Efficient KDM-CCA Secure Public-Key Encryption for Polynomial Functions

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
`{dalen17,slliu,lvlin}@sjtu.edu.cn`
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Westone Cryptologic Research Center, Beijing 100070, China

Abstract. KDM[$\mathcal{F}$]-CCA secure public-key encryption (PKE) protects the security of message $f(sk)$, with $f \in \mathcal{F}$, that is computed directly from the secret key, even if the adversary has access to a decryption oracle. An efficient KDM[$\mathcal{F}_{\text{aff}}$]-CCA secure PKE scheme for affine functions was proposed by Lu, Li and Jia (LLJ, EuroCrypt2015). We point out that their security proof cannot go through based on the DDH assumption.

In this paper, we introduce a new concept *Authenticated Encryption with Auxiliary-Input* AIAE and define for it new security notions dealing with related-key attacks, namely *IND-RKA security* and *weak INT-RKA security*. We also construct such an AIAE w.r.t. a set of restricted affine functions from the DDH assumption. With our AIAE,

- we construct the first efficient KDM[$\mathcal{F}_{\text{aff}}$]-CCA secure PKE w.r.t. affine functions with compact ciphertexts, which consist only of a constant number of group elements;
- we construct the first efficient KDM[$\mathcal{F}^{d}_{\text{poly}}$]-CCA secure PKE w.r.t. polynomial functions of bounded degree d with almost compact ciphertexts, and the number of group elements in a ciphertext is polynomial in d, independent of the security parameter.

Our PKEs are both based on the DDH & DCR assumptions, free of NIZK and free of pairing.

Keywords: Public-key encryption, Key-dependent messages, Chosen-ciphertext security, Authenticated encryption, Related-key attack

1 Introduction

Traditional Chosen-Ciphertext Attack (CCA) security of a public-key encryption (PKE) scheme considers the security of messages chosen by an adversary, even if the adversary obtains the public key pk, challenge ciphertexts of the messages, and has access to a decryption oracle (which provides decryption services to the adversary but refuses to decrypt the challenge ciphertexts). Note that the adversary cannot compute messages directly from secret keys, since it does not possess the secret keys. Therefore, CCA security does not cover the corner, where messages closely depend on the secret keys, say the secret keys themselves or functions of the secret keys. This issue was first identified in [GM84]. Later the security of key-dependent messages was formalized as KDM-security [BRS02]. KDM-security is an important notion, and has found wide applications, like hard disk encryption [BHHO08], cryptographic protocols [CL01], etc.

KDM-security w.r.t. a set of functions $\mathcal{F}$ is denoted by KDM[$\mathcal{F}$]-security. The larger $\mathcal{F}$ is, the stronger the security is. Roughly speaking, n-KDM[$\mathcal{F}$]-security of PKE considers such a scenario: an adversary is given public keys $(pk_1, pk_2, \cdots, pk_n)$ of n users and an encryption oracle. Whenever the adversary queries a function $f \in \mathcal{F}$, the encryption oracle will always reply with an encryption of a constant say 0, or always reply with an encryption of $f(sk_1, sk_2, \cdots, sk_n)$. If the adversary cannot tell which case it is, the PKE is n-KDM[$\mathcal{F}$]-CPA secure. If the adversary has also access to a decryption oracle in the scenario, then KDM[$\mathcal{F}$]-CPA security is improved to KDM[$\mathcal{F}$]-CCA security. Obviously, KDM-CCA security notion is stronger than KDM-CPA.

KDM[$\mathcal{F}$]-CPA Security. The BHHO scheme [BHHO08] was the first PKE achieving KDM[$\mathcal{F}_{\text{aff}}$]-CPA security based on the Decisional Diffie-Hellman (DDH) assumption, where $\mathcal{F}_{\text{aff}}$ denotes the set of affine functions. It was later generalized by Brakerski and Goldwasser [BG10] to KDM[$\mathcal{F}_{\text{aff}}$]-CPA secure PKE schemes based on the Subgroup Indistinguishability Assumption (including the QR and the DCR assumptions). These schemes have incompact ciphertexts containing $O(\ell)$ group elements, where ℓ denotes the security parameter.

A variant of Regev's scheme [Reg05] was shown to be KDM[$\mathcal{F}_{\text{aff}}$]-CPA secure and has compacter ciphertexts by Applebaum et al. [ACPS09].

Barak et al. [BHHI10] proposed KDM-CPA secure PKE w.r.t. a very large function set, i.e., the function set of boolean circuits of bounded size $p = p(\ell)$. However, their scheme is inflexible and highly impractical, since its encryption algorithm depends on the bound p and the number of users, and the ciphertext contains a garbled circuit of size at least $p = p(\ell)$.

Brakerski et al. [BGK11] amplified the BHHO scheme to KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA security w.r.t. the set of polynomial functions of bounded degree d. However, their ciphertext contains $O(\ell^{d+1})$ group elements.

It is Malkin et al. [MTY11] who designed the first efficient PKE scheme achieving KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA security. Their ciphertext contains only $O(d)$ group elements, thus d can be polynomial in ℓ in their case. The function set $\mathcal{F}^d_{\text{poly}}$ is characterized by a polynomial-size *Modular Arithmetic Circuit* in [MTY11].

KDM[$\mathcal{F}$]-CCA Security. KDM[$\mathcal{F}$]-CCA security of PKE is far more difficult to design than KDM[$\mathcal{F}$]-CPA security. Camenisch et al. [CCS09] gave the first solution, following Naor-Yung's paradigm, which needs a KDM-CPA secure PKE, a CCA-secure PKE and a non-interactive zero-knowledge (NIZK) proving that the two PKEs encrypt the same message.

NIZK is not practical in general, except Groth-Sahai proofs [GS08]. When following [CCS09]'s approach, the only possible way to get an efficient KDM-CCA secure PKE, is using Groth-Sahai proofs together with an efficient KDM-

CPA secure PKE. However, many existing efficient KDM-CPA secure schemes, such as [ACPS09, MTY11], are not based on pairing-friendly groups, thus not compatible with Groth-Sahai's efficient NIZK.

Another work by Galindo et al. [GHV12] is based on the Matrix DDH assumption over pairing-friendly groups. Their scheme has compact ciphertexts, but only obtains a bounded form of KDM-CCA security, i.e., the number of encryption queries is limited to be linear in the size of the secret key.

To get an efficient KDM-CCA secure PKE, Hofheinz [Hof13] proposed another approach, which uses a new tool called "lossy algebraic filter". His work results in the first PKE enjoying both KDM-CCA security and compact ciphertexts (consisting only of a constant number of group elements). However, the function set $\mathcal{F}_{\text{circ}}$ only consists of selection functions $f(sk_1, \cdots, sk_n) = sk_i$ and constant functions.

It is quite challenging to enlarge $\mathcal{F}$ for KDM[$\mathcal{F}$]-CCA security while still keeping PKE efficient. One effort was recently made by Lu, Li and Jia [LLJ15], who proposed the first efficient KDM[$\mathcal{F}_{\text{aff}}$]-CCA secure PKE with compact ciphertexts. We call their construction the LLJ scheme. There is an essential building block called "Authenticated Encryption" ($\overline{\mathsf{AE}}$) in their scheme. The KDM[$\mathcal{F}_{\text{aff}}$]-CCA security heavily relies on a so-called INT-$\mathcal{F}_{\text{aff}}$-RKA security of $\overline{\mathsf{AE}}$. INT-$\mathcal{F}_{\text{aff}}$-RKA security of $\overline{\mathsf{AE}}$ means that a PPT adversary cannot forge a fresh forgery $(f^*, \overline{\mathsf{ae}}.\mathsf{ct}^*)$ such that $\overline{\mathsf{AE}}.\mathsf{Dec}_{f^*(\mathsf{k})}(\overline{\mathsf{ae}}.\mathsf{ct}^*) \neq \perp$, even if the adversary observes multiple outputs of $\overline{\mathsf{AE}}.\mathsf{Enc}_{f_j(\mathsf{k})}(m_j)$ with his choice of (f_j, m_j). Unfortunately, we found that the INT-$\mathcal{F}_{\text{aff}}$-RKA security proof of the specific $\overline{\mathsf{AE}}$ does not go through to the DDH assumption, which in turn affects the KDM[$\mathcal{F}_{\text{aff}}$]-CCA security proof of the LLJ scheme. Our essential observation is that the DDH adversary is not able to employ the fresh forgery from the adversary of $\overline{\mathsf{AE}}$ to solve the DDH problem, since the DDH adversary does not have any trapdoor to convert the computing power (forgery) to a decision bit.

As for KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security, [CCS09]'s paradigm is the unique path to it up to now. Unfortunately, the only efficient KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA secure scheme [MTY11] does not compose well with Groth-Sahai proofs, so it has to resort to the general NIZK. Other KDM[$\mathcal{F}^d_{\text{poly}}$]-CPA secure schemes either is highly impractical [BHHI10] or has ciphertext containing $O(\ell^{d+1})$ group elements [BGK11], which grows exponentially with the degree d.

Our Contribution. We work on the design of efficient PKE with KDM[$\mathcal{F}_{\text{aff}}$]-CCA security and KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security.

– We identify the proof flaw in [LLJ15], where an efficient KDM[$\mathcal{F}_{\text{aff}}$]-CCA secure PKE was claimed. We show that for "Authenticated Encryption" ($\overline{\mathsf{AE}}$) used in the LLJ scheme, the INT-$\mathcal{F}_{\text{aff}}$-RKA security reduction to the DDH assumption does not work. This proof flaw directly affects the KDM[$\mathcal{F}_{\text{aff}}$]-CCA security proof of the LLJ scheme.
– We provide the first efficient KDM[$\mathcal{F}_{\text{aff}}$]-CCA secure PKE w.r.t. affine functions with compact ciphertexts. Our scheme has ciphertexts consisting only of a constant number of group elements and is free of NIZK.

– We provide the first efficient KDM[$\mathcal{F}_{\text{poly}}^d$]-CCA secure PKE w.r.t. polynomial functions of bounded degree d with almost compact ciphertexts. Our scheme is free of NIZK. The number of group elements in a ciphertext is polynomial in d, independent of the security parameter ℓ.

We summarize known PKEs either achieving KDM-CCA security or against function set $\mathcal{F}_{\text{poly}}^d$ in Table 1.

Table 1. Comparison between PKEs either achieving KDM-CCA security or against function set $\mathcal{F}_{\text{poly}}^d$. Here ℓ is the security parameter. $\mathcal{F}_{\text{circ}}$, $\mathcal{F}_{\text{aff}}$ and $\mathcal{F}_{\text{poly}}^d$ denote the set of selection functions, the set of affine functions and the set of polynomial functions of bounded degree d, respectively. "CCA" means the scheme is KDM-CCA secure. "Free of Pairing" asks whether the scheme is free of pairing. $|\mathsf{CT}|$ shows the size of ciphertext. $\mathbb{G}$, $\mathbb{Z}_{N^3}$, $\mathbb{Z}_{N^2}$ and $\mathbb{Z}_{\bar{N}}$ are the underlying groups. s can be any integer greater than 1. The symbol "**?**" means that the security proof is not rigorous.

Scheme	Set	CCA?	Free of Pairing?	$\lvert\mathsf{CT}\rvert$	Assumption
[BHHO08] + [CCS09]	$\mathcal{F}_{\text{aff}}$	$\surd$	–	$(6\ell+13)\lvert\mathbb{G}\rvert$	DDH
[BGK11]	$\mathcal{F}_{\text{poly}}^d$	–	$\surd$	$(\ell^{d+1})\lvert\mathbb{G}\rvert$	DDH or LWE
[MTY11]	$\mathcal{F}_{\text{poly}}^d$	–	$\surd$	$(d+2)\lvert\mathbb{Z}_{N^s}\rvert$	DCR
[Hof13]	$\mathcal{F}_{\text{circ}}$	$\surd$	–	$6\lvert\mathbb{Z}_{N^3}\rvert + 49\lvert\mathbb{G}\rvert$	DDH & DCR
[LLJ15]	$\mathcal{F}_{\text{aff}}$	**?**	$\surd$	$3\lvert\mathbb{Z}_{N^2}\rvert + 3\lvert\mathbb{Z}_{N^s}\rvert + \lvert\mathbb{Z}_{\bar{N}}\rvert$	DDH & DCR
Our scheme in §5	$\mathcal{F}_{\text{aff}}$	$\surd$	$\surd$	$9\lvert\mathbb{Z}_{N^2}\rvert + 9\lvert\mathbb{Z}_{N^s}\rvert + 2\lvert\mathbb{Z}_{\bar{N}}\rvert$	DDH & DCR
Our scheme in §6	$\mathcal{F}_{\text{poly}}^d$	$\surd$	$\surd$	$9\lvert\mathbb{Z}_{N^2}\rvert + (8d^9+1)\lvert\mathbb{Z}_{N^s}\rvert + 2\lvert\mathbb{Z}_{\bar{N}}\rvert$	DDH & DCR

Our Approach. The challenge for KDM[$\mathcal{F}$]-CCA security of PKE lies in the fact that the adversary $\mathcal{A}$ has multiple access to the encryptions of $f(sk)$ and decryption oracle $\mathsf{Dec}(sk, \cdot)$, with $f \in \mathcal{F}$ and sk the secret key. Let us consider only one secret key for simplicity. The information of sk might be leaked completely via encryptions of $f(sk)$.

To solve this problem, we follow a KEM+DEM style and construct our PKE with three building blocks: KEM, $\mathcal{E}$ and AIAE, as shown in Fig. 1.

