext-policy setting. Their construction is based on slotted IPFE and AKGS, and is similar to our scheme, but there are several differences.

1. Their scheme is formulated in the slotted setting, and is therefore more general than ours. On the other hand, they only prove IND security (that is standard for the slotted setting), and do not prove the SIM security that we establish for our scheme.
2. In their scheme, the slots are aligned when generating the two sIPFE keys, which results in a ciphertext size of $O(m(d+n))$. In our scheme, these components are separated, thereby achieving a ciphertext size of $O(d+mn)$.
3. In their security proof, they apply the MDDH_k assumption in $\mathbb{G}_1$ with $\boldsymbol{s}$ as the secret vector. However, $[\boldsymbol{s}]_2$ is used in the public slot of the secret key, and this part cannot be modified using the function-hiding property of sIPFE. Therefore, in our view, it is difficult to apply the MDDH_k assumption in $\mathbb{G}_1$ or to use $\boldsymbol{s}$ as the secret vector. In our proof, by slightly enlarging the private slot, we temporarily move each component to $\mathbb{G}_2$ and apply the MDDH_k assumption over $\mathbb{G}_2$ using a vector other than $\boldsymbol{s}$ as the secret vector.

6.4 Proof of Theorem 5

For simplicity, we present a proof in the non-hierarchical setting. Hierarchical structure can be considered in the same way as in Section 6.2. Let $\mathfrak{s}_{\mathsf{priv}} = \{\mathbf{sim}_1, \mathbf{sim}_2, \mathsf{sim}, \mathsf{ext}\}$ where $|\mathbf{sim}_1| = |\mathbf{sim}_2| = d$. Let $\tilde{\mathfrak{s}}_{\mathsf{priv}}$ consist of $\mathsf{simconst}^t$, $\mathsf{simcoef}_i^t$, $\mathsf{extconst}$, $\mathsf{extcoef}_i$, $\mathsf{tmpconst}^t$, $\mathsf{tmpcoef}_i^t$, $\widetilde{\mathsf{const}}$, and $\widetilde{\mathsf{coef}}_i$ where $i \in [n], t \in [k]$. The simulator is defined as follows:

- $\mathsf{Setup}^*(1^\lambda)$: This is the same as Setup except that it samples $\boldsymbol{\mu} \xleftarrow{\$} \mathbb{Z}_p^k$ and appends it to msk.
- $\mathsf{KeyGen}_0^*(\mathsf{msk}, \tilde{\boldsymbol{y}}_q, \boldsymbol{y}_q)$:
  1. Samples $\boldsymbol{s}_q \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\bar{\mu}_q := \langle \boldsymbol{s}_q, \boldsymbol{\mu} \rangle$.
  2. Sets the vector $\tilde{\boldsymbol{v}}_q$ as in KeyGen. Sets the vector $\boldsymbol{v}_q$ as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\boldsymbol{y}_q$	$\mathbf{0}$	0	$\bar{\mu}_q$

  3. Runs $\mathsf{isk}_q \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}_q]_2)$, $\widetilde{\mathsf{isk}}_q \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_q]_2)$.
  4. Outputs $(\tilde{\boldsymbol{y}}_q, \mathsf{isk}_q, \widetilde{\mathsf{isk}}_q)$.
- $\mathsf{Enc}^*(\mathsf{msk}, f, \mathsf{flag}, \mathcal{V})$:
  1. Samples a vector $\boldsymbol{d}$ such that $\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle = \langle \boldsymbol{d}, \boldsymbol{y}_q \rangle$ for any admissible q.
  2. Sets the vectors $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_m$ as in Enc. Sets the vector $\boldsymbol{u}$ as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\mathbf{0}$	0	1

  3. Runs $\mathsf{ict} \leftarrow \mathsf{sIPFE.Enc}(\mathsf{msk}_0, [\boldsymbol{u}]_1)$, $\widetilde{\mathsf{ict}}_r \leftarrow \mathsf{sIPFE.Enc}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{u}}_r]_1)$.
  4. Outputs $(f, \mathsf{flag}, \mathsf{ict}, \{\widetilde{\mathsf{ict}}_r\}_{r \in [m]})$.
- $\mathsf{KeyGen}_1^*(\mathsf{msk}, \tilde{\boldsymbol{y}}_q, \boldsymbol{y}_q, z_q \in \{\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle, 0\})$:
  1. Samples $\boldsymbol{s}_q \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\bar{\mu}_q := \langle \boldsymbol{s}_q, \boldsymbol{\mu} \rangle$.
  2. Sets the vector $\tilde{\boldsymbol{v}}_q$ as in KeyGen. Sets the vector $\boldsymbol{v}_q$ as follows:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\mathbf{0}$	$\mathbf{0}$	0	$z_q + \bar{\mu}_q$

  3. Runs $\mathsf{isk}_q \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_0, [\boldsymbol{v}_q]_2)$, $\widetilde{\mathsf{isk}}_q \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_q]_2)$.

4. Outputs $(\tilde{\boldsymbol{y}}_q, \mathsf{isk}_q, \widetilde{\mathsf{isk}}_q)$.

We define the intermediate security games as follows. Let Q_{pre} denote an upper bound on the number of pre-challenge queries.

$\underline{\mathsf{Game}_0}$: This is the real AD-SIM security game.

$\underline{\mathsf{Game}_1}$: In this game, Setup samples $\boldsymbol{\mu}, \boldsymbol{\eta} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{r}_t \xleftarrow{\$} \mathbb{Z}_p^{m-1}$ $(t \in [k])$ and appends them to msk. Also, we set $\bar{\mu}_q := \langle \boldsymbol{s}_q, \boldsymbol{\mu} \rangle$ and change the way that $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ for any query are generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	0	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\mathbf{0}$	$\boldsymbol{y}_q$	0	$\bar{\mu}_q$

In addition, $\tilde{\boldsymbol{u}}_r$ of the challenge ciphertext and $\tilde{\boldsymbol{v}}_q$ of any query are modified as follows:

vector	const^t	coef_i^t	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	0	0	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	0	0	0	0	0	0

vector	const^t	coef_i^t	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	0	0	0	0	0	0

By the function-hiding property of sIPFE, Game_0 and Game_1 are indistinguishable.

$\underline{\mathsf{Game}_{1,q,1}}$ $(q \in [Q_{\mathsf{pre}}])$: Let $\mathsf{Game}_{1,0,3}$ denote Game_1. In this game, from $\mathsf{Game}_{1,q-1,3}$ (defined later in the cases that $q > 1$), we change the way that $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ for the q-th pre-challenge query is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	$\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle$	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_{q'}$ $(q' < q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{s}_{q'}$	$\boldsymbol{y}_{q'}$	$\mathbf{0}$	0	$\bar{\mu}_{q'}$
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\mathbf{0}$	$\mathbf{0}$	1	$\bar{\mu}_q$
$\boldsymbol{v}_{q'}$ $(q' > q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{s}_{q'}$	$\mathbf{0}$	$\boldsymbol{y}_{q'}$	0	$\bar{\mu}_{q'}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{1,q-1,3}$ and $\mathsf{Game}_{1,q,1}$ are indistinguishable.

$\underline{\mathsf{Game}_{1,q,2}}$ $(q \in [Q_{\mathsf{pre}}])$: In this game, from $\mathsf{Game}_{1,q,1}$, we change the way that $\boldsymbol{u}$ of the challenge ciphertext is generated as

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	$\langle \boldsymbol{d}, \boldsymbol{y}_q \rangle$	1

where d is sampled in the same way as in Enc^*.

Let E_q be an event that the q-th query is admissible. In the case of E_q, we have $\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle = \langle \boldsymbol{d}, \boldsymbol{y}_q \rangle$. Thus, it follows that

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q,1}}(1^\lambda) = 0 \wedge E_q] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q,2}}(1^\lambda) = 0 \wedge E_q].$$

We consider the case of $\neg E_q$. Unlike the key-policy setting, the MDDH_k assumption on $\mathbb{G}_1$ cannot be directly applied since $[\boldsymbol{s}_q]_2$ appears in the q-th query. Instead, we achieve the replacement by temporarily moving each value to the $\mathbb{G}_2$ side and applying the MDDH_k assumption on $\mathbb{G}_2$. By the function-hiding property of sIPFE, we can change the private slots of $\tilde{\boldsymbol{u}}_r$ and $\tilde{\boldsymbol{v}}_q$ as follows:

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	0	0	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	0	0

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	0	0	0	0	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	1	$\tilde{\boldsymbol{y}}_q[i]$

Then we replace $\bar{\boldsymbol{r}}_q[\iota]$ for each $\iota \in [m-1]$ with a random value. For $\iota = 1, 2, \ldots, m-1$, we make the following changes in this order. Let $\mathbf{M}_{f,r}$ ($r \in [m]$) denote the matrices in Definition 7. By the function-hiding property of sIPFE, we firstly change the vectors as follows:

