Hindawi
Security and Communication Networks
Volume 2017, Article ID 2148534, 27 pages
https://doi.org/10.1155/2017/2148534

Research Article

Efficient KDM-CCA Secure Public-Key Encryption via Auxiliary-Input Authenticated Encryption

Shuai Han,[1,2] Shengli Liu,[1,2,3] and Lin Lyu[1,2]

[1] *Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[2] *State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China*
[3] *Westone Cryptologic Research Center, Beijing 100070, China*

Correspondence should be addressed to Shengli Liu; slliu@sjtu.edu.cn

Received 1 April 2017; Revised 13 June 2017; Accepted 6 July 2017; Published 11 December 2017

Academic Editor: Muhammad Khurram Khan

KDM[$\mathcal{F}$]-CCA security of public-key encryption (PKE) ensures the privacy of key-dependent messages $f(\mathsf{sk})$ which are closely related to the secret key sk, where $f \in \mathcal{F}$, even if the adversary is allowed to make decryption queries. In this paper, we study the design of KDM-CCA secure PKE. To this end, we develop a new primitive named *Auxiliary-Input Authenticated Encryption* (AIAE). For AIAE, we introduce two related-key attack (RKA) security notions, including *IND-RKA* and *weak-INT-RKA*. We present a generic construction of AIAE from tag-based hash proof system (HPS) and one-time secure authenticated encryption (AE) and give an instantiation of AIAE under the Decisional Diffie-Hellman (DDH) assumption. Using AIAE as an essential building block, we give two constructions of efficient KDM-CCA secure PKE based on the DDH and the Decisional Composite Residuosity (DCR) assumptions. Specifically, (i) our first PKE construction is the first one achieving KDM[$\mathcal{F}_{\text{aff}}$]-CCA security for the set of affine functions and compactness of ciphertexts simultaneously. (ii) Our second PKE construction is the first one achieving KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security for the set of polynomial functions and almost compactness of ciphertexts simultaneously. Our PKE constructions are very efficient; in particular, they are pairing-free and NIZK-free.

1. Introduction

For public-key encryption (PKE) schemes, Chosen-Ciphertext Attack (CCA) security is the de facto security notion. In the CCA security model, the adversary sees the public key and gets challenge ciphertexts, which are encryptions of messages of its choices. It is also allowed to make decryption queries and obtain the decrypted messages for ciphertexts (but not the challenge ciphertexts) of its choices. CCA security considers whether the challenge ciphertexts can protect the security of messages. Observe that the adversary does not know the secret keys; thus it is not able to submit messages that are closely related to the secret keys. Thus, there is a corner that is not covered by CCA security, that is, the security of messages which are closely dependent on the secret keys. It was Goldwasser and Micali [1] who first pointed out this problem. In 2002, the security of such key-dependent messages (KDM) was formalized by Black et al. [2]. Up to now, KDM-security has found many applications, such as anonymous credential systems [3] and hard disk encryption [4].

KDM[$\mathcal{F}$]-security means KDM-security for a set $\mathcal{F}$ of functions. Loosely speaking, in the n-KDM[$\mathcal{F}$]-security model, the adversary obtains public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ of n users and has access to an encryption oracle. Each time, the adversary submits a function f in the function set $\mathcal{F}$, the encryption oracle will encrypt $f(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ or a dummy message (say 0) and output the challenge ciphertext to the adversary. The n-KDM[$\mathcal{F}$]-CPA security stipulates that the adversary cannot distinguish the two cases, and the n-KDM[$\mathcal{F}$]-CCA security demands the indistinguishability of the two cases even if the adversary is also allowed to make decryption queries. KDM-CCA is obviously stronger than KDM-CPA security notion. Moreover, the KDM-security is stronger when the function set $\mathcal{F}$ is larger.

KDM[$\mathcal{F}$]-CPA Security. In 2008, Boneh et al. (BHHO) [4] proposed the first KDM[$\mathcal{F}_{\text{aff}}$]-CPA secure PKE construction for the affine function set $\mathcal{F}_{\text{aff}}$, from the Decisional Diffie-Hellman (DDH) assumption. Soon after, the BHHO scheme was generalized by Brakerski and Goldwasser [5], who presented KDM[$\mathcal{F}_{\text{aff}}$]-CPA secure PKE constructions under the Quadratic Residuosity (QR) assumption or the Decisional Composite Residuosity (DCR) assumption. However, these schemes suffer from incompact ciphertext, which contains $O(\lambda)$ group elements (λ denotes the security parameter throughout the paper).

Applebaum et al. [6] proved that a variant of the Regev scheme [7] is KDM[$\mathcal{F}_{\text{aff}}$]-CPA secure and enjoys compact ciphertexts, that is, encompassing only $O(1)$ group elements.

Brakerski et al. [8] provided a KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA secure PKE scheme for the polynomial function set $\mathcal{F}^d_{\text{poly}}$, which contains all polynomials whose degrees are at most d. The drawback of the scheme is incompact ciphertext, which contains $O(\lambda^{d+1})$ group elements.

Barak et al. [9] presented a KDM-CPA secure PKE for the set of Boolean circuits whose sizes are a priori bounded, which is a very large function set. Nevertheless, their scheme is neither practical nor flexible.

In 2011, Malkin et al. [10] proposed the first efficient KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA secure PKE. The ciphertext of their PKE construction is almost compact and consists of only $O(d)$ group elements.

KDM[$\mathcal{F}$]-CCA Security. The first approach to KDM-CCA security was proposed by Camenisch, Chandran, and Shoup (CCS) [11]. The CCS approach follows the Naor-Yung paradigm [12], and the building blocks are a PKE scheme with CCA security, a PKE scheme with KDM-CPA security, and a noninteractive zero-knowledge (NIZK) proof system which proves that the two PKE schemes encrypt the same message.

The Groth-Sahai proofs [13] are the only practical NIZK. To obtain efficient KDM-CCA secure PKE, we have to employ an efficient PKE scheme with KDM-CPA security and the Groth-Sahai proofs if we follow the CCS approach [11]. Unfortunately, the existing efficient PKE schemes with KDM-CPA security, like [6, 10], are not compatible with the Groth-Sahai proofs, since the underlying groups of their schemes are not pairing-friendly ones.

Galindo et al. [14] proposed a KDM-CCA secure PKE scheme from the Matrix Decisional Diffie-Hellman assumption. Their scheme enjoys compact ciphertexts, but the KDM-CCA security of their scheme is constrained (more precisely, in their KDM-CCA security model, the adversary is only allowed to have access to the encryption oracle for a number of times linear in the secret key's size).

In order to achieve both KDM-CCA security and efficiency for PKE, Hofheinz [15] developed another approach, making use of a novel primitive named "lossy algebraic filter." The PKE scheme proposed by Hofheinz enjoys the security of KDM[$\mathcal{F}_{\text{circ}}$]-CCA and the compactness of ciphertexts simultaneously, but the function set $\mathcal{F}_{\text{circ}}$ is made up of constant functions and selection functions $f(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) = \mathsf{sk}_i$.

In fact, it is a challenging job to enlarge the KDM-CCA function set $\mathcal{F}$ while keeping the efficiency of the PKE scheme. Recently, Lu et al. [16] designed the first PKE achieving both KDM[$\mathcal{F}_{\text{aff}}$]-CCA security and compact ciphertexts. Their construction is referred to as the LLJ scheme in this paper. The essential building block in their scheme is "authenticated encryption" ($\overline{\mathsf{AE}}$). The so-called INT-$\mathcal{F}_{\text{aff}}$-RKA security of $\overline{\mathsf{AE}}$ turns out to be critical to the KDM[$\mathcal{F}_{\text{aff}}$]-CCA security of the LLJ scheme. Unfortunately, their security reduction of the INT-$\mathcal{F}_{\text{aff}}$-RKA security of $\overline{\mathsf{AE}}$ to the underlying DDH assumption is flawed. Roughly speaking, the problem of their security reduction is that there is no efficient way for the DDH adversary to convert the forgery provided by the INT-$\mathcal{F}_{\text{aff}}$-RKA adversary to a decision bit for solving the DDH problem, since it has no trapdoor. See our conference version [17] for details. The failure of $\overline{\mathsf{AE}}$'s INT-$\mathcal{F}_{\text{aff}}$-RKA security reduction directly affects the validity of LLJ's KDM[$\mathcal{F}_{\text{aff}}$]-CCA security proof.

To construct efficient KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA secure PKE schemes, the CCS approach [11] is the unique way, to the best of our knowledge. However, the only efficient KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA secure PKE [10] is incompatible with the Groth-Sahai NIZK proofs [13]; thus the CCS approach must adopt a general inefficient NIZK.

Our Contribution. In this work, we focus on the design of efficient PKE schemes possessing KDM[$\mathcal{F}_{\text{aff}}$]-CCA security and KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security, respectively.

(i) We develop a new primitive named *"Auxiliary-Input Authenticated Encryption"* (AIAE). We introduce new related-key attack (RKA) security notions for it, called *IND-$\mathcal{F}'$-RKA* and *weak-INT-$\mathcal{F}'$-RKA*.

  (a) We show a general paradigm for constructing such an AIAE from a one-time secure AE and a *tag-based hash proof system* (HPS) that is universal$_2$, extracting, and key-homomorphic.

  (b) We present an instantiation of tag-based HPS under the DDH assumption. Following our paradigm, we immediately obtain a DDH-based AIAE for the set of restricted affine functions.

(ii) Using AIAE as an essential building block, we design the first PKE scheme enjoying KDM[$\mathcal{F}_{\text{aff}}$]-CCA security and compactness of ciphertexts simultaneously. Specifically, the ciphertext of our scheme contains only $O(1)$ group elements.

(iii) Furthermore, we design the first PKE scheme enjoying KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security and almost compactness of ciphertexts simultaneously. More precisely, the number of group elements contained in a ciphertext is independent of the security parameter λ.

In Table 1, we list the existing PKE schemes which either achieve KDM-CCA security or are KDM-secure for the set $\mathcal{F}^d_{\text{poly}}$ of polynomial functions.

Table 1: Comparison among PKE schemes achieving either KDM-CCA security or security against the set $\mathcal{F}^d_{\text{poly}}$ of polynomial functions. Here, we denote by λ the security parameter and by $\mathcal{F}_{\text{circ}}$, $\mathcal{F}_{\text{aff}}$, and $\mathcal{F}^d_{\text{poly}}$ the set of selection functions, the set of affine functions, and the set of polynomial functions of bounded degree d, respectively. "CCA" indicates that the scheme is KDM-CCA secure. By the symbol "**?**", we mean that the security proof is not rigorous. $\mathbb{G}$, $\mathbb{Z}_{N^2}$, $\mathbb{Z}_{N^3}$, $\mathbb{Z}_{N^s}$, and $\mathbb{Z}_{\overline{N}}$ are the underlying groups, where $s \geq 1$.

Scheme	Set	CCA?	Free of pairing?	The size of ciphertext	Assumption
BHHO08 [4] + CCS09 [11]	$\mathcal{F}_{\text{aff}}$	√	—	$(6\lambda + 13)\lvert\mathbb{G}\rvert$	DDH
BGK11 [8]	$\mathcal{F}^d_{\text{poly}}$	—	√	$(\lambda^{d+1})\lvert\mathbb{G}\rvert$	DDH or LWE
MTY11 [10]	$\mathcal{F}^d_{\text{poly}}$	—	√	$(d+2)\lvert\mathbb{Z}_{N^s}\rvert$	DCR
Hof13 [15]	$\mathcal{F}_{\text{circ}}$	√	—	$6\lvert\mathbb{Z}_{N^3}\rvert + 49\lvert\mathbb{G}\rvert$	DDH & DCR
LLJ15 [16]	$\mathcal{F}_{\text{aff}}$	**?**	√	$3\lvert\mathbb{Z}_{N^2}\rvert + 3\lvert\mathbb{Z}_{N^s}\rvert + \lvert\mathbb{Z}_{\overline{N}}\rvert$	DDH & DCR
Our scheme in Section 4	$\mathcal{F}_{\text{aff}}$	√	√	$9\lvert\mathbb{Z}_{N^2}\rvert + 9\lvert\mathbb{Z}_{N^s}\rvert + 2\lvert\mathbb{Z}_{\overline{N}}\rvert$	DDH & DCR
Our scheme in Section 5	$\mathcal{F}^d_{\text{poly}}$	√	√	$9\lvert\mathbb{Z}_{N^2}\rvert + (8d^9+1)\lvert\mathbb{Z}_{N^s}\rvert + 2\lvert\mathbb{Z}_{\overline{N}}\rvert$	DDH & DCR

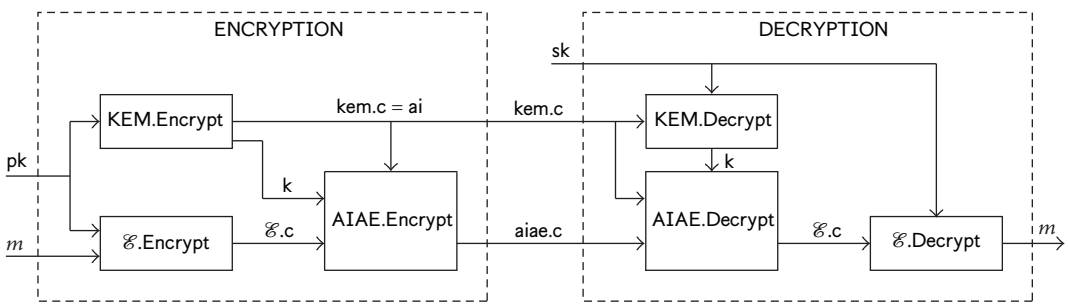

Figure 1: Our approach of PKE construction.

Overview of Our Construction. In the construction of our KDM-CCA secure PKE schemes, we adopt a key encapsulation mechanism (KEM) + data encapsulation mechanism (DEM) approach [18] and employ three building blocks: KEM, $\mathcal{E}$, and AIAE, as shown in Figure 1.

(i) KEM and $\mathcal{E}$ share the same pair of public and secret keys.

(ii) A key k is encapsulated by KEM.Encrypt, and an encapsulation kem.c is generated by KEM.Encrypt along the way.

(iii) The message m is encrypted by $\mathcal{E}$.Encrypt, and the resulting $\mathcal{E}$-ciphertext is $\mathcal{E}$.c.

(iv) The key k generated by KEM is used by AIAE.Encrypt to encrypt $\mathcal{E}$.c with auxiliary input ai $:=$ kem.c, and the resulting AIAE-ciphertext is aiae.c.

(v) The ciphertext of our PKE scheme is (kem.c, aiae.c).

Following this approach, we design KDM[$\mathcal{F}_{\text{aff}}$]-CCA and KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA secure PKE schemes, respectively, by constructing specific building blocks.

Differences to Conference Version. This paper constitutes an extended full version of [17]. The new results in this paper are as follows.

(i) In contrast to presenting a concrete construction of AIAE in the conference paper, we give a general paradigm for constructing AIAE from a one-time secure authenticated encryption (AE) and a *tag-based hash proof system* (HPS) in this paper.

(a) In Section 3.2, we show that the resulting AIAE is IND-RKA secure and weak-INT-RKA secure, as long as the underlying tag-based HPS is universal$_2$, extracting, and key-homomorphic.

(b) In Section 3.3, we give an instantiation of tag-based HPS based on the DDH assumption. Following our paradigm, we obtain a DDH-based AIAE scheme in Section 3.4.

We view the specific AIAE proposed in the conference paper as an instantiation of the general paradigm presented in this paper.

(ii) In this paper, we provide the full proofs of the theorems regarding the KDM[$\mathcal{F}_{\text{aff}}$]-CCA security and KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security of our PKEs. Compared with the conference paper, we add the proofs of Lemmas 16, 18, 25, 26, and 29, and the proof of indistinguishability between Hybrids 2 and 3 in Section 5.3.

2. Preliminaries

Throughout this paper, denote by $\lambda \in \mathbb{N}$ the security parameter. $y \leftarrow_{\$} \mathcal{Y}$ means choosing an element y from set $\mathcal{Y}$ uniformly. $y \leftarrow_{\$} \mathcal{A}(x; r)$ means executing algorithm $\mathcal{A}$ with

Proc. INITIALIZE:	**Proc.** ENCRYPT($f \in \mathcal{F}, i \in [n]$):	**Proc.** DECRYPT(pke.c, $i \in [n]$):
$\mathsf{pars} \leftarrow_\$ \mathsf{ParGen}(1^\lambda)$.	$m_1 := f(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.	If (pke.c, i) $\in \mathcal{Q}_{\mathcal{ENC}}$,
For $i \in [n]$,	$m_0 := 0^{\lvert m_1 \rvert}$.	Output $\perp$.
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_\$ \mathsf{KeyGen}(\mathsf{pars})$.	$\mathsf{pke.c} \leftarrow_\$ \mathsf{Encrypt}(\mathsf{pk}_i, m_\beta)$.	Output $\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{pke.c})$.
$\beta \leftarrow_\$ \{0, 1\}$.	$\mathcal{Q}_{\mathcal{ENC}} := \mathcal{Q}_{\mathcal{ENC}} \cup \{(\mathsf{pke.c}, i)\}$.	**Proc.** FINALIZE(β'):
Output $(\mathsf{pars}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$.	Output pke.c.	Output $(\beta' = \beta)$.

FIGURE 2: n-KDM[$\mathcal{F}$]-CCA security game.

input x and randomness r and assigning output to y. We sometimes abbreviate this to $y \leftarrow_\$ \mathcal{A}(x)$. "PPT" is short for probabilistic polynomial-time. For integers $n < m$, we denote $[n] := \{1, 2, \ldots, n\}$ and $[n, m] := \{n, n+1, \ldots, m\}$. For a security notion YY and a primitive XX, the advantage of a PPT adversary $\mathcal{A}$ is typically denoted by $\mathsf{Adv}^{YY}_{XX,\mathcal{A}}(\lambda)$ and we denote $\mathsf{Adv}^{YY}_{XX}(\lambda) := \max_{\mathrm{PPT}\mathcal{A}} \mathsf{Adv}^{YY}_{XX,\mathcal{A}}(\lambda)$. Let $\mathsf{negl}(\cdot)$ denote an unspecified negligible function.

Games. We will use games in our security definitions and proofs. Typically, a game G begins with an INITIALIZE procedure and ends with a FINALIZE procedure. In the game, there might be other procedures $\mathrm{PROC}_1, \ldots, \mathrm{PROC}_n$ which perform as oracles. All procedures are presented with pseudocode, all sets are initialized as empty sets, and all variables are initialized as empty strings. In the execution of a game G with an adversary $\mathcal{A}$, firstly $\mathcal{A}$ calls INITIALIZE and obtains its output; then $\mathcal{A}$ makes arbitrary oracle queries to PROC_i according to their specifications and obtains their outputs; finally $\mathcal{A}$ calls FINALIZE. In the end of the execution, if FINALIZE outputs b, then we write this as $\mathsf{G}^{\mathcal{A}} \Rightarrow b$. The statement $a \overset{\mathsf{G}}{=} b$ means that, in game G, a is computed as b or a equals b.

2.1. Public-Key Encryption. There are four PPT algorithms $\mathsf{PKE} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ in a public-key encryption (PKE) scheme:

(i) $\mathsf{ParGen}(1^\lambda)$ outputs a public parameter pars. We assume that pars implicitly defines a secret key space $\mathcal{SK}$ and a message space $\mathcal{M}$.

(ii) $\mathsf{KeyGen}(\mathsf{pars})$ takes pars as input and outputs a public key pk and a secret key sk.

(iii) $\mathsf{Encrypt}(\mathsf{pk}, m)$ takes pk and a message $m \in \mathcal{M}$ as input and outputs a ciphertext pke.c.

(iv) $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{pke.c})$ takes sk and a ciphertext pke.c as input and outputs either a message m or a symbol $\perp$ indicating the failure of the decryption.

We require PKE to have perfect correctness; that is, for all possible $\mathsf{pars} \leftarrow_\$ \mathsf{ParGen}(1^\lambda)$ and all $m \in \mathcal{M}$, we have

$$\Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KeyGen}(\mathsf{pars}) : \mathsf{Decrypt}(\mathsf{sk}, \mathsf{Encrypt}(\mathsf{pk}, m)) = m\right] = 1. \quad (1)$$

Definition 1 (KDM[$\mathcal{F}$]-CCA security). Let $n \in \mathbb{N}$ and let $\mathcal{F}$ denote a set of functions from $(\mathcal{SK})^n$ to $\mathcal{M}$. A scheme PKE is n-KDM[$\mathcal{F}$]-CCA secure, if for any PPT adversary $\mathcal{A}$, we have $\mathsf{Adv}^{\text{kdm-cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \lvert \Pr[n\text{-}\mathsf{KDM}[\mathcal{F}]\text{-}\mathsf{CCA}^{\mathcal{A}} \Rightarrow 1] - 1/2 \rvert \leq \mathsf{negl}(\lambda)$, where n-KDM[$\mathcal{F}$]-CCA is the security game shown in Figure 2.

2.2. Authenticated Encryption. There are three PPT algorithms $\mathsf{AE} = (\mathsf{AE.ParGen}, \mathsf{AE.Encrypt}, \mathsf{AE.Decrypt})$ in an authenticated encryption (AE) scheme:

(i) $\mathsf{AE.ParGen}(1^\lambda)$ generates a system parameter $\mathsf{pars}_{\mathsf{AE}}$. We require $\mathsf{pars}_{\mathsf{AE}}$ to be an implicit input to other algorithms and assume that $\mathsf{pars}_{\mathsf{AE}}$ implicitly defines a key space $\mathcal{K}_{\mathsf{AE}}$ and a message space $\mathcal{M}$.

(ii) $\mathsf{AE.Encrypt}(\mathsf{k}, m)$ takes a key $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$ and a message $m \in \mathcal{M}$ as input and outputs a ciphertext ae.c.

(iii) $\mathsf{AE.Decrypt}(\mathsf{k}, \mathsf{ae.c})$ takes a key $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$ and a ciphertext ae.c as input and outputs a message $m \in \mathcal{M}$ or a symbol $\perp$.

We require AE to have perfect correctness; that is, for all possible $\mathsf{pars}_{\mathsf{AE}} \leftarrow_\$ \mathsf{AE.ParGen}(1^\lambda)$, all keys $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$, and all $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{AE.Decrypt}(\mathsf{k}, \mathsf{AE.Encrypt}(\mathsf{k}, m)) = m\right] = 1. \quad (2)$$

Definition 2 (one-time security). A scheme AE is one-time secure (OT-secure), that is, IND-OT and INT-OT secure, if for any PPT $\mathcal{A}$, both $\mathsf{Adv}^{\text{ind-ot}}_{\mathsf{AE},\mathcal{A}}(\lambda) := \lvert \Pr[\mathsf{IND\text{-}OT}^{\mathcal{A}} \Rightarrow 1] - 1/2 \rvert \leq \mathsf{negl}(\lambda)$ and $\mathsf{Adv}^{\text{int-ot}}_{\mathsf{AE},\mathcal{A}}(\lambda) := \Pr[\mathsf{INT\text{-}OT}^{\mathcal{A}} \Rightarrow 1] \leq \mathsf{negl}(\lambda)$, where IND-OT and INT-OT are the security games presented in Figure 3.

2.3. Key Encapsulation Mechanism. There are three PPT algorithms $\mathsf{KEM} = (\mathsf{KEM.KeyGen}, \mathsf{KEM.Encrypt}, \mathsf{KEM.Decrypt})$ in a key encapsulation mechanism (KEM):

(i) $\mathsf{KEM.KeyGen}(1^\lambda)$ generates a public key pk and a secret key sk.

(ii) $\mathsf{KEM.Encrypt}(\mathsf{pk})$ takes pk as input and outputs a key k together with a ciphertext kem.c.

(iii) $\mathsf{KEM.Decrypt}(\mathsf{sk}, \mathsf{kem.c})$ takes sk and a ciphertext kem.c as input and outputs either a key k or a symbol $\perp$.

Proc. INITIALIZE:
$\mathsf{pars}_{\mathsf{AE}} \leftarrow_{\$} \mathsf{AE.ParGen}(1^{\lambda})$, $\mathsf{k} \leftarrow_{\$} \mathscr{K}_{\mathsf{AE}}$.
$\beta \leftarrow_{\$} \{0,1\}$.
Output $\mathsf{pars}_{\mathsf{AE}}$.

Proc. ENCRYPT(m_0, m_1): // one query
If $|m_0| \neq |m_1|$, Output $\perp$.
$\mathsf{ae.c} \leftarrow_{\$} \mathsf{AE.Encrypt}(\mathsf{k}, m_{\beta})$.
Output ae.c.

Proc. FINALIZE(β'):
Output $(\beta' = \beta)$.

(a)

Proc. INITIALIZE:
$\mathsf{pars}_{\mathsf{AE}} \leftarrow_{\$} \mathsf{AE.ParGen}(1^{\lambda})$, $\mathsf{k} \leftarrow_{\$} \mathscr{K}_{\mathsf{AE}}$.
Output $\mathsf{pars}_{\mathsf{AE}}$.

Proc. ENCRYPT(m): // one query
$\mathsf{ae.c} \leftarrow_{\$} \mathsf{AE.Encrypt}(\mathsf{k}, m)$.
Output ae.c.

Proc. FINALIZE$(\mathsf{ae.c}^*)$:
If $\mathsf{ae.c}^* = \mathsf{ae.c}$, Output 0.
Output $(\mathsf{AE.Decrypt}(\mathsf{k}, \mathsf{ae.c}^*) \neq \perp)$.

(b)

FIGURE 3: IND-OT (a) and INT-OT (b) security games.

We require KEM to have perfect correctness; that is, for all possible $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KEM.KeyGen}(1^{\lambda})$, we have

$$\Pr\left[(\mathsf{k}, \mathsf{kem.c}) \longleftarrow_{\$} \mathsf{KEM.Encrypt}(\mathsf{pk}) : \mathsf{KEM.Decrypt}(\mathsf{sk}, \mathsf{kem.c}) = \mathsf{k}\right] = 1. \quad (3)$$

2.4. Tag-Based Hash Proof System: Universal$_2$, Extracting, and Key-Homomorphism. Tag-based hash proof system (HPS) was first defined in [19]. The definition is similar to extended HPS [20], but the universal$_2$ property is slightly different.

Definition 3 (tag-based hash proof system). A tag-based hash proof system $\mathsf{THPS} = (\mathsf{THPS.Setup}, \mathsf{THPS.Pub}, \mathsf{THPS.Priv})$ is comprised of three PPT algorithms:

(i) $\mathsf{THPS.Setup}(1^{\lambda})$ outputs a parameterized instance $\mathsf{pars}_{\mathsf{THPS}}$, which implicitly defines $(\mathscr{K}, \mathscr{C}, \mathscr{V}, \mathscr{T}, \mathscr{HK}, \mathscr{PK}, \Lambda_{(\cdot)}, \mu)$, where $\mathscr{K}, \mathscr{C}, \mathscr{V}, \mathscr{T}, \mathscr{HK}, \mathscr{PK}$ are all finite sets with $\mathscr{V} \subseteq \mathscr{C}$, $\Lambda_{(\cdot)} : \mathscr{C} \times \mathscr{T} \to \mathscr{K}$ is a set of hash functions indexed by $\mathsf{hk} \in \mathscr{HK}$, and $\mu : \mathscr{HK} \to \mathscr{PK}$ is a function. We assume that μ is efficiently computable, and there are PPT algorithms sampling $\mathsf{hk} \leftarrow_{\$} \mathscr{HK}$ uniformly, sampling $C \leftarrow_{\$} \mathscr{C}$ uniformly, sampling $C \leftarrow_{\$} \mathscr{V}$ uniformly with a witness w, and checking membership in $\mathscr{C}$.

(ii) $\mathsf{THPS.Pub}(\mathsf{pk}, C, w, t)$ takes a projection key $\mathsf{pk} = \mu(\mathsf{hk}) \in \mathscr{PK}$, an element $C \in \mathscr{V}$ with a witness w, and a tag $t \in \mathscr{T}$ as input and outputs a hash value $K = \Lambda_{\mathsf{hk}}(C, t) \in \mathscr{K}$.

(iii) $\mathsf{THPS.Priv}(\mathsf{hk}, C, t)$ takes a hashing key $\mathsf{hk} \in \mathscr{HK}$, an element $C \in \mathscr{C}$, and a tag $t \in \mathscr{T}$ as input and outputs a hash value $K = \Lambda_{\mathsf{hk}}(C, t) \in \mathscr{K}$ without knowing a witness.

We require THPS to be *projective*; that is, for all $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$, all $\mathsf{hk} \in \mathscr{HK}$ and $\mathsf{pk} = \mu(\mathsf{hk}) \in \mathscr{PK}$, all $C \in \mathscr{V}$ with all witnesses w and all $t \in \mathscr{T}$, it holds that

$$\begin{aligned} \mathsf{THPS.Pub}(\mathsf{pk}, C, w, t) &= \Lambda_{\mathsf{hk}}(C, t) \\ &= \mathsf{THPS.Priv}(\mathsf{hk}, C, t). \end{aligned} \quad (4)$$

Tag-based HPS is associated with a subset membership problem. Informally speaking, it asks to distinguish the uniform distribution over $\mathscr{V}$ from the uniform distribution over $\mathscr{C} \setminus \mathscr{V}$.

Definition 4 (SMP). The Subset Membership Problem (SMP) related to THPS is hard, if for any PPT adversary $\mathscr{A}$, one has

$$\begin{aligned} \mathsf{Adv}^{\mathsf{smp}}_{\mathsf{THPS},\mathscr{A}}(\lambda) := \Big| &\Pr\left[\mathscr{A}\left(\mathsf{pars}_{\mathsf{THPS}}, C\right) = 1\right] \\ &- \Pr\left[\mathscr{A}\left(\mathsf{pars}_{\mathsf{THPS}}, C'\right) = 1\right]\Big| \leq \mathsf{negl}(\lambda), \end{aligned} \quad (5)$$

where $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$, $C \leftarrow_{\$} \mathscr{V}$, and $C' \leftarrow_{\$} \mathscr{C} \setminus \mathscr{V}$.