- We propose a new concept *"Authenticated Encryption with Auxiliary-Input"* (AIAE). We define for it new security notions dealing with related-key attacks, namely *weak INT-$\mathcal{F}'$-RKA security* and IND-$\mathcal{F}'$-RKA security.
- We design the other building blocks KEM and $\mathcal{E}$. $\mathsf{KEM.Enc}$ encapsulates a key k for AIAE, and the encapsulation $\mathsf{kem.ct}$ serves as an auxiliary input aux for $\mathsf{AIAE.Enc}$. $\mathcal{E}.\mathsf{Enc}$ encrypts m to get a ciphertext $\mathcal{E}.\mathsf{ct}$, which serves as an input for $\mathsf{AIAE.Enc}$.

We show how to achieve KDM[$\mathcal{F}$]-CCA security with our three building blocks.

– $\mathcal{E}.\mathsf{Enc}$ can behave like an entropy filter (the concept was named in [LLJ15]) for $\mathcal{F}$. That is, through some computationally indistinguishable change, some

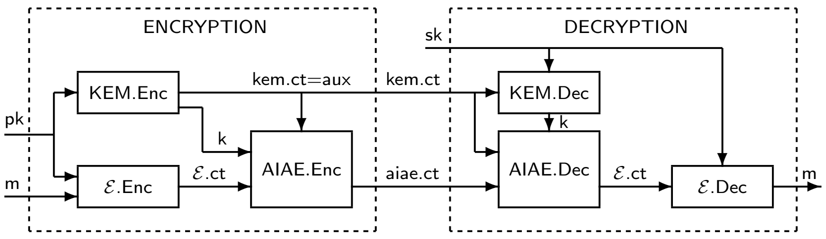

Fig. 1. Our approach of PKE construction. Here KEM and $\mathcal{E}$ share the same public/secret key pair. $\mathsf{AIAE.Enc}$ uses k output by KEM to encrypt $\mathcal{E}.\mathsf{ct}$ with auxiliary input $\mathsf{aux} := \mathsf{kem.ct}$, and outputs ciphertext $\mathsf{aiae.ct}$.

entropy of sk is always reserved even if multiple encryptions of $f_j(sk)$ are given to $\mathcal{A}$. Here $f_j \in \mathcal{F}$ is chosen by $\mathcal{A}$.

– The fresh keys k_j used by $\mathsf{AIAE.Enc}$ can be expressed as functions of a base key k^*, i.e., $\mathsf{k}_j = f'_j(\mathsf{k}^*)$, where $f'_j \in \mathcal{F}'$ for some function set $\mathcal{F}'$. We stress that $\mathcal{F}'$ might be different from $\mathcal{F}$.
– $\mathsf{KEM.Enc}$ is able to use the remaining entropy of sk to protect the base key k^*, via some computationally indistinguishable change.
– The weak INT-$\mathcal{F}'$-RKA security of AIAE guarantees: given multiple AIAE ciphertext-auxiliary input pair $(\mathsf{aiae.ct}_j, \mathsf{aux}_j)$ encrypted by $f'_j(\mathsf{k}^*)$, it is infeasible for a PPT algorithm to forge a new $(f', \mathsf{aiae.ct}, \mathsf{aux})$ satisfying (1) $\mathsf{AIAE.Dec}_{f'(\mathsf{k}^*)}(\mathsf{aiae.ct}, \mathsf{aux}) \neq \perp$; (2) if $\mathsf{aux} = \mathsf{aux}_j$ for some j then $f' = f'_j$.
– Decryption oracle can reject all invalid ciphertexts that are not properly generated by the encryption algorithm, via some computationally indistinguishable change. If the invalid ciphertext makes $\mathsf{KEM.Dec}$ decapsulate a key $f'(\mathsf{k}^*)$, $\mathsf{AIAE.Dec}$ will output $\perp$, due to its weak INT-$\mathcal{F}'$-RKA security. Otherwise, the invalid ciphertext will be rejected by $\mathcal{E}.\mathsf{Dec}$ or $\mathsf{KEM.Dec}$, due to the remaining entropy of sk. As a result, no extra information about sk is leaked.
– The IND-$\mathcal{F}'$-RKA security of AIAE ensures: given multiple AIAE ciphertext-auxiliary input pair $(\mathsf{aiae.ct}_j, \mathsf{aux}_j)$ with key $f'_j(\mathsf{k}^*)$ encrypting either m_0 or m_1, it is infeasible for a PPT algorithm to distinguish which case it is, even if $f'_j \in \mathcal{F}'$ is submitted by the algorithm.
– By the IND-$\mathcal{F}'$-RKA security of AIAE, the encryption of $\mathcal{E}.\mathsf{ct}$ can be replaced with an encryption of all zeros. Then the KDM[$\mathcal{F}$]-CCA security follows.

With this approach, we can construct PKEs possessing KDM[$\mathcal{F}_{\text{aff}}$]-CCA and KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security respectively, by designing specific building blocks.

Comparison with LLJ. We inherit the idea of utilizing RKA security of AE to achieve KDM security from LLJ. However, our approach deviates from LLJ in three aspects.

1. The structure of our scheme is different from LLJ. It is also possible to explain the LLJ scheme with three components KEM, $\mathcal{E}$ and $\overline{\mathsf{AE}}$. However,

their components were composed in a different way. In the LLJ scheme, the output $\mathsf{kem.ct}$ of KEM serves as an additional input for $\mathcal{E}.\mathsf{Enc}$. With their structure, $\mathcal{E}$ is expected to authenticate $\mathsf{kem.ct}$. In our approach, $\mathsf{kem.ct}$ is the auxiliary input of AIAE, thus can be authenticated by AIAE.

2. The syntax and security requirements of our AIAE are different from LLJ's $\overline{\mathsf{AE}}$. Their $\overline{\mathsf{AE}}$ does not support auxiliary input, and the security proof of their $\overline{\mathsf{AE}}$ instantiation has some problem, as shown in Section 3.
3. Our KEM and $\mathcal{E}$ are newly designed building blocks which compose well with our AIAE. We give two designs of $\mathcal{E}$ to support KDM[$\mathcal{F}_{\text{aff}}$]-CCA and KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security respectively.

2 Preliminaries

Let $\ell \in \mathbb{N}$ denote the security parameter. For $i, j \in \mathbb{N}$ with $i < j$, define $[i, j] := \{i, i+1, \cdots, j\}$ and $[j] := \{1, 2, \cdots, j\}$. Denote by $s \leftarrow_{\$} \mathcal{S}$ the operation of picking an element s from set $\mathcal{S}$ uniformly at random. For an algorithm $\mathcal{A}$, denote by $y \leftarrow_{\$} \mathcal{A}(x; r)$, or simply $y \leftarrow_{\$} \mathcal{A}(x)$, the operation of running $\mathcal{A}$ with input x and randomness r and assigning output to y. Let ε denote the empty string. For a primitive XX and a security notion YY, we typically denote the advantage of a PPT adversary $\mathcal{A}$ by $\mathsf{Adv}^{\text{YY}}_{\text{XX},\mathcal{A}}(\ell)$ and define $\mathsf{Adv}^{\text{YY}}_{\text{XX}}(\ell) := \max_{\text{PPT}\mathcal{A}} \mathsf{Adv}^{\text{YY}}_{\text{XX},\mathcal{A}}(\ell)$. Let $2^{-\Omega(\ell)}$ denote the value upper bounded by $2^{-c\cdot\ell}$ for some constant $c > 0$.

Games. Our security proof will be game-based security reductions. A game G starts with an Initialize procedure and ends with a Finalize procedure. There are also some optional procedures $\text{Proc}_1, \cdots, \text{Proc}_n$ performing as oracles. All procedures are described using pseudo-code, where initially all variables are empty strings ε and all sets are empty. An adversary $\mathcal{A}$ is executed in game G if it first calls Initialize, obtaining its output. Then the adversary may make arbitrary oracle-queries to procedures Proc_i according to their specification, and obtain their outputs. Finally it makes one single call to Finalize. By $\mathsf{G}^{\mathcal{A}} \Rightarrow b$ we means that G outputs b after interacting with $\mathcal{A}$, and b is in fact the output of Finalize. By $a \overset{\mathsf{G}}{=} b$ we mean that a equals b or is computed as b in game G.

2.1 Public-Key Encryption and KDM-CCA Security

A public-key encryption (PKE) scheme is made up of four PPT algorithms $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$: $\mathsf{Setup}(1^\ell)$ generates a public parameter prm, which implicitly defines a secret key space $\mathcal{SK}$ and a message space $\mathcal{M}$; $\mathsf{Gen}(\mathsf{prm})$ takes as input the public parameter prm and generates a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$; $\mathsf{Enc}(\mathsf{pk}, m)$ takes as input the public key pk and a message m, and outputs a ciphertext $\mathsf{pke.ct}$; $\mathsf{Dec}(\mathsf{sk}, \mathsf{pke.ct})$ takes as input the secret key sk and a ciphertext $\mathsf{pke.ct}$ and outputs either a message m or a failure symbol $\perp$. The correctness of PKE requires that, for all $\mathsf{prm} \leftarrow_{\$} \mathsf{Setup}(1^\ell)$, all $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(\mathsf{prm})$, all $m \in \mathcal{M}$ and all $\mathsf{pke.ct} \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}, m)$, it holds that $\mathsf{Dec}(\mathsf{sk}, \mathsf{pke.ct}) = m$.

Let $n \in \mathbb{N}$ and $\mathcal{F}$ be a family of functions from $\mathcal{SK}^n$ to $\mathcal{M}$. We define the n-KDM[$\mathcal{F}$]-CCA security via the security game in Fig. 2.

```
Procedure INITIALIZE:
$\mathsf{prm} \leftarrow_{\$} \mathsf{Setup}(1^\ell)$.
For $i \in [n]$
    $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_{\$} \mathsf{Gen}(\mathsf{prm})$.
$\beta \leftarrow_{\$} \{0,1\}$.    // challenge bit
Return $(\mathsf{prm}, \mathsf{pk}_1, \cdots, \mathsf{pk}_n)$.

Procedure FINALIZE($\beta'$):
Return $(\beta' = \beta)$.

Procedure ENC($f \in \mathcal{F}, i \in [n]$):
$m_1 := f(\mathsf{sk}_1, \cdots, \mathsf{sk}_n)$,  $m_0 := 0^{|m_1|}$.
$\mathsf{pke.ct} \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}_i, m_\beta)$.
$\mathcal{Q_{ENC}} := \mathcal{Q_{ENC}} \cup \{(\mathsf{pke.ct}, i)\}$.
Return $\mathsf{pke.ct}$.

Procedure DEC($\mathsf{pke.ct}, i \in [n]$):
If $(\mathsf{pke.ct}, i) \in \mathcal{Q_{ENC}}$, Return $\perp$.
Return $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{pke.ct})$.
```

Fig. 2. n-$\mathsf{KDM}[\mathcal{F}]$-$\mathsf{CCA}$ security game for PKE.

Definition 1 (KDM[$\mathcal{F}$]-CCA Security for PKE). *Scheme* PKE *is* n*-KDM*$[\mathcal{F}]$*-CCA secure if for any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}^{kdm\text{-}cca}_{\mathsf{PKE},\mathcal{A}}(\ell) := |\Pr[n\text{-}\mathsf{KDM}[\mathcal{F}]\text{-}\mathsf{CCA}^{\mathcal{A}} \Rightarrow 1] - 1/2|$ *is negligible in* ℓ*, where game* n-$\mathsf{KDM}[\mathcal{F}]$-$\mathsf{CCA}$ *is specified in Fig. 2.*

2.2 Key Encapsulation Mechanism

A key encapsulation mechanism (KEM) consists of three PPT algorithms $\mathsf{KEM} = (\mathsf{KEM.Gen}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$: $\mathsf{KEM.Gen}(1^\ell)$ outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$; $\mathsf{KEM.Enc}(\mathsf{pk})$ uses the public key pk to compute a key k and a ciphertext (or encapsulation) $\mathsf{kem.ct}$; $\mathsf{KEM.Dec}(\mathsf{sk}, \mathsf{kem.ct})$ takes as input the secret key sk and a ciphertext $\mathsf{kem.ct}$, and outputs either a key k or a failure symbol $\perp$. The correctness of KEM requires that, for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KEM.Gen}(1^\ell)$ and all $(\mathsf{k}, \mathsf{kem.ct}) \leftarrow_{\$} \mathsf{KEM.Enc}(\mathsf{pk})$, it holds that $\mathsf{KEM.Dec}(\mathsf{sk}, \mathsf{kem.ct}) = \mathsf{k}$.

2.3 Authenticated Encryption: One-Time Security and Related-Key Attack Security

Definition 2 (Authenticated Encryption). *An authenticated encryption (AE) scheme* $\mathsf{AE} = (\mathsf{AE.Setup}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$ *consists of three PPT algorithms:*

- $\mathsf{AE.Setup}(1^\ell)$ *outputs a system parameter* $\mathsf{prm}_{\mathsf{AE}}$*, which is an implicit input to* $\mathsf{AE.Enc}$ *and* $\mathsf{AE.Dec}$*. The parameter* $\mathsf{prm}_{\mathsf{AE}}$ *implicitly defines a message space* $\mathcal{M}$ *and a key space* $\mathcal{K}_{\mathsf{AE}}$.
- $\mathsf{AE.Enc}(\mathsf{k}, m)$ *takes as input a key* $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$ *and a message* $m \in \mathcal{M}$*, and outputs a ciphertext* $\mathsf{ae.ct}$.
- $\mathsf{AE.Dec}(\mathsf{k}, \mathsf{ae.ct})$ *takes as input a key* $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$ *and a ciphertext* $\mathsf{ae.ct}$*, and outputs a message* $m \in \mathcal{M}$ *or a rejection symbol* $\perp$.