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{N}_{r,\iota}^t[\mathsf{const}]$	$\mathbf{N}_{r,\iota}^t[\mathsf{coef}_i]$	$\mathbf{M}_{f,r}[\mathsf{const}, \iota]$	$\mathbf{M}_{f,r}[\mathsf{coef}_i, \iota]$	$\mathbf{L}_{r,>\iota}^t[\mathsf{const}]$	$\mathbf{L}_{r,>\iota}^t[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,<\iota,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,<\iota,q}[\mathsf{coef}_i]$

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	0	0	$\bar{\boldsymbol{r}}_q[\iota]$	$\bar{\boldsymbol{r}}_q[\iota]\tilde{\boldsymbol{y}}_q[i]$	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	1	$\tilde{\boldsymbol{y}}_q[i]$
$\tilde{\boldsymbol{v}}_{q'}$ $(q' \neq q)$	$\boldsymbol{s}_{q'}[t]$	$\boldsymbol{s}_{q'}[t]\tilde{\boldsymbol{y}}_{q'}[i]$	$\bar{\boldsymbol{r}}_{q'}[\iota]$	$\bar{\boldsymbol{r}}_{q'}[\iota]\tilde{\boldsymbol{y}}_{q'}[i]$	0	0	0	0

where

$$\bar{\boldsymbol{r}}_q[\iota] := \sum_{t\in[k]} \boldsymbol{s}_q[t]\boldsymbol{r}_t[\iota], \quad \bar{\boldsymbol{r}}_{q'}[\iota] := \sum_{t\in[k]} \boldsymbol{s}_{q'}[t]\boldsymbol{r}_t[\iota], \quad \bar{\boldsymbol{r}}_q[\kappa] \xleftarrow{\$} \mathbb{Z}_p \ (\kappa < \iota)$$

and for each $* \in \{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}\}$,

$$\mathbf{N}_{r,\iota}^t[*] = \mathbf{L}_r^t[*] - \mathbf{M}_{f,r}[*,\iota]\boldsymbol{r}_t[\iota],\ \mathbf{L}_{r,>\iota}^t[*] = \mathbf{L}_r^t[*] - \sum_{\kappa\le\iota} \mathbf{M}_{f,r}[*,\kappa]\boldsymbol{r}_t[\kappa],\ \bar{\mathbf{L}}_{r,<\iota,q}[*] = \sum_{\kappa<\iota} \mathbf{M}_{f,r}[*,\kappa]\bar{\boldsymbol{r}}_q[\kappa].$$

By the MDDH_k assumption, we can sample $\bar{\boldsymbol{r}}_q[\iota]$ and $\bar{\boldsymbol{r}}_{q'}[\iota]$ uniformly at random. Furthermore, by applying the MDDH_k assumption again, we can restore $\bar{\boldsymbol{r}}_{q'}[\iota]$ to the original form. By the function-hiding property of sIPFE, the vectors can be changed as follows:

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	0	0	$\mathbf{L}_{r,>\iota}^t[\mathsf{const}]$	$\mathbf{L}_{r,>\iota}^t[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,<\iota+1,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,<\iota+1,q}[\mathsf{coef}_i]$

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	0	0	0	0	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	1	$\tilde{\boldsymbol{y}}_q[i]$
$\tilde{\boldsymbol{v}}_{q'}$ $(q' \neq q)$	$\boldsymbol{s}_{q'}[t]$	$\boldsymbol{s}_{q'}[t]\tilde{\boldsymbol{y}}_{q'}[i]$	0	0	0	0	0	0

Repeating this operation until $\iota = m-1$, the vectors become

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	0	0	$\mathbf{L}_{r,>m-1}^t[\mathsf{const}]$	0	$\bar{\mathbf{L}}_{r,<m,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,<m,q}[\mathsf{coef}_i]$

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	0	0	0	0	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	1	$\tilde{\boldsymbol{y}}_q[i]$
$\tilde{\boldsymbol{v}}_{q'}$ $(q' \neq q)$	$\boldsymbol{s}_{q'}[t]$	$\boldsymbol{s}_{q'}[t]\tilde{\boldsymbol{y}}_{q'}[i]$	0	0	0	0	0	0

where $\mathbf{L}_{r,>m-1}^t[\mathsf{const}] = \boldsymbol{\alpha}[t]\cdot\mathbf{1}[r=m] + \boldsymbol{\beta}[t]\cdot\mathbf{1}[r=1]$. We can proceed to switch the rest of $\tilde{\boldsymbol{u}}_r[\mathsf{tmpconst}^t]$ in the same way. By the function-hiding property of sIPFE, they can be changed to

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}] - \mathbf{L}_{r,>m-1}^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	$\xi_{1,r}$	$\xi_{2,r}\cdot\mathbf{1}[i=1]$	0	0	$\bar{\mathbf{L}}_{r,<m,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,<m,q}[\mathsf{coef}_i]$

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	extconst	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	0	0	$\bar{\mu}_q$	$\bar{\eta}_q\cdot\mathbf{1}[i=1]$	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	1	$\tilde{\boldsymbol{y}}_q[i]$
$\tilde{\boldsymbol{v}}_{q'}$ $(q' \neq q)$	$\boldsymbol{s}_{q'}[t]$	$\boldsymbol{s}_{q'}[t]\tilde{\boldsymbol{y}}_{q'}[i]$	$\bar{\mu}_{q'}$	$\bar{\eta}_{q'}\cdot\mathbf{1}[i=1]$	0	0	0	0

where

$$\bar{\mu}_q = \langle \boldsymbol{s}_q, \boldsymbol{\mu} \rangle, \quad \bar{\eta}_q := \langle \boldsymbol{s}_q, \boldsymbol{\eta} \rangle, \quad \bar{\mu}_{q'} = \langle \boldsymbol{s}_{q'}, \boldsymbol{\mu} \rangle, \quad \bar{\eta}_{q'} := \langle \boldsymbol{s}_{q'}, \boldsymbol{\eta} \rangle,$$
$$\xi_{1,r} := \begin{cases} \mathbf{1}[r = m] & \text{if } \mathsf{flag} = 1 \\ \mathbf{1}[r = 1] & \text{if } \mathsf{flag} = 0 \end{cases}, \quad \xi_{2,r} := \begin{cases} 0 & \text{if } \mathsf{flag} = 1 \\ \mathbf{1}[r = m] & \text{if } \mathsf{flag} = 0 \end{cases}.$$

By the MDDH_k assumption, we can sample $\bar{\mu}_q$, $\bar{\mu}_{q'}$, $\bar{\eta}_q$, and $\bar{\eta}_{q'}$ uniformly at random. Furthermore, by applying the MDDH_k assumption again, we can restore $\bar{\mu}_{q'}$ and $\bar{\eta}_{q'}$ to the original form. By the function-hiding property of sIPFE, the vectors can be changed as

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	$\mathsf{extconst}$	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	0	0	0	0	$\bar{\mathbf{L}}_{r,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,q}[\mathsf{coef}_i]$

vector	$\mathsf{simconst}^t$	$\mathsf{simcoef}_i^t$	$\mathsf{extconst}$	$\mathsf{extcoef}_i$	$\mathsf{tmpconst}^t$	$\mathsf{tmpcoef}_i^t$	$\widetilde{\mathsf{const}}$	$\widetilde{\mathsf{coef}}_i$
$\tilde{\boldsymbol{v}}_q$	0	0	0	0	$\boldsymbol{s}_q[t]$	$\boldsymbol{s}_q[t]\tilde{\boldsymbol{y}}_q[i]$	1	$\tilde{\boldsymbol{y}}_q[i]$
$\tilde{\boldsymbol{v}}_{q'}$ $(q' \neq q)$	$\boldsymbol{s}_{q'}[t]$	$\boldsymbol{s}_{q'}[t]\tilde{\boldsymbol{y}}_{q'}[i]$	0	0	0	0	0	0

where $\bar{\mu}_q, \bar{\eta}_q \xleftarrow{\$} \mathbb{Z}_p, \bar{\boldsymbol{r}}_q \xleftarrow{\$} \mathbb{Z}_p^{m-1}$ and

$$(\bar{\alpha}_q, \bar{\beta}_q) = \mathsf{Embed}(\bar{\mu}_q, \bar{\eta}_q, \mathsf{flag}), \quad (\bar{\mathbf{L}}_{1,q}, \ldots, \bar{\mathbf{L}}_{m,q}) \leftarrow \mathsf{Garble}(f_q, \bar{\alpha}_q, \bar{\beta}_q; \bar{\boldsymbol{r}}_q).$$

Then by AD-IND security proof for 1-ABE [LL20], $\boldsymbol{v}_q[\mathsf{ext}]$ can be set as $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. Since $\mathsf{extconst}$ and $\mathsf{extcoef}_i$ can be reused, we do not need the additional slots $\widetilde{\mathsf{sim}}$, which appeared in the key-policy case. By the function-hiding property of sIPFE, $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ can be changed in the following way:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\boldsymbol{x}^*$	$\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle + \mu'_q$	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\mathbf{0}$	$\mathbf{0}$	1	0

Since $\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle + \mu'_q$ is uniformly random and independent of the other parameters, it can be replaced with $\langle \boldsymbol{d}, \boldsymbol{y}_q \rangle + \mu'_q$. By retracing the previous argument in reverse, it follows that

$$\left| \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q,1}}(1^\lambda) = 0 \wedge \neg E_q] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{1,q,2}}(1^\lambda) = 0 \wedge \neg E_q] \right|$$

is also negligible. Thus, $\mathsf{Game}_{1,q,1}$ and $\mathsf{Game}_{1,q,2}$ are indistinguishable.