Definition 5 (universal$_2$). THPS is called (strongly) universal$_2$, if for all possible $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$, all $\mathsf{pk} \in \mathscr{PK}$, all $C \in \mathscr{C}$, all $C' \in \mathscr{C} \setminus \mathscr{V}$, all $t, t' \in \mathscr{T}$ with $t \neq t'$, and all $K, K' \in \mathscr{K}$, it holds that

$$\begin{aligned} &\Pr\left[\Lambda_{\mathsf{hk}}\left(C', t'\right) = K' \mid \mu(\mathsf{hk}) = \mathsf{pk}, \Lambda_{\mathsf{hk}}(C, t) = K\right] \\ &\quad = \frac{1}{|\mathscr{K}|}, \end{aligned} \quad (6)$$

where the probability is over $\mathsf{hk} \leftarrow_{\$} \mathscr{HK}$.

The key difference between tag-based HPS and extended HPS lies in the definition of the universal$_2$ property [19]. Extended HPS requires (6) to hold for $(C, t) \neq (C', t')$, while tag-based HPS requires (6) to hold only for $t \neq t'$. Hence,

any (universal$_2$) extended HPS is also a (universal$_2$) tag-based HPS, but not vice versa. Tag-based HPS is essentially a weaker variant of extended HPS and admits more efficient constructions.

Dodis et al. [21] defined an extracting property for extended HPS, which requires the hash value $\Lambda_{\mathsf{hk}}(C,t)$ to be uniformly distributed over $\mathscr{K}$ for any $C \in \mathscr{C}$ and $t \in \mathscr{T}$, as long as hk is randomly chosen from $\mathscr{HK}$. Besides, Xagawa [22] considered a key-homomorphic property for extended HPS, which stipulates that $\Lambda_{\mathsf{hk}+\Delta}(C,t) = \Lambda_{\mathsf{hk}}(C,t) \cdot \Lambda_{\Delta}(C,t)$ holds for any $\mathsf{hk}, \Delta \in \mathscr{HK}$, $C \in \mathscr{C}$, and $t \in \mathscr{T}$. Here we adapt these notions to tag-based HPS.

Definition 6 (extracting). THPS is called extracting, if for all $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$, all $C \in \mathscr{C}$, all $t \in \mathscr{T}$, and all $K \in \mathscr{K}$, it holds that

$$\Pr\left[\Lambda_{\mathsf{hk}}(C,t) = K\right] = \frac{1}{|\mathscr{K}|}, \tag{7}$$

where $\mathsf{hk} \leftarrow_{\$} \mathscr{HK}$.

Definition 7 (key-homomorphism). THPS is called *key-homomorphic*, if for all $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$, which defines $(\mathscr{K}, \mathscr{C}, \mathscr{V}, \mathscr{T}, \mathscr{HK}, \mathscr{PK}, \Lambda_{(\cdot)}, \mu)$, one has the following:

(i) Both $(\mathscr{HK}, +)$ and $(\mathscr{K}, \cdot)$ are groups.

(ii) For all $C \in \mathscr{C}$ and all $t \in \mathscr{T}$, the mapping $\Lambda_{(\cdot)}(C,t) : \mathscr{HK} \to \mathscr{K}$ is a group homomorphism. That is, for all $\mathsf{hk}, \mathsf{b} \in \mathscr{HK}$ and all $a \in \mathbb{Z}$, it holds that $\Lambda_{a\cdot\mathsf{hk}+\mathsf{b}}(C,t) = (\Lambda_{\mathsf{hk}}(C,t))^a \cdot \Lambda_{\mathsf{b}}(C,t)$.

2.5. DCR, DDH, DL, and IV$_d$ Assumptions. Suppose that $\mathsf{GenN}(1^{\lambda})$ is a PPT algorithm generating $(p, q, N, \overline{N})$, where p, q are safe primes of λ-bit, $N = pq$, and $\overline{N} = 2N + 1$ is a prime. We define the following:

(i) $\mathbb{QR}_{\overline{N}} := \{a^2 \bmod \overline{N} \mid a \in \mathbb{Z}_{\overline{N}}\}$.

Then $\mathbb{QR}_{\overline{N}}$ is a cyclic group of order N. For $s \in \mathbb{N}$ and $T = 1 + N$, we define

(i) $\mathbb{QR}_{N^s} := \{a^2 \bmod N^s \mid a \in \mathbb{Z}^*_{N^s}\}$,

(ii) $\mathbb{SCR}_{N^s} := \{a^{2N^{s-1}} \bmod N^s \mid a \in \mathbb{Z}^*_{N^s}\}$,

(iii) $\mathbb{RU}_{N^s} := \{T^r \bmod N^s \mid r \in [N^{s-1}]\}$.

Then $\mathbb{SCR}_{N^s}$ is a cyclic group of order $\phi(N)/4$, and $\mathbb{QR}_{N^s} = \mathbb{SCR}_{N^s} \otimes \mathbb{RU}_{N^s}$, where $\otimes$ represents the internal direct product.

Damgård and Jurik [23] showed that the discrete logarithm $d\log_T(u) \in [N^{s-1}]$ of an element $u \in \mathbb{RU}_{N^s}$ can be efficiently computed from u and N. Observe that $\mathbb{Z}^*_{N^s} = \mathbb{Z}_2 \otimes \mathbb{Z}'_2 \otimes \mathbb{SCR}_{N^s} \otimes \mathbb{RU}_{N^s}$; thus for any $v = v(\mathbb{Z}_2) \cdot v(\mathbb{Z}'_2) \cdot v(\mathbb{SCR}_{N^s}) \cdot T^x \in \mathbb{Z}^*_{N^s}$, we have $v^{\phi(N)} = T^{x\cdot\phi(N)} \in \mathbb{RU}_{N^s}$ and

$$\frac{d\log_T\left(v^{\phi(N)}\right)}{\phi(N)} \bmod N^{s-1} = x. \tag{8}$$

Definition 8 (DCR assumption). The Decisional Composite Residuosity (DCR) assumption holds for GenN and $\mathbb{QR}_{N^s}$, if for any PPT $\mathscr{A}$, it holds that

$$\begin{aligned}&\mathsf{Adv}^{\mathrm{dcr}}_{\mathsf{GenN},\mathscr{A}}(\lambda)\\ &\quad := \left|\Pr\left[\mathscr{A}(N,u) = 1\right] - \Pr\left[\mathscr{A}(N,v) = 1\right]\right|\\ &\quad \le \mathsf{negl}(\lambda),\end{aligned} \tag{9}$$

where $(p, q, N, \overline{N}) \leftarrow_{\$} \mathsf{GenN}(1^{\lambda})$, $u \leftarrow_{\$} \mathbb{QR}_{N^s}$, and $v \leftarrow_{\$} \mathbb{SCR}_{N^s}$.

The Interactive Vector (IV$_d$) assumption is implied by the DCR assumption, as shown in [5]. Here we recall the IV$_d$ assumption according to [16].

Definition 9 (IV$_d$ assumption). The IV$_d$ assumption holds for GenN and $\mathbb{QR}_{N^s}$, if for any PPT $\mathscr{A}$, it holds that

$$\begin{aligned}&\mathsf{Adv}^{\mathrm{iv}_d}_{\mathsf{GenN},\mathscr{A}}(\lambda)\\ &\quad := \left|\Pr\left[\mathscr{A}^{\mathrm{CHAL}^b_{\mathrm{IV}_d}}(N, g_1, \ldots, g_d) = b\right] - \frac{1}{2}\right|\\ &\quad \le \mathsf{negl}(\lambda),\end{aligned} \tag{10}$$

where $(p, q, N, \overline{N}) \leftarrow_{\$} \mathsf{GenN}(1^{\lambda})$, $g_1, \ldots, g_d \leftarrow_{\$} \mathbb{SCR}_{N^s}$, $b \leftarrow_{\$} \{0,1\}$, and $\mathscr{A}$ is allowed to query the oracle $\mathrm{CHAL}^b_{\mathrm{IV}_d}(\cdot)$ adaptively. Each time, $\mathscr{A}$ can submit $(\delta_1, \ldots, \delta_d)$ to the oracle, and $\mathrm{CHAL}^b_{\mathrm{IV}_d}(\delta_1, \ldots, \delta_d)$ selects $r \leftarrow_{\$} [\lfloor N/4 \rfloor]$ randomly: if $b = 0$, the oracle outputs $(g_1^r, \ldots, g_d^r)$ to $\mathscr{A}$; otherwise it outputs $(g_1^r T^{\delta_1}, \ldots, g_d^r T^{\delta_d})$ to $\mathscr{A}$, where $T = 1 + N$.

Definition 10 (DDH assumption). The DDH assumption holds for GenN and $\mathbb{QR}_{\overline{N}}$, if for any PPT $\mathscr{A}$, it holds that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GenN},\mathscr{A}}(\lambda) &:= \left|\Pr\left[\mathscr{A}(N, p, q, g_1, g_2, g_1^x, g_2^x) = 1\right]\right.\\ &\quad \left. - \Pr\left[\mathscr{A}(N, p, q, g_1, g_2, g_1^x, g_2^y) = 1\right]\right| \le \mathsf{negl}(\lambda),\end{aligned} \tag{11}$$

where $(p, q, N, \overline{N}) \leftarrow_{\$} \mathsf{GenN}(1^{\lambda})$, $g_1, g_2 \leftarrow_{\$} \mathbb{QR}_{\overline{N}}$, $x, y \leftarrow_{\$} \mathbb{Z}_N \setminus \{0\}$.

Definition 11 (DL assumption). The Discrete Logarithm (DL) assumption holds for GenN and $\mathbb{SCR}_{N^s}$, if for any PPT $\mathscr{A}$, it holds that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GenN},\mathscr{A}}(\lambda) &:= \Pr\left[\mathscr{A}(N, p, q, g, g^x) = x\right]\\ &\le \mathsf{negl}(\lambda),\end{aligned} \tag{12}$$

where $(p, q, N, \overline{N}) \leftarrow_{\$} \mathsf{GenN}(1^{\lambda})$, $g \leftarrow_{\$} \mathbb{SCR}_{N^s}$, $x \leftarrow_{\$} [\phi(N)/4]$.

2.6. Collision-Resistant Hashing

Definition 12 (collision-resistant hashing). Let $\mathscr{H} = \{\mathsf{H} : \mathscr{X} \to \mathscr{Y}\}$ be a set of hash functions. $\mathscr{H}$ is said to be *collision-resistant*, if for any PPT $\mathscr{A}$, one has

$$\begin{aligned}\mathsf{Adv}^{\mathrm{cr}}_{\mathscr{H},\mathscr{A}}(\lambda) := \Pr\Big[\mathsf{H} \longleftarrow_{\$} \mathscr{H}, \left(x, x'\right) \longleftarrow_{\$} \mathscr{A}(\mathsf{H}) : x \\ \neq x' \wedge \mathsf{H}(x) = \mathsf{H}\left(x'\right)\Big] \leq \mathsf{negl}(\lambda).\end{aligned} \tag{13}$$

3. Auxiliary-Input Authenticated Encryption

Our PKE constructions in Sections 4 and 5 will resort to a new primitive AIAE. To serve the KDM-CCA security of our PKE construction in Figure 1, our AIAE should satisfy the following properties.

(i) AIAE must take an auxiliary input ai in both the encryption and decryption algorithms.

(ii) AIAE must have IND-$\mathscr{F}$-RKA security and weak-INT-$\mathscr{F}$-RKA security. Compared to the INT-$\mathscr{F}$-RKA security proposed in [16], the weak-INT-$\mathscr{F}$-RKA security imposes a special rule to determine whether the adversary's forgery is successful or not.

In the following, we present the syntax of *AIAE* and define its *IND-$\mathscr{F}$-RKA Security* and *Weak-INT-$\mathscr{F}$-RKA Security*. We also show a general paradigm of AIAE from tag-based HPS and give an instantiation of AIAE under the DDH assumption.

3.1. Auxiliary-Input Authenticated Encryption

Definition 13 (AIAE). There are three PPT algorithms AIAE = (AIAE.ParGen, AIAE.Encrypt, AIAE.Decrypt) in an AIAE scheme:

(i) The parameter generation algorithm $\mathsf{AIAE.ParGen}(1^\lambda)$ generates a system parameter $\mathsf{pars}_{\mathsf{AIAE}}$. We require $\mathsf{pars}_{\mathsf{AIAE}}$ to be an implicit input to other algorithms and assume that $\mathsf{pars}_{\mathsf{AIAE}}$ implicitly defines a key space $\mathscr{K}_{\mathsf{AIAE}}$, a message space $\mathscr{M}$, and an auxiliary-input space $\mathscr{AI}$.

(ii) The encryption algorithm $\mathsf{AIAE.Encrypt}(\mathsf{k}, m, \mathsf{ai})$ takes a key $\mathsf{k} \in \mathscr{K}_{\mathsf{AIAE}}$, a message $m \in \mathscr{M}$, and an auxiliary input $\mathsf{ai} \in \mathscr{AI}$ as input and outputs a ciphertext aiae.c.

(iii) The decryption algorithm $\mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai})$ takes a key $\mathsf{k} \in \mathscr{K}_{\mathsf{AIAE}}$, a ciphertext aiae.c, and an auxiliary input $\mathsf{ai} \in \mathscr{AI}$ as input and outputs a message $m \in \mathscr{M}$ or a symbol $\perp$.

We require AIAE to have perfect correctness; that is, for all possible $\mathsf{pars}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.ParGen}(1^\lambda)$, all keys $\mathsf{k} \in \mathscr{K}_{\mathsf{AIAE}}$, all messages $m \in \mathscr{M}$, and all auxiliary-inputs $\mathsf{ai} \in \mathscr{AI}$,

$$\begin{aligned}\Pr[\mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{AIAE.Encrypt}(\mathsf{k}, m, \mathsf{ai}), \mathsf{ai}) \\ = m] = 1.\end{aligned} \tag{14}$$

In fact, AIAE is a generalization of traditional AE, and traditional AE can be viewed as AIAE with $\mathscr{AI} = \emptyset$.

Definition 14 (RKA security). Denote by $\mathscr{F}$ a set of functions from $\mathscr{K}_{\mathsf{AIAE}}$ to $\mathscr{K}_{\mathsf{AIAE}}$. A scheme AIAE is IND-$\mathscr{F}$-RKA secure and weak-INT-$\mathscr{F}$-RKA secure, if for any PPT $\mathscr{A}$,

$$\begin{aligned}\mathsf{Adv}^{\text{ind-rka}}_{\mathsf{AIAE},\mathscr{A}}(\lambda) &:= \left|\Pr\left[\mathsf{IND}\text{-}\mathscr{F}\text{-}\mathsf{RKA}^{\mathscr{A}} \Longrightarrow 1\right] - \frac{1}{2}\right| \\ &\leq \mathsf{negl}(\lambda), \\ \mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE},\mathscr{A}}(\lambda) &:= \Pr\left[\mathsf{weak\text{-}INT}\text{-}\mathscr{F}\text{-}\mathsf{RKA}^{\mathscr{A}} \Longrightarrow 1\right] \\ &\leq \mathsf{negl}(\lambda),\end{aligned} \tag{15}$$

where IND-$\mathscr{F}$-RKA and weak-INT-$\mathscr{F}$-RKA are the security games presented in Figure 4.

3.2. Generic Construction of AIAE from Tag-Based HPS and OT-Secure AE. Our construction of AIAE needs the following ingredients.

(i) A tag-based hash proof system THPS = (THPS.Setup, THPS.Pub, THPS.Priv), where the hash value space is $\mathscr{K}$, the tag space is $\mathscr{T}$, and the hashing key space is $\mathscr{HK}$.

(ii) A (traditional) authenticated encryption scheme AE = (AE.ParGen, AE.Encrypt, AE.Decrypt), where the message space is $\mathscr{M}$ and the key space is $\mathscr{K}$.

(iii) A set of hash functions $\mathscr{H} = \{\mathsf{H} : \{0,1\}^* \to \mathscr{T}\}$.

We present our AIAE construction AIAE = (AIAE.ParGen, AIAE.Encrypt, AIAE.Decrypt) in Figure 5, whose key space is $\mathscr{K}_{\mathsf{AIAE}} := \mathscr{HK}$, message space is $\mathscr{M}$, and auxiliary-input space is $\mathscr{AI} := \{0,1\}^*$.

By the perfect correctness of AE, it is routine to check that AIAE has perfect correctness.

Theorem 15. *If (i)* THPS *is universal$_2$, extracting, key-homomorphic and has a hard subset membership problem, (ii)* AE *is one-time secure, and (iii) $\mathscr{H}$ is collision-resistant, then the scheme* AIAE *in Figure 5 is IND-$\mathscr{F}_{raff}$-RKA and weak-INT-$\mathscr{F}_{raff}$-RKA secure. Here $\mathscr{F}_{raff} := \{f_{(a,\mathsf{b})} : \mathsf{hk} \in \mathscr{HK} \mapsto a \cdot \mathsf{hk} + \mathsf{b} \in \mathscr{HK} \mid a \in \mathbb{Z}^*_{|\mathscr{K}|}, \mathsf{b} \in \mathscr{HK}\}$ is the set of restricted affine functions.*

Proof of Theorem 15 (IND-$\mathscr{F}_{raff}$-RKA Security). Denote by $\mathscr{A}$ a PPT adversary who is against the IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security and queries ENCRYPT oracle for at most Q_e times. We show the IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security through a series of games. For an event E, we denote by $\Pr_j[\mathsf{E}]$, $\Pr_{j'}[\mathsf{E}]$, and $\Pr_{j''}[\mathsf{E}]$ the probability of E occurring in games G_j, G'_j, and G''_j, respectively.

Game G_1. It is the original IND-$\mathscr{F}_{\mathrm{raff}}$-RKA game. Denote the event $\beta' = \beta$ by Succ. According to the definition, $\mathsf{Adv}^{\text{ind-rka}}_{\mathsf{AIAE},\mathscr{A}}(\lambda) = |\Pr_1[\mathsf{Succ}] - 1/2|$.

Proc. INITIALIZE:
$\mathsf{pars}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.ParGen}(1^{\lambda})$.
$\mathsf{k} \leftarrow_{\$} \mathscr{K}_{\mathsf{AIAE}}$.
$\beta \leftarrow_{\$} \{0,1\}$.
Output $\mathsf{pars}_{\mathsf{AIAE}}$.

Proc. ENCRYPT$(m_0, m_1, \mathsf{ai}, f \in \mathscr{F})$:
If $|m_0| \neq |m_1|$, Output $\perp$.
$\mathsf{aiae.c} \leftarrow_{\$} \mathsf{AIAE.Encrypt}(f(\mathsf{k}), m_{\beta}, \mathsf{ai})$.
Output aiae.c.

Proc. FINALIZE(β'):
Output $(\beta' = \beta)$.

(a)

Proc. INITIALIZE:
$\mathsf{pars}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.ParGen}(1^{\lambda})$, $\mathsf{k} \leftarrow_{\$} \mathscr{K}_{\mathsf{AIAE}}$.
Output $\mathsf{pars}_{\mathsf{AIAE}}$.

Proc. ENCRYPT$(m, \mathsf{ai}, f \in \mathscr{F})$:
$\mathsf{aiae.c} \leftarrow_{\$} \mathsf{AIAE.Encrypt}(f(\mathsf{k}), m, \mathsf{ai})$.
$\mathcal{Q}_{\mathscr{ENC}} := \mathcal{Q}_{\mathscr{ENC}} \cup \{(\mathsf{ai}, f, \mathsf{aiae.c})\}$.
$\mathcal{Q}_{\mathscr{AI\text{-}F}} := \mathcal{Q}_{\mathscr{AI\text{-}F}} \cup \{(\mathsf{ai}, f)\}$.
Output aiae.c.

Proc. FINALIZE$(\mathsf{ai}^*, f^* \in \mathscr{F}, \mathsf{aiae.c}^*)$:
If $(\mathsf{ai}^*, f^*, \mathsf{aiae.c}^*) \in \mathcal{Q}_{\mathscr{ENC}}$, Output 0.
// Special rule
If there exists $(\mathsf{ai}, f) \in \mathcal{Q}_{\mathscr{AI\text{-}F}}$ such that
$\mathsf{ai} = \mathsf{ai}^*$ but $f \neq f^*$, Output 0.
Output $(\mathsf{AIAE.Decrypt}(f^*(\mathsf{k}), \mathsf{aiae.c}^*, \mathsf{ai}^*) \neq \perp)$.

(b)

FIGURE 4: IND-$\mathscr{F}$-RKA (a) and weak-INT-$\mathscr{F}$-RKA (b) security games. We note that, in the weak-INT-$\mathscr{F}$-RKA game, there is a special rule (as shown in the shadow) of outputting 0 in FINALIZE.

AIAE.ParGen(1^{λ}):	AIAE.Encrypt(hk, m, ai):	AIAE.Decrypt(hk, $\langle C, \chi\rangle$, ai):
$\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$.	$C \leftarrow_{\$} \mathscr{V}$ with witness w.	If $C \notin \mathscr{C}$, Output $\perp$.
$\mathsf{pars}_{\mathsf{AE}} \leftarrow_{\$} \mathsf{AE.ParGen}(1^{\lambda})$.	$t := \mathsf{H}(C, \mathsf{ai}) \in \mathscr{T}$.	$t := \mathsf{H}(C, \mathsf{ai}) \in \mathscr{T}$.
$\mathsf{H} \leftarrow_{\$} \mathscr{H}$.	$\kappa := \Lambda_{\mathsf{hk}}(C, t) \in \mathscr{K}$.	$\kappa := \Lambda_{\mathsf{hk}}(C, t) \in \mathscr{K}$.
$\mathsf{pars}_{\mathsf{AIAE}} := (\mathsf{pars}_{\mathsf{THPS}}, \mathsf{pars}_{\mathsf{AE}}, \mathsf{H})$.	$\chi \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa, m)$.	$m/\perp \leftarrow \mathsf{AE.Decrypt}(\kappa, \chi)$.
Output $\mathsf{pars}_{\mathsf{AIAE}}$.	Output $\langle C, \chi\rangle$.	Output $m/\perp$.

FIGURE 5: Generic construction of AIAE from THPS and AE.

As for the ℓth ($\ell \in [Q_e]$) ENCRYPT query $(m_{\ell,0}, m_{\ell,1}, \mathsf{ai}_{\ell}, f_{\ell})$, where $f_{\ell} = \langle a_{\ell}, \mathsf{b}_{\ell}\rangle \in \mathscr{F}_{\mathrm{raff}}$, the challenger prepares the challenge ciphertext as follows:

(i) pick $C_{\ell} \leftarrow_{\$} \mathscr{V}$ together with witness w_{ℓ},

(ii) compute $t_{\ell} := \mathsf{H}(C_{\ell}, \mathsf{ai}_{\ell}) \in \mathscr{T}$,

(iii) compute $\kappa_{\ell} := \Lambda_{a_{\ell}\cdot\mathsf{hk}+\mathsf{b}_{\ell}}(C_{\ell}, t_{\ell}) \in \mathscr{K}$,

(iv) invoke $\chi_{\ell} \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_{\ell}, m_{\ell,\beta})$,

and it outputs the challenge ciphertext $\langle C_{\ell}, \chi_{\ell}\rangle$ to $\mathscr{A}$.

Game $\mathsf{G}_{1,j}$, $j \in [Q_e + 1]$. It is identical to G_1, except that, for the first $j-1$ times of ENCRYPT queries, that is, $\ell \in [j-1]$, the challenger chooses $\kappa_{\ell} \leftarrow_{\$} \mathscr{K}$ randomly for the AE scheme.

Clearly $\mathsf{G}_{1,1}$ is identical to G_1; thus $\Pr_1[\mathsf{Succ}] = \Pr_{1,1}[\mathsf{Succ}]$.

Game $\mathsf{G}'_{1,j}$, $j \in [Q_e]$. It is identical to $\mathsf{G}_{1,j}$, except that, for the jth ENCRYPT query, the challenger samples $C_j \leftarrow_{\$} \mathscr{C} \setminus \mathscr{V}$ uniformly.

The difference between $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$ lies in the distribution of C_j. In game $\mathsf{G}_{1,j}$, C_j is uniformly chosen from $\mathscr{V}$; in game $\mathsf{G}'_{1,j}$, C_j is uniformly chosen from $\mathscr{C} \setminus \mathscr{V}$. Any difference between $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$ results in a PPT adversary solving the subset membership problem related to THPS; thus we have that $|\Pr_{1,j}[\mathsf{Succ}] - \Pr_{1,j'}[\mathsf{Succ}]| \leq \mathsf{Adv}^{\mathrm{smp}}_{\mathsf{THPS}}(\lambda)$.

Game $\mathsf{G}''_{1,j}$, $j \in [Q_e]$. It is identical to $\mathsf{G}'_{1,j}$, except that, for the jth ENCRYPT query, the challenger chooses $\kappa_j \leftarrow_{\$} \mathscr{K}$ randomly.

Lemma 16. *For all* $j \in [Q_e]$, $Pr_{1,j'}[\mathsf{Succ}] = Pr_{1,j''}[\mathsf{Succ}]$.

Proof. For game $\mathsf{G}'_{1,j}$ and game $\mathsf{G}''_{1,j}$, the difference between them lies in the computation of κ_j in the jth ENCRYPT query. In $\mathsf{G}'_{1,j}$, κ_j is properly computed, while in $\mathsf{G}''_{1,j}$, it is chosen from $\mathscr{K}$ uniformly.

We analyze the information about the key hk that is used in game $\mathsf{G}'_{1,j}$.

(i) For the ℓth ($\ell \in [j-1]$) query, ENCRYPT does not use hk at all since κ_{ℓ} is randomly chosen from $\mathscr{K}$.

(ii) For the ℓth ($\ell \in [j+1, Q_e]$) query, ENCRYPT can use $\mathsf{pk} = \mu(\mathsf{hk})$ to compute κ_ℓ:

$$\begin{aligned}\kappa_\ell &= \Lambda_{a_\ell \cdot \mathsf{hk} + \mathsf{b}_\ell}(C_\ell, t_\ell): \quad C_\ell \longleftarrow_{\$} \mathscr{V} \text{ with witness } w_\ell \\ &= (\Lambda_{\mathsf{hk}}(C_\ell, t_\ell))^{a_\ell} \cdot \Lambda_{\mathsf{b}_\ell}(C_\ell, t_\ell): \\ &\qquad \text{via key-homomorphism} \quad (16) \\ &= (\mathsf{THPS.Pub}(\mathsf{pk}, C_\ell, w_\ell, t_\ell))^{a_\ell} \cdot \Lambda_{\mathsf{b}_\ell}(C_\ell, t_\ell): \\ &\qquad \text{via projective property.}\end{aligned}$$

(iii) For the jth query, ENCRYPT uses $\Lambda_{\mathsf{hk}}(C_j, t_j)$ to compute κ_j:

$$\begin{aligned}\kappa_j &= \Lambda_{a_j \cdot \mathsf{hk} + \mathsf{b}_j}(C_j, t_j): \quad C_j \longleftarrow_{\$} \mathscr{C} \setminus \mathscr{V} \\ &= (\Lambda_{\mathsf{hk}}(C_j, t_j))^{a_j} \cdot \Lambda_{\mathsf{b}_j}(C_j, t_j): \quad (17) \\ &\qquad \text{via key-homomorphism.}\end{aligned}$$

Since $C_j \in \mathscr{C} \setminus \mathscr{V}$, by the universal$_2$ property of THPS, $\Lambda_{\mathsf{hk}}(C_j, t_j)$ is uniformly distributed over $\mathscr{K}$ conditioned on $\mathsf{pk} = \mu(\mathsf{hk})$. Then as long as $a_j \in \mathbb{Z}^*_{|\mathscr{K}|}$, $\kappa_j = (\Lambda_{\mathsf{hk}}(C_j, t_j))^{a_j} \cdot \Lambda_{\mathsf{b}_j}(C_j, t_j)$ is also randomly distributed over $\mathscr{K}$. Consequently, $\mathsf{G}'_{1,j}$ is essentially the same as $\mathsf{G}''_{1,j}$, and $\Pr_{1,j'}[\mathsf{Succ}] = \Pr_{1,j''}[\mathsf{Succ}]$. □

Now, we show that game $\mathsf{G}''_{1,j}$ is computationally indistinguishable from game $\mathsf{G}_{1,j+1}$, $j \in [Q_e]$. Note that the divergence between $\mathsf{G}''_{1,j}$ and $\mathsf{G}_{1,j+1}$ lies in the distribution of C_j in the jth ENCRYPT query. In game $\mathsf{G}''_{1,j}$, C_j is uniformly chosen from $\mathscr{C} \setminus \mathscr{V}$; in game $\mathsf{G}_{1,j+1}$, C_j is uniformly chosen from $\mathscr{V}$. Any difference between these two games results in a PPT adversary solving the subset membership problem related to THPS; thus we have that $|\Pr_{1,j''}[\mathsf{Succ}] - \Pr_{1,j+1}[\mathsf{Succ}]| \le \mathsf{Adv}^{\mathrm{smp}}_{\mathsf{THPS}}(\lambda)$.

Game G_2. It is identical to G_{1,Q_e+1}, except that when answering ENCRYPT queries, the challenger invokes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, 0^{|m_{\ell,0}|})$.