Correctness of AE *requires that, for all* $\mathsf{prm}_{\mathsf{AE}} \leftarrow_{\$} \mathsf{AE.Setup}(1^\ell)$*, all* $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$*, all* $m \in \mathcal{M}$ *and all* $\mathsf{ae.ct} \leftarrow_{\$} \mathsf{AE.Enc}(\mathsf{k}, m)$*, it holds that* $\mathsf{AE.Dec}(\mathsf{k}, \mathsf{ae.ct}) = m$.

The security notions for AE include One-time ciphertext-indistinguishability (IND-OT) and One-time ciphertext-integrity (INT-OT). The IND-OT and INT-OT securities of AE are formalized via the security games in Fig. 3.

```
Procedure INITIALIZE:
  $\mathsf{prm_{AE}} \leftarrow_\$ \mathsf{AE.Setup}(1^\ell)$,  $\mathsf{k} \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$.
  $\beta \leftarrow_\$ \{0,1\}$.    // challenge bit
  Return $\mathsf{prm_{AE}}$.

Procedure ENC$(m_0, m_1)$: // one query
  If $|m_0| \neq |m_1|$, Return $\perp$.
  $\mathsf{ae.ct} \leftarrow_\$ \mathsf{AE.Enc}(\mathsf{k}, m_\beta)$.
  Return $\mathsf{ae.ct}$.

Procedure FINALIZE$(\beta')$:
  Return $(\beta' = \beta)$.
```

```
Procedure INITIALIZE:
  $\mathsf{prm_{AE}} \leftarrow_\$ \mathsf{AE.Setup}(1^\ell)$,  $\mathsf{k} \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$.
  Return $\mathsf{prm_{AE}}$.

Procedure ENC$(m)$: // one query
  $\mathsf{ae.ct} \leftarrow_\$ \mathsf{AE.Enc}(\mathsf{k}, m)$.
  Return $\mathsf{ae.ct}$.

Procedure FINALIZE$(\mathsf{ae.ct}^*)$:
  If $\mathsf{ae.ct}^* = \mathsf{ae.ct}$, Return 0.
  Return $(\mathsf{AE.Dec}(\mathsf{k}, \mathsf{ae.ct}^*) \neq \perp)$.
```

Fig. 3. Games IND-OT (left) and INT-OT (right) for defining securities of AE.

Definition 3 (One-Time Security for AE). *Scheme* AE *is one-time secure (OT-secure) if it is IND-OT secure and INT-OT secure, i.e., for any PPT adversary $\mathcal{A}$, both $\mathsf{Adv}^{ind\text{-}ot}_{\mathsf{AE},\mathcal{A}}(\ell) := |\Pr[\mathsf{IND\text{-}OT}^{\mathcal{A}} \Rightarrow 1] - 1/2|$ and $\mathsf{Adv}^{int\text{-}ot}_{\mathsf{AE},\mathcal{A}}(\ell) := \Pr[\mathsf{INT\text{-}OT}^{\mathcal{A}} \Rightarrow 1]$ are negligible in ℓ, where games* IND-OT *and* INT-OT *are specified in Fig. 3.*

Let $\mathcal{F}$ be a family of functions from $\mathcal{K}_{\mathsf{AE}}$ to $\mathcal{K}_{\mathsf{AE}}$. The $\mathcal{F}$-Related-Key Attack for AE scheme was formalized in [LLJ15], and RKA security notions characterize the ciphertext indistinguishability (IND-$\mathcal{F}$-RKA) and integrity (INT-$\mathcal{F}$-RKA) even if the adversary has multiple access to the encryption oracle and designates a function $f \in \mathcal{F}$ each time such that the encryption oracle uses $f(\mathsf{k})$ as the key.

```
Procedure INITIALIZE:
  $\mathsf{prm_{AE}} \leftarrow_\$ \mathsf{AE.Setup}(1^\ell)$,  $\mathsf{k} \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$.
  $\beta \leftarrow_\$ \{0,1\}$.    // challenge bit
  Return $\mathsf{prm_{AE}}$.

Procedure ENC$(m_0, m_1, f \in \mathcal{F})$:
  If $|m_0| \neq |m_1|$, Return $\perp$.
  $\mathsf{ae.ct} \leftarrow_\$ \mathsf{AE.Enc}(f(\mathsf{k}), m_\beta)$.
  Return $\mathsf{ae.ct}$.

Procedure FINALIZE$(\beta')$:
  Return $(\beta' = \beta)$.
```

```
Procedure INITIALIZE:
  $\mathsf{prm_{AE}} \leftarrow_\$ \mathsf{AE.Setup}(1^\ell)$,  $\mathsf{k} \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$.
  Return $\mathsf{prm_{AE}}$.

Procedure ENC$(m, f \in \mathcal{F})$:
  $\mathsf{ae.ct} \leftarrow_\$ \mathsf{AE.Enc}(f(\mathsf{k}), m)$.
  $\mathcal{Q}_{\mathcal{ENC}} := \mathcal{Q}_{\mathcal{ENC}} \cup \{(f, \mathsf{ae.ct})\}$.
  Return $\mathsf{ae.ct}$.

Procedure FINALIZE$(f^* \in \mathcal{F}, \mathsf{ae.ct}^*)$:
  If $(f^*, \mathsf{ae.ct}^*) \in \mathcal{Q}_{\mathcal{ENC}}$, Return 0.
  Return $(\mathsf{AE.Dec}(f^*(\mathsf{k}), \mathsf{ae.ct}^*) \neq \perp)$.
```

Fig. 4. Games IND-$\mathcal{F}$-RKA (left) and INT-$\mathcal{F}$-RKA (right) for defining securities of AE.

Definition 4 (IND-RKA and INT-RKA Securities for AE). *Scheme* AE *is IND-$\mathcal{F}$-RKA secure and INT-$\mathcal{F}$-RKA secure, if for any PPT adversary $\mathcal{A}$, both $\mathsf{Adv}^{ind\text{-}rka}_{\mathsf{AE},\mathcal{A}}(\ell) := |\Pr[\mathsf{IND}\text{-}\mathcal{F}\text{-}\mathsf{RKA}^{\mathcal{A}} \Rightarrow 1] - 1/2|$ and $\mathsf{Adv}^{int\text{-}rka}_{\mathsf{AE},\mathcal{A}}(\ell) := \Pr[\mathsf{INT}\text{-}\mathcal{F}\text{-}\mathsf{RKA}^{\mathcal{A}} \Rightarrow 1]$ are negligible in ℓ, where games* IND-$\mathcal{F}$-RKA *and* INT-$\mathcal{F}$-RKA *are specified in Fig. 4.*

2.4 DCR, DDH, DL and IV$_d$ Assumptions

Let $\mathsf{GenN}(1^\ell)$ be a PPT algorithm outputting (N, p, q), where p, q are safe primes of ℓ bits and $N = pq$, such that $\bar{N} = 2N+1$ is also a prime. Let $s \in \mathbb{N}$ and $T = 1+N$. Define $\mathbb{QR}_{N^s} := \{a^2 \bmod N^s \mid a \in \mathbb{Z}^*_{N^s}\}$, $\mathbb{SCR}_{N^s} := \{a^{2N^{s-1}} \bmod N^s \mid a \in \mathbb{Z}^*_{N^s}\}$, and $\mathbb{RU}_{N^s} := \{T^r \bmod N^s \mid r \in [N^{s-1}]\}$. 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 $\otimes$ denotes internal direct product. Let $\mathbb{QR}_{\bar{N}} := \{a^2 \bmod \bar{N} \mid a \in \mathbb{Z}_{\bar{N}}\}$, then $\mathbb{QR}_{\bar{N}}$ is a cyclic group of order $N = pq$.

For $X \in \mathbb{RU}_{N^s}$, the discrete logarithm $\mathrm{dlog}_T(X) \in [N^{s-1}]$ can be efficiently computed given only N and X [DJ01]. Note that $\mathbb{Z}^*_{N^s} = \mathbb{Z}_2 \otimes \mathbb{Z}'_2 \otimes \mathbb{SCR}_{N^s} \otimes \mathbb{RU}_{N^s}$, hence for any $u = u(\mathbb{Z}_2) \cdot u(\mathbb{Z}'_2) \cdot u(\mathbb{SCR}_{N^s}) \cdot T^x \in \mathbb{Z}^*_{N^s}$, $u^{\phi(N)} = T^{x\cdot\phi(N)} \in \mathbb{RU}_{N^s}$ and

$$\mathrm{dlog}_T(u^{\phi(N)})/\phi(N) \bmod N^{s-1} = x. \tag{1}$$

The formal definitions of the Decisional Composite Residuosity (DCR) and the Discrete Logarithm (DL) assumptions are in the full version [HLL16]. The DCR assumption implies the Interactive Vector (IV$_d$) assumption according to [BG10]. We adopt the version in [LLJ15].

Definition 5 (IV$_d$ Assumption). *The IV$_d$ Assumption holds w.r.t.* GenN *and group $\mathbb{QR}_{N^s}$ if for any PPT adversary $\mathcal{A}$, the following advantage is negligible in ℓ:*

$$\mathsf{Adv}^{iv_d}_{\mathsf{GenN},\mathcal{A}}(\ell) := \left| \Pr\left[\mathcal{A}^{\textsc{Chal}^b_{\mathrm{IV}_d}}(N, g_1, \cdots, g_d) = b\right] - 1/2 \right|,$$

where $(N, p, q) \leftarrow_\$ \mathsf{GenN}(1^\ell)$, $g_1, \cdots, g_d \leftarrow_\$ \mathbb{SCR}_{N^s}$, $b \leftarrow_\$ \{0,1\}$, and the oracle $\textsc{Chal}^b_{\mathrm{IV}_d}(\cdot)$ can be queried by $\mathcal{A}$ adaptively. $\mathcal{A}$ submits $(\delta_1, \cdots, \delta_d)$ to the oracle. $\textsc{Chal}^b_{\mathrm{IV}_d}(\delta_1, \cdots, \delta_d)$ selects random $r \leftarrow_\$ [\lfloor N/4 \rfloor]$. If $b = 0$, the oracle returns $(g_1^r, \cdots, g_d^r)$; otherwise it returns $(g_1^r T^{\delta_1}, \cdots, g_d^r T^{\delta_d})$, where $T = 1 + N$.

Definition 6 (DDH Assumption). *The Decisional Diffie-Hellman (DDH) Assumption holds w.r.t.* GenN *and group $\mathbb{QR}_{\bar{N}}$ if for any PPT adversary $\mathcal{A}$, the following advantage is negligible in ℓ:*

$$\mathsf{Adv}^{ddh}_{\mathsf{GenN},\mathcal{A}}(\ell) := \left| \Pr\left[\mathcal{A}(N,p,q,g_1,g_2,g_1^x,g_2^x) = 1\right] - \Pr\left[\mathcal{A}(N,p,q,g_1,g_2,g_1^x,g_2^y) = 1\right] \right|,$$

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

2.5 Collision Resistant Hashing and Universal Hashing

Definition 7 (Collision Resistant Hashing). *A family of functions $\mathcal{H} = \{\mathsf{H} : \mathcal{X} \longrightarrow \mathcal{Y}\}$ is collision-resistant if for any PPT adversary $\mathcal{A}$, the following advantage is negligible in ℓ:*

$$\mathsf{Adv}^{cr}_{\mathcal{H},\mathcal{A}}(\ell) := \Pr\left[\mathsf{H} \leftarrow_\$ \mathcal{H},\ (x, x') \leftarrow_\$ \mathcal{A}(\mathsf{H}) \ :\ \mathsf{H}(x) = \mathsf{H}(x') \ \wedge\ x \neq x'\right].$$

Definition 8 (Universal Hashing). *A family of functions $\mathcal{H} = \{\mathsf{H} : \mathcal{X} \longrightarrow \mathcal{Y}\}$ is universal, if for all distinct $x, x' \in \mathcal{X}$, it follows that*

$$\Pr\left[\mathsf{H} \leftarrow_\$ \mathcal{H} \ :\ \mathsf{H}(x) = \mathsf{H}(x')\right] \leq 1/|\mathcal{Y}|.$$

3 $\overline{\mathsf{AE}}$ of the LLJ Scheme and Its INT-RKA Security

The LLJ scheme [LLJ15] makes use of an important primitive "Authenticated Encryption" $\overline{\mathsf{AE}}$. Its KDM[$\mathcal{F}_{\text{aff}}$]-CCA security heavily relies on the IND-$\mathcal{F}_{\text{aff}}$-RKA security and INT-$\mathcal{F}_{\text{aff}}$-RKA security of their $\overline{\mathsf{AE}}$. LLJ claimed INT-$\mathcal{F}_{\text{aff}}$-RKA security of their $\overline{\mathsf{AE}}$, however, we point out that their security proof does not go through to the DDH assumption, which in turn affects the KDM[$\mathcal{F}_{\text{aff}}$]-CCA security proof of the LLJ scheme.

Let us briefly review LLJ's $\overline{\mathsf{AE}}$ as follows. The public parameter is $\mathsf{prm}_{\overline{\mathsf{AE}}} = (N, \bar{N}, g)$ where $N = pq$, $\bar{N} = 2N + 1$, and g is a generator of group $\mathbb{QR}_{\bar{N}}$. Let AE be an IND-OT and INT-OT secure authenticated encryption, and H be a 4-wise independent hash function. The secret key space is $\mathbb{Z}_N$.