$\underline{\mathsf{Game}_{1,q,3}}$ $(q \in [Q_{\mathsf{pre}}])$: In this game, from $\mathsf{Game}_{1,q,2}$, we change the way that $\boldsymbol{v}_q$ for the q-th pre-challenge query is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_{q'}$ $(q' < q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{s}_{q'}$	$\boldsymbol{y}_{q'}$	$\mathbf{0}$	0	$\bar{\mu}_{q'}$
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\boldsymbol{y}_q$	$\mathbf{0}$	0	$\bar{\mu}_q$
$\boldsymbol{v}_{q'}$ $(q' > q)$	$\boldsymbol{y}_{q'}$	$\boldsymbol{s}_{q'}$	$\mathbf{0}$	$\boldsymbol{y}_{q'}$	0	$\bar{\mu}_{q'}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{1,q,2}$ and $\mathsf{Game}_{1,q,3}$ are indistinguishable.

$\underline{\mathsf{Game}_2}$: In this game, from $\mathsf{Game}_{1,Q_{\mathsf{pre}},3}$, we change the way that $\boldsymbol{u}$ of the challenge ciphertext and $\boldsymbol{v}_q$ for each post-challenge query are generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{u}$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}$	$\mathbf{0}$	0	1

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\mathbf{0}$	$\mathbf{0}$	0	$\langle \boldsymbol{x}^*, \boldsymbol{y}_q \rangle + \bar{\mu}_q$

By the function-hiding property of sIPFE, $\mathsf{Game}_{1,Q_{\mathsf{pre}},3}$ and Game_2 are indistinguishable.

Game_3: This is the ideal AD-SIM security game. In this game, we change the way that $\boldsymbol{v}_q$ for each post-challenge query is generated:

vector	**vec**	**pad**	$\mathbf{sim}_1$	$\mathbf{sim}_2$	sim	ext
$\boldsymbol{v}_q$	$\boldsymbol{y}_q$	$\boldsymbol{s}_q$	$\mathbf{0}$	$\mathbf{0}$	0	$z_q + \bar{\mu}_q$

$\boldsymbol{v}_q$ is not changed when the q-th query is admissible. When it is not admissible, by using the same technique as the game hopping from $\mathsf{Game}_{1,q,1}$ to $\mathsf{Game}_{1,q,2}$, $\bar{\mu}_q$ in $\boldsymbol{v}_q[\mathsf{ext}]$ can be replaced with a fresh value $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. Thus, in both Game_2 and Game_3, $\boldsymbol{v}_q[\mathsf{ext}]$ is uniformly random and independent of the other parameters. This completes the security proof.

7 AD-SIM Secure Constructions for ABPs with Shorter Attribute Keys

Our AD-SIM secure schemes in Section 6 have attribute key size quadratic in d. In this section, we propose variant schemes, whose attribute key size is linear in d.

7.1 Constructions

We propose schemes for both the key-policy setting and the ciphertext-policy setting. We firstly present the key-policy construction.

$\mathsf{Setup}(1^\lambda)$:

1. Sets $\mathfrak{s}_{\mathsf{pub}} = \{\mathsf{vec}, \mathbf{pad}\}$, $\tilde{\mathfrak{s}}_{\mathsf{pub}} = \left\{\{\mathsf{const}^t\}_{t\in[k]}, \{\mathsf{coef}_i^t\}_{t\in[k],i\in[n]}\right\}$ where $|\mathbf{pad}| = k$. Sets $|\mathfrak{s}_{\mathsf{priv}}| = 4$, $|\tilde{\mathfrak{s}}_{\mathsf{priv}}| = n + 3$.
2. Runs $(\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$ for each $j \in [d]$.
3. Runs $(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{msk}}) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \tilde{\mathfrak{s}}_{\mathsf{pub}}, \tilde{\mathfrak{s}}_{\mathsf{priv}})$.
4. Outputs $\mathsf{mpk} := (\{\mathsf{mpk}_j\}_{j\in[d]}, \widetilde{\mathsf{mpk}})$ and $\mathsf{msk} := (\{\mathsf{msk}_j\}_{j\in[d]}, \widetilde{\mathsf{msk}})$.

$\mathsf{Enc}(\mathsf{mpk}, \tilde{\boldsymbol{x}}, \boldsymbol{x})$:

1. Samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$.
2. Sets the vectors $\{\boldsymbol{u}_j\}_{j\in[d]}$, $\tilde{\boldsymbol{u}}$ as follows:

vector	vec	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{u}_j$	$\boldsymbol{x}[j]$	$\boldsymbol{s}$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{u}}$	$\boldsymbol{s}[t]$	$\boldsymbol{s}[t]\tilde{\boldsymbol{x}}[i]$	$\mathbf{0}$

3. Runs $\mathsf{ict}_j \leftarrow \mathsf{sIPFE.SlotEnc}(\mathsf{mpk}_j, [\boldsymbol{u}_j]_1)$, $\widetilde{\mathsf{ict}} \leftarrow \mathsf{sIPFE.SlotEnc}(\widetilde{\mathsf{mpk}}, [\tilde{\boldsymbol{u}}]_1)$.
4. Outputs $(\tilde{\boldsymbol{x}}, \{\mathsf{ict}_j\}_{j\in[d]}, \widetilde{\mathsf{ict}})$.

$\mathsf{KeyGen}(\mathsf{msk}, f, \mathsf{flag} \in \{0,1\}, \boldsymbol{y})$:

1. Samples $\boldsymbol{\mu}_j, \boldsymbol{\eta}_j \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $(\boldsymbol{\alpha}_j, \boldsymbol{\beta}_j) := \mathsf{Embed}(\boldsymbol{\mu}_j, \boldsymbol{\eta}_j, \mathsf{flag})$ for each $j \in [d]$.

2. Samples $\boldsymbol{\rho}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for each $j \in [d-1]$ and sets $\boldsymbol{\rho}_d := -\boldsymbol{\rho}_1 - \cdots - \boldsymbol{\rho}_{d-1}$.
3. Runs $(\mathbf{L}_{1,j}^t, \ldots, \mathbf{L}_{m,j}^t) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}_j[t], \boldsymbol{\beta}_j[t]; \boldsymbol{r}_{t,j})$ where $\boldsymbol{r}_{t,j} \xleftarrow{\$} \mathbb{Z}_p^{m-1}$ for each $t \in [k]$ and $j \in [d]$.
4. Sets the vectors $\{\boldsymbol{v}_j\}_{j\in[d]}, \{\tilde{\boldsymbol{v}}_{r,j}\}_{r\in[m],j\in[d]}$ as follows:

vector	vec	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{v}_j$	$\boldsymbol{y}[j]$	$\boldsymbol{y}[j]\boldsymbol{\mu}_j + \boldsymbol{\rho}_j$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{v}}_{r,j}$	$\mathbf{L}_{r,j}^t[\mathsf{const}]$	$\mathbf{L}_{r,j}^t[\mathsf{coef}_i]$	$\mathbf{0}$

5. Runs $\mathsf{isk}_j \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_j]_2)$, $\widetilde{\mathsf{isk}}_{r,j} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{r,j}]_2)$.
6. Outputs $(f, \mathsf{flag}, \{\mathsf{isk}_j\}_{j\in[d]}, \{\widetilde{\mathsf{isk}}_{r,j}\}_{r\in[m],j\in[d]})$.

$\underline{\mathsf{Dec}((f, \mathsf{flag}, \{\mathsf{isk}_j\}_{j\in[d]}, \{\widetilde{\mathsf{isk}}_{r,j}\}_{r\in[m],j\in[d]}), (\tilde{\boldsymbol{x}}, \{\mathsf{ict}_j\}_{j\in[d]}, \widetilde{\mathsf{ict}}))}$:

1. Computes $[\nu_j]_T := \mathsf{sIPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}_j)$ for each $j \in [d]$.
2. Computes $[\ell_{r,j}]_T := \mathsf{sIPFE.Dec}(\widetilde{\mathsf{isk}}_{r,j}, \widetilde{\mathsf{ict}})$ for each $r \in [m], j \in [d]$.
3. Computes

$$[\mu'_j]_T := \begin{cases} [f(\tilde{\boldsymbol{x}})^{-1}\mathsf{Eval}(f, \tilde{\boldsymbol{x}}, \ell_{1,j}, \ldots, \ell_{m,j})]_T & \text{if } \mathsf{flag} = 1, \\ [\mathsf{Eval}(f, \tilde{\boldsymbol{x}}, \ell_{1,j}, \ldots, \ell_{m,j})]_T & \text{if } \mathsf{flag} = 0. \end{cases}$$

4. Computes $z := \sum_{j\in[d]} (\nu_j - \boldsymbol{y}[j]\mu'_j)$ from $[z]_T$ by brute-forcing the discrete logarithm.