In game G_{1,Q_e+1}, the challenger computes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, m_{\ell,\beta})$; in game G_2, the challenger computes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, 0^{|m_{\ell,0}|})$. Since each κ_ℓ is chosen from $\mathscr{K}$ uniformly at random, $\ell \in [Q_e]$, by a standard hybrid argument, any difference between G_{1,Q_e+1} and G_2 results in a PPT adversary against the IND-OT security of AE, so that $|\Pr_{1,Q_e+1}[\mathsf{Succ}] - \Pr_2[\mathsf{Succ}]| \le Q_e \cdot \mathsf{Adv}^{\text{ind-ot}}_{\mathsf{AE}}(\lambda)$.

Finally, in game G_2, since the challenge ciphertexts are encryptions of $0^{|m_{\ell,0}|}$, hence β is perfectly hidden to $\mathscr{A}$. So $\Pr_2[\mathsf{Succ}] = 1/2$.

Summing up, we proved the IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security.

This completes the proof of Theorem 15 (IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security).

Proof of Theorem 15 (Weak-INT-$\mathscr{F}_{raff}$-RKA Security). Denote by $\mathscr{A}$ a PPT adversary who is against the weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA security and queries ENCRYPT oracle for at most Q_e times. Similarly, the proof goes through a series of games, which are defined analogously, just like those games of the previous proof.

Game G_0. It is the original weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA game.

As for the ℓth ($\ell \in [Q_e]$) ENCRYPT query $(m_\ell, \mathsf{ai}_\ell, f_\ell)$, the challenger computes the challenge ciphertext $\langle C_\ell, \chi_\ell \rangle$ in similar steps as the previous proof and outputs $\langle C_\ell, \chi_\ell \rangle$ to $\mathscr{A}$. Moreover, the challenger will put $(\mathsf{ai}_\ell, f_\ell, \langle C_\ell, \chi_\ell \rangle)$ to a set $\mathscr{Q}_{\mathscr{ENC}}$, put $(\mathsf{ai}_\ell, f_\ell)$ to a set $\mathscr{Q}_{\mathscr{AI}\text{-}\mathscr{F}}$, and put $(C_\ell, \mathsf{ai}_\ell, t_\ell)$ to a set $\mathscr{Q}_{\mathscr{TAG}}$. In the end, the adversary outputs a forgery $(\mathsf{ai}^*, f^*, \langle C^*, \chi^* \rangle)$, where $f^* = \langle a^*, \mathsf{b}^* \rangle$, and the challenger invokes the FINALIZE procedure as follows:

(i) If $(\mathsf{ai}^*, f^*, \langle C^*, \chi^* \rangle) \in \mathscr{Q}_{\mathscr{ENC}}$, output 0.

(ii) If $\exists (\mathsf{ai}_\ell, f_\ell) \in \mathscr{Q}_{\mathscr{AI}\text{-}\mathscr{F}}$ such that $\mathsf{ai}_\ell = \mathsf{ai}^*$ but $f_\ell \ne f^*$, output 0.

(iii) If $C^* \notin \mathscr{C}$, output 0.

(iv) Compute $t^* := \mathsf{H}(C^*, \mathsf{ai}^*) \in \mathscr{T}$ and $\kappa^* := \Lambda_{a^* \cdot \mathsf{hk} + \mathsf{b}^*}(C^*, t^*) \in \mathscr{K}$.

Output $(\mathsf{AE.Decrypt}(\kappa^*, \chi^*) \ne \perp)$.

Denote the event that FINALIZE outputs 1 by Forge. According to the definition, $\mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE},\mathscr{A}}(\lambda) = \Pr_0[\mathsf{Forge}]$.

Game G_1. It is identical to G_0, except that the following rule is added to the procedure FINALIZE by the challenger:

(i) If $\exists (C_\ell, \mathsf{ai}_\ell, t_\ell) \in \mathscr{Q}_{\mathscr{TAG}}$ such that $t_\ell = t^*$ but $(C_\ell, \mathsf{ai}_\ell) \ne (C^*, \mathsf{ai}^*)$, output 0.

Since $t_\ell = \mathsf{H}(C_\ell, \mathsf{ai}_\ell)$ and $t^* = \mathsf{H}(C^*, \mathsf{ai}^*)$, any difference between G_0 and G_1 implies a hash collision of H. So $|\Pr_0[\mathsf{Forge}] - \Pr_1[\mathsf{Forge}]| \le \mathsf{Adv}^{\mathrm{cr}}_{\mathscr{H}}(\lambda)$.

Game $\mathsf{G}_{1,j}$, $j \in [Q_e + 1]$. It is identical to G_1, except that, for the first $j-1$ times of ENCRYPT queries, that is, $\ell \in [j-1]$, the challenger chooses $\kappa_\ell \leftarrow_{\$} \mathscr{K}$ uniformly for the AE scheme.

Clearly $\mathsf{G}_{1,1}$ is identical to G_1; thus $\Pr_1[\mathsf{Forge}] = \Pr_{1,1}[\mathsf{Forge}]$.

Game $\mathsf{G}'_{1,j}$, $j \in [Q_e]$. It is identical to $\mathsf{G}_{1,j}$, except that, for the jth ENCRYPT query, the challenger samples $C_j \leftarrow_{\$} \mathscr{C} \setminus \mathscr{V}$ uniformly.

The difference between $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$ lies in the distribution of C_j. In game $\mathsf{G}_{1,j}$, C_j is uniformly chosen from $\mathscr{V}$; in game $\mathsf{G}'_{1,j}$, C_j is uniformly chosen from $\mathscr{C} \setminus \mathscr{V}$. Any difference between these two games results in a PPT adversary solving the subset membership problem related to THPS. We emphasize that the PPT adversary (simulator) is able to check the occurrence of Forge in an efficient way, because the key hk can be chosen by the simulator itself. Consequently, the difference between $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$ can be reduced to the subset membership problem smoothly.

Lemma 17. *For all* $j \in [Q_e]$, $|Pr_{1,j}[\mathsf{Forge}] - Pr_{1,j'}[\mathsf{Forge}]| \leq \mathsf{Adv}^{smp}_{\mathsf{THPS}}(\lambda)$.

Proof. To bound the difference between $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$, we build an efficient adversary $\mathscr{B}$ solving the subset membership problem. Given $(\mathsf{pars}_{\mathsf{THPS}}, C)$, where $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$, $\mathscr{B}$ aims to distinguish $C \leftarrow_{\$} \mathscr{V}$ from $C \leftarrow_{\$} \mathscr{C} \setminus \mathscr{V}$.

$\mathscr{B}$ simulates $\mathsf{G}_{1,j}$ or $\mathsf{G}'_{1,j}$ for $\mathscr{A}$. Firstly, $\mathscr{B}$ invokes $\mathsf{pars}_{\mathsf{AE}} \leftarrow_{\$} \mathsf{AE.ParGen}(1^{\lambda})$, picks $\mathsf{H} \leftarrow_{\$} \mathscr{H}$ randomly, and sends $\mathsf{pars}_{\mathsf{AIAE}} := (\mathsf{pars}_{\mathsf{THPS}}, \mathsf{pars}_{\mathsf{AE}}, \mathsf{H})$ to $\mathscr{A}$. Next, $\mathscr{B}$ chooses $\mathsf{hk} \leftarrow_{\$} \mathscr{H}\mathscr{K}$.

As for the ℓth ($\ell \in [Q_e]$) ENCRYPT query $(m_\ell, \mathsf{ai}_\ell, f_\ell)$, where $f_\ell = \langle a_\ell, \mathsf{b}_\ell \rangle \in \mathscr{F}_{\mathrm{raff}}$, $\mathscr{B}$ prepares the challenge ciphertext $\langle C_\ell, \chi_\ell \rangle$ in the following way.

(i) If $\ell \in [j-1]$, $\mathscr{B}$ computes $\langle C_\ell, \chi_\ell \rangle$ just like that in both $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$. That is, $\mathscr{B}$ chooses $C_\ell \leftarrow_{\$} \mathscr{V}$ with witness w_ℓ, chooses $\kappa_\ell \leftarrow_{\$} \mathscr{K}$ randomly, and invokes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, m_\ell)$.

(ii) If $\ell \in [j+1, Q_e]$, $\mathscr{B}$ computes $\langle C_\ell, \chi_\ell \rangle$ just like that in both $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$. That is, $\mathscr{B}$ chooses $C_\ell \leftarrow_{\$} \mathscr{V}$ with witness w_ℓ, computes $t_\ell := \mathsf{H}(C_\ell, \mathsf{ai}_\ell)$ and $\kappa_\ell := \Lambda_{a_\ell \cdot \mathsf{hk} + \mathsf{b}_\ell}(C_\ell, t_\ell)$, and invokes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, m_\ell)$.

(iii) If $\ell = j$, $\mathscr{B}$ embeds its own challenge C to C_j, that is, $C_j := C$. Then it computes $t_j := \mathsf{H}(C_j, \mathsf{ai}_j)$, $\kappa_j := \Lambda_{a_j \cdot \mathsf{hk} + \mathsf{b}_j}(C_j, t_j)$, and invokes $\chi_j \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_j, m_j)$.

$\mathscr{B}$ outputs the challenge ciphertext $\langle C_\ell, \chi_\ell \rangle$ to $\mathscr{A}$. Moreover, $\mathscr{B}$ puts $(\mathsf{ai}_\ell, f_\ell, \langle C_\ell, \chi_\ell \rangle)$ to $\mathscr{Q}_{\mathscr{E}\mathscr{N}\mathscr{C}}$, $(\mathsf{ai}_\ell, f_\ell)$ to $\mathscr{Q}_{\mathscr{A}\mathscr{I}\text{-}\mathscr{F}}$, and $(C_\ell, \mathsf{ai}_\ell, t_\ell)$ to $\mathscr{Q}_{\mathscr{T}\mathscr{A}\mathscr{G}}$.

Obviously, $\mathscr{B}$ simulates $\mathsf{G}_{1,j}$ in the case of $C \leftarrow_{\$} \mathscr{V}$ and simulates $\mathsf{G}'_{1,j}$ in the case of $C \leftarrow_{\$} \mathscr{C} \setminus \mathscr{V}$.

Finally, $\mathscr{A}$ sends a forgery $(\mathsf{ai}^*, f^*, \langle C^*, \chi^* \rangle)$ to $\mathscr{B}$, with $f^* = \langle a^*, \mathsf{b}^* \rangle \in \mathscr{F}_{\mathrm{raff}}$. Then $\mathscr{B}$ decides whether FINALIZE outputs 1 or not with the help of hk.

(i) If $(\mathsf{ai}^*, f^*, \langle C^*, \chi^* \rangle) \in \mathscr{Q}_{\mathscr{E}\mathscr{N}\mathscr{C}}$, $\mathscr{B}$ outputs 0 (to its own challenger).

(ii) If $\exists(\mathsf{ai}_\ell, f_\ell) \in \mathscr{Q}_{\mathscr{A}\mathscr{I}\text{-}\mathscr{F}}$ such that $\mathsf{ai}_\ell = \mathsf{ai}^*$ but $f_\ell \neq f^*$, $\mathscr{B}$ outputs 0.

(iii) If $C^* \notin \mathscr{C}$, $\mathscr{B}$ outputs 0.

(iv) $\mathscr{B}$ computes $t^* := \mathsf{H}(C^*, \mathsf{ai}^*) \in \mathscr{T}$.

(v) If $\exists(C_\ell, \mathsf{ai}_\ell, t_\ell) \in \mathscr{Q}_{\mathscr{T}\mathscr{A}\mathscr{G}}$ such that $t_\ell = t^*$ but $(C_\ell, \mathsf{ai}_\ell) \neq (C^*, \mathsf{ai}^*)$, $\mathscr{B}$ outputs 0.

(vi) $\mathscr{B}$ computes $\kappa^* := \Lambda_{a^* \cdot \mathsf{hk} + \mathsf{b}^*}(C^*, t^*) \in \mathscr{K}$ and outputs $(\mathsf{AE.Decrypt}(\kappa^*, \chi^*) \neq \perp)$.

With the help of hk, $\mathscr{B}$ is able to perfectly simulate FINALIZE, just like that in both $\mathsf{G}_{1,j}$ and $\mathsf{G}'_{1,j}$. Moreover, $\mathscr{B}$ outputs 1 to its own challenger if and only if the event Forge occurs.

As a result, we have that $|\Pr_{1,j}[\mathsf{Forge}] - \Pr_{1,j'}[\mathsf{Forge}]| \leq \mathsf{Adv}^{\mathrm{smp}}_{\mathsf{THPS},\mathscr{B}}(\lambda)$. □

Game $\mathsf{G}''_{1,j}$, $j \in [Q_e]$. It is identical to $\mathsf{G}'_{1,j}$, except that, for the jth ENCRYPT query, the challenger chooses $\kappa_j \leftarrow_{\$} \mathscr{K}$ randomly.

Lemma 18. *For all* $j \in [Q_e]$, $Pr_{1,j'}[\mathsf{Forge}] \leq Pr_{1,j''}[\mathsf{Forge}] + \mathsf{Adv}^{int\text{-}ot}_{\mathsf{AE}}(\lambda)$.

Proof. For game $\mathsf{G}'_{1,j}$ and game $\mathsf{G}''_{1,j}$, the difference between them lies in the computation of κ_j in the jth ENCRYPT query. In $\mathsf{G}'_{1,j}$, κ_j is properly computed; in $\mathsf{G}''_{1,j}$, κ_j is chosen from $\mathscr{K}$ uniformly.

We consider the information about the key hk that is used in $\mathsf{G}'_{1,j}$.

(i) For the ℓth ($\ell \in [j-1]$) query, ENCRYPT does not use hk at all since κ_ℓ is randomly chosen from $\mathscr{K}$.

(ii) For the ℓth ($\ell \in [j+1, Q_e]$) query, similar to the proof of Lemma 16, ENCRYPT can use $\mathsf{pk} = \mu(\mathsf{sk})$ to compute κ_ℓ.

(iii) For the jth query, similar to the proof of Lemma 16, ENCRYPT uses $\Lambda_{\mathsf{hk}}(C_j, t_j)$ to compute κ_j:

$$\begin{aligned} \kappa_j &= \Lambda_{a_j \cdot \mathsf{hk} + \mathsf{b}_j}\left(C_j, t_j\right): \quad C_j \longleftarrow_{\$} \mathscr{C} \setminus \mathscr{V} \\ &= \left(\Lambda_{\mathsf{hk}}\left(C_j, t_j\right)\right)^{a_j} \cdot \Lambda_{\mathsf{b}_j}\left(C_j, t_j\right): \\ &\qquad \text{via key-homomorphism.} \end{aligned} \tag{18}$$

(iv) The FINALIZE procedure, which defines the event Forge, uses $\Lambda_{\mathsf{hk}}(C^*, t^*)$ to compute κ^*:

$$\begin{aligned} \kappa^* &= \Lambda_{a^* \cdot \mathsf{hk} + \mathsf{b}^*}\left(C^*, t^*\right) \\ &= \left(\Lambda_{\mathsf{hk}}\left(C^*, t^*\right)\right)^{a^*} \cdot \Lambda_{\mathsf{b}^*}\left(C^*, t^*\right): \\ &\qquad \text{via key-homomorphism.} \end{aligned} \tag{19}$$

We divide the event Forge into the following two subevents:

(i) Subevent: $\mathsf{Forge} \wedge t_j \neq t^*$. Let us first consider the event $t_j \neq t^*$. We show that

$$\Pr_{1,j'}\left[t_j \neq t^*\right] = \Pr_{1,j''}\left[t_j \neq t^*\right]. \tag{20}$$

By the fact that $C_j \in \mathscr{C} \setminus \mathscr{V}$ and by the universal$_2$ property of THPS, $\Lambda_{\mathsf{hk}}(C_j, t_j)$ is uniformly distributed over $\mathscr{K}$ conditioned on $\mathsf{pk} = \mu(\mathsf{hk})$. Then as long as $a_j \in \mathbb{Z}^*_{|\mathscr{K}|}$, $\kappa_j = (\Lambda_{\mathsf{hk}}(C_j, t_j))^{a_j} \cdot \Lambda_{\mathsf{b}_j}(C_j, t_j)$ is also randomly distributed over $\mathscr{K}$. Hence, $\mathsf{G}'_{1,j}$ is the same as $\mathsf{G}''_{1,j}$ before $\mathscr{A}$ queries FINALIZE, and consequently, $t_j \neq t^*$ occurs with the same probability in $\mathsf{G}'_{1,j}$ and $\mathsf{G}''_{1,j}$.

Next we consider the event Forge conditioned on $t_j \neq t^*$. We show that

$$\Pr_{1,j'}\left[\mathsf{Forge} \mid t_j \neq t^*\right] = \Pr_{1,j''}\left[\mathsf{Forge} \mid t_j \neq t^*\right]. \tag{21}$$

Since $t_j \neq t^*$ and $C_j \in \mathscr{C} \setminus \mathscr{V}$, by the universal$_2$ property of THPS, $\Lambda_{\mathsf{hk}}(C_j, t_j)$ is uniformly distributed over $\mathscr{K}$ conditioned on $\mathsf{pk} = \mu(\mathsf{hk})$ and $\Lambda_{\mathsf{hk}}(C^*, t^*)$. With a similar argument, κ_j is also randomly distributed over $\mathscr{K}$. Hence, $\mathsf{G}'_{1,j}$ is the same as $\mathsf{G}''_{1,j}$ when $t_j \neq t^*$, and consequently, the probability that Forge occurs in $\mathsf{G}'_{1,j}$ and $\mathsf{G}''_{1,j}$ conditioned on $t_j \neq t^*$ is the same.

In conclusion, we have that

$$\begin{aligned}\Pr_{1,j'}\left[\mathsf{Forge} \wedge t_j \neq t^*\right] &= \Pr_{1,j''}\left[\mathsf{Forge} \wedge t_j \neq t^*\right] \\ &\leq \Pr_{1,j''}\left[\mathsf{Forge}\right].\end{aligned} \tag{22}$$

(ii) Subevent: $\mathsf{Forge} \wedge t_j = t^*$. By the new rule added in game G_1, Forge and $t_j = t^*$ will imply $(C_j, \mathsf{ai}_j) = (C^*, \mathsf{ai}^*)$. In addition, Forge and $\mathsf{ai}_j = \mathsf{ai}^*$ will imply that $f_j = f^*$, due to the special rule in the weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA game (see Figure 4). Then it is straightforward to check that $\Lambda_{\mathsf{hk}}(C_j, t_j) = \Lambda_{\mathsf{hk}}(C^*, t^*)$ and

$$\begin{aligned}\kappa_j &= \left(\Lambda_{\mathsf{hk}}\left(C_j, t_j\right)\right)^{a_j} \cdot \Lambda_{\mathsf{b}_j}\left(C_j, t_j\right) \\ &= \left(\Lambda_{\mathsf{hk}}\left(C^*, t^*\right)\right)^{a^*} \cdot \Lambda_{\mathsf{b}^*}\left(C^*, t^*\right) = \kappa^*.\end{aligned} \tag{23}$$

Since $C_j \in \mathscr{C} \setminus \mathscr{V}$, by the universal$_2$ property of THPS, $\Lambda_{\mathsf{hk}}(C_j, t_j)$ $(= \Lambda_{\mathsf{hk}}(C^*, t^*))$ is uniformly distributed over $\mathscr{K}$ conditioned on $\mathsf{pk} = \mu(\mathsf{hk})$. Then as long as a_j (which equals a^*) $\in \mathbb{Z}^*_{|\mathscr{K}|}$, κ_j (which equals κ^*) is also randomly distributed over $\mathscr{K}$. Also in this subevent, $(\mathsf{ai}^*, f^*, C^*) = (\mathsf{ai}_j, f_j, C_j)$ implies $\chi^* \neq \chi_j$; thus the probability of $\mathsf{AE.Decrypt}(\kappa^*, \chi^*) \neq \perp$ is bounded by $\mathsf{Adv}^{\text{int-ot}}_{\mathsf{AE}}(\lambda)$. So we have the following claim. We present the full description of the reduction in Appendix A.

Claim 19. One has $\Pr_{1,j'}[\mathsf{Forge} \wedge t_j = t^*] \leq \mathsf{Adv}^{\text{int-ot}}_{\mathsf{AE}}(\lambda)$.

Combining the above two subevents together, Lemma 18 follows. □

Now, we show that game $\mathsf{G}''_{1,j}$ is computationally indistinguishable from game $\mathsf{G}_{1,j+1}$, $j \in [Q_e]$. Note that the divergence between $\mathsf{G}''_{1,j}$ and $\mathsf{G}_{1,j+1}$ lies in the distribution of C_j in the jth ENCRYPT query. In game $\mathsf{G}''_{1,j}$, C_j is uniformly chosen from $\mathscr{C} \setminus \mathscr{V}$; in game $\mathsf{G}_{1,j+1}$, C_j is uniformly chosen from $\mathscr{V}$. Similar to Lemma 17, any difference between these two games results in a PPT adversary solving the subset membership problem related to THPS; thus we have that $|\Pr_{1,j''}[\mathsf{Forge}] - \Pr_{1,j+1}[\mathsf{Forge}]| \leq \mathsf{Adv}^{\text{smp}}_{\mathsf{THPS}}(\lambda)$.

Finally, in game G_{1,Q_e+1}, note that the challenger does not use hk to compute κ_ℓ at all; thus hk is uniformly random to $\mathscr{A}$. Consequently, in the FINALIZE procedure, we have

$$\kappa^* = \left(\Lambda_{\mathsf{hk}}\left(C^*, t^*\right)\right)^{a^*} \cdot \Lambda_{\mathsf{b}^*}\left(C^*, t^*\right). \tag{24}$$

By the extracting property of THPS, $\Lambda_{\mathsf{hk}}(C^*, t^*)$ is uniformly random over $\mathscr{K}$. Therefore, as long as $a^* \in \mathbb{Z}^*_{|\mathscr{K}|}$, κ^* is uniformly random over $\mathscr{K}$ as well. Hence, the probability of $\mathsf{AE.Decrypt}(\kappa^*, \chi^*) \neq \perp$ is bounded by $\mathsf{Adv}^{\text{int-ot}}_{\mathsf{AE}}(\lambda)$, and we have $\Pr_{1,Q_e+1}[\mathsf{Forge}] \leq \mathsf{Adv}^{\text{int-ot}}_{\mathsf{AE}}(\lambda)$.

In all, we proved the weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA security.

This completes the proof of Theorem 15 (weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA security).

Remark 20. We emphasize that the special rule in the weak-INT-$\mathscr{F}$-RKA game (cf. Figure 4) plays an essential role in proving Lemma 18. Below is the reason.

Without this special rule, the adversary is allowed to submit f^* $(= \langle a^*, \mathsf{b}^*\rangle)$ which is different from f_j $(= \langle a_j, \mathsf{b}_j\rangle)$, even if $\mathsf{ai}^* = \mathsf{ai}_j$ holds. In this case, we cannot expect to employ the INT-OT security of the underlying AE scheme to show that the second subevent $(\mathsf{Forge} \wedge t_j = t^*)$ occurs with only a negligible probability. To demonstrate the problem clearly, suppose that the adversary $\mathscr{A}$ submits $f_j = \langle a_j, \mathsf{b}_j\rangle$ in the jth ENCRYPT query and submits $f^* = \langle a^*, \mathsf{b}^*\rangle = \langle a_j, \mathsf{b}_j + \Delta\rangle$ in the FINALIZE procedure, where Δ is a constant. Then we have

$$\begin{aligned}\kappa^* &= \left(\Lambda_{\mathsf{hk}}\left(C_j, t_j\right)\right)^{a_j} \cdot \Lambda_{\mathsf{b}_j+\Delta}\left(C_j, t_j\right) \\ &= \left(\Lambda_{\mathsf{hk}}\left(C_j, t_j\right)\right)^{a_j} \cdot \Lambda_{\mathsf{b}_j}\left(C_j, t_j\right) \cdot \Lambda_{\Delta}\left(C_j, t_j\right) \\ &= \kappa_j \cdot \Lambda_{\Delta}\left(C_j, t_j\right),\end{aligned} \tag{25}$$

where the second equality follows from the key-homomorphism of THPS. Thus, κ^* *and* κ_j *are closely related but may not be equal;* in particular, the quotient κ^*/κ_j $(= \Lambda_\Delta(C_j, t_j))$ is a constant.

Consequently, it is hard for us to show that the subevent $\mathsf{Forge} \wedge t_j = t^*$ occurs with a negligible probability. The reason is as follows. To show that it is infeasible for any PPT adversary $\mathscr{A}$, who obtains $\chi_j \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_j, m_j)$ in the jth ENCRYPT query, to generate an AE-ciphertext χ^* satisfying $\mathsf{AE.Decrypt}(\kappa^*, \chi^*)$ $(= \mathsf{AE.Decrypt}(\kappa_j \cdot \Lambda_\Delta(C_j, t_j), \chi^*)) \neq \perp$, it seems that INT-RKA security of AE is required to some extent. We definitely cannot require INT-RKA security for the underlying AE scheme, since we are constructing (weak) INT-RKA secure (AI)AE scheme AIAE. As a result, it is hard to prove Lemma 18 without our special rule in the weak-INT-$\mathscr{F}$-RKA game.

3.3. Tag-Based HPS from the DDH Assumption. Qin et al. [19] gave a construction of tag-based HPS from the d-LIN assumption. Here we construct a key-homomorphic $\mathsf{THPS}_{\mathsf{DDH}}$ under the DDH assumption in Figure 6. With a routine check, the projective property of $\mathsf{THPS}_{\mathsf{DDH}}$ follows.

Theorem 21. $\mathsf{THPS}_{\mathsf{DDH}}$ *in Figure 6 is universal$_2$, extracting, and key-homomorphic. Moreover, the subset membership problem related to* $\mathsf{THPS}_{\mathsf{DDH}}$ *is hard under the DDH assumption for* GenN *and* $\mathbb{QR}_{\overline{N}}$.

Proof of Theorem 21.

Universal$_2$. Suppose that $C = (g_1^{w_1}, g_2^{w_2}) \in \mathscr{C}$, $C' = (g_1^{w'_1}, g_2^{w'_2}) \in \mathscr{C} \setminus \mathscr{V}$, and $t, t' \in \mathscr{T}$ with $t \neq t'$. For $\mathsf{hk} = (k_1, k_2, k_3,$

$\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^{\lambda})$:

$(p, q, N, \overline{N}) \leftarrow_{\$} \mathsf{GenN}(1^{\lambda})$,

i.e., p, q are safe primes, $N := pq$, $\overline{N} := 2N + 1$ is a prime.

$g_1, g_2 \leftarrow_{\$} \mathbb{QR}_{\overline{N}}$.

Output $\mathsf{pars}_{\mathsf{THPS}} := (N, p, q, \overline{N}, g_1, g_2)$, which implicitly defines $(\mathscr{K}, \mathscr{C}, \mathscr{V}, \mathscr{T}, \mathscr{HK}, \mathscr{PK}, \Lambda_{(\cdot)}, \mu)$:

$\mathscr{K} := \mathbb{QR}_{\overline{N}}$. $\mathscr{C} := \mathbb{QR}_{\overline{N}}^2 \backslash \{(1,1)\}$. $\mathscr{V} := \{(g_1^w, g_2^w) \mid w \in \mathbb{Z}_N \backslash \{0\}\}$.

$\mathscr{T} := \mathbb{Z}_N$. $\mathscr{HK} := (\mathbb{Z}_N)^4$. $\mathscr{PK} := \mathbb{QR}_{\overline{N}}^2$.

For $\mathsf{hk} = (k_1, k_2, k_3, k_4) \in \mathscr{HK}, C = (c_1, c_2) \in \mathscr{C}, t \in \mathscr{T}, \Lambda_{\mathsf{hk}}(C, t) := c_1^{k_1+k_3 t} c_2^{k_2+k_4 t} \in \mathscr{K}$.

For $\mathsf{hk} = (k_1, k_2, k_3, k_4) \in \mathscr{HK}, \mathsf{pk} = \mu(\mathsf{hk}) := (g_1^{k_1} g_2^{k_2}, g_1^{k_3} g_2^{k_4}) \in \mathscr{PK}$.

$K \leftarrow \mathsf{THPS.Pub}(\mathsf{pk}, C, w, t)$:	$K \leftarrow \mathsf{THPS.Priv}(\mathsf{hk}, C, t)$:
Parse $\mathsf{pk} = (h_1, h_2) \in \mathscr{PK}$.	Parse $\mathsf{hk} = (k_1, k_2, k_3, k_4) \in \mathscr{HK}$ and $C = (c_1, c_2) \in \mathscr{C}$.
Output $K := h_1^w h_2^{wt}$.	Output $K := c_1^{k_1+k_3 t} c_2^{k_2+k_4 t}$.

Figure 6: Construction of $\mathsf{THPS}_{\mathsf{DDH}}$.

$k_4) \leftarrow_{\$} (\mathbb{Z}_N)^4$, we analyze the distribution of $\Lambda_{\mathsf{hk}}(C', t')$ conditioned on $\mathsf{pk} = \mu(\mathsf{hk})$ and $\Lambda_{\mathsf{hk}}(C, t)$.

Denote $d := d \log_{g_1} g_2 \in \mathbb{Z}_N$. Firstly $\mathsf{pk} = \mu(\mathsf{hk}) = (g_1^{k_1} g_2^{k_2}, g_1^{k_3} g_2^{k_4}) = (g_1^{k_1+dk_2}, g_1^{k_3+dk_4})$, which may leak the values of $k_1 + dk_2$ and $k_3 + dk_4$.

Next

$$\begin{aligned}\Lambda_{\mathsf{hk}}(C, t) &= \left(g_1^{w_1}\right)^{k_1+k_3 t} \cdot \left(g_2^{w_2}\right)^{k_2+k_4 t} \\ &= g_1^{\overbrace{(w_1 k_1 + w_2 d k_2) + t \cdot (w_1 k_3 + w_2 d k_4)}^{\triangleq X}},\end{aligned} \tag{26}$$

which may further leak the value of X.