- $\overline{\mathsf{AE}}.\mathsf{Enc}(k, m)$ computes $u = g^r$ with $r \leftarrow_\$ \mathbb{Z}_N$, $\kappa = \mathsf{H}(u^k, u)$ and invokes $\chi \leftarrow_\$ \mathsf{AE}.\mathsf{Enc}(\kappa, m)$. It outputs the ciphertext $\langle u, \chi\rangle$.
- $\overline{\mathsf{AE}}.\mathsf{Dec}(k, \langle u, \chi\rangle)$ computes $\kappa = \mathsf{H}(u^k, u)$ and outputs $m/\perp \leftarrow \mathsf{AE}.\mathsf{Dec}(\kappa, \chi)$.

In the LLJ scheme, $\overline{\mathsf{AE}}$ should have RKA security w.r.t. $\mathcal{F}_{\text{aff}} = \{f : k \longmapsto ak + b \mid a \neq 0\}$. Let us check their security proof. See Table 2. The proof idea is to use the DDH assumption to make sure that each κ_λ, $\lambda \in [Q_e]$, is random to the adversary. Then the INT-OT of AE guarantees that the adversary cannot make a fresh forgery $\big(f^* = (a^*, b^*), \langle u^*, \chi^*\rangle\big)$ such that $\overline{\mathsf{AE}}.\mathsf{Dec}(a^*k + b^*, \langle u^*, \chi^*\rangle) \neq \perp$.

In [LLJ15], the indistinguishability of Game 1.$(i-1)$ and Game 1.i is reduced to the DDH assumption. A PPT algorithm $\mathcal{B}$ is constructed to solve the DDH problem by employing an INT-$\mathcal{F}_{\text{aff}}$-RKA adversary $\mathcal{A}$. Given the challenge (g, g^{r_i}, g^k, Z), $\mathcal{B}$ wants to tell whether $Z = g^{kr_i}$ or $Z = g^{z_i}$ for a random z_i. $\mathcal{B}$ simulates the INT-$\mathcal{F}_{\text{aff}}$-RKA game for $\mathcal{A}$ by computing $\kappa_i = \mathsf{H}(Z^{a_i} g^{r_i b_i}, g^{r_i})$. If $Z = g^{kr_i}$, $\mathcal{B}$ simulates Game 1.$(i-1)$ for $\mathcal{A}$; if $Z = g^{z_i}$, $\mathcal{B}$ simulates Game 1.i.

The problem is now that $\mathcal{B}$ does not know the value of secret key k (it knows g^k). When $\mathcal{A}$ submits a fresh forgery $\big(f^* = (a^*, b^*), \langle u^*, \chi^*\rangle\big)$, $\mathcal{B}$ is not able to see whether $\overline{\mathsf{AE}}.\mathsf{Dec}(a^*k + b^*, \langle u^*, \chi^*\rangle) \neq \perp$ or not without the knowledge of k. More precisely, $\mathcal{B}$ can not compute $\kappa^* = \mathsf{H}(u^{*a^*k+b^*}, u^*) = \mathsf{H}\big((u^{*k})^{a^*} \cdot u^{*b^*}, u^*\big)$ from g^k and u^*, unless it is able to compute the CDH value u^{*k} from g^k and u^*. Without κ^*, it is hard for $\mathcal{B}$ to decide whether $\mathsf{AE}.\mathsf{Dec}(\kappa^*, \chi^*) \neq \perp$ or not. In other words, $\mathcal{B}$ cannot find an efficient (PPT) way to transform the computing power (forgery) of $\mathcal{A}$ into its own decisional power (decision bit) to determine (g, g^{r_i}, g^k, Z) to be a DDH tuple or a random tuple. The failure of the INT-$\mathcal{F}_{\text{aff}}$-RKA security proof results in the failure of the KDM[$\mathcal{F}_{\text{aff}}$]-CCA proof of the LLJ scheme since INT-$\mathcal{F}_{\text{aff}}$-RKA security is used to prevent a KDM[$\mathcal{F}_{\text{aff}}$]-CCA adversary from learning more information about the secret key by querying some invalid ciphertexts for decryption.

4 Authenticated Encryption with Auxiliary-Input

We do not see any hope of successfully fixing the security proof of the LLJ's $\overline{\mathsf{AE}}$ in [LLJ15]. Alternatively, we resort to a different building block, namely AIAE.

Table 2. INT-$\mathcal{F}_{\text{aff}}$-RKA security proof of $\overline{\mathsf{AE}}$ in the LLJ scheme; we point out a flaw in the security reduction from Game 1.$(i-1)$ to Game 1.i, denoted by "**?**".

	$\textsc{Enc}(m_\lambda, f_\lambda = (a_\lambda, b_\lambda))$ oracle, $\lambda \in [Q_e]$, where Q_e is the number of encryption queries	Assumptions
Game 0	$r_\lambda \leftarrow_\$ \mathbb{Z}_N$; $u_\lambda := g^{r_\lambda}$; $\kappa_\lambda := \mathsf{H}(u_\lambda^{(a_\lambda k + b_\lambda)}, u_\lambda)$; $\chi_\lambda \leftarrow_\$ \mathsf{AE.Enc}(\kappa_\lambda, m_\lambda)$; return $\overline{\mathsf{ae}}.\mathsf{ct}_\lambda := \langle u_\lambda, \chi_\lambda \rangle$.	–
Game 1	Same as Game 0 except $\kappa_\lambda := \mathsf{H}((g^{kr_\lambda})^{a_\lambda} g^{r_\lambda b_\lambda}, g^{r_\lambda})$.	Game 1 = Game 0
Game 1.i	For $\lambda = 1, \cdots, i$, the same as Game 1 except $\kappa_\lambda := \mathsf{H}((g^{z_\lambda})^{a_\lambda} g^{r_\lambda b_\lambda}, g^{r_\lambda})$ with $z_\lambda \leftarrow_\$ \mathbb{Z}_N$; For $\lambda = i+1, \cdots Q_e$, the same as Game 1.	DDH (**?**)
Game 2	Game 2 = Game 1.Q_e	INT-OT of AE

The intuition is as follows. If LLJ's $\overline{\mathsf{AE}}$ is regarded as (ElGamal + OT-AE), we can design a new AIAE as (Kurosawa-Desmedt [KD04] + OT-AE). But a new problem with our design arises: the secret key of KEM [KD04] consists of several elements, i.e., $\mathsf{k} = (k_1, k_2, k_3, k_4)$. The affine function of k is too complicated to prove the INT-$\mathcal{F}_{\text{aff}}$-RKA security. Fortunately, (a weak) INT-RKA security follows w.r.t. a smaller restricted affine function set $\mathcal{F}_{\text{raff}} = \big\{f : (k_1, k_2, k_3, k_4) \longmapsto a \cdot (k_1, k_2, k_3, k_4) + (b_1, b_2, b_3, b_4) \;\big|\; a \neq 0\big\}$.

To make AIAE serve KDM-CCA security of our PKE construction in Fig. 1, we have the following requirements.

- AIAE must have auxiliary input aux.
- A weak INT-$\mathcal{F}$-RKA security is defined for AIAE. Compared to INT-$\mathcal{F}$-RKA security, the weak version has an additional special rule for the adversary's forgery $(\mathsf{aux}^*, f^*, \mathsf{aiae.ct}^*)$ to be successful: if the adversary has already queried (m, aux^*, f) to the encryption oracle $\textsc{Enc}$, it must hold that $f^* = f$.

Next, we introduce the formal definitions of *Authenticated Encryption with Auxiliary-Input*, its *IND-$\mathcal{F}$-RKA Security* and *Weak INT-$\mathcal{F}$-RKA Security*.

4.1 AIAE and Its Related-Key Attack Security

Definition 9 (AIAE). *An auxiliary-input authenticated encryption (AIAE) scheme* $\mathsf{AIAE} = (\mathsf{AIAE.Setup}, \mathsf{AIAE.Enc}, \mathsf{AIAE.Dec})$ *consists of three PPT algorithms:*

- $\mathsf{AIAE.Setup}(1^\ell)$ *outputs a system parameter* $\mathsf{prm}_{\mathsf{AIAE}}$*, which is an implicit input to* $\mathsf{AIAE.Enc}$ *and* $\mathsf{AIAE.Dec}$*. The parameter* $\mathsf{prm}_{\mathsf{AIAE}}$ *implicitly defines a message space* $\mathcal{M}$*, a key space* $\mathcal{K}_{\mathsf{AIAE}}$ *and an auxiliary-input space* $\mathcal{AUX}$*.*
- $\mathsf{AIAE.Enc}(\mathsf{k}, m, \mathsf{aux})$ *takes as input a key* $\mathsf{k} \in \mathcal{K}_{\mathsf{AIAE}}$*, a message* $m \in \mathcal{M}$ *and an auxiliary input* $\mathsf{aux} \in \mathcal{AUX}$*, and outputs a ciphertext* $\mathsf{aiae.ct}$*.*
- $\mathsf{AIAE.Dec}(\mathsf{k}, \mathsf{aiae.ct}, \mathsf{aux})$ *takes as input a key* $\mathsf{k} \in \mathcal{K}_{\mathsf{AE}}$*, a ciphertext* $\mathsf{aiae.ct}$ *and an auxiliary input* $\mathsf{aux} \in \mathcal{AUX}$*, and outputs a message* $m \in \mathcal{M}$ *or a rejection symbol* $\perp$*.*

Correctness of AIAE *requires that, for all* $\mathsf{prm}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.Setup}(1^{\ell})$*, all* $\mathsf{k} \in \mathcal{K}_{\mathsf{AIAE}}$*, all* $m \in \mathcal{M}$*, all* $\mathsf{aux} \in \mathcal{AUX}$ *and all* $\mathsf{aiae.ct} \leftarrow_{\$} \mathsf{AIAE.Enc}(\mathsf{k}, m, \mathsf{aux})$*, we have that* $\mathsf{AIAE.Dec}(\mathsf{k}, \mathsf{aiae.ct}, \mathsf{aux}) = m$*.*

If the auxiliary-input space $\mathcal{AUX} = \emptyset$ for all possible parameters $\mathsf{prm}_{\mathsf{AIAE}}$, the above definition is reduced to traditional AE.

Let $\mathcal{F}$ be a family of functions from $\mathcal{K}_{\mathsf{AIAE}}$ to $\mathcal{K}_{\mathsf{AIAE}}$. We define the related-key security notions for AIAE via Fig. 5.

```
Procedure INITIALIZE:
$\mathsf{prm}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.Setup}(1^{\ell})$,  $\mathsf{k} \leftarrow_{\$} \mathcal{K}_{\mathsf{AIAE}}$.
$\beta \leftarrow_{\$} \{0,1\}$.            // challenge bit
Return $\mathsf{prm}_{\mathsf{AIAE}}$.

Procedure ENC($m_0, m_1, \mathsf{aux}, f \in \mathcal{F}$):
If $|m_0| \neq |m_1|$, Return $\perp$.
$\mathsf{aiae.ct} \leftarrow_{\$} \mathsf{AIAE.Enc}(f(\mathsf{k}), m_{\beta}, \mathsf{aux})$.
Return $\mathsf{aiae.ct}$.

Procedure FINALIZE($\beta'$):
Return $(\beta' = \beta)$.
```

```
Procedure INITIALIZE:
$\mathsf{prm}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.Setup}(1^{\ell})$,  $\mathsf{k} \leftarrow_{\$} \mathcal{K}_{\mathsf{AIAE}}$.
Return $\mathsf{prm}_{\mathsf{AIAE}}$.

Procedure ENC($m, \mathsf{aux}, f \in \mathcal{F}$):
$\mathsf{aiae.ct} \leftarrow_{\$} \mathsf{AIAE.Enc}(f(\mathsf{k}), m, \mathsf{aux})$.
$\mathcal{Q}_{\mathcal{ENC}} := \mathcal{Q}_{\mathcal{ENC}} \cup \{(\mathsf{aux}, f, \mathsf{aiae.ct})\}$.
$\mathcal{Q}_{\mathcal{AUXF}} := \mathcal{Q}_{\mathcal{AUXF}} \cup \{(\mathsf{aux}, f)\}$.
Return $\mathsf{aiae.ct}$.

Procedure FINALIZE($\mathsf{aux}^*, f^* \in \mathcal{F}, \mathsf{aiae.ct}^*$):
If $(\mathsf{aux}^*, f^*, \mathsf{aiae.ct}^*) \in \mathcal{Q}_{\mathcal{ENC}}$, Return 0.
// Special rule:
If there exists $(\mathsf{aux}, f) \in \mathcal{Q}_{\mathcal{AUXF}}$ such that
    $\mathsf{aux} = \mathsf{aux}^*$ but $f \neq f^*$, Return 0.
Return $(\mathsf{AIAE.Dec}(f^*(\mathsf{k}), \mathsf{aiae.ct}^*, \mathsf{aux}^*) \neq \perp)$.
```

Fig. 5. Games $\mathsf{IND}\text{-}\mathcal{F}\text{-}\mathsf{RKA}$ (left) and $\mathsf{weak}\text{-}\mathsf{INT}\text{-}\mathcal{F}\text{-}\mathsf{RKA}$ (right) for defining securities of auxiliary-input authenticated encryption scheme AIAE. We note that the weak INT-$\mathcal{F}$-RKA security needs a special rule to return 0 in FINALIZE as shown in the shadow.