Correctness

When "$\mathsf{flag} = 1$ and $f(\tilde{\boldsymbol{x}}) \neq 0$" or "$\mathsf{flag} = 0$ and $f(\tilde{\boldsymbol{x}}) = 0$", we have $\mu'_j = \langle \boldsymbol{\mu}_j, \boldsymbol{s} \rangle$. Then it follows that

$$\sum_{j\in[d]} (\nu_j - \boldsymbol{y}[j]\mu'_j) = \sum_{j\in[d]} (\boldsymbol{x}[j]\boldsymbol{y}[j] + \langle \boldsymbol{\rho}_j, \boldsymbol{s} \rangle) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle.$$

The hierarchical structures can be added similarly to Section 6.1. Setup samples random vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k \xleftarrow{\$} \mathbb{Z}_p^k$. For $t' \in [k], i' \in [n]$, we define the following vectors:

vector	vec	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{v}_{\mathsf{base},t'}$	0	$\boldsymbol{b}_{t'}$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{v}}_{\mathsf{base},t'}$	$\boldsymbol{b}_{t'}[t]$	0	$\mathbf{0}$
$\tilde{\boldsymbol{v}}_{\mathsf{base},t',i'}$	0	$\boldsymbol{b}_{t'}[t] \cdot \mathbf{1}[i = i']$	$\mathbf{0}$

$\mathsf{bk}_{\mathsf{base},t',j} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_{\mathsf{base},t'}]_2)$, $\widetilde{\mathsf{bk}}_{\mathsf{base},t'} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{\mathsf{base},t'}]_2)$, $\widetilde{\mathsf{bk}}_{\mathsf{base},t',i'} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{\mathsf{base},t',i'}]_2)$ are added to the master public key.

$\mathsf{KeyGen}_{\mathsf{Att}}$ is the same as KeyGen except that we set $\boldsymbol{v}_j[\mathsf{vec}] = 1$ and $\boldsymbol{v}_j[\mathbf{pad}] = \boldsymbol{\mu}_j$. Let $\left(f, \mathsf{flag}, \{\mathsf{isk}_j^{\mathsf{Att}}\}_{j\in[d]}, \left\{\widetilde{\mathsf{isk}}_{r,j}^{\mathsf{Att}}\right\}_{r\in[m],j\in[d]}\right)$ denote the attribute key. $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}$ samples $\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\eta}}_j \xleftarrow{\$} \mathbb{Z}_p^k$, sets $(\hat{\boldsymbol{\alpha}}_j, \hat{\boldsymbol{\beta}}_j) := \mathsf{Embed}(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\eta}}_j, \mathsf{flag})$, and runs $(\hat{\mathbf{L}}_{1,j}^{t'}, \ldots, \hat{\mathbf{L}}_{m,j}^{t'}) \leftarrow \mathsf{Garble}(f, \hat{\boldsymbol{\alpha}}_j[t'], \hat{\boldsymbol{\beta}}_j[t'])$ for each $j \in [d]$. It also samples zero-sum vectors $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_d \in \mathbb{Z}_p^k$ and computes

$$\mathsf{isk}_j \leftarrow \left(\boldsymbol{y}[j] \boxdot \mathsf{isk}_j^{\mathsf{Att}}\right) \boxplus \left(\sum_{t'\in[k]} (\boldsymbol{y}[j]\hat{\boldsymbol{\mu}}_j[t'] + \boldsymbol{\rho}_j[t']) \boxdot \mathsf{bk}_{\mathsf{base},t',j}\right),$$

$$\widetilde{\mathsf{isk}}_{r,j} \leftarrow \widetilde{\mathsf{isk}}_{r,j}^{\mathsf{Att}} \boxplus \left(\sum_{t'\in[k]} \hat{\mathbf{L}}_{r,j}^{t'}[\mathsf{const}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \hat{\mathbf{L}}_{r,j}^{t'}[\mathsf{coef}_{i'}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right).$$

$\mathsf{KeyGen}_{\mathsf{Vec}}$ samples zero-sum vectors $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_d \in \mathbb{Z}_p^k$ and runs $\mathsf{isk}_j^{\mathsf{Vec}} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_j]_2)$ where $\boldsymbol{v}_j[\mathsf{vec}] = \boldsymbol{y}[j], \boldsymbol{v}_j[\mathbf{pad}] = \boldsymbol{\rho}_j, \boldsymbol{v}_j[\mathfrak{s}_{\mathsf{priv}}] = \mathbf{0}$ for each $j \in [d]$. $\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}$ samples zero-sum vectors $\boldsymbol{\rho}'_1, \ldots, \boldsymbol{\rho}'_d$ and computes

$$\mathsf{isk}_j \leftarrow \mathsf{isk}_j^{\mathsf{Vec}} \boxplus \left(\sum_{t' \in [k]} (\boldsymbol{y}[j]\boldsymbol{\mu}_j[t'] + \boldsymbol{\rho}'_j[t']) \boxdot \mathsf{bk}_{\mathsf{base},t',j} \right),$$

$$\widetilde{\mathsf{isk}}_{r,j} \leftarrow \left(\sum_{t' \in [k]} \mathbf{L}_{r,j}^{t'}[\mathsf{const}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'} \right) \boxplus \left(\sum_{t' \in [k], i' \in [n]} \mathbf{L}_{r,j}^{t'}[\mathsf{coef}_{i'}] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'} \right)$$

where $\boldsymbol{\mu}_j, \{\mathbf{L}_{r,j}^{t'}\}_{r\in[m],t'\in[k]}$ for each j are generated in the same way as in KeyGen.

We then present the ciphertext-policy construction.

$\underline{\mathsf{Setup}(1^\lambda)}$:

1. Sets $\mathfrak{s}_{\mathsf{pub}} = \{\mathsf{vec}, \mathbf{pad}\}$, $\tilde{\mathfrak{s}}_{\mathsf{pub}} = \left\{\{\mathsf{const}^t\}_{t\in[k]}, \{\mathsf{coef}_i^t\}_{t\in[k],i\in[n]}\right\}$ where $|\mathbf{pad}| = k$. Sets $|\mathfrak{s}_{\mathsf{priv}}| = 4, |\tilde{\mathfrak{s}}_{\mathsf{priv}}| = 2(k+1)(n+1)$.
2. Runs $(\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \mathfrak{s}_{\mathsf{pub}}, \mathfrak{s}_{\mathsf{priv}})$ for each $j \in [d]$.
3. Runs $(\widetilde{\mathsf{mpk}}, \widetilde{\mathsf{msk}}) \leftarrow \mathsf{sIPFE.Setup}(1^\lambda, \tilde{\mathfrak{s}}_{\mathsf{pub}}, \tilde{\mathfrak{s}}_{\mathsf{priv}})$.
4. Outputs $\mathsf{mpk} := (\{\mathsf{mpk}_j\}_{j\in[d]}, \widetilde{\mathsf{mpk}})$ and $\mathsf{msk} := (\{\mathsf{msk}_j\}_{j\in[d]}, \widetilde{\mathsf{msk}})$.

$\underline{\mathsf{Enc}(\mathsf{mpk}, f, \mathsf{flag} \in \{0,1\}, \boldsymbol{x})}$:

1. Samples $\boldsymbol{\mu}, \boldsymbol{\eta} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $(\boldsymbol{\alpha}, \boldsymbol{\beta}) := \mathsf{Embed}(\boldsymbol{\mu}, \boldsymbol{\eta}, \mathsf{flag})$.
2. For each $t \in [k]$, runs $(\mathbf{L}_1^t, \ldots, \mathbf{L}_m^t) \leftarrow \mathsf{Garble}(f, \boldsymbol{\alpha}[t], \boldsymbol{\beta}[t]; \boldsymbol{r}_t)$ where $\boldsymbol{r}_t \xleftarrow{\$} \mathbb{Z}_p^{m-1}$.
3. Sets the vectors $\{\boldsymbol{u}_j\}_{j\in[d]}, \{\tilde{\boldsymbol{u}}_r\}_{r\in[m]}$ as follows:

vector	vec	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{u}_j$	$\boldsymbol{x}[j]$	$\boldsymbol{\mu}$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{u}}_r$	$\mathbf{L}_r^t[\mathsf{const}]$	$\mathbf{L}_r^t[\mathsf{coef}_i]$	$\mathbf{0}$