Similarly,

$$\begin{aligned}\Lambda_{\mathsf{hk}}\left(C', t'\right) &= \left(g_1^{w_1'}\right)^{k_1+k_3 t'} \cdot \left(g_2^{w_2'}\right)^{k_2+k_4 t'} \\ &= g_1^{\overbrace{(w_1' k_1 + w_2' d k_2) + t' \cdot (w_1' k_3 + w_2' d k_4)}^{\triangleq Y}}.\end{aligned} \tag{27}$$

By the fact that $C' = (g_1^{w_1'}, g_2^{w_2'}) \notin \mathscr{V}$, we have $w_1' \neq w_2'$. Then as long as $t \neq t'$, Y is independent of $k_1 + dk_2$, $k_3 + dk_4$, and X, and consequently, Y is uniformly distributed over $\mathbb{Z}_N$.

Therefore, conditioned on $\mathsf{pk} = \mu(\mathsf{hk})$ and $\Lambda_{\mathsf{hk}}(C, t)$, $\Lambda_{\mathsf{hk}}(C', t')$ $(= g_1^Y)$ is randomly distributed over $\mathscr{K} = \mathbb{QR}_{\overline{N}}$.

Extracting. Suppose that $C = (g_1^{w_1}, g_2^{w_2}) \in \mathscr{C}$ and $t \in \mathscr{T}$. For $\mathsf{hk} = (k_1, k_2, k_3, k_4) \leftarrow_{\$} (\mathbb{Z}_N)^4$, we analyze the distribution of $\Lambda_{\mathsf{hk}}(C, t)$.

By (26), $\Lambda_{\mathsf{hk}}(C, t) = g_1^X$ with $X = (w_1 k_1 + w_2 d k_2) + t \cdot (w_1 k_3 + w_2 d k_4)$. Since $C = (g_1^{w_1}, g_2^{w_2}) \in \mathscr{C}$, we have $(w_1, w_2) \neq (0, 0)$. Then when (k_1, k_2, k_3, k_4) is randomly chosen from $(\mathbb{Z}_N)^4$, X is uniformly distributed over $\mathbb{Z}_N$. Consequently, $\Lambda_{\mathsf{hk}}(C, t)$ is randomly distributed over $\mathscr{K} = \mathbb{QR}_{\overline{N}}$.

Key-Homomorphism. For all $\mathsf{hk} = (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$, all $a \in \mathbb{Z}$, all $\mathsf{b} = (b_1, b_2, b_3, b_4) \in (\mathbb{Z}_N)^4$, all $C = (c_1, c_2) \in \mathscr{C}$, and all $t \in \mathscr{T}$, we have $a \cdot \mathsf{hk} + \mathsf{b} = a \cdot (k_1, k_2, k_3, k_4) + (b_1, b_2, b_3, b_4) = (ak_1 + b_1, ak_2 + b_2, ak_3 + b_3, ak_4 + b_4)$. Then it follows that

$$\begin{aligned}\Lambda_{a \cdot \mathsf{hk}+\mathsf{b}}(C, t) &= c_1^{(ak_1+b_1)+(ak_3+b_3)t} \cdot c_2^{(ak_2+b_2)+(ak_4+b_4)t} \\ &= \left(c_1^{k_1+k_3 t} c_2^{k_2+k_4 t}\right)^a \cdot \left(c_1^{b_1+b_3 t} c_2^{b_2+b_4 t}\right) \\ &= \Lambda_{\mathsf{hk}}(C, t)^a \cdot \Lambda_{\mathsf{b}}(C, t).\end{aligned} \tag{28}$$

Subset Membership Problem. The subset membership problem related to $\mathsf{THPS}_{\mathsf{DDH}}$ requires that $(\mathsf{pars}_{\mathsf{THPS}} = (N, p, q, \overline{N}, g_1, g_2), C = (g_1^w, g_2^w))$ is computationally indistinguishable from $(\mathsf{pars}_{\mathsf{THPS}} = (N, p, q, \overline{N}, g_1, g_2), C' = (g_1^{w_1}, g_2^{w_2}))$, where $C \leftarrow_{\$} \mathscr{V}$ and $C' \leftarrow_{\$} \mathscr{C} \setminus \mathscr{V}$. It trivially holds under the DDH assumption for GenN and $\mathbb{QR}_{\overline{N}}$. □

3.4. Instantiation: $\mathsf{AIAE}_{\mathsf{DDH}}$ from DDH-Based $\mathsf{THPS}_{\mathsf{DDH}}$ and OT-Secure AE. When plugging the $\mathsf{THPS}_{\mathsf{DDH}}$ (cf. Figure 6) into the paradigm in Figure 5, we immediately obtain an AIAE scheme $\mathsf{AIAE}_{\mathsf{DDH}}$ under the DDH assumption, as shown in Figure 7. The key space is $\mathscr{K}_{\mathsf{AIAE}} = (\mathbb{Z}_N)^4$.

By combining Theorem 15 with Theorem 21, we have the following corollary regarding the RKA security of $\mathsf{AIAE}_{\mathsf{DDH}}$.

Corollary 22. *If (i) the DDH assumption holds for* GenN *and* $\mathbb{QR}_{\overline{N}}$*, (ii)* AE *is one-time secure, and (iii)* $\mathscr{H}$ *is collision-resistant, then the scheme* $\mathsf{AIAE}_{\mathsf{DDH}}$ *in Figure 7 is IND-*$\mathscr{F}_{raff}$*-RKA and weak-INT-*$\mathscr{F}_{raff}$*-RKA secure. Here* $\mathscr{F}_{raff} := \{f_{(a,\mathsf{b})} : (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4 \mapsto (ak_1 + b_1, ak_2 + b_2, ak_3 + b_3, ak_4 + b_4) \in (\mathbb{Z}_N)^4 \mid a \in \mathbb{Z}_N^*, \mathsf{b} = (b_1, b_2, b_3, b_4) \in (\mathbb{Z}_N)^4\}$.

Remark 23. Our $\mathsf{AIAE}_{\mathsf{DDH}}$ enjoys the following property: $\kappa = c_1^{k_1+k_3 t} \cdot c_2^{k_2+k_4 t}$ will be randomly distributed over $\mathbb{QR}_{\overline{N}}$, as long as any element k_j in $\mathsf{k} = (k_1, k_2, k_3, k_4)$ is uniformly chosen. As a result, the one-time security of AE will guarantee that $\mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai}) = \perp$ holds for any $(\mathsf{aiae.c}, \mathsf{ai})$ except with probability $\mathsf{Adv}_{\mathsf{AE}}^{\text{int-ot}}(\lambda) \leq \mathsf{Adv}_{\mathsf{AIAE}_{\mathsf{DDH}}}^{\text{weak-int-rka}}(\lambda)$. This

$\mathsf{AIAE.ParGen}(1^\lambda)$:	
$(p, q, N, \overline{N}) \leftarrow_\$ \mathsf{GenN}(1^\lambda)$, i.e., p, q are safe primes, $N := pq$, $\overline{N} := 2N + 1$ is a prime.	
$g_1, g_2 \leftarrow_\$ \mathbb{QR}_{\overline{N}}$. $\mathsf{pars}_{\mathsf{AE}} \leftarrow_\$ \mathsf{AE.ParGen}(1^\lambda)$. $\mathsf{H} \leftarrow_\$ \mathscr{H}$.	
Output $\mathsf{pars}_{\mathsf{AIAE}} := (N, p, q, \overline{N}, g_1, g_2, \mathsf{pars}_{\mathsf{AE}}, \mathsf{H})$.	
$\mathsf{AIAE.Encrypt}(\mathsf{k}, m, \mathsf{ai})$:	$\mathsf{AIAE.Decrypt}(\mathsf{k}, \langle c_1, c_2, \chi\rangle, \mathsf{ai})$:
Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.	Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.
$w \leftarrow_\$ \mathbb{Z}_N \backslash \{0\}$. $(c_1, c_2) := (g_1^w, g_2^w) \in \mathbb{QR}_{\overline{N}}^2$.	If $(c_1, c_2) \notin \mathbb{QR}_{\overline{N}}^2 \vee (c_1, c_2) = (1, 1)$, Output $\perp$.
$t := \mathsf{H}(c_1, c_2, \mathsf{ai}) \in \mathbb{Z}_N$.	$t := \mathsf{H}(c_1, c_2, \mathsf{ai}) \in \mathbb{Z}_N$.
$\kappa := c_1^{k_1+k_3t} \cdot c_2^{k_2+k_4t} \in \mathbb{QR}_{\overline{N}}$.	$\kappa := c_1^{k_1+k_3t} \cdot c_2^{k_2+k_4t} \in \mathbb{QR}_{\overline{N}}$.
$\chi \leftarrow_\$ \mathsf{AE.Encrypt}(\kappa, m)$.	Output $\mathsf{AE.Decrypt}(\kappa, \chi)$.
Output $\langle c_1, c_2, \chi\rangle$.	

Figure 7: Construction of $\mathsf{AIAE}_{\mathsf{DDH}}$ from AE and $\mathsf{THPS}_{\mathsf{DDH}}$.

fact will be used in the security proof of the PKE schemes presented in Sections 4 and 5.

4. PKE with n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA Security

Denote by $\mathsf{AIAE}_{\mathsf{DDH}} = (\mathsf{AIAE.ParGen}, \mathsf{AIAE.Encrypt}, \mathsf{AIAE.Decrypt})$ the DDH-based AIAE scheme in Figure 7, where the key space is $(\mathbb{Z}_N)^4$. We need two other building blocks, following the approach in Figure 1.

KEM: to be compatible with this $\mathsf{AIAE}_{\mathsf{DDH}}$, we have to design a KEM encapsulating a key tuple $(k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.

$\mathscr{E}$: to support the set $\mathscr{F}_{\mathrm{aff}}$ of affine functions, we have to construct a special public-key encryption $\mathscr{E}$, so that after a computationally indistinguishable change, $\mathscr{E}$.Encrypt can serve as an entropy filter for the affine function set.

The proposed PKE scheme $\mathsf{PKE} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is presented in Figure 8, in which the shadowed parts highlight algorithms of KEM and $\mathscr{E}$.

The correctness of PKE is guaranteed by the correctness of $\mathsf{AIAE}_{\mathsf{DDH}}$, $\mathscr{E}$, and KEM.

Theorem 24. *If (i) the DCR assumption holds for* GenN *and* $\mathbb{QR}_{N^s}$, *(ii)* $\mathsf{AIAE}_{\mathsf{DDH}}$ *is IND-*$\mathscr{F}_{raff}$*-RKA and weak-INT-*$\mathscr{F}_{raff}$*-RKA secure, and (iii) the DL assumption holds for* GenN *and* $\mathbb{SCR}_{N^s}$, *then the proposed scheme* PKE *in Figure 8 is n-KDM[*$\mathscr{F}_{aff}$*]-CCA secure.*

Proof of Theorem 24. Denote by $\mathscr{A}$ a PPT adversary who is against the n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA security, querying ENCRYPT oracle for at most Q_e times and DECRYPT oracle for at most Q_d times. The theorem is proved through a series of games. A rough description of differences between adjacent games is summarized in Table 2.

In the proof, G_1-G_2 deals with the n-user case; G_3-G_4 is used to eliminate the utilization of the (mod N) part of $(x_j, y_j)_{j=1}^4$ in the ENCRYPT oracle; the aim of G_5-G_6 is to use $(x_j, y_j)_{j=1}^4 \bmod N$ to hide a base key $\mathsf{k}^* = (k_1^*, \ldots, k_4^*)$ of $\mathsf{AIAE}_{\mathsf{DDH}}$ in the ENCRYPT oracle; G_7-G_8 is used to eliminate the utilization of $(x_j, y_j)_{j=1}^4 \bmod N$ in the DECRYPT oracle; in G_9-G_{10}, the IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security of $\mathsf{AIAE}_{\mathsf{DDH}}$ leads to the n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA security, because $\mathsf{k}^* = (k_1^*, \ldots, k_4^*)$ now is concealed by $(x_j, y_j)_{j=1}^4 \bmod N$ perfectly.

Game G_0. It is the n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA game. Denote the event $\beta' = \beta$ by Succ. According to the definition, $\mathsf{Adv}_{\mathsf{PKE},\mathscr{A}}^{\mathsf{kdm\text{-}cca}}(\lambda) = |\Pr_0[\mathsf{Succ}] - 1/2|$.

For the ith user, $i \in [n]$, let $\mathsf{pk}_i = (h_{i,1}, \ldots, h_{i,4})$ and $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4})$ denote the corresponding public key and secret key, respectively.

Game G_1. It is identical to G_0, except the way of answering the DECRYPT query $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle,\ i \in [n])$. More precisely, the challenger outputs $\perp$ if $\langle \mathsf{ai}, \mathsf{aiae.c}\rangle = \langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle$ for some $\ell \in [Q_e]$, where $\langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle$ is the challenge ciphertext of the ℓth ENCRYPT oracle query (f_ℓ, i_ℓ).

Case 1 $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle, i) = (\langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle, i_\ell)$. DECRYPT will output $\perp$ in G_0 since $(\langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle, i_\ell) \in \mathcal{Q}_{\mathscr{E}\mathscr{N}\mathscr{C}}$ is prohibited by DECRYPT.

Case 2 $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle = \langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle$ but $i \neq i_\ell)$. We show that, in G_0, DECRYPT will output $\perp$, due to $e_{\ell,1} u_{\ell,1}^{x_{i,1}} u_{\ell,2}^{y_{i,1}} \notin \mathbb{RU}_{N^2}$, with overwhelming probability. Recall that $u_{\ell,1} = g_1^{r_\ell}$, $u_{\ell,2} = g_2^{r_\ell}$, $e_{\ell,1} = h_{i_\ell,1}^{r_\ell} T^{k_{\ell,1}}$, so

$$\begin{aligned} e_{\ell,1} u_{\ell,1}^{x_{i,1}} u_{\ell,2}^{y_{i,1}} &= h_{i_\ell,1}^{r_\ell} T^{k_{\ell,1}} \cdot \left(g_1^{r_\ell}\right)^{x_{i,1}} \left(g_2^{r_\ell}\right)^{y_{i,1}} \\ &= \left(h_{i_\ell,1} h_{i,1}^{-1}\right)^{r_\ell} T^{k_{\ell,1}} \mod N^2, \end{aligned} \tag{29}$$

where $h_{i_\ell,1}$ and $h_{i,1}$ are parts of public keys of i_ℓth user and ith user, respectively, and are uniformly random over $\mathbb{SCR}_{N^s}$.

$\mathsf{pars} \leftarrow_{\$} \mathsf{ParGen}(1^{\lambda})$:

$\mathsf{pars}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.ParGen}(1^{\lambda})$, where

$\mathsf{pars}_{\mathsf{AIAE}} = (N, p, q, \overline{N}, \overline{g}_1, \overline{g}_2, \mathsf{pars}_{\mathsf{AE}}, \mathsf{H})$,

$N = pq, \overline{N} = 2N + 1, \overline{g}_1, \overline{g}_2 \in \mathbb{QR}_{\overline{N}}$.

$\mathsf{pars}'_{\mathsf{AIAE}} := (N, \overline{N}, \overline{g}_1, \overline{g}_2, \mathsf{pars}_{\mathsf{AE}}, \mathsf{H})$.

$g_1, g_2, g_3, g_4, g_5 \leftarrow_{\$} \mathbb{SCR}_{N^s}$.

Output $\mathsf{pars} := (\mathsf{pars}'_{\mathsf{AIAE}}, g_1, g_2, g_3, g_4, g_5)$.

$\langle \mathsf{ai}, \mathsf{aiae.c} \rangle \leftarrow_{\$} \mathsf{Encrypt}(\mathsf{pk}, m)$: $\quad m \in [N^{s-1}]$

// $(\mathsf{k}, \mathsf{ai}) \leftarrow_{\$} \mathsf{KEM.Encrypt}(\mathsf{pk})$:

$\mathsf{k} = (k_1, k_2, k_3, k_4) \leftarrow_{\$} (\mathbb{Z}_N)^4$.

$r \leftarrow_{\$} \left[\left\lfloor \frac{N}{4} \right\rfloor\right]$.

$(u_1, u_2, u_3, u_4, u_5) := (g_1^r, g_2^r, g_3^r, g_4^r, g_5^r) \bmod N^2$.

$(e_1, e_2, e_3, e_4) := (h_1^r T^{k_1}, h_2^r T^{k_2}, h_3^r T^{k_3}, h_4^r T^{k_4}) \bmod N^2$.

$\mathsf{ai} := (u_1, \ldots, u_5, e_1, \ldots, e_4)$.

// $\mathscr{E}.\mathsf{c} \leftarrow_{\$} \mathscr{E}.\mathsf{Encrypt}(\mathsf{pk}, m)$:

$\tilde{r}_1, \tilde{r}_2, \tilde{r}_3, \tilde{r}_4 \leftarrow_{\$} \left[\left\lfloor \frac{N}{4} \right\rfloor\right]$.

$(\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, \tilde{u}_4, \tilde{u}_5, \tilde{u}_6, \tilde{u}_7, \tilde{u}_8) := (g_1^{\tilde{r}_1}, g_2^{\tilde{r}_1}, g_2^{\tilde{r}_2}, g_3^{\tilde{r}_2}, g_3^{\tilde{r}_3}, g_4^{\tilde{r}_3}, g_4^{\tilde{r}_4}, g_5^{\tilde{r}_4}) \bmod N^s$.

$\tilde{e} := h_1^{\tilde{r}_1} h_2^{\tilde{r}_2} h_3^{\tilde{r}_3} h_4^{\tilde{r}_4} T^m \bmod N^s$.

$t := g_1^m \bmod N \in \mathbb{Z}_N$.

$\mathscr{E}.\mathsf{c} := (\tilde{u}_1, \ldots, \tilde{u}_8, \tilde{e}, t)$.

$\mathsf{aiae.c} \leftarrow_{\$} \mathsf{AIAE.Encrypt}(\mathsf{k}, \mathscr{E}.\mathsf{c}, \mathsf{ai})$.

Output $\langle \mathsf{ai}, \mathsf{aiae.c} \rangle$.

$(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KeyGen}(\mathsf{pars})$:

$x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \leftarrow_{\$} \left[\left\lfloor \frac{N^2}{4} \right\rfloor\right]$.

$(h_1, h_2, h_3, h_4) := (g_1^{-x_1} g_2^{-y_1}, g_2^{-x_2} g_3^{-y_2}, g_3^{-x_3} g_4^{-y_3}, g_4^{-x_4} g_5^{-y_4}) \bmod N^s$.

$\mathsf{pk} := (h_1, h_2, h_3, h_4)$.

$\mathsf{sk} := (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$.

Output $(\mathsf{pk}, \mathsf{sk})$.

$m/\perp \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \langle \mathsf{ai}, \mathsf{aiae.c} \rangle)$:

// $\mathsf{k}/\perp \leftarrow \mathsf{KEM.Decrypt}(\mathsf{sk}, \mathsf{ai})$:

Parse $\mathsf{ai} = (u_1, \ldots, u_5, e_1, \ldots, e_4)$.

If $e_1 u_1^{x_1} u_2^{y_1}, e_2 u_2^{x_2} u_3^{y_2}, e_3 u_3^{x_3} u_4^{y_3}, e_4 u_4^{x_4} u_5^{y_4} \in \mathbb{RU}_{N^2}$

$(k_1, k_2, k_3, k_4) := (d \log_T(e_1 u_1^{x_1} u_2^{y_1}), d \log_T(e_2 u_2^{x_2} u_3^{y_2}), d \log_T(e_3 u_3^{x_3} u_4^{y_3}), d \log_T(e_4 u_4^{x_4} u_5^{y_4})) \bmod N$.

$\mathsf{k} := (k_1, k_2, k_3, k_4)$.

Else, Output $\perp$.

$\mathscr{E}.\mathsf{c}/\perp \leftarrow \mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai})$.

// $m/\perp \leftarrow \mathscr{E}.\mathsf{Decrypt}(\mathsf{sk}, \mathscr{E}.\mathsf{c})$:

Parse $\mathscr{E}.\mathsf{c} = (\tilde{u}_1, \ldots, \tilde{u}_8, \tilde{e}, t)$.

If $\widetilde{eu}_1^{x_1} \tilde{u}_2^{y_1} \tilde{u}_3^{x_2} \tilde{u}_4^{y_2} \tilde{u}_5^{x_3} \tilde{u}_6^{y_3} \tilde{u}_7^{x_4} \tilde{u}_8^{y_4} \in \mathbb{RU}_{N^s}$

$m := d \log_T(\widetilde{eu}_1^{x_1} \tilde{u}_2^{y_1} \tilde{u}_3^{x_2} \tilde{u}_4^{y_2} \tilde{u}_5^{x_3} \tilde{u}_6^{y_3} \tilde{u}_7^{x_4} \tilde{u}_8^{y_4}) \bmod N^{s-1}$.

If $t = g_1^m \bmod N$, Output m.

Else, Output $\perp$.

Figure 8: Construction of PKE from $\mathsf{AIAE}_{\mathsf{DDH}}$. The shadowed parts highlight algorithms of KEM and $\mathscr{E}$. Here p, q in $\mathsf{pars}_{\mathsf{AIAE}}$ are not provided in $\mathsf{pars}'_{\mathsf{AIAE}}$, since they are not used in $\mathsf{AIAE.Encrypt}$ and $\mathsf{AIAE.Decrypt}$ of $\mathsf{AIAE}_{\mathsf{DDH}}$.

So $h_{i_\ell,1} h_{i,1}^{-1} \neq 1$; hence $e_{\ell,1} u_{\ell,1}^{x_{i,1}} u_{\ell,2}^{y_{i,1}} \notin \mathbb{RU}_{N^2}$, except with negligible probability $2^{-\Omega(\lambda)}$.

Thus G_0 and G_1 are the same except with probability at most $Q_d \cdot 2^{-\Omega(\lambda)}$ according to the union bound, and $|\Pr_0[\mathsf{Succ}] - \Pr_1[\mathsf{Succ}]| \leq Q_d \cdot 2^{-\Omega(\lambda)}$.

Game G_2. It is identical to G_1, except the way the challenger samples the secret keys $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4})$, $i \in [n]$. In game G_2, the challenger first chooses $(x_1, y_1, \ldots, x_4, y_4)$ and $(\overline{x}_{i,1}, \overline{y}_{i,1}, \ldots, \overline{x}_{i,4}, \overline{y}_{i,4})$ randomly from $[\lfloor N^2/4 \rfloor]$; next it computes $(x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4}) := (x_1, y_1, \ldots, x_4, y_4) + (\overline{x}_{i,1}, \overline{y}_{i,1}, \ldots, \overline{x}_{i,4}, \overline{y}_{i,4}) \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$.

Obviously, the secret keys $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4})$ are uniformly distributed. Hence G_2 is identical to G_1, and $\Pr_1[\mathsf{Succ}] = \Pr_2[\mathsf{Succ}]$.

Game G_3. It is identical to G_2, except the way the challenger responds to the ℓth ($\ell \in [Q_e]$) ENCRYPT query (f_ℓ, i_ℓ). In game G_3, instead of using the public key $\mathsf{pk}_{i_\ell} = (h_{i_\ell,1}, \ldots, h_{i_\ell,4})$, the challenger uses the secret key $\mathsf{sk}_{i_\ell} = (x_{i_\ell,1}, y_{i_\ell,1}, \ldots, x_{i_\ell,4}, y_{i_\ell,4})$ to prepare $(e_{\ell,1}, \ldots, e_{\ell,4})$ and $\tilde{e}_\ell$ in the following way:

(i)

$$(e_{\ell,1}, \ldots, e_{\ell,4}) := \left(u_{\ell,1}^{-x_{i_\ell,1}} u_{\ell,2}^{-y_{i_\ell,1}} T^{k_{\ell,1}}, \ldots, u_{\ell,4}^{-x_{i_\ell,4}} u_{\ell,5}^{-y_{i_\ell,4}} T^{k_{\ell,4}}\right) \bmod N^2, \tag{30}$$

(ii)

$$\tilde{e}_\ell := \tilde{u}_{\ell,1}^{-x_{i_\ell,1}} \tilde{u}_{\ell,2}^{-y_{i_\ell,1}} \tilde{u}_{\ell,3}^{-x_{i_\ell,2}} \tilde{u}_{\ell,4}^{-y_{i_\ell,2}} \tilde{u}_{\ell,5}^{-x_{i_\ell,3}} \tilde{u}_{\ell,6}^{-y_{i_\ell,3}} \tilde{u}_{\ell,7}^{-x_{i_\ell,4}} \tilde{u}_{\ell,8}^{-y_{i_\ell,4}} T^{m_\beta} \bmod N^s. \tag{31}$$

Note that for $j \in [4]$,

$$e_{\ell,j} \overset{\mathsf{G}_2}{=} h_{i_\ell,j}^{r_\ell} T^{k_{\ell,j}} = \left(g_j^{-x_{i_\ell,j}} g_{j+1}^{-y_{i_\ell,j}}\right)^{r_\ell} T^{k_{\ell,j}} \overset{\mathsf{G}_3}{=} u_{\ell,j}^{-x_{i_\ell,j}} u_{\ell,j+1}^{-y_{i_\ell,j}} T^{k_{\ell,j}} \bmod N^2,$$

Table 2: Brief description of the security proof of Theorem 24.

	Changes between adjacent games	Assumptions
G_0	The original n-KDM-CCA security game.	—
G_1	DECRYPT: reject if $\langle \mathsf{ai}, \mathsf{aiae.c}\rangle = \langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle$ for some $\ell \in [Q_e]$.	$\mathsf{G}_0 \approx_s \mathsf{G}_1$
G_2	INITIALIZE: sample secret keys with $(x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4}) := (x_1, y_1, \ldots, x_4, y_4) + (\overline{x}_{i,1}, \overline{y}_{i,1}, \ldots, \overline{x}_{i,4}, \overline{y}_{i,4})$.	$\mathsf{G}_1 = \mathsf{G}_2$
G_3	ENCRYPT(f_ℓ, i_ℓ): use the secret keys to run KEM.Encrypt and $\mathscr{E}$.Encrypt	$\mathsf{G}_2 = \mathsf{G}_3$
G_4	ENCRYPT(f_ℓ, i_ℓ): when ENCRYPT oracle encrypts affine function of secret keys, $\mathscr{E}$.c is computed with $(\widetilde{u}_{\ell,j})_{j\in[8]} := (g_1^{\widetilde{r}_{\ell,1}} T^{\delta_1}, \ldots, g_5^{\widetilde{r}_{\ell,4}} T^{\delta_8})$ instead of $(g_1^{\widetilde{r}_{\ell,1}}, \ldots, g_5^{\widetilde{r}_{\ell,4}})$. ENCRYPT does not use $(x_j, y_j)_{j=1}^4 \bmod N$ any more if $(\delta_j)_{j\in[8]}$ is carefully chosen.	$\mathsf{G}_3 \approx_c \mathsf{G}_4$ by IV_5
G_5	ENCRYPT(f_ℓ, i_ℓ): kem.ct (= ai) of KEM.Encrypt is computed with $(u_{\ell,j})_{j\in[5]} := ((g_j^{r^*} T^{\alpha_j})^{r_\ell})_{j\in[5]}$ instead of $(g_j^{r_\ell})_{j\in[5]}$. Now KEM.Encrypt encapsulates four keys $(k_{\ell,j} - r_\ell \cdot (\alpha_j x_{i,j} + \alpha_{j+1} y_{i,j}))_{j=1}^4 \bmod N$ but $(k_{\ell,j})_{j=1}^4$ is the key used in AIAE.Encrypt.	$\mathsf{G}_4 \approx_c \mathsf{G}_5$ by IV_5
G_6	ENCRYPT(f_ℓ, i_ℓ): sample $k_{\ell,j} := r_\ell k_j^* + s_{\ell,j}$ for $j \in [4]$. Now KEM.Encrypt encapsulates four keys $(r_\ell(k_j^* - \alpha_j x_j - \alpha_{j+1} y_j) - r_\ell(\alpha_j \overline{x}_{i,j} + \alpha_{j+1} \overline{y}_{i,j}) + s_{\ell,j})_{j=1}^4$ but $(r_\ell k_j^* + s_{\ell,j})_{j=1}^4$ is the key used in AIAE.Encrypt.	$\mathsf{G}_5 = \mathsf{G}_6$
G_7	DECRYPT: use $\phi(N)$ and secret keys to answer decryption queries.	$\mathsf{G}_6 = \mathsf{G}_7$
G_8	DECRYPT: add an additional rejection rule. Reject if $\mathsf{Bad}' := (\exists u_j \notin \mathbb{SCR}_{N^2})$ or $\widetilde{\mathsf{Bad}} := (\forall u_j \in \mathbb{SCR}_{N^2}) \wedge (\exists \widetilde{u}_j \notin \mathbb{SCR}_{N^s})$ happens. Bad' and $\widetilde{\mathsf{Bad}}$ can be detected by using $\phi(N)$. Now only the $(\bmod \phi(N)/4)$ part of secret keys and $\phi(N)$ are used in DECRYPT. The randomness of $(\alpha_j x_j + \alpha_{j+1} y_j)_{j=1}^4 \bmod N$ perfectly hides $(k_1^*, \ldots, k_4^*)$ in ENCRYPT, thus $(k_1^*, \ldots, k_4^*)$ is uniform. $(r_\ell k_j^* + s_{\ell,j})_{j=1}^4$ is the key used in AIAE.Encrypt. Bad' may lead to a fresh successful forgery for $\mathsf{AIAE}_{\mathsf{DDH}}$.	$\mathsf{G}_7 = \mathsf{G}_8$ if neither Bad' nor $\widetilde{\mathsf{Bad}}$ happens. $\Pr[\mathsf{Bad}'] = \mathsf{negl}$ due to weak INT-$\mathscr{F}_{\mathrm{raff}}$-RKA security of $\mathsf{AIAE}_{\mathsf{DDH}}$
G_9	INITIALIZE: sample an independent random tuple $(\overline{k}_1^*, \ldots, \overline{k}_4^*)$. ENCRYPT$(f_\ell, i_\ell)$: use $(r_\ell \overline{k}_j^* + s_{\ell,j})_{j=1}^4$ in AIAE.Encrypt.	$\mathsf{G}_8 = \mathsf{G}_9$ to the adversary
G_{10}	ENCRYPT: encrypt zeros instead of the affine function of secret keys. $\widetilde{\mathsf{Bad}}$ happens with negligible probability, since $t \neq g_1^m \bmod N$ in DECRYPT. Adversary $\mathscr{A}$ wins with probability 1/2.	$\mathsf{G}_9 \approx_c \mathsf{G}_{10}$ by IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security of $\mathsf{AIAE}_{\mathsf{DDH}}$. $\Pr[\widetilde{\mathsf{Bad}}] = \mathsf{negl}$

$$\begin{aligned}\widetilde{e}_\ell &\overset{\mathsf{G}_2}{=} h_{i_\ell,1}^{\widetilde{r}_{\ell,1}} \cdots h_{i_\ell,4}^{\widetilde{r}_{\ell,4}} T^{m_\beta} \\ &= \left(g_1^{-x_{i_\ell,1}} g_2^{-y_{i_\ell,1}}\right)^{\widetilde{r}_{\ell,1}} \cdots \left(g_4^{-x_{i_\ell,4}} g_5^{-y_{i_\ell,4}}\right)^{\widetilde{r}_{\ell,4}} T^{m_\beta} \\ &\overset{\mathsf{G}_3}{=} \widetilde{u}_{\ell,1}^{-x_{i_\ell,1}} \widetilde{u}_{\ell,2}^{-y_{i_\ell,1}} \cdots \widetilde{u}_{\ell,7}^{-x_{i_\ell,4}} \widetilde{u}_{\ell,8}^{-y_{i_\ell,4}} T^{m_\beta} \mod N^s.\end{aligned} \tag{32}$$

Thus, G_3 is the same as G_2, and $\Pr_2[\mathsf{Succ}] = \Pr_3[\mathsf{Succ}]$.