Definition 10 (IND-$\mathcal{F}$-RKA and Weak INT-$\mathcal{F}$-RKA Securities for AIAE). *Scheme* AIAE *is IND-$\mathcal{F}$-RKA secure and weak INT-$\mathcal{F}$-RKA secure, if for any PPT adversary* $\mathcal{A}$*, both* $\mathsf{Adv}^{ind\text{-}rka}_{\mathsf{AIAE},\mathcal{A}}(\ell) := |\Pr[\mathsf{IND}\text{-}\mathcal{F}\text{-}\mathsf{RKA}^{\mathcal{A}} \Rightarrow 1] - 1/2|$ *and* $\mathsf{Adv}^{weak\text{-}int\text{-}rka}_{\mathsf{AIAE},\mathcal{A}}(\ell) := \Pr[\mathsf{weak}\text{-}\mathsf{INT}\text{-}\mathcal{F}\text{-}\mathsf{RKA}^{\mathcal{A}} \Rightarrow 1]$ *are negligible in* ℓ*, where games* $\mathsf{IND}\text{-}\mathcal{F}\text{-}\mathsf{RKA}$ *and* $\mathsf{weak}\text{-}\mathsf{INT}\text{-}\mathcal{F}\text{-}\mathsf{RKA}$ *are specified in Fig. 5.*

4.2 AIAE from OT-secure AE and DDH Assumption

Let $\mathsf{AE} = (\mathsf{AE.Setup}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$ be a traditional (without auxiliary-input) authenticated encryption scheme with key space $\mathcal{K}_{\mathsf{AE}}$ and message space $\mathcal{M}$. Let $\mathcal{H}_1 = \{\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_N\}$ and $\mathcal{H}_2 = \{\mathsf{H}_2 : \mathbb{QR}_{\bar{N}} \rightarrow \mathcal{K}_{\mathsf{AE}}\}$ be two families of hash functions with $|\mathcal{K}_{\mathsf{AE}}|/|\mathbb{QR}_{\bar{N}}|$ $(= |\mathcal{K}_{\mathsf{AE}}|/N) \leq 2^{-\Omega(\ell)}$. The proposed scheme $\mathsf{AIAE} = (\mathsf{AIAE.Setup}, \mathsf{AIAE.Enc}, \mathsf{AIAE.Dec})$ with key space $\mathcal{K}_{\mathsf{AIAE}} = (\mathbb{Z}_N)^4$, message space $\mathcal{M}$ and auxiliary-input space $\mathcal{AUX} = \{0,1\}^*$ is defined in Fig. 6.

The correctness of AIAE follows from the correctness of AE directly. Note that the factors p, q of N in $\mathsf{prm}_{\mathsf{AIAE}}$ are not needed in the encryption and decryption algorithms of AIAE. Jumping ahead, the factors p, q are necessary when the

```
$\underline{\mathsf{prm}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.Setup}(1^{\ell})\text{:}}$
$(N, p, q) \leftarrow_{\$} \mathsf{GenN}(1^{\ell})$, i.e., pick two $\ell$-bit safe primes $p$ and $q$, such that $2pq + 1$
    is also a prime, and $N := pq$.
$\bar{N} := 2N + 1 = 2pq + 1.$  $g_1, g_2 \leftarrow_{\$} \mathbb{QR}_{\bar{N}}$.  $\mathsf{H}_1 \leftarrow_{\$} \mathcal{H}_1$,  $\mathsf{H}_2 \leftarrow_{\$} \mathcal{H}_2$.
Return $\mathsf{prm}_{\mathsf{AIAE}} := (N, p, q, \bar{N}, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2)$.
```

```
$\underline{\langle c_1, c_2, \chi \rangle \leftarrow_{\$} \mathsf{AIAE.Enc}(\mathsf{k}, m, \mathsf{aux})\text{:}}$
Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in \mathbb{Z}_N^4$.
$w \leftarrow_{\$} \mathbb{Z}_N \backslash \{0\}$.  $(c_1, c_2) := (g_1^w, g_2^w) \in \mathbb{QR}_{\bar{N}}^2$.
$t := \mathsf{H}_1(c_1, c_2, \mathsf{aux}) \in \mathbb{Z}_N$.
$\kappa := \mathsf{H}_2\big(c_1^{k_1 + k_3 t} \cdot c_2^{k_2 + k_4 t}\big) \in \mathcal{K}_{\mathsf{AE}}$.
$\chi \leftarrow_{\$} \mathsf{AE.Enc}(\kappa, m)$.
Return $\langle c_1, c_2, \chi \rangle$.
```

```
$\underline{m/\perp \leftarrow \mathsf{AIAE.Dec}\big(\mathsf{k}, \langle c_1, c_2, \chi \rangle, \mathsf{aux}\big)\text{:}}$
Parse $\mathsf{k} = (k_1, k_2, k_3, k_4) \in \mathbb{Z}_N^4$.
If $(c_1, c_2) \notin \mathbb{QR}_{\bar{N}}^2 \vee (c_1, c_2) = (1, 1)$,
    Return $\perp$.
$t := \mathsf{H}_1(c_1, c_2, \mathsf{aux}) \in \mathbb{Z}_N$.
$\kappa := \mathsf{H}_2\big(c_1^{k_1 + k_3 t} \cdot c_2^{k_2 + k_4 t}\big) \in \mathcal{K}_{\mathsf{AE}}$.
Return $m/\perp \leftarrow \mathsf{AE.Dec}(\kappa, \chi)$.
```

Fig. 6. Construction of the DDH-based AIAE from AE.

security of the PKEs presented in Sections 5 and 6 is reduced to the security of AIAE. We now show the RKA-security of AIAE through the following theorem.

Theorem 1. *If the underlying scheme* AE *is OT-secure, the DDH assumption holds w.r.t.* GenN *and* $\mathbb{QR}_{\bar{N}}$, $\mathcal{H}_1$ *is collision resistant and* $\mathcal{H}_2$ *is universal, then the resulting scheme* AIAE *in Fig. 6 is IND-*$\mathcal{F}_{raff}$*-RKA and weak INT-*$\mathcal{F}_{raff}$*-RKA secure, where the restricted affine function set is defined as* $\mathcal{F}_{raff} := \big\{ f_{(a,\mathsf{b})} : (k_1, k_2, k_3, k_4) \in \mathbb{Z}_N^4 \longmapsto (ak_1 + b_1, ak_2 + b_2, ak_3 + b_3, ak_4 + b_4) \in \mathbb{Z}_N^4 \bigm| a \in \mathbb{Z}_N^*,\ \mathsf{b} = (b_1, b_2, b_3, b_4) \in \mathbb{Z}_N^4 \big\}$.

Proof of IND-$\mathcal{F}_{\mathrm{raff}}$-RKA security of AIAE in Theorem 1. The proof proceeds with a sequence of games. Suppose that $\mathcal{A}$ is a PPT adversary against the IND-$\mathcal{F}_{\mathrm{raff}}$-RKA security of AIAE, who makes at most Q_e times of Enc queries. Let $\Pr_i[\cdot]$ (resp., $\Pr_{i'}[\cdot]$) denote the probability of a particular event occurring in game G_i (resp., game G'_i).

– Game G_1: This is the original IND-$\mathcal{F}_{\mathrm{raff}}$-RKA security game. Let Win denote the event that $\beta' = \beta$. Then by definition, $\mathsf{Adv}^{ind\text{-}rka}_{\mathsf{AIAE},\mathcal{A}}(\ell) = \big| \Pr_1[\mathsf{Win}] - \frac{1}{2} \big|$.
  Denote $\mathsf{prm}_{\mathsf{AIAE}} = (N, p, q, \bar{N}, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2)$ and $\mathsf{k} = (k_1, k_2, k_3, k_4)$. To answer the λ-th ($\lambda \in [Q_e]$) Enc query $(m_{\lambda,0}, m_{\lambda,1}, \mathsf{aux}_\lambda, f_\lambda)$, where $f_\lambda = \langle a_\lambda, \mathsf{b}_\lambda = (b_{\lambda,1}, b_{\lambda,2}, b_{\lambda,3}, b_{\lambda,4}) \rangle \in \mathcal{F}_{\mathrm{raff}}$, the challenger proceeds as follows:
  1. pick $w_\lambda \leftarrow_{\$} \mathbb{Z}_N \backslash \{0\}$ and compute $(c_{\lambda,1}, c_{\lambda,2}) := (g_1^{w_\lambda}, g_2^{w_\lambda}) \in \mathbb{QR}_{\bar{N}}^2$,
  2. compute a tag $t_\lambda := \mathsf{H}_1(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda) \in \mathbb{Z}_N$,
  3. compute an encryption key for AE scheme using a related key $f_\lambda(\mathsf{k})$:
  $$\kappa_\lambda := \mathsf{H}_2\big(c_{\lambda,1}^{(a_\lambda k_1 + b_{\lambda,1}) + (a_\lambda k_3 + b_{\lambda,3}) t_\lambda} \cdot c_{\lambda,2}^{(a_\lambda k_2 + b_{\lambda,2}) + (a_\lambda k_4 + b_{\lambda,4}) t_\lambda}\big) \in \mathcal{K}_{\mathsf{AE}},$$
  4. invoke $\chi_\lambda \leftarrow_{\$} \mathsf{AE.Enc}(\kappa_\lambda, m_{\lambda,\beta})$,

  and returns the challenge ciphertext $\langle c_{\lambda,1}, c_{\lambda,2}, \chi_\lambda \rangle$ to the adversary $\mathcal{A}$.
– Game $\mathsf{G}_{1,i}$, $i \in [Q_e + 1]$: This game is the same as game G_1, except that, the challenger does not use secret key k to answer the λ-th ($\lambda \in [i - 1]$) Enc query at all, and instead, it changes steps 1, 3 to steps 1′, 3′ as follows:

1′. pick $w_{\lambda,1}, w_{\lambda,2} \leftarrow_{\$} \mathbb{Z}_N \backslash \{0\}$ and compute $(c_{\lambda,1}, c_{\lambda,2}) := (g_1^{w_{\lambda,1}}, g_2^{w_{\lambda,2}})$,
3′. choose an encryption key $\kappa_\lambda \leftarrow_{\$} \mathcal{K}_{\mathsf{AE}}$ randomly for the AE scheme.
The challenger still answers the λ-th ($\lambda \in [i, Q_e]$) Enc query as in G_1, i.e., using steps 1, 3.

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

– Game $\mathsf{G}'_{1,i}$, $i \in [Q_e]$: This game is the same as game $\mathsf{G}_{1,i}$, except that the challenger answers the i-th Enc query using steps 1′, 3 (rather than steps 1, 3 in game $\mathsf{G}_{1,i}$).

The only difference between $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$ is the distribution of $(g_1, g_2, c_{i,1}, c_{i,2})$. In game $\mathsf{G}_{1,i}$, $(g_1, g_2, c_{i,1}, c_{i,2})$ is a DDH tuple, while in game $\mathsf{G}'_{1,i}$, it is a random tuple. It is straightforward to construct a PPT adversary to solve the DDH problem w.r.t. GenN and $\mathbb{QR}_{\bar{N}}$, thus we have that $\left| \Pr_{1,i}[\mathsf{Win}] - \Pr_{1,i'}[\mathsf{Win}] \right| \leq \mathsf{Adv}^{ddh}_{\mathsf{GenN}}(\ell)$.

We analyze the difference between $\mathsf{G}'_{1,i}$ and $\mathsf{G}_{1,i+1}$ via the following lemma. Its proof is provided in the full version [HLL16].

Lemma 1. *For all $i \in [Q_e]$, $\left| \Pr_{1,i'}[\mathsf{Win}] - \Pr_{1,i+1}[\mathsf{Win}] \right| \leq \frac{1}{N-1} + 2^{-\Omega(\ell)}$.*

– Game G_2: This game is the same as game G_{1,Q_e+1}, except that, to answer the λ-th ($\lambda \in [Q_e]$) Enc query, the challenger changes step 4 to step 4′:
4′. invoke $\chi_\lambda \leftarrow_{\$} \mathsf{AE.Enc}(\kappa_\lambda, 0^{|m_{\lambda,0}|})$.

In game G_{1,Q_e+1}, the challenger computes the AE encryption of $m_{\lambda,\beta}$ under encryption key κ_λ in Enc, while in game G_2 it computes the AE encryption of $0^{|m_{\lambda,0}|}$ in Enc. Both in games G_{1,Q_e+1} and G_2, we have that each κ_λ is chosen uniformly from $\mathcal{K}_{\mathsf{AE}}$ and independent of other parts of the game. Therefore we can reduce the differences between G_{1,Q_e+1} and G_2 to the IND-OT security of AE by a standard hybrid argument, and have that $\left| \Pr_{1,Q_e+1}[\mathsf{Win}] - \Pr_2[\mathsf{Win}] \right| \leq Q_e \cdot \mathsf{Adv}^{ind\text{-}ot}_{\mathsf{AE}}(\ell)$.

Now in game G_2, since the challenger always encrypts the constant message $0^{|m_{\lambda,0}|}$, the challenge bit β is completely hidden. Then $\Pr_2[\mathsf{Win}] = 1/2$.

Taking all things together, the IND-$\mathcal{F}_{\text{raff}}$-RKA security of AIAE follows. ∎

Proof of Weak INT-$\mathcal{F}_{\text{raff}}$-RKA security of AIAE in Theorem 1. Again, we prove it through a sequence of games. These games are defined almost the same as those in the previous proof. Suppose that $\mathcal{A}$ is a PPT adversary against the weak INT-$\mathcal{F}_{\text{raff}}$-RKA security of AIAE, who makes at most Q_e times of Enc queries.

– Game G_0: This is the original weak-INT-$\mathcal{F}_{\text{raff}}$-RKA security game.