4. Runs $\mathsf{ict}_j \leftarrow \mathsf{sIPFE.SlotEnc}(\mathsf{mpk}_j, [\boldsymbol{u}_j]_1)$, $\widetilde{\mathsf{ict}}_r \leftarrow \mathsf{sIPFE.SlotEnc}(\widetilde{\mathsf{mpk}}, [\tilde{\boldsymbol{u}}_r]_1)$.
5. Outputs $(f, \mathsf{flag}, \{\mathsf{ict}_j\}_{j\in[d]}, \{\widetilde{\mathsf{ict}}_r\}_{r\in[m]})$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \tilde{\boldsymbol{y}}, \boldsymbol{y})}$:

1. Samples $\boldsymbol{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for each $j \in [d]$.
2. Samples $\boldsymbol{\rho}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for each $j \in [d-1]$ and sets $\boldsymbol{\rho}_d := -\boldsymbol{\rho}_1 - \cdots - \boldsymbol{\rho}_{d-1}$.
3. Sets the vectors $\{\boldsymbol{v}_j\}_{j\in[d]}, \{\tilde{\boldsymbol{v}}_j\}_{j\in[d]}$ as follows:

vector	vec	**pad**	$\mathfrak{s}_{\mathsf{priv}}$
$\boldsymbol{v}_j$	$\boldsymbol{y}[j]$	$\boldsymbol{y}[j]\boldsymbol{s}_j + \boldsymbol{\rho}_j$	$\mathbf{0}$

vector	const^t	coef_i^t	$\tilde{\mathfrak{s}}_{\mathsf{priv}}$
$\tilde{\boldsymbol{v}}_j$	$\boldsymbol{s}_j[t]$	$\boldsymbol{s}_j[t]\tilde{\boldsymbol{y}}[i]$	$\mathbf{0}$

4. Runs $\mathsf{isk}_j \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_j]_2)$, $\widetilde{\mathsf{isk}}_j \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_j]_2)$.
5. Outputs $(\tilde{\boldsymbol{y}}, \{\mathsf{isk}_j\}_{j\in[d]}, \{\widetilde{\mathsf{isk}}_j\}_{j\in[d]})$.

$\underline{\mathsf{Dec}((\tilde{\boldsymbol{y}}, \{\mathsf{isk}_j\}_{j\in[d]}, \{\widetilde{\mathsf{isk}}_j\}_{j\in[d]}), (f, \mathsf{flag}, \{\mathsf{ict}_j\}_{j\in[d]}, \{\widetilde{\mathsf{ict}}_r\}_{r\in[m]}))}$:

1. Computes $[\nu_j]_T := \mathsf{sIPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}_j)$ for each $j \in [d]$.
2. Computes $[\ell_{r,j}]_T := \mathsf{sIPFE.Dec}(\widetilde{\mathsf{isk}}_j, \widetilde{\mathsf{ict}}_r)$ for each $r \in [m], j \in [d]$.
3. Computes

$$[\mu'_j]_T := \begin{cases} [f(\tilde{\boldsymbol{y}})^{-1}\mathsf{Eval}(f, \tilde{\boldsymbol{y}}, \ell_{1,j}, \ldots, \ell_{m,j})]_T & \text{if } \mathsf{flag} = 1, \\ [\mathsf{Eval}(f, \tilde{\boldsymbol{y}}, \ell_{1,j}, \ldots, \ell_{m,j})]_T & \text{if } \mathsf{flag} = 0. \end{cases}$$

4. Computes $z := \sum_{j\in[d]} (\nu_j - \boldsymbol{y}[j]\mu'_j)$ from $[z]_T$ by brute-forcing the discrete logarithm.

To introduce the hierarchical structures, as in the key-policy setting, $\{\mathsf{bk}_{\mathsf{base},t',j}\}_{t'\in[k],j\in[d]}$, $\{\widetilde{\mathsf{bk}}_{\mathsf{base},t'}\}_{t'\in[k]}$, and $\{\widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\}_{t'\in[k],i'\in[n]}$ are added to the master public key.

$\mathsf{KeyGen}_{\mathsf{Att}}$ is the same as KeyGen except that we set $\boldsymbol{v}_j[\mathsf{vec}] = 1$ and $\boldsymbol{v}_j[\mathbf{pad}] = \boldsymbol{s}_j$. Let $\left(\tilde{\boldsymbol{y}}, \{\mathsf{isk}_j^{\mathsf{Att}}\}_{j\in[d]}, \left\{\widetilde{\mathsf{isk}}_j^{\mathsf{Att}}\right\}_{j\in[d]}\right)$ denote the attribute key. $\mathsf{Derive}_{\mathsf{Att}\to\mathsf{Vec}}$ samples $\hat{\boldsymbol{s}}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for each $j \in [d]$ and zero-sum vectors $\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_d \in \mathbb{Z}_p^k$. Then it computes

$$\mathsf{isk}_j \leftarrow \left(\boldsymbol{y}[j] \boxdot \mathsf{isk}_j^{\mathsf{Att}}\right) \boxplus \left(\sum_{t'\in[k]} (\boldsymbol{y}[j]\hat{\boldsymbol{s}}_j[t'] + \boldsymbol{\rho}_j[t']) \boxdot \mathsf{bk}_{\mathsf{base},t',j}\right),$$

$$\widetilde{\mathsf{isk}}_j \leftarrow \widetilde{\mathsf{isk}}_j^{\mathsf{Att}} \boxplus \left(\sum_{t'\in[k]} \hat{\boldsymbol{s}}_j[t'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \hat{\boldsymbol{s}}_j[t']\tilde{\boldsymbol{y}}[i'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right).$$

$\mathsf{KeyGen}_{\mathsf{Vec}}$ samples zero-sum vectors $\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_d \in \mathbb{Z}_p^k$ and runs $\mathsf{isk}_j^{\mathsf{Vec}} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_j]_2)$ where $\boldsymbol{v}_j[\mathsf{vec}] = \boldsymbol{y}[j], \boldsymbol{v}_j[\mathbf{pad}] = \boldsymbol{\rho}_j, \boldsymbol{v}_j[\mathfrak{s}_{\mathsf{priv}}] = \mathbf{0}$ for each $j \in [d]$. $\mathsf{Derive}_{\mathsf{Vec}\to\mathsf{Att}}$ samples $\boldsymbol{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$ for each $j \in [d]$ and zero-sum vectors $\boldsymbol{\rho}'_1, \dots, \boldsymbol{\rho}'_d$. Then it computes

$$\mathsf{isk}_j \leftarrow \mathsf{isk}_j^{\mathsf{Vec}} \boxplus \left(\sum_{t'\in[k]} (\boldsymbol{y}[j]\boldsymbol{s}_j[t'] + \boldsymbol{\rho}'_j[t']) \boxdot \mathsf{bk}_{\mathsf{base},t',j}\right),$$

$$\widetilde{\mathsf{isk}}_j \leftarrow \left(\sum_{t'\in[k]} \boldsymbol{s}_j[t'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t'}\right) \boxplus \left(\sum_{t'\in[k],i'\in[n]} \boldsymbol{s}_j[t']\tilde{\boldsymbol{y}}[i'] \boxdot \widetilde{\mathsf{bk}}_{\mathsf{base},t',i'}\right).$$

We give the security proof for the key-policy scheme in the next section. We omit the security proof for the ciphertext-policy scheme since it is a simple combination of Section 6.4 and the proof for the key-policy scheme.

Theorem 6. Assume that sIPFE is function-hiding, the AKGS is piecewise secure, and the MDDH_k assumption holds. Then the (hierarchical) key-policy AB-IPFE scheme and the (hierarchical) ciphertext-policy AB-IPFE scheme are AD-SIM secure.

7.2 Proof of Theorem 6

Let $\mathfrak{s}_{\mathsf{priv}} = \{\mathsf{sim}_1, \mathsf{sim}_2, \mathsf{sim}, \mathsf{ext}\}$ and $\tilde{\mathfrak{s}}_{\mathsf{priv}} = \{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \widetilde{\mathbf{sim}}\}$ where $|\widetilde{\mathbf{sim}}| = 2$. We construct the simulator as follows:

- $\mathsf{Setup}^*(1^\lambda)$: This is the same as Setup except that it samples $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k$ and appends it to msk.
- $\mathsf{KeyGen}_0^*(\mathsf{msk}, f_q, \mathsf{flag}_q \in \{0,1\}, \boldsymbol{y}_q)$:
  1. Samples $\boldsymbol{\mu}_{j,q}, \boldsymbol{\eta}_{j,q} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\boldsymbol{\alpha}_{j,q}, \boldsymbol{\beta}_{j,q}, (\mathbf{L}^t_{1,j,q}, \dots, \mathbf{L}^t_{m,j,q})$ for $j \in [d]$ as in KeyGen.
  2. Samples $\bar{\mu}_{j,q}, \bar{\eta}_{j,q} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\bar{\alpha}_{j,q}, \bar{\beta}_{j,q}, (\bar{\mathbf{L}}^t_{1,j,q}, \dots, \bar{\mathbf{L}}^t_{m,j,q})$ for $j \in [d]$ by using Embed and Garble.
  3. Samples $\boldsymbol{\rho}_{j,q} \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [d-1]$ and sets $\boldsymbol{\rho}_{d,q} := -\boldsymbol{\rho}_{1,q} - \cdots - \boldsymbol{\rho}_{d-1,q}$.
  4. Samples $\bar{\rho}_{j,q} \xleftarrow{\$} \mathbb{Z}_p$ for $j \in [d-1]$ and sets $\bar{\rho}_{d,q} := -\bar{\rho}_{1,q} - \cdots - \bar{\rho}_{d-1,q}$.
  5. Sets the vectors $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ as follows:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	$\boldsymbol{y}_q[j]$	0	0	$\boldsymbol{y}_q[j]\bar{\mu}_{j,q} + \bar{\rho}_{j,q}$