Game G_4. It is identical to G_3, except the way the challenger responds to the ℓth ($\ell \in [Q_e]$) ENCRYPT query (f_ℓ, i_ℓ). In game G_4, in the case of $\beta = 1$, $(\widetilde{u}_{\ell,1}, \ldots, \widetilde{u}_{\ell,8})$ and $\widetilde{e}_\ell$ are computed without the use of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$:

(i)

$$\begin{aligned}(\widetilde{u}_{\ell,1}, \ldots, \widetilde{u}_{\ell,8}) := \Big(&g_1^{\widetilde{r}_{\ell,1} T^{\sum_{i=1}^n a_{i,1}}}, g_2^{\widetilde{r}_{\ell,1} T^{\sum_{i=1}^n b_{i,1}}}, \\ &g_2^{\widetilde{r}_{\ell,2} T^{\sum_{i=1}^n a_{i,2}}}, g_3^{\widetilde{r}_{\ell,2} T^{\sum_{i=1}^n b_{i,2}}}, g_3^{\widetilde{r}_{\ell,3} T^{\sum_{i=1}^n a_{i,3}}}, g_4^{\widetilde{r}_{\ell,3} T^{\sum_{i=1}^n b_{i,3}}}, \\ &g_4^{\widetilde{r}_{\ell,4} T^{\sum_{i=1}^n a_{i,4}}}, g_5^{\widetilde{r}_{\ell,4} T^{\sum_{i=1}^n b_{i,4}}}\Big) \mod N^s,\end{aligned} \tag{33}$$

(ii)

$$\widetilde{e}_\ell := h_{i_\ell,1}^{\widetilde{r}_{\ell,1}} \cdots h_{i_\ell,4}^{\widetilde{r}_{\ell,4}} T^{\sum_{i=1}^n \sum_{j=1}^4 (a_{i,j}(\overline{x}_{i,j} - \overline{x}_{i_\ell,j}) + b_{i,j}(\overline{y}_{i,j} - \overline{y}_{i_\ell,j})) + c} \mod N^s, \tag{34}$$

where $f_\ell = (\{a_{i,1}, b_{i,1}, \ldots, a_{i,4}, b_{i,4}\}_{i\in[n]}, c) \in \mathscr{F}_{\mathrm{aff}}$.

Note that

$$\begin{aligned}\widetilde{e}_\ell &\overset{\mathsf{G}_4}{=} \prod_{j=1}^4 h_{i_\ell,j}^{\widetilde{r}_{\ell,j}} \cdot T^{\sum_{i=1}^n \sum_{j=1}^4 (a_{i,j}(\overline{x}_{i,j} - \overline{x}_{i_\ell,j}) + b_{i,j}(\overline{y}_{i,j} - \overline{y}_{i_\ell,j})) + c} \\ &= \prod_{j=1}^4 h_{i_\ell,j}^{\widetilde{r}_{\ell,j}} \cdot T^{\sum_{i=1}^n \sum_{j=1}^4 (a_{i,j}(x_{i,j} - x_{i_\ell,j}) + b_{i,j}(y_{i,j} - y_{i_\ell,j})) + c}\end{aligned}$$

$$
\begin{aligned}
&= \prod_{j=1}^{4}\left(g_j^{-x_{i_\ell,j}} g_{j+1}^{-y_{i_\ell,j}}\right)^{\tilde{r}_{\ell,j}} \cdot T^{m_1-\sum_{i=1}^{n}\sum_{j=1}^{4}(a_{i,j}x_{i_\ell,j}+b_{i,j}y_{i_\ell,j})} \\
&= \prod_{j=1}^{4}\left(g_j^{\tilde{r}_{\ell,j}} T^{\sum_{i=1}^{n} a_{i,j}}\right)^{-x_{i_\ell,j}} \left(g_{j+1}^{\tilde{r}_{\ell,j}} T^{\sum_{i=1}^{n} b_{i,j}}\right)^{-y_{i_\ell,j}} \cdot T^{m_1} \\
&= \tilde{u}_{\ell,1}^{-x_{i_\ell,1}} \tilde{u}_{\ell,2}^{-y_{i_\ell,1}} \cdots \tilde{u}_{\ell,7}^{-x_{i_\ell,4}} \tilde{u}_{\ell,8}^{-y_{i_\ell,4}} T^{m_1} \mod N^s,
\end{aligned} \tag{35}
$$

where the third equality follows from $m_1 = \sum_{i=1}^{n}(a_{i,1}x_{i,1} + b_{i,1}y_{i,1} + \cdots + a_{i,4}x_{i,4} + b_{i,4}y_{i,4}) + c$.

We analyze the difference between G_3 and G_4 via the following lemma.

Lemma 25. *One has* $|\Pr_3[\mathsf{Succ}] - \Pr_4[\mathsf{Succ}]| \le \mathsf{Adv}_{\mathsf{GenN}}^{iv_5}(\lambda)$.

Proof. According to the last line of (35), the way that $\tilde{e}_\ell$ is computed from $(\tilde{u}_{\ell,1}, \ldots, \tilde{u}_{\ell,8})$ is the same in G_3 and G_4. Therefore the only divergence between G_3 and G_4 lies in $(\tilde{u}_{\ell,1}, \ldots, \tilde{u}_{\ell,8})$.

We show that any difference between G_3 and G_4 results in a PPT adversary $\mathscr{B}_1$ solving the IV_5 problem. $\mathscr{B}_1$ is provided with $(N, g_1, \ldots, g_5)$ and has access to its $\textsc{chal}_{\mathrm{IV}_5}^{b}$ oracle. $\mathscr{B}_1$ simulates game G_3 or game G_4 for $\mathscr{A}$. Firstly, $\mathscr{B}_1$ prepares pars and generates $(\mathsf{pk}_i, \mathsf{sk}_i)$, $i \in [n]$, as in G_3 and G_4. As for the ℓth ($\ell \in [Q_e]$) $\textsc{encrypt}$ query (f_ℓ, i_ℓ) from $\mathscr{A}$, where $f_\ell = (\{a_{i,1}, b_{i,1}, \ldots, a_{i,4}, b_{i,4}\}_{i\in[n]}, c) \in \mathscr{F}_{\mathrm{aff}}$, $\mathscr{B}_1$ proceeds as follows: it queries its own $\textsc{chal}_{\mathrm{IV}_5}^{b}$ oracle with $(\sum_{i=1}^{n} a_{i,1}, \sum_{i=1}^{n} b_{i,1}, *, *, *)$, $(*, \sum_{i=1}^{n} a_{i,2}, \sum_{i=1}^{n} b_{i,2}, *, *)$, $(*, *, \sum_{i=1}^{n} a_{i,3}, \sum_{i=1}^{n} b_{i,3}, *)$, $(*, *, *, \sum_{i=1}^{n} a_{i,4}, \sum_{i=1}^{n} b_{i,4})$, where the symbol "$*$" denotes dummy messages. Then $\mathscr{B}_1$ obtains its challenges $(\tilde{u}_{\ell,1}, \tilde{u}_{\ell,2}, \tilde{*}, \tilde{*}, \tilde{*})$, $(\tilde{*}, \tilde{u}_{\ell,3}, \tilde{u}_{\ell,4}, \tilde{*}, \tilde{*})$, $(\tilde{*}, \tilde{*}, \tilde{u}_{\ell,5}, \tilde{u}_{\ell,6}, \tilde{*})$, $(\tilde{*}, \tilde{*}, \tilde{*}, \tilde{u}_{\ell,7}, \tilde{u}_{\ell,8})$ and neglects "$\tilde{*}$" terms. According to the definition of $\textsc{chal}_{\mathrm{IV}_5}^{b}$ oracle, $(\tilde{u}_{\ell,1}, \ldots, \tilde{u}_{\ell,8})$ is one of the following:

Case 1 ($b = 0$). $(g_1^{\tilde{r}_{\ell,1}}, g_2^{\tilde{r}_{\ell,1}}, g_2^{\tilde{r}_{\ell,2}}, g_3^{\tilde{r}_{\ell,2}}, g_3^{\tilde{r}_{\ell,3}}, g_4^{\tilde{r}_{\ell,3}}, g_4^{\tilde{r}_{\ell,4}}, g_5^{\tilde{r}_{\ell,4}})$.

Case 2 ($b = 1$). $(g_1^{\tilde{r}_{\ell,1}} T^{\sum_{i=1}^{n} a_{i,1}}, g_2^{\tilde{r}_{\ell,1}} T^{\sum_{i=1}^{n} b_{i,1}}, g_2^{\tilde{r}_{\ell,2}} T^{\sum_{i=1}^{n} a_{i,2}}, g_3^{\tilde{r}_{\ell,2}} T^{\sum_{i=1}^{n} b_{i,2}}, g_3^{\tilde{r}_{\ell,3}} T^{\sum_{i=1}^{n} a_{i,3}}, g_4^{\tilde{r}_{\ell,3}} T^{\sum_{i=1}^{n} b_{i,3}}, g_4^{\tilde{r}_{\ell,4}} T^{\sum_{i=1}^{n} a_{i,4}}, g_5^{\tilde{r}_{\ell,4}} T^{\sum_{i=1}^{n} b_{i,4}})$.

Next $\mathscr{B}_1$ uses the obtained $(\tilde{u}_{\ell,1}, \ldots, \tilde{u}_{\ell,8})$ and the secret keys to compute $\tilde{e}_\ell$ via (35) for $\mathscr{A}$. In the meantime, $\mathscr{B}_1$ can also simulate $\textsc{decrypt}$ for $\mathscr{A}$ since it knows the secret keys. Finally, $\mathscr{B}_1$ outputs 1 if the event Succ occurs.

In Case 1, $\mathscr{B}_1$ simulates game G_3 perfectly for $\mathscr{A}$; in Case 2, $\mathscr{B}_1$ simulates game G_4 perfectly for $\mathscr{A}$. Any difference between $\Pr_3[\mathsf{Succ}]$ and $\Pr_4[\mathsf{Succ}]$ results in $\mathscr{B}_1$'s advantage over the IV_5 problem. Thus Lemma 25 follows. □

Game G_5. It is identical to G_4, except for the following differences. In the $\textsc{initialize}$ procedure of game G_5, the challenger picks $r^* \leftarrow_\$ [\lfloor N/4 \rfloor]$ and $\alpha_1, \ldots, \alpha_5 \leftarrow_\$ \mathbb{Z}_N$ randomly. As for the ℓth ($\ell \in [Q_e]$) $\textsc{encrypt}$ query (f_ℓ, i_ℓ), the challenger computes $(u_{\ell,1}, \ldots, u_{\ell,5})$ as follows:

(i) $(u_{\ell,1}, \ldots, u_{\ell,5}) := ((g_1^{r^*} T^{\alpha_1})^{r_\ell}, \ldots, (g_5^{r^*} T^{\alpha_5})^{r_\ell}) \mod N^2$.

The only difference between G_4 and G_5 is the distribution of $(u_{\ell,1}, \ldots, u_{\ell,5})$. In game G_4, $(u_{\ell,1}, \ldots, u_{\ell,5}) = (g_1^{r_\ell}, \ldots, g_5^{r_\ell}) \mod N^2$, while in game G_5, $(u_{\ell,1}, \ldots, u_{\ell,5}) = ((g_1^{r^*} T^{\alpha_1})^{r_\ell}, \ldots, (g_5^{r^*} T^{\alpha_5})^{r_\ell}) \mod N^2$. Just like Lemma 25, any difference between G_4 and G_5 results in a PPT adversary solving IV_5 problem by invoking $\mathscr{A}$. Therefore, $|\Pr_4[\mathsf{Succ}] - \Pr_5[\mathsf{Succ}]| \le \mathsf{Adv}_{\mathsf{GenN}}^{\mathrm{iv}_5}(\lambda)$.

Game G_6. It is identical to G_5, except for the following differences. In the $\textsc{initialize}$ procedure of game G_6, the challenger picks $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ randomly. As for the ℓth ($\ell \in [Q_e]$) $\textsc{encrypt}$ query (f_ℓ, i_ℓ), the challenger computes $\mathsf{k}_\ell = (k_{\ell,1}, k_{\ell,2}, k_{\ell,3}, k_{\ell,4})$ and $(e_{\ell,1}, \ldots, e_{\ell,4})$ in a different way:

(i) Pick $\mathsf{s}_\ell = (s_{\ell,1}, s_{\ell,2}, s_{\ell,3}, s_{\ell,4}) \leftarrow_\$ (\mathbb{Z}_N)^4$ and $r_\ell \leftarrow_\$ [\lfloor N/4 \rfloor]$ randomly, and compute $\mathsf{k}_\ell = (k_{\ell,1}, k_{\ell,2}, k_{\ell,3}, k_{\ell,4}) := (r_\ell k_1^* + s_{\ell,1}, \ldots, r_\ell k_4^* + s_{\ell,4})$.

(ii)

$$
\begin{aligned}
(e_{\ell,1}, \ldots, e_{\ell,4}) := \Big(& h_{i_\ell,1}^{r^* r_\ell} T^{r_\ell \cdot (k_1^* - \alpha_1 x_{i_\ell,1} - \alpha_2 y_{i_\ell,1}) + s_{\ell,1}}, \ldots, \\
& h_{i_\ell,4}^{r^* r_\ell} T^{r_\ell \cdot (k_4^* - \alpha_4 x_{i_\ell,4} - \alpha_5 y_{i_\ell,4}) + s_{\ell,4}} \Big).
\end{aligned} \tag{36}
$$

Clearly k_ℓ is uniformly random over $(\mathbb{Z}_N)^4$, just like that in game G_5. In the meantime, for $j \in [4]$, we have

$$
\begin{aligned}
e_{\ell,j} &\overset{\mathsf{G}_5}{=} u_{\ell,j}^{-x_{i_\ell,j}} u_{\ell,j+1}^{-y_{i_\ell,j}} T^{k_{\ell,j}} \\
&= \left(g_j^{r^*} T^{\alpha_j}\right)^{-r_\ell \cdot x_{i_\ell,j}} \left(g_{j+1}^{r^*} T^{\alpha_{j+1}}\right)^{-r_\ell \cdot y_{i_\ell,j}} T^{k_{\ell,j}} \\
&= \left(g_j^{-x_{i_\ell,j}} g_{j+1}^{-y_{i_\ell,j}}\right)^{r^* r_\ell} T^{k_{\ell,j} - r_\ell \cdot (\alpha_j x_{i_\ell,j} + \alpha_{j+1} y_{i_\ell,j})} \\
&\overset{\mathsf{G}_6}{=} h_{i_\ell,j}^{r^* r_\ell} T^{r_\ell \cdot (k_j^* - \alpha_j x_{i_\ell,j} - \alpha_{j+1} y_{i_\ell,j}) + s_{\ell,j}} \mod N^2.
\end{aligned} \tag{37}
$$

Thus, G_6 is the same as G_5, and $\Pr_5[\mathsf{Succ}] = \Pr_6[\mathsf{Succ}]$.

Game G_7. It is identical to G_6, except the way the challenger answers the $\textsc{decrypt}$ oracle queries $(\langle \mathsf{ai}, \mathsf{aiae.c} \rangle, i \in [n])$. In game G_7, it uses $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4})$ and $\phi(N) = (p-1)(q-1)$ to decrypt $\langle \mathsf{ai}, \mathsf{aiae.c} \rangle$, where $\mathsf{ai} = (u_1, \ldots, u_5, e_1, \ldots, e_4)$. More precisely, it computes $\mathsf{k} = (k_1, \ldots, k_4)$ and m in the following way:

(i)

$$
\begin{aligned}
&(\alpha_1', \ldots, \alpha_5') \\
&\quad := \left(\frac{d \log_T\left(u_1^{\phi(N)}\right)}{\phi(N)}, \ldots, \frac{d \log_T\left(u_5^{\phi(N)}\right)}{\phi(N)} \right) \mod N,
\end{aligned}
$$

$$\left(\gamma_1', \ldots, \gamma_4'\right) := \left(\frac{d\log_T\left(e_1^{\phi(N)}\right)}{\phi(N)}, \ldots, \frac{d\log_T\left(e_4^{\phi(N)}\right)}{\phi(N)}\right) \bmod N,$$

$$\begin{aligned}\mathsf{k} &= (k_1, \ldots, k_4) \\ &:= \left(\alpha_1' x_{i,1} + \alpha_2' y_{i,1} + \gamma_1', \ldots, \alpha_4' x_{i,4} + \alpha_5' y_{i,4} + \gamma_4'\right) \bmod N,\end{aligned} \tag{38}$$

(ii)

$$\mathscr{E}.\mathsf{c} = (\tilde{u}_1, \ldots, \tilde{u}_8, \tilde{e}, t) \,/\, \perp \longleftarrow \mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai}), \tag{39}$$

(iii)

$$\begin{aligned}&(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_8) \\ &\quad := \left(\frac{d\log_T\left(\tilde{u}_1^{\phi(N)}\right)}{\phi(N)}, \ldots, \frac{d\log_T\left(\tilde{u}_8^{\phi(N)}\right)}{\phi(N)}\right) \bmod N^{s-1}, \\ &\tilde{\gamma} := \frac{d\log_T\left(\tilde{e}^{\phi(N)}\right)}{\phi(N)} \bmod N^{s-1}, \\ &m \\ &\quad := \tilde{\alpha}_1 x_{i,1} + \tilde{\alpha}_2 y_{i,1} + \tilde{\alpha}_3 x_{i,2} + \tilde{\alpha}_4 y_{i,2} + \tilde{\alpha}_5 x_{i,3} + \tilde{\alpha}_6 y_{i,3} \\ &\qquad + \tilde{\alpha}_7 x_{i,4} + \tilde{\alpha}_8 y_{i,4} + \tilde{\gamma} \mod N^{s-1}.\end{aligned} \tag{40}$$

According to (8), for $j \in [4]$, we have that

$$\begin{aligned}k_j &\overset{\mathsf{G}_6}{=} d\log_T\left(e_j u_j^{x_{i,j}} u_{j+1}^{y_{i,j}}\right) \\ &= \frac{d\log_T\left(e_j u_j^{x_{i,j}} u_{j+1}^{y_{i,j}}\right)^{\phi(N)}}{\phi(N)} \bmod N \\ &= \frac{d\log_T\left(u_j^{\phi(N)\cdot x_{i,j}}\right)}{\phi(N)} + \frac{d\log_T\left(u_{j+1}^{\phi(N)\cdot y_{i,j}}\right)}{\phi(N)} \\ &\quad + \frac{d\log_T\left(e_j^{\phi(N)}\right)}{\phi(N)} \\ &\overset{\mathsf{G}_7}{=} \underbrace{\frac{d\log_T\left(u_j^{\phi(N)}\right)}{\phi(N)}}_{\alpha_j'} \cdot x_{i,j} + \underbrace{\frac{d\log_T\left(u_{j+1}^{\phi(N)}\right)}{\phi(N)}}_{\alpha_{j+1}'} \cdot y_{i,j} \\ &\quad + \underbrace{\frac{d\log_T\left(e_j^{\phi(N)}\right)}{\phi(N)}}_{\gamma_j'},\end{aligned}$$

$$\begin{aligned}m &\overset{\mathsf{G}_6}{=} d\log_T\left(\widetilde{eu}_1^{x_{i,1}} \tilde{u}_2^{y_{i,1}} \tilde{u}_3^{x_{i,2}} \tilde{u}_4^{y_{i,2}} \tilde{u}_5^{x_{i,3}} \tilde{u}_6^{y_{i,3}} \tilde{u}_7^{x_{i,4}} \tilde{u}_8^{y_{i,4}}\right) \bmod N^{s-1} \\ &\overset{\mathsf{G}_7}{=} \underbrace{\frac{d\log_T\left(\tilde{u}_1^{\phi(N)}\right)}{\phi(N)}}_{\tilde{\alpha}_1} \cdot x_{i,1} + \cdots + \underbrace{\frac{d\log_T\left(\tilde{u}_8^{\phi(N)}\right)}{\phi(N)}}_{\tilde{\alpha}_8} \\ &\quad \cdot y_{i,4} + \underbrace{\frac{d\log_T\left(\tilde{e}^{\phi(N)}\right)}{\phi(N)}}_{\tilde{\gamma}}.\end{aligned} \tag{41}$$

Hence G_7 is essentially the same as G_6, and $\Pr_6[\mathsf{Succ}] = \Pr_7[\mathsf{Succ}]$.

Game G_8. It is identical to G_7, except the way of answering the DECRYPT oracle queries ($\langle \mathsf{ai}, \mathsf{aiae.c}\rangle$, $i \in [n]$). More precisely, a rejection rule is added in DECRYPT:

(i) If $\alpha_1' \neq 0 \vee \cdots \vee \alpha_5' \neq 0 \vee \tilde{\alpha}_1 \neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0$, output $\perp$.

Denote by Bad the event that $\mathscr{A}$ ever queries the DECRYPT oracle with ($\langle \mathsf{ai}, \mathsf{aiae.c}\rangle$, $i \in [n]$), satisfying

$$\begin{aligned}&e_1 u_1^{x_{i,1}} u_2^{y_{i,1}}, \ldots, e_4 u_4^{x_{i,4}} u_5^{y_{i,4}} \in \mathbb{RU}_{N^2} \\ &\quad \wedge \mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai}) \neq \perp\end{aligned} \tag{42}$$

$$\begin{aligned}&\wedge \widetilde{eu}_1^{x_{i,1}} \tilde{u}_2^{y_{i,1}} \tilde{u}_3^{x_{i,2}} \tilde{u}_4^{y_{i,2}} \tilde{u}_5^{x_{i,3}} \tilde{u}_6^{y_{i,3}} \tilde{u}_7^{x_{i,4}} \tilde{u}_8^{y_{i,4}} \in \mathbb{RU}_{N^s} \\ &\wedge t = g_1^m \mod N\end{aligned} \tag{43}$$

$$\wedge \left(\alpha_1' \neq 0 \vee \cdots \vee \alpha_5' \neq 0 \vee \tilde{\alpha}_1 \neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0\right). \tag{44}$$

Obviously, G_8 is identical to G_7 unless Bad occurs. Thus, $|\Pr_7[\mathsf{Succ}] - \Pr_8[\mathsf{Succ}]| \leq \Pr_8[\mathsf{Bad}]$.

To show the computational indistinguishability of G_7 and G_8, we must prove that $\Pr_8[\mathsf{Bad}]$ is negligible. To this end, Bad is divided into two subevents:

(i) Bad': $\mathscr{A}$ ever queries the DECRYPT oracle with ($\langle \mathsf{ai}, \mathsf{aiae.c}\rangle$, $i \in [n]$), satisfying

$$\text{Conditions } (42), (43) \wedge \left(\alpha_1' \neq 0 \vee \cdots \vee \alpha_5' \neq 0\right). \tag{45}$$

(ii) $\widetilde{\mathsf{Bad}}$: $\mathscr{A}$ ever queries the DECRYPT oracle with ($\langle \mathsf{ai}, \mathsf{aiae.c}\rangle$, $i \in [n]$), satisfying

$$\begin{aligned}&\text{Conditions } (42), (43) \wedge \left(\alpha_1' = \cdots = \alpha_5' = 0\right) \\ &\quad \wedge \left(\tilde{\alpha}_1 \neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0\right).\end{aligned} \tag{46}$$

Obviously, $\Pr_8[\mathsf{Bad}] \leq \Pr_8[\mathsf{Bad}'] + \Pr_8[\widetilde{\mathsf{Bad}}]$. We will defer the analysis of $\Pr_8[\widetilde{\mathsf{Bad}}]$ to subsequent games. Through the following lemma, we provide the analysis of $\Pr_8[\mathsf{Bad}']$.

Lemma 26. *One has* $\Pr_8[\mathsf{Bad}'] \leq 2Q_d \cdot \mathsf{Adv}^{\textit{weak-int-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$.

Proof. In DECRYPT of game G_8, the challenger will reply $\perp$ to $\mathscr{A}$ unless $\alpha'_1 = \cdots = \alpha'_5 = 0$ and $\tilde{\alpha}_1 = \cdots = \tilde{\alpha}_8 = 0$. Consequently, the $(\bmod \phi(N)/4)$ part of sk_i, that is, $(x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4}) \bmod \phi(N)/4$, $i \in [n]$, and the value of $\phi(N)$, is enough for answering DECRYPT queries. In particular, the values of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ are not necessary in DECRYPT.

Bad' is further divided into the following two subevents:

(i) Bad'-1: $\mathscr{A}$ ever queries the DECRYPT oracle with $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle,\ i \in [n])$, satisfying

$$\text{Conditions } (42), (43) \wedge \left(\alpha'_1 \neq 0 \vee \cdots \vee \alpha'_5 \neq 0\right) \wedge \left(\exists j \in [4], \frac{\alpha'_j}{\alpha_j} \neq \frac{\alpha'_{j+1}}{\alpha_{j+1}} \bmod N\right). \tag{47}$$

(ii) Bad'-2: $\mathscr{A}$ ever queries the DECRYPT oracle with $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle,\ i \in [n])$, satisfying

$$\text{Conditions } (42), (43) \wedge \left(\alpha'_1 \neq 0 \vee \cdots \vee \alpha'_5 \neq 0\right) \wedge \left(\frac{\alpha'_1}{\alpha_1} = \cdots = \frac{\alpha'_5}{\alpha_5} \bmod N\right). \tag{48}$$

Recall that $(\alpha_1, \ldots, \alpha_5)$ are chosen in INITIALIZE.

We will consider the two subevents in game G_8 separately via the following two claims.

Claim 27. One has $\Pr_8[\mathsf{Bad}'\text{-1}] \leq Q_d \cdot \mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$.

Proof. In game G_8, the values of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ are not needed in DECRYPT, and the computation of $t_\ell = g_1^{m_\beta} \bmod N$ in ENCRYPT only makes use of $(x_1, y_1, \ldots, x_4, y_4) \bmod \phi(N)/4$. Thus the only information about $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ leaked to $\mathscr{A}$ is through the computation of $(e_{\ell,1}, \ldots, e_{\ell,4})$ in ENCRYPT, which may leak the values of $(\alpha_1 x_1 + \alpha_2 y_1)$, $(\alpha_2 x_2 + \alpha_3 y_2)$, $(\alpha_3 x_3 + \alpha_4 y_3)$, $(\alpha_4 x_4 + \alpha_5 y_4) \bmod N$: for $j \in [4]$,

$$\begin{aligned} e_{\ell,j} &= h_{i_\ell,j}^{r^* r_\ell} T^{r_\ell \cdot (k_j^* - \alpha_j x_{i_\ell,j} - \alpha_{j+1} y_{i_\ell,j}) + s_{\ell,j}} \bmod N^2 \\ &= h_{i_\ell,j}^{r^* r_\ell} T^{r_\ell \cdot \underbrace{(k_j^* - \alpha_j x_j - \alpha_{j+1} y_j}_{\triangleq \widehat{k}_j} - \alpha_j \overline{x}_{i_\ell,j} - \alpha_{j+1} \overline{y}_{i_\ell,j}) + s_{\ell,j}} \\ &\qquad \bmod N^2. \end{aligned} \tag{49}$$

If Bad'-1 occurs, for concreteness, say that $\alpha'_1/\alpha_1 \neq \alpha'_2/\alpha_2 \bmod N$, then

$$\begin{aligned} k_1 &= \alpha'_1 x_{i,1} + \alpha'_2 y_{i,1} + \gamma'_1 \\ &= \alpha'_1 x_1 + \alpha'_2 y_1 + \alpha'_1 \overline{x}_{i,1} + \alpha'_2 \overline{y}_{i,1} + \gamma'_1 \bmod N, \end{aligned} \tag{50}$$

where k_1 is independent of $(\alpha_1 x_1 + \alpha_2 y_1) \bmod N$, thus uniformly distributed over $\mathbb{Z}_N$ from $\mathscr{A}$'s view. By Remark 23, for $\mathsf{k} = (k_1, k_2, k_3, k_4)$ where $k_1 \leftarrow_\$ \mathbb{Z}_N$, the probability of $\mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai}) \neq \perp$ is upper bounded by $\mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$.