Denote $\mathsf{prm}_{\mathsf{AIAE}} = (N, p, q, \bar{N}, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2)$ and $\mathsf{k} = (k_1, k_2, k_3, k_4)$. To answer the λ-th ($\lambda \in [Q_e]$) Enc query $(m_\lambda, \mathsf{aux}_\lambda, f_\lambda)$, the challenger proceeds with steps 1∼4, similar to the previous proof, and returns the challenge ciphertext $\langle c_{\lambda,1}, c_{\lambda,2}, \chi_\lambda \rangle$ to the adversary $\mathcal{A}$. Moreover, the challenger will put $\big(\mathsf{aux}_\lambda, f_\lambda, \langle c_{\lambda,1}, c_{\lambda,2}, \chi_\lambda \rangle\big)$ to a set $\mathcal{Q}_{\mathcal{ENC}}$, put $(\mathsf{aux}_\lambda, f_\lambda)$ to a set $\mathcal{Q}_{\mathcal{AUXF}}$, and put $(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda, t_\lambda)$ to a set $\mathcal{Q}_{\mathcal{TAG}}$. Finally, the adversary outputs a forgery $\big(\mathsf{aux}^*, f^* = \langle a^*, \mathsf{b}^* = (b_1^*, b_2^*, b_3^*, b_4^*) \rangle, \langle c_1^*, c_2^*, \chi^* \rangle\big)$.

Let Forge be the event that the following Finalize procedure outputs 1:

- If $\big(\mathsf{aux}^*, f^*, \langle c_1^*, c_2^*, \chi^*\rangle\big) \in \mathcal{Q}_{\mathcal{ENC}}$, Return 0.
- If there exists $(\mathsf{aux}_\lambda, f_\lambda) \in \mathcal{Q}_{\mathcal{AUXF}}$ such that $\mathsf{aux}_\lambda = \mathsf{aux}^*$ but $f_\lambda \neq f^*$, Return 0.
- If $(c_1^*, c_2^*) \notin \mathbb{QR}_{\bar{N}}^2 \vee (c_1^*, c_2^*) = (1,1)$, Return 0.
- $t^* := \mathsf{H}_1(c_1^*, c_2^*, \mathsf{aux}^*)$, $\kappa^* := \mathsf{H}_2\big(c_1^{*(a^*k_1+b_1^*)+(a^*k_3+b_3^*)t^*} \cdot c_2^{*(a^*k_2+b_2^*)+(a^*k_4+b_4^*)t^*}\big)$. Return $(\mathsf{AE.Dec}(\kappa^*, \chi^*) \neq \perp)$.

By definition, it follows that, $\mathsf{Adv}^{weak\text{-}int\text{-}rka}_{\mathsf{AIAE},\mathcal{A}}(\ell) = \Pr_0[\mathsf{Forge}]$.

– Game G_1: This game is the same as game G_0, except that, the challenger adds the following new rule to the FINALIZE procedure:
  - If there exists $(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda, t_\lambda) \in \mathcal{Q}_{\mathcal{TAG}}$ such that $t_\lambda = t^*$ but $(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda) \neq (c_1^*, c_2^*, \mathsf{aux}^*)$, Return 0.

  Since $t_\lambda = \mathsf{H}_1(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda)$ and $t^* = \mathsf{H}_1(c_1^*, c_2^*, \mathsf{aux}^*)$, any difference between G_0 and G_1 will imply a collision of H_1. Thus $\big|\Pr_0[\mathsf{Forge}] - \Pr_1[\mathsf{Forge}]\big| \leq \mathsf{Adv}^{cr}_{\mathcal{H}_1}(\ell)$.

– Game $\mathsf{G}_{1,i}$, $i \in [Q_e+1]$: This game is the same as game G_1, except that, the challenger does not use secret key k to answer the λ-th ($\lambda \in [i-1]$) ENC query at all, and instead, it changes the steps 1, 3 to the steps 1′, 3′ respectively, as in the previous proof.

  Clearly $\Pr_1[\mathsf{Forge}] = \Pr_{1,1}[\mathsf{Forge}]$.

– Game $\mathsf{G}'_{1,i}$, $i \in [Q_e]$: This game is the same as game $\mathsf{G}_{1,i}$, except that the challenger answers the i-th ENC query using steps 1′, 3 (rather than steps 1, 3 in game $\mathsf{G}_{1,i}$), as in the previous proof.

  The only difference between $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$ is the distribution of $(g_1, g_2, c_{i,1}, c_{i,2})$. In game $\mathsf{G}_{1,i}$, $(g_1, g_2, c_{i,1}, c_{i,2})$ is a DDH tuple, while in game $\mathsf{G}'_{1,i}$, it is a random tuple. It is straightforward to construct a PPT adversary to solve the DDH problem w.r.t. GenN and $\mathbb{QR}_{\bar{N}}$. We stress that the PPT adversary (simulator) can detect the occurrence of event Forge efficiently since it can choose the secret key $\mathsf{k} = (k_1, k_2, k_3, k_4)$ itself. Thus we can reduce the difference between $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$ to the DDH assumption smoothly.

Lemma 2. *For all* $i \in [Q_e]$, $\big|\Pr_{1,i}[\mathsf{Forge}] - \Pr_{1,i'}[\mathsf{Forge}]\big| \leq \mathsf{Adv}^{ddh}_{\mathsf{GenN}}(\ell)$.

Proof. We construct a PPT adversary $\mathcal{B}$ to solve the DDH problem. $\mathcal{B}$ is given $(N, p, q, g_1, g_2, g_1^{x_1}, g_2^{x_2})$, where $(N, p, q) \leftarrow_\$ \mathsf{GenN}(1^\ell)$, $g_1, g_2 \leftarrow_\$ \mathbb{QR}_{\bar{N}}$, and aims to distinguish whether $x_1 = x_2 \leftarrow_\$ \mathbb{Z}_N \setminus \{0\}$ or $x_1, x_2 \leftarrow_\$ \mathbb{Z}_N \setminus \{0\}$.

$\mathcal{B}$ will simulate game $\mathsf{G}_{1,i}$ or $\mathsf{G}'_{1,i}$ for adversary $\mathcal{A}$. First, $\mathcal{B}$ picks $\mathsf{H}_1 \leftarrow_\$ \mathcal{H}_1$, $\mathsf{H}_2 \leftarrow_\$ \mathcal{H}_2$ randomly, sets $\mathsf{prm}_{\mathsf{AIAE}} := (N, p, q, \bar{N} = 2N+1, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2)$ and sends $\mathsf{prm}_{\mathsf{AIAE}}$ to $\mathcal{A}$. Then $\mathcal{B}$ generates the secret key $\mathsf{k} = (k_1, k_2, k_3, k_4)$ itself.

To answer the λ-th ($\lambda \in [Q_e]$) ENC query $(m_\lambda, \mathsf{aux}_\lambda, f_\lambda)$, where $f_\lambda = \langle a_\lambda, \mathsf{b}_\lambda = (b_{\lambda,1}, b_{\lambda,2}, b_{\lambda,3}, b_{\lambda,4})\rangle \in \mathcal{F}_{\text{raff}}$, $\mathcal{B}$ proceeds as follows:

- If $\lambda \in [i-1]$, $\mathcal{B}$ proceeds the same as in $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$. That is, $\mathcal{B}$ picks $w_{\lambda,1}, w_{\lambda,2} \leftarrow_\$ \mathbb{Z}_N \setminus \{0\}$ randomly and sets $(c_{\lambda,1}, c_{\lambda,2}) := (g_1^{w_{\lambda,1}}, g_2^{w_{\lambda,2}})$. Then $\mathcal{B}$ chooses $\kappa_\lambda \leftarrow_\$ \mathcal{K}_{\mathsf{AE}}$ and invokes $\chi_\lambda \leftarrow_\$ \mathsf{AE.Enc}(\kappa_\lambda, m_\lambda)$.
- If $\lambda \in [i+1, Q_e]$, $\mathcal{B}$ proceeds the same as in $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$. That is, $\mathcal{B}$ picks $w_\lambda \leftarrow_\$ \mathbb{Z}_N \setminus \{0\}$ randomly and sets $(c_{\lambda,1}, c_{\lambda,2}) := (g_1^{w_\lambda}, g_2^{w_\lambda})$. Then $\mathcal{B}$ computes $t_\lambda := \mathsf{H}_1(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda)$, $\kappa_\lambda := \mathsf{H}_2\big(c_{\lambda,1}^{(a_\lambda k_1+b_{\lambda,1})+(a_\lambda k_3+b_{\lambda,3})t_\lambda} \cdot c_{\lambda,2}^{(a_\lambda k_2+b_{\lambda,2})+(a_\lambda k_4+b_{\lambda,4})t_\lambda}\big)$, and invokes $\chi_\lambda \leftarrow_\$ \mathsf{AE.Enc}(\kappa_\lambda, m_\lambda)$.

- If $\lambda = i$, $\mathcal{B}$ embedded its DDH challenge to $(c_{i,1}, c_{i,2}) := (g_1^{x_1}, g_2^{x_2})$. Then it computes $t_i := \mathsf{H}_1(c_{i,1}, c_{i,2}, \mathsf{aux}_i)$, $\kappa_i := \mathsf{H}_2\big(c_{i,1}^{(a_i k_1 + b_{i,1}) + (a_i k_3 + b_{i,3}) t_i} \cdot c_{i,2}^{(a_i k_2 + b_{i,2}) + (a_i k_4 + b_{i,4}) t_i}\big)$, and invokes $\chi_i \leftarrow_{\$} \mathsf{AE.Enc}(\kappa_i, m_i)$.

$\mathcal{B}$ returns the challenge ciphertext $\langle c_{\lambda,1}, c_{\lambda,2}, \chi_\lambda \rangle$ to $\mathcal{A}$, and puts $\big(\mathsf{aux}_\lambda, f_\lambda, \langle c_{\lambda,1}, c_{\lambda,2}, \chi_\lambda \rangle\big)$ to $\mathcal{Q}_{\mathcal{ENC}}$, $(\mathsf{aux}_\lambda, f_\lambda)$ to $\mathcal{Q}_{\mathcal{AUXF}}$, and $(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda, t_\lambda)$ to $\mathcal{Q}_{\mathcal{TAG}}$.

In the case of that $(N, p, q, g_1, g_2, g_1^{x_1}, g_2^{x_2})$ is a DDH tuple, i.e., $x_1 = x_2 \leftarrow_{\$} \mathbb{Z}_N \setminus \{0\}$, $\mathcal{B}$ simulates game $\mathsf{G}_{1,i}$ perfectly for $\mathcal{A}$; in the case of that $(N, p, q, g_1, g_2, g_1^{x_1}, g_2^{x_2})$ is a random tuple, i.e., $x_1, x_2 \leftarrow_{\$} \mathbb{Z}_N \setminus \{0\}$, $\mathcal{B}$ simulates game $\mathsf{G}'_{1,i}$ perfectly for $\mathcal{A}$.

Finally $\mathcal{B}$ receives a forgery $\big(\mathsf{aux}^*, f^*, \langle c_1^*, c_2^*, \chi^* \rangle\big)$ from $\mathcal{A}$, where $f^* = \langle a^*, \mathsf{b}^* = (b_1^*, b_2^*, b_3^*, b_4^*) \rangle \in \mathcal{F}_{\text{raff}}$. $\mathcal{B}$ determines whether or not the FINALIZE procedure outputs 1 using the secret key $\mathsf{k} = (k_1, k_2, k_3, k_4)$. That is,

- If $\big(\mathsf{aux}^*, f^*, \langle c_1^*, c_2^*, \chi^* \rangle\big) \in \mathcal{Q}_{\mathcal{ENC}}$, $\mathcal{B}$ outputs 0 (to its DDH challenger).
- If there exists $(\mathsf{aux}_\lambda, f_\lambda) \in \mathcal{Q}_{\mathcal{AUXF}}$ such that $\mathsf{aux}_\lambda = \mathsf{aux}^*$ but $f_\lambda \neq f^*$, $\mathcal{B}$ outputs 0.
- If $(c_1^*, c_2^*) \notin \mathbb{QR}_{\bar{N}}^2 \vee (c_1^*, c_2^*) = (1, 1)$, $\mathcal{B}$ outputs 0.
- $t^* := \mathsf{H}_1(c_1^*, c_2^*, \mathsf{aux}^*)$, $\kappa^* := \mathsf{H}_2\big(c_1^{*(a^* k_1 + b_1^*) + (a^* k_3 + b_3^*) t^*} \cdot c_2^{*(a^* k_2 + b_2^*) + (a^* k_4 + b_4^*) t^*}\big)$.
- If there exists $(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda, t_\lambda) \in \mathcal{Q}_{\mathcal{TAG}}$ such that $t_\lambda = t^*$ but $(c_{\lambda,1}, c_{\lambda,2}, \mathsf{aux}_\lambda) \neq (c_1^*, c_2^*, \mathsf{aux}^*)$, $\mathcal{B}$ outputs 0.
- Output $(\mathsf{AE.Dec}(\kappa^*, \chi^*) \neq \perp)$.

With the secret key $\mathsf{k} = (k_1, k_2, k_3, k_4)$, $\mathcal{B}$ simulates FINALIZE perfectly, the same as in games $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$, and $\mathcal{B}$ outputs 1 to its DDH challenger if and only if FINALIZE outputs 1, i.e., the event Forge occurs.

As a consequence, $\big|\Pr_{1,i}[\mathsf{Forge}] - \Pr_{1,i'}[\mathsf{Forge}]\big| \leq \mathsf{Adv}^{ddh}_{\mathsf{GenN},\mathcal{B}}(\ell)$. ▌

We analyze the difference between $\mathsf{G}'_{1,i}$ and $\mathsf{G}_{1,i+1}$ via the following lemma, and the proof is in the full version [HLL16] due to the lack of space.

Lemma 3. *For all* $i \in [Q_e]$, $\Pr_{1,i'}[\mathsf{Forge}] \leq \Pr_{1,i+1}[\mathsf{Forge}] + \mathsf{Adv}^{int\text{-}ot}_{\mathsf{AE}}(\ell) + \frac{1}{(N-1)} + 2^{-\Omega(\ell)}$.