  6. Sets the vectors $\{\tilde{\boldsymbol{v}}_{r,j,q}\}_{r\in[m],j\in[d]}$ as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{v}}_{r,j,q}$	$\mathbf{L}^t_{r,j,q}[\mathsf{const}]$	$\mathbf{L}^t_{r,j,q}[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{coef}_i]$	$\mathbf{0}$

7. Runs $\mathsf{isk}_{j,q} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_{j,q}]_2)$, $\widetilde{\mathsf{isk}}_{r,j,q} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{r,j,q}]_2)$.
8. Outputs $(f_q, \mathsf{flag}_q, \{\mathsf{isk}_{j,q}\}_{j\in[d]}, \{\widetilde{\mathsf{isk}}_{r,j,q}\}_{r\in[m],j\in[d]})$.

- $\mathsf{Enc}^*(\mathsf{msk}, \tilde{\boldsymbol{x}}, \mathcal{V})$:

1. Samples a vector $\boldsymbol{d}$ such that $\langle \boldsymbol{x}^*, \boldsymbol{y}_q\rangle = \langle \boldsymbol{d}, \boldsymbol{y}_q\rangle$ for any admissible q.
2. Sets the vector $\tilde{\boldsymbol{u}}$ as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{u}}$	0	0	1	$\tilde{\boldsymbol{x}}[i]$	$\mathbf{0}$

3. Sets the vector $\boldsymbol{u}_j$ as follows:

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{u}_j$	0	$\mathbf{0}$	$\boldsymbol{d}[j]$	0	0	1

4. Runs $\mathsf{ict}_j \leftarrow \mathsf{sIPFE.Enc}(\mathsf{msk}_j, [\boldsymbol{u}_j]_1)$, $\widetilde{\mathsf{ict}} \leftarrow \mathsf{sIPFE.Enc}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{u}}]_1)$.
5. Outputs $(\tilde{\boldsymbol{x}}, \{\mathsf{ict}_j\}_{j\in[d]}, \widetilde{\mathsf{ict}})$.

- $\mathsf{KeyGen}_1^*(\mathsf{msk}, f_q, \mathsf{flag}_q \in \{0,1\}, \boldsymbol{y}_q, z_q \in \{\langle \boldsymbol{x}^*, \boldsymbol{y}_q\rangle, 0\})$:

1. Samples $\boldsymbol{\mu}_{j,q}, \boldsymbol{\eta}_{j,q} \xleftarrow{\$} \mathbb{Z}_p^k$ and sets $\boldsymbol{\alpha}_{j,q}, \boldsymbol{\beta}_{j,q}, (\mathbf{L}^t_{1,j,q}, \ldots, \mathbf{L}^t_{m,j,q})$ for $j \in [d]$ as in KeyGen.
2. Samples $\bar{\mu}_{j,q}, \bar{\eta}_{j,q} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\bar{\alpha}_{j,q}, \bar{\beta}_{j,q}, (\bar{\mathbf{L}}_{1,j,q}, \ldots, \bar{\mathbf{L}}_{m,j,q})$ for $j \in [d]$ by using Embed and Garble.
3. Samples $\boldsymbol{\rho}_{j,q} \xleftarrow{\$} \mathbb{Z}_p^k$ for $j \in [d-1]$ and sets $\boldsymbol{\rho}_{d,q} := -\boldsymbol{\rho}_{1,q} - \cdots - \boldsymbol{\rho}_{d-1,q}$.
4. Samples $\bar{\rho}_{j,q} \xleftarrow{\$} \mathbb{Z}_p$ for $j \in [d-1]$ and sets $\bar{\rho}_{d,q} := -\bar{\rho}_{1,q} - \cdots - \bar{\rho}_{d-1,q}$.
5. Sets the vectors $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ as follows:

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	0	0	0	$z_q \cdot \mathbf{1}[j=1] + \boldsymbol{y}_q[j]\bar{\mu}_{j,q} + \bar{\rho}_{j,q}$

6. Sets the vectors $\{\tilde{\boldsymbol{v}}_{r,j,q}\}_{r\in[m],j\in[d]}$ as follows:

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{v}}_{r,j,q}$	$\mathbf{L}^t_{r,j,q}[\mathsf{const}]$	$\mathbf{L}^t_{r,j,q}[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{coef}_i]$	$\mathbf{0}$

7. Runs $\mathsf{isk}_{j,q} \leftarrow \mathsf{sIPFE.KeyGen}(\mathsf{msk}_j, [\boldsymbol{v}_{j,q}]_2)$, $\widetilde{\mathsf{isk}}_{r,j,q} \leftarrow \mathsf{sIPFE.KeyGen}(\widetilde{\mathsf{msk}}, [\tilde{\boldsymbol{v}}_{r,j,q}]_2)$.
8. Outputs $(f_q, \mathsf{flag}_q, \{\mathsf{isk}_{j,q}\}_{j\in[d]}, \{\widetilde{\mathsf{isk}}_{r,j,q}\}_{r\in[m],j\in[d]})$.

We define the intermediate security games as follows. Let Q_{pre} denote an upper bound on the number of pre-challenge queries.

$\underline{\mathsf{Game}_0}$: This is the real AD-SIM security game.

$\underline{\mathsf{Game}_1}$: In this game, we change the way that $\{\boldsymbol{u}_j\}_{j\in[d]}$ of the challenge ciphertext and $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ for any query are generated:

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{u}_j$	0	$\mathbf{0}$	$\boldsymbol{d}[j]$	$\boldsymbol{x}^*[j]$	0	1

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	$\mathbf{0}$	$\boldsymbol{y}_q[j]$	0	$\boldsymbol{y}_q[j]\bar{\mu}_{j,q} + \bar{\rho}_{j,q}$

Also, $\tilde{\boldsymbol{u}}$ of the challenge ciphertext and $\tilde{\boldsymbol{v}}_{r,j,q}$ for any query are modified as

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{u}}$	0	0	1	$\tilde{\boldsymbol{x}}[i]$	$\mathbf{0}$

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{v}}_{r,j,q}$	$\mathbf{L}^t_{r,j,q}[\mathsf{const}]$	$\mathbf{L}^t_{r,j,q}[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{coef}_i]$	$\mathbf{0}$

where

$$\bar{\mu}_{j,q} = \langle \boldsymbol{\mu}_{j,q}, \boldsymbol{s}\rangle, \quad \bar{\eta}_{j,q} = \langle \boldsymbol{\eta}_{j,q}, \boldsymbol{s}\rangle, \quad (\bar{\alpha}_{j,q}, \bar{\beta}_{j,q}) = \mathsf{Embed}(\bar{\mu}_{j,q}, \bar{\eta}_{j,q}, \mathsf{flag}_q),$$
$$\bar{\rho}_{j,q} = \langle \boldsymbol{\rho}_{j,q}, \boldsymbol{s}\rangle, \quad \bar{\boldsymbol{r}}_{j,q} = \sum_{t\in[k]} \boldsymbol{r}_{t,j,q}\boldsymbol{s}[t], \quad (\bar{\mathbf{L}}_{1,j,q}, \ldots, \bar{\mathbf{L}}_{m,j,q}) \leftarrow \mathsf{Garble}(f_q, \bar{\alpha}_{j,q}, \bar{\beta}_{j,q}; \bar{\boldsymbol{r}}_{j,q}).$$

By the function-hiding property of sIPFE, Game_0 and Game_1 are indistinguishable.

$\underline{\mathsf{Game}_2}$: In this game, we sample $\bar{\mu}_{j,q}, \bar{\eta}_{j,q} \xleftarrow{\$} \mathbb{Z}_p$, $\bar{\boldsymbol{r}}_{j,q} \xleftarrow{\$} \mathbb{Z}_p^{m-1}$, $\bar{\rho}_{1,q}, \ldots, \bar{\rho}_{d-1,q} \xleftarrow{\$} \mathbb{Z}_p$ and set

$$(\bar{\alpha}_{j,q}, \bar{\beta}_{j,q}) = \mathsf{Embed}(\bar{\mu}_{j,q}, \bar{\eta}_{j,q}, \mathsf{flag}_q), \quad (\bar{\mathbf{L}}_{1,j,q}, \ldots, \bar{\mathbf{L}}_{m,j,q}) \leftarrow \mathsf{Garble}(f_q, \bar{\alpha}_{j,q}, \bar{\beta}_{j,q}; \bar{\boldsymbol{r}}_{j,q}),$$

and $\bar{\rho}_{d,q} = -\bar{\rho}_{1,q} - \cdots - \bar{\rho}_{d-1,q}$. By the MDDH_k assumption, Game_1 and Game_2 are indistinguishable.