Then $\Pr_8[\mathsf{Bad}'\text{-1}] \leq Q_d \cdot \mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$ by a union bound. □

Claim 28. One has $\Pr_8[\mathsf{Bad}'\text{-2}] \leq Q_d \cdot \mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$.

Proof. Similar to the discussion in the proof for the previous claim, in game G_8, the only information about $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ and $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ involved is through ENCRYPT, which uses the value of $\widehat{k}_1 := (k_1^* - \alpha_1 x_1 - \alpha_2 y_1)$, $\widehat{k}_2 := (k_2^* - \alpha_2 x_2 - \alpha_3 y_2)$, $\widehat{k}_3 := (k_3^* - \alpha_3 x_3 - \alpha_4 y_3)$, $\widehat{k}_4 := (k_4^* - \alpha_4 x_4 - \alpha_5 y_4) \bmod N$ via computing $(e_{\ell,1}, \ldots, e_{\ell,4})$ (see (49)) and also uses $\mathsf{k}_\ell = r_\ell \cdot (k_1^*, k_2^*, k_3^*, k_4^*) + (s_{\ell,1}, \ldots, s_{\ell,4})$ as the encryption key of $\mathsf{AIAE.Encrypt}$.

Note that because of the randomness of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$, $(\widehat{k}_1, \widehat{k}_2, \widehat{k}_3, \widehat{k}_4)$ are uniformly distributed and independent of $(k_1^*, k_2^*, k_3^*, k_4^*)$. Therefore it is possible to construct an algorithm to simulate DECRYPT and ENCRYPT of game G_8 without $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ and $(x_1, y_1, \ldots, x_4, y_4) \bmod N$. The algorithm can also simulate $\mathsf{AIAE.Encrypt}$ as long as it has access to a weak-INT-$\mathscr{F}_{\text{raff}}$-RKA encryption oracle of the $\mathsf{AIAE}_{\mathsf{DDH}}$ scheme.

More precisely, we construct a PPT adversary $\mathscr{B}_2(\mathsf{pars}_{\mathsf{AIAE}})$, which has access to $\text{ENCRYPT}_{\mathsf{AIAE}}$ oracle, against the weak-INT-$\mathscr{F}_{\text{raff}}$-RKA security of the $\mathsf{AIAE}_{\mathsf{DDH}}$ scheme, where $\mathsf{pars}_{\mathsf{AIAE}} = (N, p, q, \ldots)$. $\mathscr{B}_2$ does not choose $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ in INITIALIZE any more, and it implicitly sets k^* to be the encryption key used by its weak-INT-$\mathscr{F}_{\text{raff}}$-RKA challenger. $\mathscr{B}_2$ does not choose $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ either, and instead, it chooses $\widehat{\mathsf{k}} = (\widehat{k}_1, \widehat{k}_2, \widehat{k}_3, \widehat{k}_4)$ uniformly from $(\mathbb{Z}_N)^4$. $\mathscr{B}_2$ picks $(x_1, y_1, \ldots, x_4, y_4) \bmod \phi(N)/4$ and $(\overline{x}_{i,1}, \overline{y}_{i,1}, \ldots, \overline{x}_{i,4}, \overline{y}_{i,4}) \in [\lfloor N^2/4 \rfloor]$, $i \in [n]$, randomly. To simulate ENCRYPT, $\mathscr{B}_2$ can use $(\overline{x}_{i_\ell,j}, \overline{y}_{i_\ell,j}, \widehat{k}_j)_{j=1}^4$ to compute $(e_{\ell,j})_{j=1}^4$ via (49) and use $(\overline{x}_{i,j}, \overline{y}_{i,j})_{j=1}^4$, $i \in [n]$, to compute $\tilde{e}_\ell$. Note that $\mathscr{B}_2$ is able to compute $t_\ell = g_1^{m_\beta} \bmod N$, even if $\beta = 1$, because it knows the $(\bmod \phi(N)/4)$ part of sk_i, that is, $(x_j, y_j)_{j=1}^4 \bmod \phi(N)/4$ and $(\overline{x}_{i,j}, \overline{y}_{i,j})_{j=1}^4 \bmod \phi(N)/4$, $i \in [n]$. Then $\mathscr{B}_2$ submits $(\mathscr{E}.\mathsf{c}_\ell, \mathsf{ai}_\ell, \langle r_\ell, \mathsf{s}_\ell = (s_{\ell,1}, \ldots, s_{\ell,4})\rangle)$ to its own $\text{ENCRYPT}_{\mathsf{AIAE}}$ oracle and obtains $\mathsf{aiae.c}_\ell$. The final ciphertext is $\langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle$. According to the weak-INT-$\mathscr{F}_{\text{raff}}$-RKA security game, the $\text{ENCRYPT}_{\mathsf{AIAE}}$ oracle will encrypt $\mathscr{E}.\mathsf{c}_\ell$ with the auxiliary input ai_ℓ under the transformed key $\mathsf{k}_\ell = r_\ell \cdot \mathsf{k}^* + \mathsf{s}_\ell$; that is, the $\text{ENCRYPT}_{\mathsf{AIAE}}$ oracle behaves as $\mathsf{AIAE.Encrypt}(\mathsf{k}_\ell, \mathscr{E}.\mathsf{c}_\ell, \mathsf{ai}_\ell)$. Thus $\mathscr{B}_2$'s simulation of ENCRYPT is identical to G_8. For DECRYPT, $\mathscr{B}_2$ answers decryption queries with the $(\bmod \phi(N)/4)$ part of all the secret keys and $\phi(N) = (p-1)(q-1)$, just like G_8.

Suppose that $\mathscr{A}$ ever queries the DECRYPT oracle with $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle,\ i \in [n])$, such that Bad'-2 occurs. For concreteness, say that $r := \alpha'_1/\alpha_1 = \cdots = \alpha'_5/\alpha_5 \neq 0 \bmod N$, then for $j \in [4]$,

$$k_j = \alpha'_j x_{i,j} + \alpha'_{j+1} y_{i,j} + \gamma'_j = r \cdot \left(\alpha_j x_{i,j} + \alpha_{j+1} y_{i,j}\right) + \gamma'_j \bmod N$$

$$
\begin{aligned}
&= r \cdot k_j^* - r \cdot \left(k_j^* - \alpha_j x_{i,j} - \alpha_{j+1} y_{i,j}\right) + \gamma_j' \mod N \\
&= r \cdot k_j^* - r \\
&\quad \cdot \left(\underbrace{k_j^* - \alpha_j x_j - \alpha_{j+1} y_j}_{=\widehat{k}_j} - \alpha_j \overline{x}_{i,j} - \alpha_{j+1} \overline{y}_{i,j} \right) \\
&\quad + \gamma_j' \mod N \\
&= r \cdot k_j^* \underbrace{- r \cdot \left(\widehat{k}_j - \alpha_j \overline{x}_{i,j} - \alpha_{j+1} \overline{y}_{i,j}\right) + \gamma_j'}_{\triangleq s_j} \\
&= r \cdot k_j^* + s_j \mod N.
\end{aligned} \tag{51}
$$

Thus $\mathsf{k} = (k_1, \ldots, k_4) = r \cdot \mathsf{k}^* + \mathsf{s}$, where $\mathsf{s} := (s_1, \ldots, s_4)$. $\mathscr{B}_2$ can compute $\langle r, \mathsf{s} = (s_1, \ldots, s_4)\rangle$ as above using $(\overline{x}_{i,j}, \overline{y}_{i,j}, \widehat{k}_j)_{j=1}^4$ and outputs $(\mathsf{ai}, \langle r, \mathsf{s}\rangle, \mathsf{aiae.c})$ to its weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA challenger as a forgery. We analyze the success probability of $\mathscr{B}_2$ as follows.

(i) Firstly, a valid decryption query from $\mathscr{A}$ satisfies $\langle \mathsf{ai}, \mathsf{aiae.c}\rangle \neq \langle \mathsf{ai}_\ell, \mathsf{aiae.c}_\ell\rangle$ for all $\ell \in [Q_e]$; thus $(\mathsf{ai}, \langle r, \mathsf{s}\rangle, \mathsf{aiae.c}) \neq (\mathsf{ai}_\ell, \langle r_\ell, \mathsf{s}_\ell\rangle, \mathsf{aiae.c}_\ell)$ will hold for all $\ell \in [Q_e]$; that is, $\mathscr{B}_2$ always outputs a fresh forgery.

(ii) Secondly, if $\mathsf{ai} = \mathsf{ai}_\ell$ for some $\ell \in [Q_e]$, then it is easy to have that $\alpha_1' = \alpha_1 \cdot r_\ell, \ldots, \alpha_5' = \alpha_5 \cdot r_\ell$ and thus $r = r_\ell$. Furthermore for $j \in [4]$, it clearly holds that $\gamma_j' = r_\ell \cdot (\widehat{k}_j - \alpha_j \overline{x}_{i,j} - \alpha_{j+1} \overline{y}_{i,j}) + s_{\ell,j}$ (cf. (49)); thus $s_j = -r \cdot (\widehat{k}_j - \alpha_j \overline{x}_{i,j} - \alpha_{j+1} \overline{y}_{i,j}) + \gamma_j' = s_{\ell,j}$ and $\mathsf{s} = \mathsf{s}_\ell$. That is, if $\mathsf{ai} = \mathsf{ai}_\ell$ for some $\ell \in [Q_e]$, then it holds that $\langle r, \mathsf{s}\rangle = \langle r_\ell, \mathsf{s}_\ell\rangle$. Obviously it satisfies the special rule required for the weak-INT-$\mathscr{F}_{\mathrm{raff}}$-RKA security.

(iii) Finally, if Bad'-2 occurs in this decryption query, then $\mathsf{AIAE.Decrypt}(\mathsf{k}, \mathsf{aiae.c}, \mathsf{ai}) \neq \perp$, where $\mathsf{k} = r \cdot \mathsf{k}^* + \mathsf{s}$, will imply that $\mathscr{B}_2$'s forgery is successful.

By a union bound, we have that $\Pr_8[\mathsf{Bad}'\text{-}2] \leq Q_d \cdot \mathsf{Adv}^{\text{weak-int-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}, \mathscr{B}_2}(\lambda)$. □

In conclusion, Lemma 26 follows from the above two claims.

This completes the proof of Lemma 26.

Game G_9. It is identical to G_8, except for the following differences. In the INITIALIZE procedure of game G_9, the challenger picks an independent $\overline{\mathsf{k}}^* = (\overline{k}_1^*, \overline{k}_2^*, \overline{k}_3^*, \overline{k}_4^*) \leftarrow_\$ (\mathbb{Z}_N)^4$ besides $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$. As for the ℓth ($\ell \in [Q_e]$) ENCRYPT oracle query (f_ℓ, i_ℓ), the challenger employs a different key for $\mathsf{AIAE}_{\mathsf{DDH}}$ in the computation of $\mathsf{aiae.c}_\ell$:

(i) $\overline{\mathsf{k}}_\ell := (r_\ell \overline{k}_1^* + s_{\ell,1}, \ldots, r_\ell \overline{k}_4^* + s_{\ell,4})$;

(ii) $\mathsf{aiae.c}_\ell \leftarrow_\$ \mathsf{AIAE.Encrypt}(\overline{\mathsf{k}}_\ell, \mathscr{E}.\mathsf{c}_\ell, \mathsf{ai}_\ell)$.

We stress that the challenger still employs $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ in the computation of $(e_{\ell,1}, \ldots, e_{\ell,4})$.

In G_8, the only place that involves the value of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ is in the computation of $(e_{\ell,1}, \ldots, e_{\ell,4})$ in the ENCRYPT oracle. Specifically, for $j \in [4]$,

$$
\begin{aligned}
e_{\ell,j} &= h_{i_\ell,j}^{r^* r_\ell} T^{r_\ell \cdot (k_j^* - \alpha_j x_{i_\ell,j} - \alpha_{j+1} y_{i_\ell,j}) + s_{\ell,j}} \mod N^2 \\
&= h_{i_\ell,j}^{r^* r_\ell} T^{r_\ell \cdot (k_j^* - \alpha_j x_j - \alpha_{j+1} y_j - \alpha_j \overline{x}_{i_\ell,j} - \alpha_{j+1} \overline{y}_{i_\ell,j}) + s_{\ell,j}} \\
&\qquad \mod N^2.
\end{aligned} \tag{52}
$$

Note that the computation of $t_\ell = g_1^{m_\beta} \bmod N$ in the ENCRYPT oracle only involves $(x_1, y_1, \ldots, x_4, y_4) \bmod \phi(N)/4$. Moreover, observe that neither $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ nor $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ is used in DECRYPT. Hence, $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ is perfectly hidden by $(x_1, y_1, \ldots, x_4, y_4) \bmod N$.

Therefore, the challenger could always employ another $\overline{\mathsf{k}}^* = (\overline{k}_1^*, \ldots, \overline{k}_4^*)$ in the computation of $\overline{\mathsf{k}}_\ell$ and utilize $\overline{\mathsf{k}}_\ell$ in the $\mathsf{AIAE}_{\mathsf{DDH}}$'s encryption in the ENCRYPT oracle, as in G_9.

Then game G_8 and game G_9 are essentially the same from $\mathscr{A}$'s view, so $\Pr_8[\mathsf{Succ}] = \Pr_9[\mathsf{Succ}]$ and $\Pr_8[\widetilde{\mathsf{Bad}}] = \Pr_9[\widetilde{\mathsf{Bad}}]$.

Game G_{10}. It is identical to G_9, except the way the challenger answers the ℓth ($\ell \in [Q_e]$) ENCRYPT oracle query (f_ℓ, i_ℓ). More precisely, in game G_{10}, the challenger computes $\mathsf{aiae.c}_\ell$ in the following way:

(i) $\mathsf{aiae.c}_\ell \leftarrow_\$ \mathsf{AIAE.Encrypt}(\overline{\mathsf{k}}_\ell, 0^{\lambda_{\mathscr{M}}}, \mathsf{ai}_\ell)$.

Observe that, in G_9 and G_{10}, $\overline{\mathsf{k}}^*$ is employed only in the $\mathsf{AIAE}_{\mathsf{DDH}}$ encryption, where it uses $\overline{\mathsf{k}}_\ell = r_\ell \cdot \overline{\mathsf{k}}^* + \mathsf{s}_\ell$ as the encryption key with $\mathsf{s}_\ell = (s_{\ell,1}, \ldots, s_{\ell,4})$. Any difference between G_9 and G_{10} results in a PPT adversary against the IND-$\mathscr{F}_{\mathrm{raff}}$-RKA security of the $\mathsf{AIAE}_{\mathsf{DDH}}$ scheme. Therefore, $|\Pr_9[\mathsf{Succ}] - \Pr_{10}[\mathsf{Succ}]| \leq \mathsf{Adv}^{\text{ind-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$ and $|\Pr_9[\widetilde{\mathsf{Bad}}] - \Pr_{10}[\widetilde{\mathsf{Bad}}]| \leq \mathsf{Adv}^{\text{ind-rka}}_{\mathsf{AIAE}_{\mathsf{DDH}}}(\lambda)$.

Finally in G_{10}, the challenger always computes the $\mathsf{AIAE}_{\mathsf{DDH}}$ encryption of $0^{\lambda_{\mathscr{M}}}$ in the ENCRYPT oracle, so β is perfectly hidden from $\mathscr{A}$'s view. Thus, $\Pr_{10}[\mathsf{Succ}] = 1/2$.

To complete the proof of Theorem 24, we only need to prove the following lemma.

Lemma 29. *One has* $\Pr_{10}[\widetilde{\mathsf{Bad}}] \leq (Q_d + 1) \cdot 2^{-\Omega(\lambda)} + \mathsf{Adv}^{dl}_{\mathsf{GenN}}(\lambda)$.

Proof. In G_{10}, neither DECRYPT nor ENCRYPT uses the values of $(x_1, y_1, \ldots, x_4, y_4) \bmod \phi(N)/4$. The only information leaked about them lies in the public keys pk_i, $i \in [n]$, which reveal the values of $(w_1 x_1 + w_2 y_1)$, $(w_2 x_2 + w_3 y_2)$, $(w_3 x_3 + w_4 y_3)$, $(w_4 x_4 + w_5 y_4) \bmod \phi(N)/4$, where we denote $w_j := d \log_g g_j \bmod \phi(N)/4$ for some base $g \in \mathbb{SCR}_{N^s}$, $j \in [5]$.

$\widetilde{\mathsf{Bad}}$ is further divided into the following disjoint two subevents:

(i) $\widetilde{\mathsf{Bad}}$-1: $\mathscr{A}$ ever queries the DECRYPT oracle with $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle, i \in [n])$, satisfying

$$\begin{aligned}&\text{Conditions}\,(42), (43) \wedge \left(\alpha_1' = \cdots = \alpha_5' = 0\right) \wedge (\tilde{\alpha}_1 \\ &\neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0) \wedge \left(\frac{\tilde{\alpha}_1}{w_1} \neq \frac{\tilde{\alpha}_2}{w_2} \vee \frac{\tilde{\alpha}_3}{w_2} \neq \frac{\tilde{\alpha}_4}{w_3} \vee \frac{\tilde{\alpha}_5}{w_3}\right. \\ &\left.\neq \frac{\tilde{\alpha}_6}{w_4} \vee \frac{\tilde{\alpha}_7}{w_4} \neq \frac{\tilde{\alpha}_8}{w_5}\right).\end{aligned} \tag{53}$$

(ii) $\widetilde{\mathsf{Bad}}$-2: $\mathscr{A}$ ever queries the DECRYPT oracle with $(\langle \mathsf{ai}, \mathsf{aiae.c}\rangle, i \in [n])$, satisfying

$$\begin{aligned}&\text{Conditions}\,(42), (43) \wedge \left(\alpha_1' = \cdots = \alpha_5' = 0\right) \wedge (\tilde{\alpha}_1 \\ &\neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0) \wedge \left(\frac{\tilde{\alpha}_1}{w_1} = \frac{\tilde{\alpha}_2}{w_2} \wedge \frac{\tilde{\alpha}_3}{w_2} = \frac{\tilde{\alpha}_4}{w_3} \wedge \frac{\tilde{\alpha}_5}{w_3}\right. \\ &\left.= \frac{\tilde{\alpha}_6}{w_4} \wedge \frac{\tilde{\alpha}_7}{w_4} = \frac{\tilde{\alpha}_8}{w_5}\right).\end{aligned} \tag{54}$$

We will analyze the two subevents in game G_{10} separately via the following two claims.

Claim 30. One has $\Pr_{10}[\widetilde{\mathsf{Bad}}\text{-1}] \leq Q_d \cdot 2^{-\Omega(\lambda)}$.

Proof. If $\widetilde{\mathsf{Bad}}$-1 occurs, for concreteness, say that $\tilde{\alpha}_1/w_1 \neq \tilde{\alpha}_2/w_2$, then

$$\begin{aligned}g_1^m &= g_1^{\tilde{\alpha}_1 x_{i,1} + \tilde{\alpha}_2 y_{i,1} + \cdots} \\ &= g_1^{\tilde{\alpha}_1 x_1 + \tilde{\alpha}_2 y_1 + \tilde{\alpha}_1 \bar{x}_{i,1} + \tilde{\alpha}_2 \bar{y}_{i,1} + \cdots \bmod \phi(N)/4} \quad \bmod N,\end{aligned} \tag{55}$$

and $(\tilde{\alpha}_1 x_1 + \tilde{\alpha}_2 y_1) \bmod \phi(N)/4$ is independent of $(w_1 x_1 + w_2 y_1) \bmod \phi(N)/4$. Thus g_1^m is uniformly distributed over $\mathbb{SCR}_{N^s}$ from $\mathscr{A}$'s view, and $t = g_1^m \bmod N$ will not hold except with negligible probability $2^{-\Omega(\lambda)}$.

Then according to a union bound, $\Pr_{10}[\widetilde{\mathsf{Bad}}\text{-1}] \leq Q_d \cdot 2^{-\Omega(\lambda)}$. □

Claim 31. One has $\Pr_{10}[\widetilde{\mathsf{Bad}}\text{-2}] \leq \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GenN}}(\lambda) + 2^{-\Omega(\lambda)}$.

Proof. In game G_{10}, if $\widetilde{\mathsf{Bad}}$-2 occurs, then we can construct a PPT adversary $\mathscr{B}_3(N, p, q, g, h)$ to compute the discrete logarithm of h based on g, where $g, h \in \mathbb{SCR}_{N^s}$. With (N, p, q, g, h), $\mathscr{B}_3$ simulates INITIALIZE as follows. $\mathscr{B}_3$ picks z_j, z_j' uniformly from $[\phi(N)/4]$ and sets $g_j := g^{z_j} h^{z_j'}$ for $j \in [5]$. Then g_j is uniformly distributed over $\mathbb{SCR}_{N^s}$. Next, $\mathscr{B}_3$ samples secret keys and computes public keys just the same way as INITIALIZE in G_{10}. Since $\mathscr{B}_3$ knows all the secret keys together with $\phi(N) = (p-1)(q-1)$, $\mathscr{B}_3$ can perfectly simulates ENCRYPT and DECRYPT the same way as G_{10} does. Furthermore, z_j' is hidden by z_j perfectly from $\mathscr{A}$' view. If we denote $w := d \log_g h \bmod \phi(N)/4$, then for $j \in [5]$, $w_j = d \log_g g_j = z_j + w z_j' \bmod \phi(N)/4$.

If $\widetilde{\mathsf{Bad}}$-2 occurs in DECRYPT, for concreteness, say that $\tilde{\alpha}_1/w_1 = \tilde{\alpha}_2/w_2 \neq 0 \bmod \phi(N)/4$, that is, $g_1^{\tilde{\alpha}_2} = g_2^{\tilde{\alpha}_1} \neq 1$, then $\mathscr{B}_3$ can compute w by solving the equation $w_1 \tilde{\alpha}_2 = w_2 \tilde{\alpha}_1 \bmod \phi(N)/4$, or equivalently,

$$z_1 \tilde{\alpha}_2 + w z_1' \tilde{\alpha}_2 = z_2 \tilde{\alpha}_1 + w z_2' \tilde{\alpha}_1 \bmod \frac{\phi(N)}{4}. \tag{56}$$

Since z_j' is hidden from the point of view of $\mathscr{A}$, $(z_1' \tilde{\alpha}_2 - z_2' \tilde{\alpha}_1) \bmod \phi(N)/4$ is multiplicative invertible except with negligible probability $2^{-\Omega(\lambda)}$. Thus $\mathscr{B}_3$ will succeed in computing the discrete logarithm of h based on g and output $w = (z_1' \tilde{\alpha}_2 - z_2' \tilde{\alpha}_1)^{-1} \cdot (z_2 \tilde{\alpha}_1 - z_1 \tilde{\alpha}_2) \bmod \phi(N)/4$ to its challenger. Clearly, we have $\Pr_{10}[\widetilde{\mathsf{Bad}}\text{-2}] \leq \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GenN},\mathscr{B}_3}(\lambda) + 2^{-\Omega(\lambda)}$. □

In conclusion, Lemma 29 follows from the above two claims.

This completes the proof of Lemma 29.

In all, we proved the n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA security.

This completes the proof of Theorem 24.

5. PKE with n-KDM$[\mathscr{F}^d_{\mathrm{poly}}]$-CCA Security

5.1. The Basic Idea. We extend the construction of n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA secure PKE to that of n-KDM$[\mathscr{F}^d_{\mathrm{poly}}]$-CCA secure PKE. We allow adversaries to submit polynomial function in $\mathscr{F}^d_{\mathrm{poly}}$ in the form of *modular arithmetic circuit* (MAC) [10], which is a polynomial-sized circuit computing $f \in \mathscr{F}^d_{\mathrm{poly}}$. We stress that there is no a priori bound on the size of modular arithmetic circuits. The only requirement is that the degree d of the polynomials is a priori bounded. We still follow the approach in Figure 1 in our PKE construction. Indeed, we use the same $\mathsf{AIAE}_{\mathsf{DDH}}$ and KEM as those in the previous n-KDM$[\mathscr{F}_{\mathrm{aff}}]$-CCA secure PKE in Figure 8. We only need to construct a new $\mathscr{E}$ to serve as an entropy filter for the polynomial function set. Moreover, the new $\mathscr{E}$ should employ the same pair of public and secret keys with KEM. That is, we have $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \ldots, x_{i,4}, y_{i,4})$ and $\mathsf{pk}_i = (h_{i,1}, \ldots, h_{i,4})$ with $h_{i,1} = g_1^{-x_{i,1}} g_2^{-y_{i,1}}, \ldots, h_{i,4} = g_4^{-x_{i,4}} g_5^{-y_{i,4}} \bmod N^s$, for $i \in [n]$.

5.2. Reducing Polynomials of 8n Variables to Polynomials of 8 Variables

How to Reduce 8n-Variable Polynomial f_ℓ. In the n-$\mathsf{KDM}[\mathscr{F}^d_{\mathrm{poly}}]$-$\mathsf{CCA}$ security game, the adversary is allowed to query the ENCRYPT oracle with $(f_\ell, i_\ell \in [n])$ for $\ell \in [Q_e]$. Note that the function f_ℓ is a polynomial in the n secret keys $(x_{i,j}, y_{i,j})_{i \in [n], j \in [4]}$; thus f_ℓ has $8n$ variables and is of degree at most d. The bad news is that f_ℓ contains as many as $\binom{8n+d}{8n} = \Theta(d^{8n})$ monomial functions. Note that this number can be exponentially large.

The good news is that we found an efficient way to greatly reduce the number of monomials from $\Theta(d^{8n})$ to $\Theta(d^8)$. In particular, the polynomial $f_\ell((x_{i,j}, y_{i,j})_{i \in [n], j \in [4]})$ can always be changed to a polynomial $f_\ell'((x_{i_\ell,j}, y_{i_\ell,j})_{j \in [4]})$ of 8 variables,

consisting of at most $\binom{8+d}{8} = \Theta(d^8)$ monomial functions. Now this number is polynomial in λ.

The efficient method for reducing the $8n$-variable polynomial f_ℓ is as follows. In the INITIALIZE procedure, sk_i could be computed as $x_{i,j} := x_j + \overline{x}_{i,j}$ and $y_{i,j} := y_j + \overline{y}_{i,j} \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$ and $j \in [4]$. By using $(\overline{x}_{i,j}, \overline{y}_{i,j})_{i\in[n],j\in[4]}$, $(x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}$ could be represented as shifts of $(x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}$; that is,

$$\begin{aligned} x_{i,j} &= x_{i_\ell,j} + \overline{x}_{i,j} - \overline{x}_{i_\ell,j}, \\ y_{i,j} &= y_{i_\ell,j} + \overline{y}_{i,j} - \overline{y}_{i_\ell,j}. \end{aligned} \tag{57}$$

Consequently, f_ℓ in $8n$ variables $(x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}$ can be reduced to f'_ℓ in 8 variables $(x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}$; that is,

$$\begin{aligned} f_\ell\left(\left(x_{i,j}, y_{i,j}\right)_{i\in[n],j\in[4]}\right) &= f_\ell\left(\left(\underbrace{x_{i_\ell,j} + \overline{x}_{i,j} - \overline{x}_{i_\ell,j}}_{x_{i,j}},\right.\right. \\ &\left.\left.\underbrace{y_{i_\ell,j} + \overline{y}_{i,j} - \overline{y}_{i_\ell,j}}_{y_{i,j}}\right)_{i\in[n],j\in[4]}\right) = f'_\ell\left(\left(x_{i_\ell,j},\right.\right. \\ &\left.\left. y_{i_\ell,j}\right)_{j\in[4]}\right) = \sum_{0\le c_1+\cdots+c_8\le d} a_{(c_1,\ldots,c_8)} \\ &\cdot x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8}. \end{aligned} \tag{58}$$

The degree of the resulting polynomial f'_ℓ is still upper bounded by d. Moreover, the coefficients $a_{(c_1,\ldots,c_8)}$ of f'_ℓ are completely determined by the shifts $(\overline{x}_{i,j}, \overline{y}_{i,j})_{i\in[n],j\in[4]}$.

How to Determine Coefficients $a_{(c_1,\ldots,c_8)}$ for f'_ℓ Efficiently with Only $(\overline{x}_{i,j}, \overline{y}_{i,j})_{i\in[n],j\in[4]}$. In order to compute the coefficients $a_{(c_1,\ldots,c_8)}$ of f'_ℓ, we can repeat the following procedure:

(i) Choose $(x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}$ uniformly.

(ii) Feed modular arithmetic circuit (which functions as f_ℓ) with $(x_{i_\ell,j} + \overline{x}_{i,j} - \overline{x}_{i_\ell,j}, y_{i_\ell,j} + \overline{y}_{i,j} - \overline{y}_{i_\ell,j})_{i\in[n],j\in[4]}$ as input. We stress that $(\overline{x}_{i,j}, \overline{y}_{i,j})_{i\in[n],j\in[4]}$ are always the ones chosen in INITIALIZE.

(iii) Record the output of the circuit.