Now in game G_{1,Q_e+1}, the challenger does not use the secret key k to compute κ_λ at all, hence $\mathsf{k} = (k_1, k_2, k_3, k_4)$ is uniformly random to the adversary $\mathcal{A}$. As a result, in the FINALIZE procedure defining the event Forge,

$$\kappa^* = \mathsf{H}_2\big(\underbrace{g_1^{a^* \cdot ((w_1^* k_1 + w_2^* w k_2) + t^* \cdot (w_1^* k_3 + w_2^* w k_4))} \cdot g_1^{(w_1^* b_1^* + w_2^* w b_2^*) + t^* \cdot (w_1^* b_3^* + w_2^* w b_4^*)}}_{\triangleq Y}\big),$$

where $w = \mathrm{dlog}_{g_1} g_2 \in \mathbb{Z}_N$ and $(w_1^*, w_2^*) = (\mathrm{dlog}_{g_1} c_1^*, \mathrm{dlog}_{g_2} c_2^*) \in \mathbb{Z}_N^2 \setminus \{(0,0)\}$. The term $(w_1^* k_1 + w_2^* w k_2)$ is uniformly distributed over $\mathbb{Z}_N$. Then as long as $a^* \in \mathbb{Z}_N^*$, Y will be uniformly distributed over $\mathbb{QR}_{\bar{N}}$ and independent of H_2. By the Leftover Hash Lemma, $\kappa^* = \mathsf{H}_2(Y)$ is statistically close to the uniform distribution over $\mathcal{K}_{\mathsf{AE}}$. Thus $\mathsf{AE.Dec}(\kappa^*, \chi^*) \neq \perp$ will hold with probability at most $\mathsf{Adv}^{int\text{-}ot}_{\mathsf{AE}}(\ell)$. Then $\Pr_{1,Q_e+1}[\mathsf{Forge}] \leq \mathsf{Adv}^{int\text{-}ot}_{\mathsf{AE}}(\ell) + 2^{-\Omega(\ell)}$.

Taking all things together, the weak INT-$\mathcal{F}_{\text{raff}}$-RKA security of AIAE follows. ▌

Remark. We stress that the problem in the INT-$\mathcal{F}_{\text{aff}}$-RKA security proof of LLJ's $\overline{\mathsf{AE}}$ does not appear here. The weak INT-$\mathcal{F}_{\text{raff}}$-RKA security of our AIAE can be reduced to the DDH assumption smoothly. More precisely, in the security analysis of games $\mathsf{G}_{1,i}$ and $\mathsf{G}'_{1,i}$ (cf. Lemma 2), the simulator chooses the secret key itself and uses it to detect the occurrence of event Forge efficiently. Therefore the simulator can always make use of the difference between $\Pr_{1,i}[\mathsf{Forge}]$ and $\Pr_{1,i'}[\mathsf{Forge}]$ to solve the DDH problem.

5 PKE with n-KDM[$\mathcal{F}_{\text{aff}}$]-CCA Security

Let $\mathsf{AIAE} = (\mathsf{AIAE.Setup}, \mathsf{AIAE.Enc}, \mathsf{AIAE.Dec})$ be the DDH-based auxiliary-input authenticated encryption scheme constructed from OT-secure AE, with key space $(\mathbb{Z}_N)^4$ and a suitable message space $\mathcal{M}$ (cf. Fig. 6). Following our approach in Fig. 1, we have to design the other two building blocks.

- KEM: With respect to this AIAE, we design a KEM which can encapsulate a key tuple $(k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$.
- $\mathcal{E}$: With respect to the affine function $\mathcal{F}_{\text{aff}}$, we design a public-key encryption $\mathcal{E}$ such that $\mathcal{E}.\mathsf{Enc}$ can be changed to an entropy filter for affine functions in a computationally indistinguishable way.

The proposed $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is defined in Fig. 7, where the shadowed parts describe algorithms of building blocks KEM and $\mathcal{E}$.

The correctness of PKE follows from the correctness of AIAE, $\mathcal{E}$ and KEM directly. We now show its KDM-CCA-security through the following theorem.

Theorem 2. *If the underlying scheme* AIAE *is IND-$\mathcal{F}_{raff}$-RKA and weak INT-$\mathcal{F}_{raff}$-RKA secure, the DCR assumption holds w.r.t.* GenN *and group* $\mathbb{QR}_{N^s}$*, and the DL Assumption holds w.r.t.* GenN *and group* $\mathbb{SCR}_{N^s}$*, then the resulting scheme* PKE *in Fig. 7 is n-KDM[$\mathcal{F}_{aff}$]-CCA secure.*

Proof of Theorem 2. Suppose that $\mathcal{A}$ is a PPT adversary against the n-KDM[$\mathcal{F}_{\text{aff}}$]-CCA security of PKE, who makes at most Q_e times of $\textsc{Enc}$ queries and Q_d times of $\textsc{Dec}$ queries. We prove the theorem by defining a sequence of games. Before presenting the full detailed proof, we first give a high-level description how n-KDM[$\mathcal{F}_{\text{aff}}$]-CCA security is achieved.

(1) For the n secret key tuples, each tuple can be divided into two parts: for $i \in [n]$, $\mathsf{sk}_i = (x_{i,j}, y_{i,j})_{j=1}^4 = \big((x_{i,j}, y_{i,j})_{j=1}^4 \bmod N,\ (x_{i,j}, y_{i,j})_{j=1}^4 \bmod \phi(N)/4\big)$.
(2) Each secret key tuple can be generated by adding a random shift $(\overline{x}_{i,j}, \overline{y}_{i,j})_{j=1}^4$ to a fixed base $(x_j, y_j)_{j=1}^4$, i.e., $\mathsf{sk}_i = (x_{i,j}, y_{i,j})_{j=1}^4 := (x_j, y_j)_{j=1}^4 + (\overline{x}_{i,j}, \overline{y}_{i,j})_{j=1}^4$.
(3) Every public key tuple $\mathsf{pk}_i = (h_{i,1}, \cdots, h_{i,4})$ only reveals information about the (mod $\phi(N)/4$) part of the secret key tuple sk_i.
(4) For each encryption query from the adversary (f_λ, i_λ), if the $\textsc{Enc}$ oracle encrypts $f_\lambda(\mathsf{sk}_1, \cdots, \mathsf{sk}_n)$, the ciphertext might reveal information about sk_i through $\mathcal{E}.\mathsf{ct}$. We have to change this fact such that the leaked information about sk_i in $\textsc{Enc}$ is bounded.

```
$\underline{\mathsf{prm} \leftarrow_{\$} \mathsf{Setup}(1^{\ell})}$:
$\mathsf{prm}_{\mathsf{AIAE}} \leftarrow_{\$} \mathsf{AIAE.Setup}(1^{\ell})$, where
    $\mathsf{prm}_{\mathsf{AIAE}} = (N, p, q, \bar{N}, \bar{g}_1, \bar{g}_2, \mathsf{H}_1, \mathsf{H}_2)$,
    $N = pq$, $\bar{N} = 2N + 1$, $\bar{g}_1, \bar{g}_2 \in \mathbb{QR}_{\bar{N}}$.
$\mathsf{prm}'_{\mathsf{AIAE}} := (N, \bar{N}, \bar{g}_1, \bar{g}_2, \mathsf{H}_1, \mathsf{H}_2)$.
$g_1, g_2, g_3, g_4, g_5 \leftarrow_{\$} \mathbb{SCR}_{N^s}$.
Return $\mathsf{prm} := (\mathsf{prm}'_{\mathsf{AIAE}}, g_1, g_2, g_3, g_4, g_5)$.
```

```
$\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Gen}(\mathsf{prm})}$:
$x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \leftarrow_{\$} \left[\lfloor \frac{N^2}{4} \rfloor\right]$.
$(h_1, h_2, h_3, h_4) := (g_1^{-x_1} g_2^{-y_1}, g_2^{-x_2} g_3^{-y_2}$,
    $g_3^{-x_3} g_4^{-y_3}, g_4^{-x_4} g_5^{-y_4}) \bmod N^s$.
$\mathsf{pk} := (h_1, h_2, h_3, h_4)$.
$\mathsf{sk} := (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$.
Return $(\mathsf{pk}, \mathsf{sk})$.
```

```
$\underline{\langle \mathsf{aux}, \mathsf{aiae.ct} \rangle \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}, m)}$:  $m \in [N^{s-1}]$
// $(\mathsf{k}, \mathsf{aux}) \leftarrow_{\$} \mathsf{KEM.Enc}(\mathsf{pk})$:
$\mathsf{k} = (k_1, k_2, k_3, k_4) \leftarrow_{\$} \mathbb{Z}_N^4$.
$r \leftarrow_{\$} \left[\lfloor \frac{N}{4} \rfloor\right]$.
$(u_1, u_2, u_3, u_4, u_5) := (g_1^r, g_2^r, g_3^r, g_4^r, g_5^r)$
    $\bmod N^2$.
$(e_1, e_2, e_3, e_4) := (h_1^r T^{k_1}, h_2^r T^{k_2}, h_3^r T^{k_3}$,
    $h_4^r T^{k_4}) \bmod N^2$.
$\mathsf{aux} := (u_1, \cdots, u_5, e_1, \cdots, e_4)$.
// $\mathcal{E}.\mathsf{ct} \leftarrow_{\$} \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$:
$\tilde{r}_1, \tilde{r}_2, \tilde{r}_3, \tilde{r}_4 \leftarrow_{\$} \left[\lfloor \frac{N}{4} \rfloor\right]$.
$(\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, \tilde{u}_4, \tilde{u}_5, \tilde{u}_6, \tilde{u}_7, \tilde{u}_8) := (g_1^{\tilde{r}_1}, g_2^{\tilde{r}_1}$,
    $g_2^{\tilde{r}_2}, g_3^{\tilde{r}_2}, g_3^{\tilde{r}_3}, g_4^{\tilde{r}_3}, g_4^{\tilde{r}_4}, g_5^{\tilde{r}_4}) \bmod N^s$.
$\tilde{e} := h_1^{\tilde{r}_1} h_2^{\tilde{r}_2} h_3^{\tilde{r}_3} h_4^{\tilde{r}_4} T^m \bmod N^s$.
$t := g_1^m \bmod N \in \mathbb{Z}_N$.
$\mathcal{E}.\mathsf{ct} := (\tilde{u}_1, \cdots, \tilde{u}_8, \tilde{e}, t)$.
$\mathsf{aiae.ct} \leftarrow_{\$} \mathsf{AIAE.Enc}(\mathsf{k}, \mathcal{E}.\mathsf{ct}, \mathsf{aux})$.
Return $\langle \mathsf{aux}, \mathsf{aiae.ct} \rangle$.
```

```
$\underline{m/\perp \leftarrow \mathsf{Dec}(\mathsf{sk}, \langle \mathsf{aux}, \mathsf{aiae.ct} \rangle)}$:
// $\mathsf{k}/\perp \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}, \mathsf{aux})$:
Parse $\mathsf{aux} = (u_1, \cdots, u_5, e_1, \cdots, e_4)$.
If $e_1 u_1^{x_1} u_2^{y_1}, e_2 u_2^{x_2} u_3^{y_2}, e_3 u_3^{x_3} u_4^{y_3}$,
    $e_4 u_4^{x_4} u_5^{y_4} \in \mathbb{RU}_{N^2}$
    $(k_1, k_2, k_3, k_4) := (\mathrm{dlog}_T(e_1 u_1^{x_1} u_2^{y_1})$,
      $\mathrm{dlog}_T(e_2 u_2^{x_2} u_3^{y_2}), \mathrm{dlog}_T(e_3 u_3^{x_3} u_4^{y_3})$,
      $\mathrm{dlog}_T(e_4 u_4^{x_4} u_5^{y_4})) \bmod N$.
    $\mathsf{k} := (k_1, k_2, k_3, k_4)$.
Else, Return $\perp$.
$\mathcal{E}.\mathsf{ct}/\perp \leftarrow \mathsf{AIAE.Dec}(\mathsf{k}, \mathsf{aiae.ct}, \mathsf{aux})$.
// $m/\perp \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, \mathcal{E}.\mathsf{ct})$:
Parse $\mathcal{E}.\mathsf{ct} = (\tilde{u}_1, \cdots, \tilde{u}_8, \tilde{e}, t)$.
If $\tilde{e} \tilde{u}_1^{x_1} \tilde{u}_2^{y_1} \tilde{u}_3^{x_2} \tilde{u}_4^{y_2} \tilde{u}_5^{x_3} \tilde{u}_6^{y_3} \tilde{u}_7^{x_4} \tilde{u}_8^{y_4} \in \mathbb{RU}_{N^s}$
    $m := \mathrm{dlog}_T(\tilde{e} \tilde{u}_1^{x_1} \tilde{u}_2^{y_1} \tilde{u}_3^{x_2} \tilde{u}_4^{y_2} \tilde{u}_5^{x_3} \tilde{u}_6^{y_3}$
      $\tilde{u}_7^{x_4} \tilde{u}_8^{y_4}) \bmod N^{s-1}$.
    If $t = g_1^m \bmod N$, Return $m$.
Else, Return $\perp$.
```

Fig. 7. Construction of PKE from AIAE. The shadowed parts describe algorithms of building blocks KEM and $\mathcal{E}$. Here p, q contained in $\mathsf{prm}_{\mathsf{AIAE}}$ are not provided in $\mathsf{prm}'_{\mathsf{AIAE}}$, since they are not necessary in the encryption and decryption algorithms of AIAE.