$\underline{\mathsf{Game}_{2,q,1}}$ ($q \in [Q_{\mathsf{pre}}]$): Let $\mathsf{Game}_{2,0,3}$ denote Game_2. In this game, from $\mathsf{Game}_{2,q-1,3}$ (defined later in the cases that $q > 1$), we change the way that $\{\boldsymbol{u}_j\}_{j\in[d]}$ of the challenge ciphertext and $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ for the q-th pre-challenge query is generated:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{u}_j$	0	$\mathbf{0}$	$\boldsymbol{d}[j]$	$\boldsymbol{x}^*[j]$	$\boldsymbol{x}^*[j]\boldsymbol{y}_q[j] + \bar{\rho}_{j,q}$	1

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q'}$ $(q' < q)$	$\boldsymbol{y}_{q'}[j]$	$\boldsymbol{y}_{q'}[j]\boldsymbol{\mu}_{j,q'} + \boldsymbol{\rho}_{j,q'}$	$\boldsymbol{y}_{q'}[j]$	0	0	$\boldsymbol{y}_{q'}[j]\bar{\mu}_{j,q'} + \bar{\rho}_{j,q'}$
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	0	0	1	$\boldsymbol{y}_q[j]\bar{\mu}_{j,q}$
$\boldsymbol{v}_{j,q'}$ $(q' > q)$	$\boldsymbol{y}_{q'}[j]$	$\boldsymbol{y}_{q'}[j]\boldsymbol{\mu}_{j,q'} + \boldsymbol{\rho}_{j,q'}$	0	$\boldsymbol{y}_{q'}[j]$	0	$\boldsymbol{y}_{q'}[j]\bar{\mu}_{j,q'} + \bar{\rho}_{j,q'}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{2,q-1,3}$ and $\mathsf{Game}_{2,q,1}$ are indistinguishable.

$\underline{\mathsf{Game}_{2,q,2}}$ ($q \in [Q_{\mathsf{pre}}]$): In this game, from $\mathsf{Game}_{2,q,1}$, we change the way that $\{\boldsymbol{u}_j\}_{j\in[d]}$ of the challenge ciphertext is generated:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{u}_j$	$\mathbf{0}$	$\mathbf{0}$	$\boldsymbol{d}[j]$	$\boldsymbol{x}^*[j]$	$\boldsymbol{d}[j]\boldsymbol{y}_q[j] + \bar{\rho}_{j,q}$	1

Let E_q be an event that the q-th query is admissible. In the case of E_q, we have $\langle \boldsymbol{x}^*, \boldsymbol{y}_q\rangle = \langle \boldsymbol{d}, \boldsymbol{y}_q\rangle$. By setting $\bar{\rho}'_{j,q} := \bar{\rho}_{j,q} + (\boldsymbol{x}^*[j] - \boldsymbol{d}[j])\boldsymbol{y}_q[j]$, we have $\bar{\rho}'_{1,q} + \cdots + \bar{\rho}'_{d,q} = 0$ and $\boldsymbol{x}^*[j]\boldsymbol{y}_q[j] + \bar{\rho}_{j,q} = \boldsymbol{d}[j]\boldsymbol{y}_q[j] + \bar{\rho}'_{j,q}$. Thus, it follows that

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,1}}(1^\lambda) = 0 \wedge E_q] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,2}}(1^\lambda) = 0 \wedge E_q].$$

We consider the case of $\neg E_q$. By AD-IND security of the underlying ABE [LL20], $\boldsymbol{v}_{j,q}[\mathsf{ext}]$ in $\mathsf{Game}_{2,q,1}$ can be set as $\mu'_{j,q}$ where $\mu'_{j,q} \xleftarrow{\$} \mathbb{Z}_p$ is an independent fresh value. By the function-hiding property of sIPFE, $\{\boldsymbol{u}_j\}_{j\in[d]}$ of the challenge ciphertext and $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ can be changed in the following way:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{u}_j$	0	$\mathbf{0}$	$\boldsymbol{d}[j]$	$\boldsymbol{x}^*[j]$	$\boldsymbol{x}^*[j]\boldsymbol{y}_q[j] + \bar{\rho}_{j,q} + \boldsymbol{y}_q[j]\mu'_{j,q}$	1

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	0	0	1	0

If $\boldsymbol{y}_q = \mathbf{0}$, we have $\boldsymbol{x}^*[j]\boldsymbol{y}_q[j] = \boldsymbol{d}[j]\boldsymbol{y}_q[j] = 0$ for any $j \in [d]$. Otherwise, $\{\boldsymbol{x}^*[j]\boldsymbol{y}_q[j] + \bar{\rho}_{j,q} + \boldsymbol{y}_q[j]\mu'_{j,q}\}_{j\in[d]}$ is uniformly random and independent of the other parameters, it can be replaced with $\boldsymbol{d}[j]\boldsymbol{y}_q[j] + \bar{\rho}_{j,q} + \boldsymbol{y}_q[j]\mu'_{j,q}$. By retracing the previous argument in reverse, it follows that

$$\left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,1}}(1^\lambda) = 0 \wedge \neg E_q] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game}_{2,q,2}}(1^\lambda) = 0 \wedge \neg E_q]\right|$$

is also negligible. Thus, $\mathsf{Game}_{2,q,1}$ and $\mathsf{Game}_{2,q,2}$ are indistinguishable.

$\underline{\mathsf{Game}_{2,q,3}}$ ($q \in [Q_{\mathsf{pre}}]$): In this game, from $\mathsf{Game}_{2,q,2}$, we change the way that $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ for the q-th pre-challenge query is generated:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q'}$ $(q' < q)$	$\boldsymbol{y}_{q'}[j]$	$\boldsymbol{y}_{q'}[j]\boldsymbol{\mu}_{j,q'} + \boldsymbol{\rho}_{j,q'}$	$\boldsymbol{y}_{q'}[j]$	0	0	$\boldsymbol{y}_{q'}[j]\bar{\mu}_{j,q'} + \bar{\rho}_{j,q'}$
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	$\boldsymbol{y}_q[j]$	0	0	$\boldsymbol{y}_q[j]\bar{\mu}_{j,q} + \bar{\rho}_{j,q}$
$\boldsymbol{v}_{j,q'}$ $(q' > q)$	$\boldsymbol{y}_{q'}[j]$	$\boldsymbol{y}_{q'}[j]\boldsymbol{\mu}_{j,q'} + \boldsymbol{\rho}_{j,q'}$	0	$\boldsymbol{y}_{q'}[j]$	0	$\boldsymbol{y}_{q'}[j]\bar{\mu}_{j,q'} + \bar{\rho}_{j,q'}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{2,q,2}$ and $\mathsf{Game}_{2,q,3}$ are indistinguishable.

$\underline{\mathsf{Game}_3}$: In this game, from $\mathsf{Game}_{2,Q_{\mathsf{pre}},3}$, we change the way that $\{\boldsymbol{u}_j\}_{j\in[d]}$ of the challenge ciphertext and $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ for each post-challenge query is generated:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{u}_j$	0	$\mathbf{0}$	$\boldsymbol{d}[j]$	0	0	1

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	0	0	0	$\boldsymbol{x}^*[j]\boldsymbol{y}_q[j] + \boldsymbol{y}_q[j]\bar{\mu}_{j,q} + \bar{\rho}_{j,q}$

By the function-hiding property of sIPFE, $\mathsf{Game}_{2,Q_{\mathsf{pre}},3}$ and Game_3 are indistinguishable.

$\underline{\mathsf{Game}_4}$: In this game, we change the way that $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ for each post-challenge query is generated:

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{y}_q[j]\boldsymbol{\mu}_{j,q} + \boldsymbol{\rho}_{j,q}$	0	0	0	$z_q \cdot \mathbf{1}[j = 1] + \boldsymbol{y}_q[j]\bar{\mu}_{j,q} + \bar{\rho}_{j,q}$

By randomness of $\{\bar{\rho}_{j,q}\}_{j\in[d]}$, the distribution of $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ is not changed when the q-th query is admissible. When it is not, by AD-IND security of the underlying ABE [LL20], $\bar{\mu}_{j,q}$ can be replaced with a fresh value $\mu'_{j,q} \xleftarrow{\$} \mathbb{Z}_p$, which is independent of $(\bar{\alpha}_{j,q}, \bar{\beta}_{j,q})$. If $\boldsymbol{y}_q = \mathbf{0}$, $\{\boldsymbol{v}_{j,q}\}_{j\in[d]}$ is not changed. Otherwise, in both Game_3 and Game_4, each $\{\boldsymbol{v}_{j,q}[\mathsf{ext}]\}_{j\in[d]}$ is uniformly random and independent of the other parameters. This completes the security proof in the non-hierarchical case.