Repeating the above procedure about $\binom{8+d}{8} = \Theta(d^8)$ times, all the coefficients $a_{(c_1,\ldots,c_8)}$ can be extracted through solving a linear system of equations:

$$\begin{aligned} &f_\ell\left(\left(x_{i_\ell,j} + \overline{x}_{i,j} - \overline{x}_{i_\ell,j}, y_{i_\ell,j} + \overline{y}_{i,j} - \overline{y}_{i_\ell,j}\right)_{i\in[n],j\in[4]}\right) \\ &\quad = \sum_{0\le c_1+\cdots+c_8\le d} a_{(c_1,\ldots,c_8)} \\ &\quad\cdot x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8}. \end{aligned} \tag{59}$$

The overall time complexity for computing the coefficients $a_{(c_1,\ldots,c_8)}$ is polynomial in λ.

5.3. How to Design $\mathscr{E}$: A Warmup. To illustrate the ideas behind our construction, we take a simple case as consideration: construct $\mathscr{E}$ for a concrete type of monomial function; that is,

$$\begin{aligned} &f'_\ell\left(\left(x_{i_\ell,j}, y_{i_\ell,j}\right)_{j\in[4]}\right) \\ &\quad = a \cdot x_{i_\ell,1} y_{i_\ell,1} x_{i_\ell,2} y_{i_\ell,2} x_{i_\ell,3} y_{i_\ell,3} x_{i_\ell,4} y_{i_\ell,4}. \end{aligned} \tag{60}$$

Algorithms $\mathscr{E}$.Encrypt and $\mathscr{E}$.Decrypt are shown in Figure 9.

Security Proof. Now we sketch the proof of KDM-CCA security for this concrete type of monomial functions, that is, $a \cdot x_{i_\ell,1} y_{i_\ell,1} x_{i_\ell,2} y_{i_\ell,2} x_{i_\ell,3} y_{i_\ell,3} x_{i_\ell,4} y_{i_\ell,4}$. The proof is similar to that for Theorem 24 (cf. Table 2). The only difference lies in games G_3-G_4, which are related to the building block $\mathscr{E}$. Next, we will replace G_3-G_4 with the following hybrids (i.e., Hybrid 1–Hybrid 3), as shown in Figure 10. Concretely, the $\mathscr{E}$.Encrypt part of ENCRYPT is changed in a computationally indistinguishable way, so that it can serve as an entropy filter for this concrete monomial function, reserving the entropy of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$.

Suppose that the adversary submits $(f_\ell, i_\ell \in [n])$ to the ENCRYPT oracle. Our purpose is to eliminate the use of $(x_j, y_j)_{j=1}^4 \bmod N$ in the computation of $\mathscr{E}.\mathsf{Encrypt}(\mathsf{pk}_{i_\ell}, f_\ell((x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}))$, so the entropy of $(x_j, y_j)_{j=1}^4 \bmod N$ is reserved.

Hybrid 0. In the INITIALIZE procedure, the secret keys are computed as $x_{i,j} := x_j + \overline{x}_{i,j}$ and $y_{i,j} := y_j + \overline{y}_{i,j} \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$, $j \in [4]$. This hybrid is identical to G_2 in the proof of Theorem 24.

Hybrid 1. Using $(\overline{x}_{i,j}, \overline{y}_{i,j})_{i\in[n],j\in[4]}$, reduce $(f_\ell, i_\ell \in [n])$ to $(f'_\ell, i_\ell \in [n])$, and calculate the coefficient a of f'_ℓ, such that

$$\begin{aligned} &f'_\ell\left(\left(x_{i_\ell,j}, y_{i_\ell,j}\right)_{j\in[4]}\right) \\ &\quad = a \cdot x_{i_\ell,1} y_{i_\ell,1} x_{i_\ell,2} y_{i_\ell,2} x_{i_\ell,3} y_{i_\ell,3} x_{i_\ell,4} y_{i_\ell,4}. \end{aligned} \tag{61}$$

Hybrid 2. Implement $\mathscr{E}$.Encrypt using $\mathsf{sk}_{i_\ell} = (x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}$. This hybrid corresponds to G_3 in the proof of Theorem 24.

(i) Invoke $\mathscr{E}$.Encrypt to set up table.

(ii) Invoke $\mathscr{E}$.Decrypt to compute $\widehat{v}_0, \ldots, \widehat{v}_8$ from table.

(iii) Employ $\widehat{v}_8$ rather than $\widetilde{v}_8$ in the computation of $\widetilde{e}$, that is, $\widetilde{e} := \widehat{v}_8 \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N^s$, and compute $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N$.

Clearly, $\widehat{v}_0, \ldots, \widehat{v}_8$ computed via $\mathscr{E}$.Decrypt are the same as $\widetilde{v}_0, \ldots, \widetilde{v}_8$ computed via $\mathscr{E}$.Encrypt. Therefore, this is just a conceptual change.

Hybrid 3. This hybrid corresponds to G_4 in the proof of Theorem 24.

(i) table is computed similarly as that in $\mathscr{E}$.Encrypt, except for a small difference. More precisely, in table,

$\mathscr{E}.\mathsf{c} \leftarrow_{\$} \mathscr{E}.\mathsf{Encrypt}(\mathsf{pk} = (h_1, h_2, h_3, h_4), m)$:

For $l \in \{0, 1, \ldots, 8\}$,

$\tilde{r}_{l,1}, \tilde{r}_{l,2}, \tilde{r}_{l,3}, \tilde{r}_{l,4} \leftarrow_{\$} \left[\left\lfloor \frac{N}{4} \right\rfloor\right]$.

$(\tilde{u}_{l,1}, \ldots, \tilde{u}_{l,8}) := (g_1^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,4}}, g_5^{\tilde{r}_{l,4}}) \bmod N^s$.

$\tilde{v}_l := h_1^{\tilde{r}_{l,1}} h_2^{\tilde{r}_{l,2}} h_3^{\tilde{r}_{l,3}} h_4^{\tilde{r}_{l,4}} \bmod N^s$.

$$\mathsf{table} := \begin{array}{|c|c|c|c|} \hline \tilde{u}_{0,1} & \tilde{u}_{0,2} & \cdots & \tilde{u}_{0,8} \\ \hline \tilde{u}_{1,1} \cdot \tilde{v}_0 & \tilde{u}_{1,2} & \cdots & \tilde{u}_{1,8} \\ \hline \tilde{u}_{2,1} & \tilde{u}_{2,2} \cdot \tilde{v}_1 & \cdots & \tilde{u}_{2,8} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \tilde{u}_{8,1} & \tilde{u}_{8,2} & \cdots & \tilde{u}_{8,8} \cdot \tilde{v}_7 \\ \hline \end{array}$$

$\tilde{e} := \tilde{v}_8 \cdot T^m \bmod N^s$.

$t := g_1^m \bmod N \in \mathbb{Z}_N$.

Output $\mathscr{E}.\mathsf{c} := (\mathsf{table}, \tilde{e}, t)$.

$m/\perp \leftarrow \mathscr{E}.\mathsf{Decrypt}(\mathsf{sk} = (x_1, y_1, \ldots, x_4, y_4), \mathscr{E}.\mathsf{c})$:

Parse $\mathscr{E}.\mathsf{c} = (\mathsf{table}, \tilde{e}, t)$.

$$\text{Parse table} = \begin{array}{|c|c|c|c|} \hline \hat{u}_{0,1} & \hat{u}_{0,2} & \cdots & \hat{u}_{0,8} \\ \hline \hat{u}_{1,1} & \hat{u}_{1,2} & \cdots & \hat{u}_{1,8} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \hat{u}_{8,1} & \hat{u}_{8,2} & \cdots & \hat{u}_{8,8} \\ \hline \end{array}$$

$\hat{v}_0 := \hat{u}_{0,1}^{-x_1} \hat{u}_{0,2}^{-y_1} \hat{u}_{0,3}^{-x_2} \hat{u}_{0,4}^{-y_2} \cdots \hat{u}_{0,7}^{-x_4} \hat{u}_{0,8}^{-y_4}$.

$\hat{v}_1 := (\hat{u}_{1,1}/\hat{v}_0)^{-x_1} \hat{u}_{1,2}^{-y_1} \hat{u}_{1,3}^{-x_2} \hat{u}_{1,4}^{-y_2} \cdots \hat{u}_{1,7}^{-x_4} \hat{u}_{1,8}^{-y_4}$.

$\hat{v}_2 := \hat{u}_{2,1}^{-x_1} (\hat{u}_{2,2}/\hat{v}_1)^{-y_1} \hat{u}_{2,3}^{-x_2} \hat{u}_{2,4}^{-y_2} \cdots \hat{u}_{2,7}^{-x_4} \hat{u}_{2,8}^{-y_4}$.

$\vdots$

$\hat{v}_8 := \hat{u}_{8,1}^{-x_1} \hat{u}_{8,2}^{-y_1} \hat{u}_{8,3}^{-x_2} \hat{u}_{8,4}^{-y_2} \cdots \hat{u}_{8,7}^{-x_4} (\hat{u}_{8,8}/\hat{v}_7)^{-y_4}$.

If $\tilde{e}/\hat{v}_8 \in \mathbb{RU}_{N^s}$, $m := d \log_T(\tilde{e}/\hat{v}_8) \bmod N^{s-1}$.

If $t = g_1^m \bmod N$, Output m.

Otherwise, Output $\perp$.

FIGURE 9: $\mathscr{E}$ designed for a concrete type of monomial functions $a \cdot x_{i_\ell,1} y_{i_\ell,1} x_{i_\ell,2} y_{i_\ell,2} x_{i_\ell,3} y_{i_\ell,3} x_{i_\ell,4} y_{i_\ell,4}$.

$\mathscr{E}.\mathsf{c} \leftarrow_{\$} \mathscr{E}.\mathsf{Encrypt}(f_\ell, i_\ell \in [n])$: // Hybrid 1, Hybrid 2, Hybrid 3, Hybrid 3 (Equivalent Form)

For $l \in \{0, 1, \ldots, 8\}$,

$\tilde{r}_{l,1}, \tilde{r}_{l,2}, \tilde{r}_{l,3}, \tilde{r}_{l,4} \leftarrow_{\$} \left[\left\lfloor \frac{N}{4} \right\rfloor\right]$.

$(\tilde{u}_{l,1}, \ldots, \tilde{u}_{l,8}) := (g_1^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,4}}, g_5^{\tilde{r}_{l,4}}) \bmod N^s$.

$\tilde{v}_l := h_1^{\tilde{r}_{l,1}} h_2^{\tilde{r}_{l,2}} h_3^{\tilde{r}_{l,3}} h_4^{\tilde{r}_{l,4}} \bmod N^s$.

$$\mathsf{table} := \begin{array}{|c|c|c|c|c|} \hline \tilde{u}_{0,1} & \tilde{u}_{0,2} & \tilde{u}_{0,3} & \cdots & \tilde{u}_{0,8} \\ \hline \tilde{u}_{1,1} \cdot \tilde{v}_0 \ \boxed{\tilde{u}_{1,1} T^a \cdot \tilde{v}_0} & \tilde{u}_{1,2} & \tilde{u}_{1,3} & \cdots & \tilde{u}_{1,8} \\ \hline \tilde{u}_{2,1} & \tilde{u}_{2,2} \cdot \tilde{v}_1 & \tilde{u}_{2,3} & \cdots & \tilde{u}_{2,8} \\ \hline \tilde{u}_{3,1} & \tilde{u}_{3,2} & \tilde{u}_{3,3} \cdot \tilde{v}_2 & \cdots & \tilde{u}_{3,8} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline \tilde{u}_{8,1} & \tilde{u}_{8,2} & \tilde{u}_{8,3} & \cdots & \tilde{u}_{8,8} \cdot \tilde{v}_7 \\ \hline \end{array} \triangleq \begin{array}{|c|c|c|c|c|} \hline \hat{u}_{0,1} & \hat{u}_{0,2} & \hat{u}_{0,3} & \cdots & \hat{u}_{0,8} \\ \hline \hat{u}_{1,1} & \hat{u}_{1,2} & \hat{u}_{1,3} & \cdots & \hat{u}_{1,8} \\ \hline \hat{u}_{2,1} & \hat{u}_{2,2} & \hat{u}_{2,3} & \cdots & \hat{u}_{2,8} \\ \hline \hat{u}_{3,1} & \hat{u}_{3,2} & \hat{u}_{3,3} & \cdots & \hat{u}_{3,8} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline \hat{u}_{8,1} & \hat{u}_{8,2} & \hat{u}_{8,3} & \cdots & \hat{u}_{8,8} \\ \hline \end{array}.$$

$\hat{v}_0 := \hat{u}_{0,1}^{-x_{i_\ell,1}} \hat{u}_{0,2}^{-y_{i_\ell,1}} \hat{u}_{0,3}^{-x_{i_\ell,2}} \hat{u}_{0,4}^{-y_{i_\ell,2}} \hat{u}_{0,5}^{-x_{i_\ell,3}} \hat{u}_{0,6}^{-y_{i_\ell,3}} \hat{u}_{0,7}^{-x_{i_\ell,4}} \hat{u}_{0,8}^{-y_{i_\ell,4}} \bmod N^s$.

$\hat{v}_1 := (\hat{u}_{1,1}/\hat{v}_0)^{-x_{i_\ell,1}} \hat{u}_{1,2}^{-y_{i_\ell,1}} \hat{u}_{1,3}^{-x_{i_\ell,2}} \hat{u}_{1,4}^{-y_{i_\ell,2}} \hat{u}_{1,5}^{-x_{i_\ell,3}} \hat{u}_{1,6}^{-y_{i_\ell,3}} \hat{u}_{1,7}^{-x_{i_\ell,4}} \hat{u}_{1,8}^{-y_{i_\ell,4}} \bmod N^s$.

$\hat{v}_2 := \hat{u}_{2,1}^{-x_{i_\ell,1}} (\hat{u}_{2,2}/\hat{v}_1)^{-y_{i_\ell,1}} \hat{u}_{2,3}^{-x_{i_\ell,2}} \hat{u}_{2,4}^{-y_{i_\ell,2}} \hat{u}_{2,5}^{-x_{i_\ell,3}} \hat{u}_{2,6}^{-y_{i_\ell,3}} \hat{u}_{2,7}^{-x_{i_\ell,4}} \hat{u}_{2,8}^{-y_{i_\ell,4}} \bmod N^s$.

$\vdots$

$\hat{v}_8 := \hat{u}_{8,1}^{-x_{i_\ell,1}} \hat{u}_{8,2}^{-y_{i_\ell,1}} \hat{u}_{8,3}^{-x_{i_\ell,2}} \hat{u}_{8,4}^{-y_{i_\ell,2}} \hat{u}_{8,5}^{-x_{i_\ell,3}} \hat{u}_{8,6}^{-y_{i_\ell,3}} \hat{u}_{8,7}^{-x_{i_\ell,4}} (\hat{u}_{8,8}/\hat{v}_7)^{-y_{i_\ell,4}} \bmod N^s$.

$\tilde{e} := \tilde{v}_8 \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j \in [4]})} \bmod N^s$.

$\tilde{e} := \hat{v}_8 \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j \in [4]})} \bmod N^s$.

$\tilde{e} := \tilde{v}_8 \bmod N^s$.

$t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j \in [4]})} \bmod N \in \mathbb{Z}_N$.

Output $\mathscr{E}.\mathsf{c} := (\mathsf{table}, \tilde{e}, t)$.

FIGURE 10: Security proof of $\mathscr{E}.\mathsf{Encrypt}$ as an entropy filter for concrete monomials $a \cdot x_{i_\ell,1} y_{i_\ell,1} x_{i_\ell,2} y_{i_\ell,2} x_{i_\ell,3} y_{i_\ell,3} x_{i_\ell,4} y_{i_\ell,4}$.

the entry located in row 1 and column 1 is now computed as $\widehat{u}_{1,1} = (\widetilde{u}_{1,1}T^a) \cdot \widetilde{v}_0$ rather than $\widehat{u}_{1,1} = \widetilde{u}_{1,1} \cdot \widetilde{v}_0$. By the IV_5 assumption, this difference is computationally undetectable (see Appendix B for a formal analysis).

(ii) Invoke $\mathscr{E}$.Decrypt to compute $\widehat{v}_0, \ldots, \widehat{v}_8$ from table.

(iii) Compute $\widetilde{e} := \widehat{v}_8 \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N^s$, and $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N$.

Through a routine calculation, we have $\widehat{v}_0 = \widetilde{v}_0, \widehat{v}_1 = \widetilde{v}_1 \cdot T^{-ax_{i_\ell,1}}, \widehat{v}_2 = \widetilde{v}_2 \cdot T^{-ax_{i_\ell,1}y_{i_\ell,1}}, \ldots, \widehat{v}_8 = \widetilde{v}_8 \cdot T^{-ax_{i_\ell,1}y_{i_\ell,1}\cdots x_{i_\ell,4}y_{i_\ell,4}} = \widetilde{v}_8 \cdot T^{-f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})}$; hence $\widetilde{e} = \widehat{v}_8 \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} = \widetilde{v}_8$.

Consequently, Hybrid 3 can be implemented in an equivalent way.

Hybrid 3 (Equivalent Form). (i) table is computed similarly as that in $\mathscr{E}$.Encrypt, except for a small difference. More precisely, the entry located in row 1 and column 1 in table is now computed as $\widehat{u}_{1,1} = (\widetilde{u}_{1,1}T^a) \cdot \widetilde{v}_0$ rather than $\widehat{u}_{1,1} = \widetilde{u}_{1,1} \cdot \widetilde{v}_0$.

(ii) Compute $\widetilde{e} := \widetilde{v}_8 \bmod N^s$, and $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}) \bmod \phi(N)/4} \bmod N$.

Now $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ is not used in $\mathscr{E}$.Encrypt any more.

After these computationally indistinguishable changes, the $\mathscr{E}$.Encrypt part of the ENCRYPT oracle reserves the entropy of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$.

Similarly, we can change the DECRYPT oracle in a computationally indistinguishable way, so that $(x_j, y_j)_{j=1}^4 \bmod N$ is not involved at all. More precisely, DECRYPT uses only the $(\bmod \phi(N)/4)$ part of secret key and $\phi(N)$. This change corresponds to G_7-G_8 in the proof of Theorem 24. Loosely speaking, $\phi(N)$ is used to ensure that all entries in table are elements in $\mathbb{SCR}_{N^s}$. If this is not the case, DECRYPT rejects immediately. Consequently, the DECRYPT oracle leaks nothing about $(x_1, y_1, \ldots, x_4, y_4) \bmod N$. We can also show the computational indistinguishability of this change, through a similar analysis as that of Pr[Bad] in the proof of Theorem 24.

5.4. The General $\mathscr{E}$ Designed for $\mathscr{F}^d_{\mathrm{poly}}$. In Section 5.3, we presented the construction of $\mathscr{E}$ for a concrete type of monomial functions. Generally, a polynomial function f'_ℓ of degree d might contain as many as $\binom{8+d}{8} = \Theta(d^8)$ monomials. In order to construct a general $\mathscr{E}$ for the set $\mathscr{F}^d_{\mathrm{poly}}$ of polynomial functions, we must handle all types of monomial functions. To this end, we generate a table for each type of nonconstant monomial and associate it with a $\widetilde{v}$, which is named as a *title*. Algorithms $\mathscr{E}$.Encrypt and $\mathscr{E}$.Decrypt are shown in Figure 11.

Neglecting the coefficients of monomials, there are $\binom{8+d}{8} - 1$ types of nonconstant monomial functions whose degrees are at most d. For each nonconstant monomial type $x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8}$, we can associate it with a degree tuple $\mathsf{c} = (c_1, \ldots, c_8)$. Let $\mathcal{S}$ denote the set of all such degree tuples, that is, $\mathcal{S} := \{\mathsf{c} = (c_1, \ldots, c_8) \mid 1 \le c_1 + \cdots + c_8 \le d\}$.

For each degree tuple $\mathsf{c} = (c_1, \ldots, c_8) \in \mathcal{S}$, which corresponds to the monomial $x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8}$, we generate $\mathsf{table}^{(\mathsf{c})}$ and $\widetilde{v}^{(\mathsf{c})}$ by invoking the algorithm TableGen shown in Figure 11. Finally in $\widetilde{e}$, T^m is hidden by the product of all the titles.

Meanwhile, with the help of the secret key $\mathsf{sk} = (x_1, y_1, \ldots, x_4, y_4)$, we can recover $\widehat{v}^{(\mathsf{c})} = \widetilde{v}^{(\mathsf{c})}$ from $\mathsf{table}^{(\mathsf{c})}$ by invoking the algorithm CalculateV in Figure 11. Thus, the titles $(\widetilde{v}^{(\mathsf{c})})_{\mathsf{c}\in\mathcal{S}}$ could always be extracted from $(\mathsf{table}^{(\mathsf{c})})_{\mathsf{c}\in\mathcal{S}}$ one by one, and finally m is recovered.

Security Proof. We sketch the proof of KDM[$\mathscr{F}^d_{\mathrm{poly}}$]-CCA security for the set of polynomial functions. The proof is also similar to that for Theorem 24 (cf. Table 2). The only difference lies in games G_3-G_4. Next, we will replace G_3-G_4 with the following hybrids (Hybrid 1–Hybrid 3). Specifically, the $\mathscr{E}$.Encrypt part of ENCRYPT is changed in a computationally indistinguishable way, so that it can serve as an entropy filter for polynomial functions of degree at most d, reserving the entropy of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$.

Suppose that the adversary submits $(f_\ell, i_\ell \in [n])$ to the ENCRYPT oracle. Our purpose is to eliminate the use of $(x_j, y_j)_{j=1}^4 \bmod N$ in the computation of $\mathscr{E}.\mathsf{Encrypt}(\mathsf{pk}_{i_\ell}, f_\ell((x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}))$, so the entropy of $(x_j, y_j)_{j=1}^4 \bmod N$ is reserved.

Hybrid 0. In the INITIALIZE procedure, the secret keys are computed as $x_{i,j} := x_j + \overline{x}_{i,j}$ and $y_{i,j} := y_j + \overline{y}_{i,j} \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$, $j \in [4]$. This hybrid is identical to G_2 in the proof of Theorem 24.

Hybrid 1. Using $(\overline{x}_{i,j}, \overline{y}_{i,j})_{i\in[n],j\in[4]}$, reduce $(f_\ell, i_\ell \in [n])$ to $(f'_\ell, i_\ell \in [n])$, and compute the coefficients $a_{(c_1,\ldots,c_8)}$ of f'_ℓ, as discussed in Section 5.2. Then

$$f'_\ell\left(\left(x_{i_\ell,j}, y_{i_\ell,j}\right)_{j\in[4]}\right) = \sum_{(c_1,\ldots,c_8)\in\mathcal{S}} a_{(c_1,\ldots,c_8)} \cdot x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8} + \delta, \tag{62}$$

where δ is the constant term $a_{(0,\ldots,0)}$ of f'_ℓ.

Hybrid 2. Implement $\mathscr{E}$.Encrypt using $\mathsf{sk}_{i_\ell} = (x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}$. This hybrid corresponds to G_3 in the proof of Theorem 24.

(i) For each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathcal{S}$

(1) invoke $(\mathsf{table}^{(\mathsf{c})}, \widetilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}_{i_\ell}, \mathsf{c})$,

(2) invoke $\widehat{v}^{(\mathsf{c})} \leftarrow \mathsf{CalculateV}(\mathsf{sk}_{i_\ell}, \mathsf{table}^{(\mathsf{c})}, \mathsf{c})$.

(ii) Employ $(\widehat{v}^{(\mathsf{c})})_{\mathsf{c}\in\mathcal{S}}$ rather than $(\widetilde{v}^{(\mathsf{c})})_{\mathsf{c}\in\mathcal{S}}$ in the computation of $\widetilde{e}$, that is, $\widetilde{e} := \prod_{\mathsf{c}\in\mathcal{S}} \widehat{v}^{(\mathsf{c})} \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N^s$, and compute $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N$.

$\mathscr{E}.\mathsf{c} \leftarrow_{\$} \mathscr{E}.\mathsf{Encrypt}(\mathsf{pk}, m)$:	$m/\perp \leftarrow \mathscr{E}.\mathsf{Decrypt}(\mathsf{sk}, \mathscr{E}.\mathsf{c})$:
For each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathscr{S}$	Parse $\mathscr{E}.\mathsf{c} := ((\mathsf{table}^{(\mathsf{c})})_{c \in \mathscr{S}}, \tilde{e}, t)$.
$(\mathsf{table}^{(\mathsf{c})}, \tilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}, \mathsf{c})$.	For each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathscr{S}$
$\tilde{e} := \prod_{\mathsf{c} \in \mathscr{S}} \tilde{v}^{(\mathsf{c})} \cdot T^m \bmod N^s$.	$\hat{v}^{(\mathsf{c})} \leftarrow \mathsf{CalculateV}(\mathsf{sk}, \mathsf{table}^{(\mathsf{c})}, \mathsf{c})$.
$t := g_1^m \bmod N \in \mathbb{Z}_N$.	If $\tilde{e} \cdot (\prod_{\mathsf{c} \in \mathscr{S}} \hat{v}^{(\mathsf{c})})^{-1} \in \mathbb{RU}_{N^s}$
Output $\mathscr{E}.\mathsf{c} := ((\mathsf{table}^{(\mathsf{c})})_{\mathsf{c} \in \mathscr{S}}, \tilde{e}, t)$.	$m := d \log_T(\tilde{e} \cdot (\prod_{\mathsf{c} \in \mathscr{S}} \hat{v}^{(\mathsf{c})})^{-1}) \bmod N^{s-1}$.
	If $t = g_1^m \bmod N$, Output m.
	Otherwise, Output $\perp$.

(a)

$\mathsf{TableGen}(\mathsf{pk} = (h_1, h_2, h_3, h_4), \mathsf{c} = (c_1, \ldots, c_8))$:

For each $l \in \{0, 1, \ldots, \sum_{j=1}^{8} c_j\}$

$\tilde{r}_{l,1}, \tilde{r}_{l,2}, \tilde{r}_{l,3}, \tilde{r}_{l,4} \leftarrow_{\$} \left[\left\lfloor \frac{N}{4} \right\rfloor\right]$.

$(\tilde{u}_{l,1}, \ldots, \tilde{u}_{l,8}) := (g_1^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,4}}, g_5^{\tilde{r}_{l,4}}) \bmod N^s$.

$\tilde{v}_l := h_1^{\tilde{r}_{l,1}} h_2^{\tilde{r}_{l,2}} h_3^{\tilde{r}_{l,3}} h_4^{\tilde{r}_{l,4}} \bmod N^s$.

$\mathsf{table}^{(\mathsf{c})} :=$

$\tilde{u}_{0,1}$	$\tilde{u}_{0,2}$	$\cdots$	$\tilde{u}_{0,8}$	
$\tilde{u}_{1,1} \cdot \tilde{v}_0$	$\tilde{u}_{1,2}$	$\cdots$	$\tilde{u}_{1,8}$	c_1 rows
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\tilde{u}_{c_1,1} \cdot \tilde{v}_{c_1-1}$	$\tilde{u}_{c_1,2}$	$\cdots$	$\tilde{u}_{c_1,8}$	
$\tilde{u}_{c_1+1,1}$	$\tilde{u}_{c_1+1,1} \cdot \tilde{v}_{c_1}$	$\cdots$	$\tilde{u}_{c_1+1,8}$	c_2 rows
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\tilde{u}_{c_1+c_2,1}$	$\tilde{u}_{c_1+c_2,2} \cdot \tilde{v}_{c_1+c_2-1}$	$\cdots$	$\tilde{u}_{c_1+c_2,8}$	
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\tilde{u}_{\sum_{j=1}^{7} c_j+1,1}$	$\tilde{u}_{\sum_{j=1}^{7} c_j+1,2}$	$\cdots$	$\tilde{u}_{\sum_{j=1}^{7} c_j+1,8} \cdot \tilde{v}_{\sum_{j=1}^{7} c_j}$	c_8 rows
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
$\tilde{u}_{\sum_{j=1}^{8} c_j,1}$	$\tilde{u}_{\sum_{j=1}^{8} c_j,2}$	$\cdots$	$\tilde{u}_{\sum_{j=1}^{8} c_j,8} \cdot \tilde{v}_{\sum_{j=1}^{8} c_j-1}$	

Output $(\mathsf{table}^{(\mathsf{c})}, \tilde{v}^{(\mathsf{c})} := \tilde{v}_{\sum_{j=1}^{8} c_j})$.

(b)

$\mathsf{CalculateV}(\mathsf{sk} = (x_1, y_1 \ldots, x_4, y_4), \mathsf{table}^{(\mathsf{c})}, \mathsf{c} = (c_1, \ldots, c_8))$:

Parse $\mathsf{table}^{(\mathsf{c})} = \{\boxed{\hat{u}_{l,1} \mid \hat{u}_{l,2} \mid \cdots \mid \hat{u}_{l,8}}\}_{l \in \{0,1,\ldots,\sum_{j=1}^{8} c_j\}}$.

$\hat{v}_0 := \hat{u}_{0,1}^{-x_1} \hat{u}_{0,2}^{-y_1} \hat{u}_{0,3}^{-x_2} \hat{u}_{0,4}^{-y_2} \hat{u}_{0,5}^{-x_3} \hat{u}_{0,6}^{-y_3} \hat{u}_{0,7}^{-x_4} \hat{u}_{0,8}^{-y_4}$.

For each $l \in \{1, \ldots, c_1\}$

$\hat{v}_l := (\hat{u}_{l,1}/\hat{v}_{l-1})^{-x_1} \hat{u}_{l,2}^{-y_1} \hat{u}_{l,3}^{-x_2} \hat{u}_{l,4}^{-y_2} \hat{u}_{l,5}^{-x_3} \hat{u}_{l,6}^{-y_3} \hat{u}_{l,7}^{-x_4} \hat{u}_{l,8}^{-y_4}$.

For each $l \in \{c_1 + 1, \ldots, c_1 + c_2\}$

$\hat{v}_l := \hat{u}_{l,1}^{-x_1} (\hat{u}_{l,2}/\hat{v}_{l-1})^{-y_1} \hat{u}_{l,3}^{-x_2} \hat{u}_{l,4}^{-y_2} \hat{u}_{l,5}^{-x_3} \hat{u}_{l,6}^{-y_3} \hat{u}_{l,7}^{-x_4} \hat{u}_{l,8}^{-y_4}$.