We then consider the hierarchical structure. We can see that the additional components in the master public key do not compromise security in the same way as the proof of Theorem 4. The oracle corresponding to $\mathsf{KeyGen}_{\mathsf{Att}}$ can be simulated similarly to KeyGen_0^* and KeyGen_1^*. Concretely, we set

vector	vec	**pad**	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	1	$\boldsymbol{\mu}_{j,q}$	1	0	0	$\bar{\mu}_{j,q}$

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{v}}_{r,j,q}$	$\mathbf{L}^t_{r,j,q}[\mathsf{const}]$	$\mathbf{L}^t_{r,j,q}[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{coef}_i]$	$\mathbf{0}$

for pre-challenge queries and

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	1	$\boldsymbol{\mu}_{j,q}$	0	0	0	$z_{j,q} + \bar{\mu}_{j,q}$

vector	const^t	coef_i^t	const	coef_i	$\widetilde{\mathsf{sim}}$
$\tilde{\boldsymbol{v}}_{r,j,q}$	$\mathbf{L}^t_{r,j,q}[\mathsf{const}]$	$\mathbf{L}^t_{r,j,q}[\mathsf{coef}_i]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{const}]$	$\bar{\mathbf{L}}_{r,j,q}[\mathsf{coef}_i]$	$\mathbf{0}$

for post-challenge queries where $z_{j,q} = \boldsymbol{x}^*[j]$ if it is admissible and otherwise $z_{j,q} = 0$. The oracle corresponding to $\mathsf{KeyGen}_{\mathsf{Vec}}$ can be simulated by setting

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{\rho}_{j,q}$	$\boldsymbol{y}_q[j]$	0	0	$\bar{\rho}_{j,q}$

for pre-challenge queries and

vector	vec	$\mathbf{pad}$	sim_1	sim_2	sim	ext
$\boldsymbol{v}_{j,q}$	$\boldsymbol{y}_q[j]$	$\boldsymbol{\rho}_{j,q}$	0	0	0	$z_q \cdot \mathbf{1}[j=1] + \bar{\rho}_{j,q}$

for post-challenge queries.

References

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval, "Simple functional encryption schemes for inner products," PKC 2015, pages 733-751, 2015.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry, "Differing-inputs obfuscation and applications," Cryptology ePrint Archive, Paper 2013/689, 2013.

[ACGU20] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu, "Inner-product functional encryption with fine-grained access control," ASIACRYPT 2020, Part III, pages 467-497, 2020. (Full version: Cryptology ePrint Archive, Paper 2020/577)

[AGW20] Michel Abdalla, Junqing Gong, and Hoeteck Wee, "Functional encryption for attribute-weighted sums from k-Lin," CRYPTO 2020, Part I, pages 685-716, 2020.

[ATY23] Shweta Agrawal, Junichi Tomida, and Anshu Yadav, "Attribute-based multi-input FE (and more) for attribute-weighted sums," CRYPTO 2023, Part IV, pages 464-497, 2023.

[BCG+17] Zvika Brakerski, Nishanth Chandran, Vipul Goyal, Aayush Jain, Amit Sahai, and Gil Segev, "Hierarchical functional encryption," ITCS 2017, 2017.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy, "Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits," EUROCRYPT 2014, pages 533–556, 2014.

[BS25] Anushree Belel and Junji Shikata, "Efficient identity-based inner product functional encryption from RLWE," INDOCRYPT 2025, pages 285-311, 2025.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters, "Functional encryption: Definitions and challenges," TCC 2011, pages 253-273, 2011.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee, "Improved dual system ABE in prime-order groups via predicate encodings," EUROCRYPT 2015, Part II, pages 595-624, 2015.

[CGW18] Jie Chen, Junqing Gong, and Hoeteck Wee, "Improved inner-product encryption with adaptive security and full attribute-hiding," ASIACRYPT 2018, Part II, pages 673-702, 2018.

[CT26] Valerio Cini and Erkan Tairi, "(Fine-Grained) Unbounded Inner-Product Functional Encryption from LWE," Cryptology ePrint Archive, Paper 2026/062, 2026.

[DDM+23] Uddipana Dowerah, Subhranil Dutta, Aikaterini Mitrokotsa, Sayantan Mukherjee, and Tapas Pal, "Unbounded predicate inner product functional encryption from pairings," Journal of Cryptology 36.3: 29, 2023.

[DMPT25] Subhranil Dutta, Aikaterini Mitrokotsa, Tapas Pal, and Jenit Tomy, "Multi-client attribute-based unbounded inner product functional encryption, and more," PKC 2025, pages 3-37, 2025.

[DP21] Pratish Datta and Tapas Pal, "(Compact) adaptively secure FE for attribute-weighted sums from k-Lin," ASIACRYPT 2021, Part IV, pages 434-467, 2021.

[DP23] Pratish Datta and Tapas Pal, "Decentralized multi-authority attribute-based inner-product FE: large universe and unbounded," PKC 2023, Part I, pages 587-621, 2023.

[DPY24] Pratish Datta, Tapas Pal, and Shota Yamada, "Registered FE beyond predicates: (attribute-based) linear functions and more," ASIACRYPT 2024, Part I, pages 65-104, 2024.

[DST26] Pratish Datta, Robert Schädlich, and Erkan Tairi, "Decentralized multi-authority (attribute-based) traitor tracing," Cryptology ePrint Archive, Paper 2026/1268, 2026.

[EHK+17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar, "An algebraic framework for Diffie–Hellman assumptions," Journal of Cryptology 30.1, pages 242-288, 2017.

[ET25] Tenma Edamura, Atsushi Takayasu, "Tight adaptive simulation security for identity-based inner-product FE in the (quantum) random oracle model," PKC 2025, Part III, pages 3-36, 2025.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters, "Attribute-based encryption for fine-grained access control of encrypted data," ACMCCS 2006, pages 89-98, 2006.

[GS02] Craig Gentry and Alice Silverberg, "Hierarchical ID-based cryptography," ASIACRYPT 2002, pages 548-566, 2002.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee, "Attribute-based encryption for circuits," Journal of the ACM 62.6 pages 1-33, 2015.

[HL02] Jeremy Horwitz and Ben Lynn, "Toward hierarchical identity-based encryption," EUROCRYPT 2002, pages 466–481, 2002.

[IW14] Yuval Ishai and Hoeteck Wee, "Partial garbling schemes and their applications," ICALP 2014, pages 650-662, 2014.

[Lin17] Huijia Lin, "Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs," CRYPTO 2017, Part I, pages 599-629, 2017.

[LJ20] Roman Langrehr and Pan Jiaxin, "Tightly secure hierarchical identity-based encryption," Journal of Cryptology 33.4, pages 1787-1821, 2020.

[LL20] Huijia Lin and Ji Luo, "Compact adaptively secure ABE from k-Lin: beyond NC^1 and towards NL," EUROCRYPT 2020, Part III, pages 247-277, 2020.

[LLW21] Qiqi Lai, Feng-Hao Liu, and Zhedong Wang, "New lattice two-stage sampling technique and its applications to functional encryption–stronger security and smaller ciphertexts," EUROCRYPT 2021, Part I, pages 498-527, 2021.

[LV16] Huijia Lin and Vinod Vaikuntanathan, "Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings," IEEE FOCS 2016, pages 11-20, 2016.

[LW11] Allison Lewko and Brent Waters , "Unbounded HIBE and attribute-based encryption," EUROCRYPT 2011, pages 547-567, 2011.

[NPP22] Ky Nguyen, Duong Hieu Phan, and David Pointcheval, "Multi-client functional encryption with fine-grained access control," ASIACRYPT 2022, Part I, pages 95-125, 2022.

[NPP25] Ky Nguyen, Duong Hieu Phan, and David Pointcheval, "Multi-client functional encryption with public inputs and strong security," PKC 2025, Part III, pages 68-101, 2025.

[NPS25] Ky Nguyen, David Pointcheval, and Robert Schädlich, "Dynamic decentralized functional encryption: generic constructions with strong security," PKC 2025, Part IV, pages 344-379, 2025.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima, "Hierarchical predicate encryption for inner-products," ASIACRYPT 2009, pages 214-231, 2009.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," CRYPTO 2010, pages 191-208, 2010.

[O'N10] Adam O'Neill, "Definitional issues in functional encryption," Cryptology ePrint Archive, Paper 2010/556, 2010.

[Wee14] Hoeteck Wee, "Dual system encryption via predicate encodings," TCC 2014, pages 616-637, 2014.

[ZZZ+25] Ziqi Zhu, Jun Zhao, Kai Zhang, Junqing Gong, and Haifeng Qian, "When KGC meets curator: new paradigm of registered ABE and FE," ASIACRYPT 2025, Part VI, pages 35-67, 2025.