$\vdots$

For each $l \in \{\sum_{j=1}^{7} c_j + 1, \ldots, \sum_{j=1}^{8} c_j\}$

$\hat{v}_l := \hat{u}_{l,1}^{-x_1} \hat{u}_{l,2}^{-y_1} \hat{u}_{l,3}^{-x_2} \hat{u}_{l,4}^{-y_2} \hat{u}_{l,5}^{-x_3} \hat{u}_{l,6}^{-y_3} \hat{u}_{l,7}^{-x_4} (\hat{u}_{l,8}/\hat{v}_{l-1})^{-y_4}$.

Output $\hat{v}^{(\mathsf{c})} := \hat{v}_{\sum_{j=1}^{8} c_j}$.

(c)

Figure 11: (a) $\mathscr{E}.\mathsf{Encrypt}$ (left) and $\mathscr{E}.\mathsf{Decrypt}$ (right) of $\mathscr{E}$ designed for $\mathscr{F}^d_{\mathrm{poly}}$; (b) $\mathsf{TableGen}$, which generates $\mathsf{table}^{(\mathsf{c})}$ together with a title $\tilde{v}^{(\mathsf{c})}$; (c) $\mathsf{CalculateV}$, which calculates a title $\hat{v}^{(\mathsf{c})}$ from $\mathsf{table}^{(\mathsf{c})}$ using secret key.

Clearly, for each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathcal{S}$, $\widehat{v}^{(\mathsf{c})}$ computed via $\mathsf{CalculateV}$ is the same as $\widetilde{v}^{(\mathsf{c})}$ computed via $\mathsf{TableGen}$. Therefore, this change is just conceptual.

Hybrid 3. This hybrid corresponds to G_4 in the proof of Theorem 24.

(i) For each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathcal{S}$

(1) $\mathsf{table}^{(\mathsf{c})}$ is computed by $(\mathsf{table}^{(\mathsf{c})}, \widetilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}_{i_\ell}, \mathsf{c})$, except for a small difference; more precisely, in $\mathsf{table}^{(\mathsf{c})}$, the entry located in row 1 and column $j := \min\{i \mid 1 \le i \le 8, c_i \ne 0\}$ is now computed as $\widehat{u}_{1,j} = (\widetilde{u}_{1,j} T^{a_{(c_1,\ldots,c_8)}}) \cdot \widetilde{v}_0$ rather than $\widehat{u}_{1,j} = \widetilde{u}_{1,j} \cdot \widetilde{v}_0$; by the IV_5 assumption, this difference is computationally undetectable,

(2) extract $\widehat{v}^{(\mathsf{c})}$ from the (modified) $\mathsf{table}^{(\mathsf{c})}$ by invoking $\widehat{v}^{(\mathsf{c})} \leftarrow \mathsf{CalculateV}(\mathsf{sk}_{i_\ell}, \mathsf{table}^{(\mathsf{c})}, \mathsf{c})$.

(ii) Compute $\widetilde{e} := \prod_{\mathsf{c}\in\mathcal{S}} \widehat{v}^{(\mathsf{c})} \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N^s$, and $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \bmod N$.

Through a routine calculation, for each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathcal{S}$, we have

$$\widehat{v}^{(\mathsf{c})} = \widetilde{v}^{(\mathsf{c})} \cdot T^{-a_{(c_1,\ldots,c_8)}} x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8}. \tag{63}$$

Hence,

$$\begin{aligned}\widetilde{e} &= \prod_{\mathsf{c}\in\mathcal{S}} \widehat{v}^{(\mathsf{c})} \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} \\ &= \prod_{\mathsf{c}\in\mathcal{S}} \widetilde{v}^{(\mathsf{c})} \cdot \prod_{\mathsf{c}\in\mathcal{S}} T^{-a_{(c_1,\ldots,c_8)}} x_{i_\ell,1}^{c_1} y_{i_\ell,1}^{c_2} x_{i_\ell,2}^{c_3} y_{i_\ell,2}^{c_4} x_{i_\ell,3}^{c_5} y_{i_\ell,3}^{c_6} x_{i_\ell,4}^{c_7} y_{i_\ell,4}^{c_8} \\ &\quad \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]})} = \prod_{\mathsf{c}\in\mathcal{S}} \widetilde{v}^{(\mathsf{c})} \cdot T^{\delta}.\end{aligned} \tag{64}$$

Consequently, Hybrid 3 can be implemented in an equivalent way.

Hybrid 3 (Equivalent Form). (i) For each $\mathsf{c} = (c_1, \ldots, c_8) \in \mathcal{S}$

$\mathsf{table}^{(\mathsf{c})}$ is computed by $(\mathsf{table}^{(\mathsf{c})}, \widetilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}_{i_\ell}, \mathsf{c})$, except for a small difference. More precisely, in $\mathsf{table}^{(\mathsf{c})}$, the entry located in row 1 and column $j := \min\{i \mid 1 \le i \le 8, c_i \ne 0\}$ is now computed as $\widehat{u}_{1,j} = (\widetilde{u}_{1,j} T^{a_{(c_1,\ldots,c_8)}}) \cdot \widetilde{v}_0$ rather than $\widehat{u}_{1,j} = \widetilde{u}_{1,j} \cdot \widetilde{v}_0$.

(ii) Compute $\widetilde{e} := \prod_{\mathsf{c}\in\mathcal{S}} \widetilde{v}^{(\mathsf{c})} \cdot T^{\delta} \bmod N^s$, and $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j\in[4]}) \bmod \phi(N)/4} \bmod N$.

Now $(x_1, y_1, \ldots, x_4, y_4) \bmod N$ is not used in $\mathcal{E}.\mathsf{Encrypt}$ any more.

After these computationally indistinguishable changes, the $\mathcal{E}.\mathsf{Encrypt}$ part of the $\textsc{encrypt}$ oracle reserves the entropy of $(x_1, y_1, \ldots, x_4, y_4) \bmod N$.

With a similar argument as that in Section 5.3, we can change the $\textsc{decrypt}$ oracle in a computationally indistinguishable way, so that $(x_j, y_j)_{j=1}^4 \bmod N$ is not employed at all.

Appendix

A. Proof of Claim 19

We build a PPT adversary $\mathcal{B}$ against the INT-OT security of AE. Suppose that the INT-OT challenger picks a key $\widehat{\kappa} \leftarrow_{\$} \mathcal{K}$ randomly. $\mathcal{B}$ is given $\mathsf{pars}_{\mathsf{AE}}$ and has access to the oracle $\textsc{encrypt}_{\mathsf{AE}}(\cdot) = \mathsf{AE.Encrypt}(\widehat{\kappa}, \cdot)$ for one time.

Firstly, $\mathcal{B}$ prepares $\mathsf{pars}_{\mathsf{AIAE}}$ in the same way as in $\mathsf{G}'_{1,j}$. That is, invoke $\mathsf{pars}_{\mathsf{THPS}} \leftarrow_{\$} \mathsf{THPS.Setup}(1^\lambda)$, pick $\mathsf{H} \leftarrow_{\$} \mathcal{H}$ randomly, and set $\mathsf{pars}_{\mathsf{AIAE}} := (\mathsf{pars}_{\mathsf{THPS}}, \mathsf{pars}_{\mathsf{AE}}, \mathsf{H})$. $\mathcal{B}$ sends $\mathsf{pars}_{\mathsf{AIAE}}$ to $\mathcal{A}$. Besides, $\mathcal{B}$ chooses $\mathsf{hk} \leftarrow_{\$} \mathcal{HK}$.

As for the ℓth ($\ell \in [Q_e]$) $\textsc{encrypt}$ query $(m_\ell, \mathsf{ai}_\ell, f_\ell)$, where $f_\ell = \langle a_\ell, \mathsf{b}_\ell \rangle \in \mathcal{F}_{\mathrm{raff}}$, $\mathcal{B}$ prepares the challenge ciphertext $\langle C_\ell, \chi_\ell \rangle$ in the following way.

(i) If $\ell \in [j-1]$, $\mathcal{B}$ computes $\langle C_\ell, \chi_\ell \rangle$ just like that in $\mathsf{G}'_{1,j}$. That is, $\mathcal{B}$ picks $C_\ell \leftarrow_{\$} \mathcal{V}$ with witness w_ℓ, chooses $\kappa_\ell \leftarrow_{\$} \mathcal{K}$, and invokes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, m_\ell)$.

(ii) If $\ell \in [j+1, Q_e]$, $\mathcal{B}$ computes $\langle C_\ell, \chi_\ell \rangle$ just like that in $\mathsf{G}'_{1,j}$. That is, $\mathcal{B}$ picks $C_\ell \leftarrow_{\$} \mathcal{V}$ with witness w_ℓ, computes $t_\ell := \mathsf{H}(C_\ell, \mathsf{ai}_\ell)$ and $\kappa_\ell := \Lambda_{a_\ell \cdot \mathsf{hk} + \mathsf{b}_\ell}(C_\ell, t_\ell)$, and invokes $\chi_\ell \leftarrow_{\$} \mathsf{AE.Encrypt}(\kappa_\ell, m_\ell)$.

(iii) If $\ell = j$, $\mathcal{B}$ does not use the key hk at all, and instead, it will resort to its own $\textsc{encrypt}_{\mathsf{AE}}(\cdot)$ oracle. More precisely, $\mathcal{B}$ picks $C_j \leftarrow_{\$} \mathcal{C} \setminus \mathcal{V}$ randomly and computes $t_j := \mathsf{H}(C_j, \mathsf{ai}_j)$. Then $\mathcal{B}$ implicitly sets $\kappa_j = \widehat{\kappa}$ as the key used by its challenger and queries its $\textsc{encrypt}_{\mathsf{AE}}(\cdot)$ oracle with m_j and gets the challenge χ_j.

According to the $\textsc{encrypt}_{\mathsf{AE}}(\cdot)$ oracle, we have $\chi_j \leftarrow_{\$} \mathsf{AE.Encrypt}(\widehat{\kappa}, m_j)$. As discussed in the proof of Lemma 18, κ_j is uniformly random in $\mathsf{G}'_{1,j}$. Therefore, the simulation of $\mathcal{B}$ is the same as that in $\mathsf{G}'_{1,j}$.

$\mathcal{B}$ outputs the challenge ciphertext $\langle C_\ell, \chi_\ell \rangle$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ puts $(\mathsf{ai}_\ell, f_\ell, \langle C_\ell, \chi_\ell \rangle)$ to $\mathcal{Q}_{\mathcal{ENC}}$, $(\mathsf{ai}_\ell, f_\ell)$ to $\mathcal{Q}_{\mathcal{AI\text{-}F}}$, and $(C_\ell, \mathsf{ai}_\ell, t_\ell)$ to $\mathcal{Q}_{\mathcal{TAG}}$.

Finally, $\mathcal{A}$ sends a forgery $(\mathsf{ai}^*, f^*, \langle C^*, \chi^* \rangle)$ to $\mathcal{B}$, with $f^* = \langle a^*, \mathsf{b}^* \rangle \in \mathcal{F}_{\mathrm{raff}}$. $\mathcal{B}$ prepares its own forgery with respect to the AE scheme as follows.

(i) If $(\mathsf{ai}^*, f^*, \langle C^*, \chi^* \rangle) \in \mathcal{Q}_{\mathcal{ENC}}$, $\mathcal{B}$ aborts the game.

(ii) If $\exists(\mathsf{ai}_\ell, f_\ell) \in \mathcal{Q}_{\mathcal{AI\text{-}F}}$ such that $\mathsf{ai}_\ell = \mathsf{ai}^*$ but $f_\ell \ne f^*$, $\mathcal{B}$ aborts the game.

(iii) If $C^* \notin \mathcal{C}$, $\mathcal{B}$ aborts the game.

(iv) $\mathcal{B}$ computes $t^* := \mathsf{H}(C^*, \mathsf{ai}^*) \in \mathcal{T}$.

(v) If $\exists(C_\ell, \mathsf{ai}_\ell, t_\ell) \in \mathcal{Q}_{\mathcal{TAG}}$ such that $t_\ell = t^*$ but $(C_\ell, \mathsf{ai}_\ell) \ne (C^*, \mathsf{ai}^*)$, $\mathcal{B}$ aborts the game.

(vi) If $t^* \ne t_j$, $\mathcal{B}$ aborts the game. If $t^* = t_j$, $\mathcal{B}$ outputs χ^* to its INT-OT challenger.

We analyze $\mathscr{B}$'s success probability. As discussed in the proof of Lemma 18, the subevent $\mathsf{Forge} \wedge t_j = t^*$ will imply that $(\mathsf{ai}^*, f^*, C^*) = (\mathsf{ai}_j, f_j, C_j)$, $\chi^* \neq \chi_j$, $\kappa^* = \kappa_j$, and $\mathsf{AE.Decrypt}(\kappa^*, \chi^*) \neq \perp$. Since $\mathscr{B}$ implicitly sets $\kappa_j = \widehat{\kappa}$ as the key used by its challenger, then $\chi^* \neq \chi_j$, $\kappa^* = \kappa_j$, and $\mathsf{AE.Decrypt}(\kappa^*, \chi^*) \neq \perp$ implies that $\chi^* \neq \chi_j$ and $\mathsf{AE.Decrypt}(\widehat{\kappa}, \chi^*) \neq \perp$; that is, the χ^* output by $\mathscr{B}$ is a fresh forgery.

In summary, $\mathscr{B}$ perfectly simulates $\mathsf{G}'_{1,j}$ for $\mathscr{A}$ and outputs a fresh forgery as long as the subevent $\mathsf{Forge} \wedge t_j = t^*$ occurs. Thus, we have that $\Pr_{1,j'}[\mathsf{Forge} \wedge t_j = t^*] \leq \mathsf{Adv}^{\text{int-ot}}_{\mathsf{AE},\mathscr{B}}(\lambda)$. This completes the proof of Claim 19.

B. Proof of Indistinguishability between Hybrids 2 and 3 in Section 5.3

To show the indistinguishability between Hybrids 2 and 3, we build a PPT adversary $\mathscr{B}^{\text{CHAL}^b_{\text{IV}_5}}(N, g_1, \ldots, g_5)$ to solve the IV_5 problem. Firstly, $\mathscr{B}$ generates secret and public keys in INITIALIZE as Hybrid 0 does. When $\mathscr{A}$ submits an encryption query $(f_\ell, i_\ell \in [n])$, $\mathscr{B}$ reduces $(f_\ell, i_\ell \in [n])$ to $(f'_\ell, i_\ell \in [n])$ as Hybrid 1 does and obtains the coefficient a. Then $\mathscr{B}$ simulates $\mathscr{E}$.Encrypt as follows.

(i) For the 0th row of table, $\mathscr{B}$ computes $(\widetilde{u}_{0,1}, \ldots, \widetilde{u}_{0,8})$ and $\widetilde{v}_0$ as in Hybrids 2 and 3.

(ii) For the 1st row, $\mathscr{B}$ queries its own $\text{CHAL}^b_{\text{IV}_5}$ oracle with $(a, 0, *, *, *)$ and obtains its challenge $(\widetilde{u}^*_{1,1}, \widetilde{u}^*_{1,2}, \widetilde{*}, \widetilde{*}, \widetilde{*})$; that is,

$$\text{Case } (b = 0): (\widetilde{u}^*_{1,1}, \widetilde{u}^*_{1,2}) = (g_1^{\widetilde{r}_{1,1}}, g_2^{\widetilde{r}_{1,1}}) = (\widetilde{u}_{1,1}, \widetilde{u}_{1,2}) \text{ or}$$

$$\text{Case } (b = 1): (\widetilde{u}^*_{1,1}, \widetilde{u}^*_{1,2}) = (g_1^{\widetilde{r}_{1,1}} T^a, g_2^{\widetilde{r}_{1,1}}) = (\widetilde{u}_{1,1} T^a, \widetilde{u}_{1,2}).$$

$\mathscr{B}$ sets $\widehat{u}_{1,1} := \widetilde{u}^*_{1,1} \cdot \widetilde{v}_0$, which is $\widehat{u}_{1,1} = \widetilde{u}_{1,1} \cdot \widetilde{v}_0$ if $b = 0$ and $\widehat{u}_{1,1} = \widetilde{u}_{1,1} T^a \cdot \widetilde{v}_0$ if $b = 1$. Then $\mathscr{B}$ generates the remaining elements $(\widetilde{u}_{1,3}, \ldots, \widetilde{u}_{1,8})$ in the 1st row of table using its public keys and sets the 1st row of table to be

$$\boxed{\widehat{u}_{1,1} = \widetilde{u}^*_{1,1} \cdot \widetilde{v}_0 \mid \widetilde{u}^*_{1,2} \mid \widetilde{u}_{1,3} \mid \cdots \mid \widetilde{u}_{1,8}}. \tag{B.1}$$

$\mathscr{B}$ also computes $\widetilde{v}^*_1$ from $(\widetilde{u}^*_{1,1}, \widetilde{u}^*_{1,2}, \widetilde{u}_{1,3}, \ldots, \widetilde{u}_{1,8})$ via $\widetilde{v}^*_1 := \widetilde{u}_{1,1}^{*-x_{i_\ell,1}} \widetilde{u}_{1,2}^{*-y_{i_\ell,1}} \widetilde{u}_{1,3}^{-x_{i_\ell,2}} \cdots \widetilde{u}_{1,8}^{-y_{i_\ell,4}}$, which equals

$$\text{Case } (b = 0): \widetilde{v}^*_1 = \widetilde{v}_1 \text{ or}$$
$$\text{Case } (b = 1): \widetilde{v}^*_1 = \widetilde{v}_1 T^{-a \cdot x_{i_\ell,1}}.$$

(iii) For the 2nd row, $\mathscr{B}$ queries its own $\text{CHAL}^b_{\text{IV}_5}$ oracle with $(0, a \cdot x_{i_\ell,1}, *, *, *)$; remember that $\mathscr{B}$ has the secret keys and obtains its challenge $(\widetilde{u}^*_{2,1}, \widetilde{u}^*_{2,2}, \widetilde{*}, \widetilde{*}, \widetilde{*})$; that is,

$$\text{Case } (b = 0): (\widetilde{u}^*_{2,1}, \widetilde{u}^*_{2,2}) = (g_1^{\widetilde{r}_{2,1}}, g_2^{\widetilde{r}_{2,1}}) = (\widetilde{u}_{2,1}, \widetilde{u}_{2,2}) \text{ or}$$

$$\text{Case } (b = 1): (\widetilde{u}^*_{2,1}, \widetilde{u}^*_{2,2}) = (g_1^{\widetilde{r}_{2,1}}, g_2^{\widetilde{r}_{2,1}} T^{a \cdot x_{i_\ell,1}}) = (\widetilde{u}_{2,1}, \widetilde{u}_{2,2} T^{a \cdot x_{i_\ell,1}}).$$

$\mathscr{B}$ sets $\widehat{u}_{2,2} := \widetilde{u}^*_{2,2} \cdot \widetilde{v}^*_1$; that is, $\widehat{u}_{2,2} = \widetilde{u}_{2,2} \cdot \widetilde{v}_1$ if $b = 0$ and $\widehat{u}_{2,2} = (\widetilde{u}_{2,2} T^{a \cdot x_{i_\ell,1}})(\widetilde{v}_1 T^{-a \cdot x_{i_\ell,1}}) = \widetilde{u}_{2,2} \cdot \widetilde{v}_1$ if $b = 1$. Thus $\widehat{u}_{2,2} = \widetilde{u}_{2,2} \cdot \widetilde{v}_1$ in both cases. Then $\mathscr{B}$ generates the remaining elements $(\widetilde{u}_{2,3}, \ldots, \widetilde{u}_{2,8})$ in the 2nd row of table using its public keys and sets the 2nd row of table to be

$$\boxed{\widetilde{u}^*_{2,1} \mid \widehat{u}_{2,2} = \widetilde{u}^*_{2,2} \cdot \widetilde{v}^*_1 \mid \widetilde{u}_{2,3} \mid \cdots \mid \widetilde{u}_{2,8}}. \tag{B.2}$$

$\mathscr{B}$ also computes $\widetilde{v}^*_2$ from $(\widetilde{u}^*_{2,1}, \widetilde{u}^*_{2,2}, \widetilde{u}_{2,3}, \ldots, \widetilde{u}_{2,8})$ via $\widetilde{v}^*_2 := \widetilde{u}_{2,1}^{*-x_{i_\ell,1}} \widetilde{u}_{2,2}^{*-y_{i_\ell,1}} \widetilde{u}_{2,3}^{-x_{i_\ell,2}} \cdots \widetilde{u}_{2,8}^{-y_{i_\ell,4}}$, which equals

$$\text{Case } (b = 0): \widetilde{v}^*_2 = \widetilde{v}_2 \text{ or}$$
$$\text{Case } (b = 1): \widetilde{v}^*_2 = \widetilde{v}_2 T^{-a \cdot x_{i_\ell,1} y_{i_\ell,1}}.$$

(iv) For the 3rd row, $\mathscr{B}$ queries its own $\text{CHAL}^b_{\text{IV}_5}$ oracle with $(*, a \cdot x_{i_\ell,1} y_{i_\ell,1}, 0, *, *)$ and obtains its challenge $(\widetilde{*}, \widetilde{u}^*_{3,3}, \widetilde{u}^*_{3,4}, \widetilde{*}, \widetilde{*})$; that is,

$$\text{Case } (b = 0): (\widetilde{u}^*_{3,3}, \widetilde{u}^*_{3,4}) = (g_2^{\widetilde{r}_{3,2}}, g_3^{\widetilde{r}_{3,2}}) = (\widetilde{u}_{3,3}, \widetilde{u}_{3,4}) \text{ or}$$

$$\text{Case } (b = 1): (\widetilde{u}^*_{3,3}, \widetilde{u}^*_{3,4}) = (g_2^{\widetilde{r}_{3,2}} T^{a \cdot x_{i_\ell,1} y_{i_\ell,1}}, g_3^{\widetilde{r}_{3,2}}) = (\widetilde{u}_{3,3} T^{a \cdot x_{i_\ell,1} y_{i_\ell,1}}, \widetilde{u}_{3,4}).$$

$\mathscr{B}$ sets $\widehat{u}_{3,3} := \widetilde{u}^*_{3,3} \cdot \widetilde{v}^*_2$; similarly, it is easy to check that $\widehat{u}_{3,3} = \widetilde{u}_{3,3} \cdot \widetilde{v}_2$ in both cases. Then $\mathscr{B}$ generates the remaining elements in the 3rd row of table using its public keys and sets the 3rd row of table to be

$$\boxed{\widetilde{u}_{3,1} \mid \widetilde{u}_{3,2} \mid \widehat{u}_{3,3} = \widetilde{u}^*_{3,3} \cdot \widetilde{v}^*_2 \mid \widetilde{u}^*_{3,4} \mid \widetilde{u}_{3,5} \mid \cdots \mid \widetilde{u}_{3,8}}. \tag{B.3}$$

$\mathscr{B}$ also computes $\widetilde{v}^*_3$ from $(\widetilde{u}_{3,1}, \widetilde{u}_{3,2}, \widetilde{u}^*_{3,3}, \widetilde{u}^*_{3,4}, \widetilde{u}_{3,5}, \ldots, \widetilde{u}_{3,8})$ via $\widetilde{v}^*_3 := \widetilde{u}_{3,1}^{-x_{i_\ell,1}} \widetilde{u}_{3,2}^{-y_{i_\ell,1}} \widetilde{u}_{3,3}^{*-x_{i_\ell,2}} \widetilde{u}_{3,4}^{*-y_{i_\ell,2}} \widetilde{u}_{3,5}^{-x_{i_\ell,3}} \cdots \widetilde{u}_{3,8}^{-y_{i_\ell,4}}$, which equals

$$\text{Case } (b = 0): \widetilde{v}^*_3 = \widetilde{v}_3 \text{ or}$$
$$\text{Case } (b = 1): \widetilde{v}^*_3 = \widetilde{v}_3 T^{-a \cdot x_{i_\ell,1} y_{i_\ell,1} x_{i_\ell,2}}.$$

(v) For the 4~8th rows, $\mathscr{B}$ computes table similarly as above.

(vi) Finally, $\mathscr{B}$ computes $\widehat{v}_0, \ldots, \widehat{v}_8$ from table, just as in Hybrids 2 and 3 (also as the original $\mathscr{E}$.Decrypt algorithm), and computes $\widetilde{e} := \widehat{v}_8 \cdot T^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j \in [4]})} \bmod N^s$, $t := g_1^{f'_\ell((x_{i_\ell,j}, y_{i_\ell,j})_{j \in [4]})} \bmod N$ using the secret keys.

If $b = 0$, $\mathscr{B}$ perfectly simulates Hybrid 2. If $b = 1$, $\mathscr{B}$ perfectly simulates Hybrid 3. Any difference between Hybrids 2 and 3 results in $\mathscr{B}$'s advantage over the IV_5 problem.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China Grant nos. 61672346 and 61373153.

References

[1] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[2] J. Black, P. Rogaway, and T. Shrimpton, "Encryption-scheme security in the presence of key-dependent messages," in *Selected Areas in Cryptography*, K. Nyberg and H. M. Heys, Eds., vol. 2595 of *Lecture Notes in Computer Science*, pp. 62–75, Springer, 2003.

[3] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology*, B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, pp. 93–118, Springer, 2001.

[4] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky, "Circular-secure encryption from decision Diffie-Hellman," in *Advances in Cryptology*, D. Wagner, Ed., vol. 5157 of *Lecture Notes in Computer Science*, pp. 108–125, Springer, 2008.

[5] Z. Brakerski and S. Goldwasser, "Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: quadratic residuosity strikes back)," in *Advances in Cryptology*, T. Rabin, Ed., vol. 6223 of *Lecture Notes in Computer Science*, pp. 1–20, Springer, 2010.

[6] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *Advances in Cryptology*, S. Halevi, Ed., vol. 5677 of *Lecture Notes in Computer Science*, pp. 595–618, Springer, 2009.

[7] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC '05)*, H. N. Gabow and R. Fagin, Eds., pp. 84–93, ACM, 2005.

[8] Z. Brakerski, S. Goldwasser, and Y. T. Kalai, "Black-box circular-secure encryption beyond affine functions," in *Theory of Cryptography*, Y. Ishai, Ed., vol. 6597 of *Lecture Notes in Computer Science*, pp. 201–218, Springer, 2011.

[9] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai, "Bounded key-dependent message security," in *Advances in Cryptology*, H. Gilbert, Ed., vol. 6110 of *Lecture Notes in Computer Science*, pp. 423–444, Springer, 2010.

[10] T. Malkin, I. Teranishi, and M. Yung, "Efficient circuit-size independent public key encryption with KDM security," in *Advances in Cryptology*, K. G. Paterson, Ed., vol. 6632 of *Lecture Notes in Computer Science*, pp. 507–526, Springer, 2011.

[11] J. Camenisch, N. Chandran, and V. Shoup, "A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks," in *Advances in Cryptology*, A. Joux, Ed., vol. 5479 of *Lecture Notes in Computer Science*, pp. 351–368, Springer, 2009.

[12] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC '90)*, H. Ortiz, Ed., pp. 427–437, May 1990.

[13] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology*, N. P. Smart, Ed., vol. 4965 of *Lecture Notes in Computer Science*, pp. 415–432, Springer, 2008.

[14] D. Galindo, J. Herranz, and J. Villar, "Identity-based encryption with master key-dependent message security and leakage-resilience," in *European Symposium on Research in Computer Security*, S. Foresti, M. Yung, and F. Martinelli, Eds., vol. 7459 of *Lecture Notes in Computer Science*, pp. 627–642, 2012.

[15] D. Hofheinz, "Circular chosen-ciphertext security with compact ciphertexts," in *Advances in Cryptology*, T. Johansson and P. Q. Nguyen, Eds., vol. 7881 of *Lecture Notes in Computer Science*, pp. 520–536, Springer, 2013.

[16] X. Lu, B. Li, and D. Jia, "KDM-CCA security from RKA secure authenticated encryption," in *Advances in Cryptology. Part I*, E. Oswald and M. Fischlin, Eds., vol. 9056 of *Lecture Notes in Computer Science*, pp. 559–583, Springer, 2015.

[17] S. Han, S. Liu, and L. Lyu, "Efficient KDM-CCA secure public-key encryption for polynomial functions," in *Annual International Conference on the Theory and Applications of Cryptology and Information Security*, J. H. Cheon and T. Takagi, Eds., vol. 10032 of *Lecture Notes in Computer Science*, pp. 307–338, Springer, 2016.

[18] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 167–226, 2003.

[19] B. Qin, S. Liu, and K. Chen, "Efficient chosen-ciphertext secure public-key encryption scheme with high leakage-resilience," *IET Information Security*, vol. 9, no. 1, pp. 32–42, 2015.

[20] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Advances in Cryptology*, L. R. Knudsen, Ed., vol. 2332 of *Lecture Notes in Computer Science*, pp. 45–64, Springer, 2002.

[21] Y. Dodis, E. Kiltz, K. Pietrzak, and D. Wichs, "Message authentication, revisited," in *Advances in Cryptology*, D. Pointcheval and T. Johansson, Eds., vol. 7237 of *Lecture Notes in Computer Science*, pp. 355–374, Springer, 2012.

[22] K. Xagawa, "Message authentication codes secure against additively related-key attacks," in *Proceedings of the Symposium on Cryptography and Information Security (SCIS '13)*, 2013.

[23] I. DamgÅrd and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system," in *Public Key Cryptography*, K. Kim, Ed., vol. 1992 of *Lecture Notes in Computer Science*, pp. 119–136, Springer, 2001.