- By IV_d assumption, we can change the generation of $\mathcal{E}.\mathsf{ct}$ by oracle Enc such that it does not reveal any information about $(x_j, y_j)_{j=1}^4 \bmod N$, i.e., the (mod N) part of the base secret key tuple.
- By IV_d assumption, we can change the generation of $\mathsf{kem.ct}(= \mathsf{aux})$ by Enc such that it encapsulates a different key, other than the key used in AIAE.Enc. If AIAE.Enc uses key $(r_\lambda k_j^* + s_{\lambda,j})_{j=1}^4$, then KEM.Enc encapsulates $\big(r_\lambda(k_j^* - \alpha_j x_j - \alpha_{j+1} y_j) - r_\lambda(\alpha_j \bar{x}_{i_\lambda,j} + \alpha_{j+1} \bar{y}_{i_\lambda,j}) + s_{\lambda,j}\big)_{j=1}^4 \bmod N$. Thus, $(k_1^*, \cdots, k_4^*)$ is now protected by $(x_j, y_j)_{j=1}^4 \bmod N$.

(5) Oracle Dec might also leak information about $(x_j, y_j)_{j=1}^4 \bmod N$. Therefore, we change how oracle Dec works so that decryption does not use $(x_j, y_j)_{j=1}^4 \bmod N$ any more. Observe that as long as the ciphertext queried

by the adversary satisfies $\forall j \in [5], u_j \in \mathbb{SCR}_{N^2}$ and $\forall j \in [8], \tilde{u}_j \in \mathbb{SCR}_{N^s}$, DEC can use $\phi(N)$ and the (mod $\phi(N)/4$) part of secret key for decryption.

– If $\exists j \in [5], u_j \notin \mathbb{SCR}_{N^2}$ in the ciphertext queried by the adversary, we expect that $\mathsf{AIAE.Dec}$ will reject, due to its weak INT-$\mathcal{F}_{\text{raff}}$-RKA security.
– If $\exists j \in [8], \tilde{u}_j \notin \mathbb{SCR}_{N^s}$ in the ciphertext queried by the adversary, we expect decryption will result in $t \neq g_1^m \bmod N$, so $\mathcal{E}.\mathsf{Dec}$ will reject.

(6) Consequently, both $(x_j, y_j)_{j=1}^4 \bmod N$ and $(k_1^*, \cdots, k_4^*)$ are random to the adversary, and $\mathsf{AIAE.Enc}$ always uses the restricted affine function of $(k_1^*, \cdots, k_4^*)$ for encryption. Then IND-$\mathcal{F}_{\text{raff}}$-RKA security of AIAE implies the n-KDM$[\mathcal{F}_{\text{aff}}]$-CCA security.

In the proof, G_1-G_2 are dedicated to deal with the n-user case; the aim of G_3-G_4 is to eliminate the use of the (mod N) part of $(x_j, y_j)_{j=1}^4$ in ENC; the aim of G_5-G_6 is to use $(x_j, y_j)_{j=1}^4 \bmod N$ to hide the AIAE's base key $(k_1^*, \cdots, k_4^*)$ in ENC, however, DEC may still leak the information about $(x_j, y_j)_{j=1}^4 \bmod N$; the aim of G_7-G_8 is to eliminate the use of $(x_j, y_j)_{j=1}^4 \bmod N$ in DEC; finally, in G_9-G_{10}, the IND-$\mathcal{F}_{\text{raff}}$-RKA security of AIAE is used to prove the n-KDM$[\mathcal{F}_{\text{aff}}]$-CCA security of PKE, since $(k_1^*, \cdots, k_4^*)$ is perfectly hided by $(x_j, y_j)_{j=1}^4 \bmod N$.

– Game G_0: This is the original n-$\mathsf{KDM}[\mathcal{F}_{\text{aff}}]$-$\mathsf{CCA}$ game. Let Win denote the event that $\beta' = \beta$. Then by definition, $\mathsf{Adv}_{\mathsf{PKE},\mathcal{A}}^{kdm\text{-}cca}(\ell) = \left|\Pr_0[\mathsf{Win}] - \frac{1}{2}\right|$.

Denote by $\mathsf{pk}_i = (h_{i,1}, \cdots, h_{i,4})$ and $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \cdots, x_{i,4}, y_{i,4})$ the public and secret keys of the i-th user respectively, $i \in [n]$.

– Game G_1: This game is the same as game G_0, except that, when answering the DEC query $(\langle \mathsf{aux}, \mathsf{aiae.ct}\rangle, i \in [n])$, the challenger outputs $\perp$ if $\langle \mathsf{aux}, \mathsf{aiae.ct}\rangle = \langle \mathsf{aux}_\lambda, \mathsf{aiae.ct}_\lambda\rangle$ for some $\lambda \in [Q_e]$, where $\langle \mathsf{aux}_\lambda, \mathsf{aiae.ct}_\lambda\rangle$ is the challenge ciphertext for the λ-th ENC query (f_λ, i_λ).

Case 1: $(\langle \mathsf{aux}, \mathsf{aiae.ct}\rangle, i) = (\langle \mathsf{aux}_\lambda, \mathsf{aiae.ct}_\lambda\rangle, i_\lambda)$.

DEC will output $\perp$ in game G_0 since $(\langle \mathsf{aux}_\lambda, \mathsf{aiae.ct}_\lambda\rangle, i_\lambda)$ is prohibited.

Case 2: $\langle \mathsf{aux}, \mathsf{aiae.ct}\rangle = \langle \mathsf{aux}_\lambda, \mathsf{aiae.ct}_\lambda\rangle$ but $i \neq i_\lambda$.

We show that in game G_0, DEC will output $\perp$, due to $e_{\lambda,1} u_{\lambda,1}^{x_{i,1}} u_{\lambda,2}^{y_{i,1}} \notin \mathbb{RU}_{N^2}$, with overwhelming probability. Recall that $u_{\lambda,1} = g_1^{r_\lambda}, u_{\lambda,2} = g_2^{r_\lambda}, e_{\lambda,1} = h_{i_\lambda,1}^{r_\lambda} T^{k_{\lambda,1}}$, so

$$e_{\lambda,1} u_{\lambda,1}^{x_{i,1}} u_{\lambda,2}^{y_{i,1}} = h_{i_\lambda,1}^{r_\lambda} T^{k_{\lambda,1}} \cdot (g_1^{r_\lambda})^{x_{i,1}} (g_2^{r_\lambda})^{y_{i,1}} = (h_{i_\lambda,1} h_{i,1}^{-1})^{r_\lambda} T^{k_{\lambda,1}} \bmod N^2,$$

where $h_{i_\lambda,1}$ and $h_{i,1}$ are parts of public key of different users i_λ and i respectively and are uniformly distributed over $\mathbb{SCR}_{N^s}$. So $h_{i_\lambda,1} h_{i,1}^{-1} \neq 1$, hence $e_{\lambda,1} u_{\lambda,1}^{x_{i,1}} u_{\lambda,2}^{y_{i,1}} \notin \mathbb{RU}_{N^2}$, except with probability $2^{-\Omega(\ell)}$.

By a union bound, G_0 and G_1 are identical except with probability $Q_d \cdot 2^{-\Omega(\ell)}$, therefore $\left|\Pr_0[\mathsf{Win}] - \Pr_1[\mathsf{Win}]\right| \leq Q_d \cdot 2^{-\Omega(\ell)}$.

– Game G_2: This game is the same as game G_1, except that, the challenger samples the secret keys $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \cdots, x_{i,4}, y_{i,4})$, $i \in [n]$, in a different way. First, it chooses random $(x_1, y_1, \cdots, x_4, y_4)$ and $(\overline{x}_{i,1}, \overline{y}_{i,1}, \cdots, \overline{x}_{i,4}, \overline{y}_{i,4})$, $i \in [n]$, from $\left[\lfloor N^2/4 \rfloor\right]$, then it computes $(x_{i,1}, y_{i,1}, \cdots, x_{i,4}, y_{i,4}) = (x_1, y_1, \cdots, x_4, y_4) + (\overline{x}_{i,1}, \overline{y}_{i,1}, \cdots, \overline{x}_{i,4}, \overline{y}_{i,4}) \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$.

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

– Game G_3: This game is the same as game G_2, except that, when responding to the adversary's λ-th ($\lambda \in [Q_e]$) Enc query (f_λ, i_λ), instead of using the public keys $\mathsf{pk}_{i_\lambda} = (h_{i_\lambda,1}, \cdots, h_{i_\lambda,4})$, the challenger uses the secret keys $\mathsf{sk}_{i_\lambda} = (x_{i_\lambda,1}, y_{i_\lambda,1}, \cdots, x_{i_\lambda,4}, y_{i_\lambda,4})$ to prepare $(e_{\lambda,1}, \cdots, e_{\lambda,4})$ and $\tilde{e}_\lambda$ as follows:
  - $(e_{\lambda,1}, \cdots, e_{\lambda,4}) := (u_{\lambda,1}^{-x_{i_\lambda,1}} u_{\lambda,2}^{-y_{i_\lambda,1}} T^{k_{\lambda,1}}, \cdots, u_{\lambda,4}^{-x_{i_\lambda,4}} u_{\lambda,5}^{-y_{i_\lambda,4}} T^{k_{\lambda,4}}) \bmod N^2$,
  - $\tilde{e}_\lambda := \tilde{u}_{\lambda,1}^{-x_{i_\lambda,1}} \tilde{u}_{\lambda,2}^{-y_{i_\lambda,1}} \tilde{u}_{\lambda,3}^{-x_{i_\lambda,2}} \tilde{u}_{\lambda,4}^{-y_{i_\lambda,2}} \tilde{u}_{\lambda,5}^{-x_{i_\lambda,3}} \tilde{u}_{\lambda,6}^{-y_{i_\lambda,3}} \tilde{u}_{\lambda,7}^{-x_{i_\lambda,4}} \tilde{u}_{\lambda,8}^{-y_{i_\lambda,4}} T^{m_\beta} \bmod N^s$.

  Observe that for $j \in \{1, 2, 3, 4\}$,

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

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

  Thus G_3 is identical to G_2, and $\Pr_2[\mathsf{Win}] = \Pr_3[\mathsf{Win}]$.

– Game G_4: This game is the same as game G_3, except that, in the case of the challenge bit $\beta = 1$, to answer the λ-th ($\lambda \in [Q_e]$) Enc query (f_λ, i_λ), the challenger does not use $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ to compute $\tilde{e}_\lambda$ any more, and instead, it computes $(\tilde{u}_{\lambda,1}, \cdots, \tilde{u}_{\lambda,8})$ and $\tilde{e}_\lambda$ as follows:
  - $(\tilde{u}_{\lambda,1}, \cdots, \tilde{u}_{\lambda,8}) := (g_1^{\tilde{r}_{\lambda,1}} T^{\sum_{i=1}^n a_{i,1}}, g_2^{\tilde{r}_{\lambda,1}} T^{\sum_{i=1}^n b_{i,1}}, g_2^{\tilde{r}_{\lambda,2}} T^{\sum_{i=1}^n a_{i,2}}, g_3^{\tilde{r}_{\lambda,2}} \cdot T^{\sum_{i=1}^n b_{i,2}}, g_3^{\tilde{r}_{\lambda,3}} T^{\sum_{i=1}^n a_{i,3}}, g_4^{\tilde{r}_{\lambda,3}} T^{\sum_{i=1}^n b_{i,3}}, g_4^{\tilde{r}_{\lambda,4}} T^{\sum_{i=1}^n a_{i,4}}, g_5^{\tilde{r}_{\lambda,4}} T^{\sum_{i=1}^n b_{i,4}})$,
  - $\tilde{e}_\lambda := h_{i_\lambda,1}^{\tilde{r}_{\lambda,1}} \cdots h_{i_\lambda,4}^{\tilde{r}_{\lambda,4}} T^{\sum_{i=1}^n \sum_{j=1}^4 (a_{i,j}(\bar{x}_{i,j} - \bar{x}_{i_\lambda,j}) + b_{i,j}(\bar{y}_{i,j} - \bar{y}_{i_\lambda,j})) + c} \bmod N^s$,

  where $f_\lambda = (\{a_{i,1}, b_{i,1}, \cdots, a_{i,4}, b_{i,4}\}_{i \in [n]}, c) \in \mathcal{F}_{\text{aff}}$.

  Observe that,

$$\begin{aligned} \tilde{e}_\lambda &\overset{\mathsf{G}_4}{=} \prod_{j=1}^4 h_{i_\lambda,j}^{\tilde{r}_{\lambda,j}} \cdot T^{\sum_{i=1}^n \sum_{j=1}^4 (a_{i,j}(\bar{x}_{i,j} - \bar{x}_{i_\lambda,j}) + b_{i,j}(\bar{y}_{i,j} - \bar{y}_{i_\lambda,j})) + c} \\ &= \prod_{j=1}^4 h_{i_\lambda,j}^{\tilde{r}_{\lambda,j}} \cdot T^{\sum_{i=1}^n \sum_{j=1}^4 (a_{i,j}(x_{i,j} - x_{i_\lambda,j}) + b_{i,j}(y_{i,j} - y_{i_\lambda,j})) + c} \\ &= \prod_{j=1}^4 (g_j^{-x_{i_\lambda,j}} g_{j+1}^{-y_{i_\lambda,j}})^{\tilde{r}_{\lambda,j}} \cdot T^{m_1 - \sum_{i=1}^n \sum_{j=1}^4 (a_{i,j} x_{i_\lambda,j} + b_{i,j} y_{i_\lambda,j})} \\ &= \prod_{j=1}^4 \left(g_j^{\tilde{r}_{\lambda,j}} T^{\sum_{i=1}^n a_{i,j}}\right)^{-x_{i_\lambda,j}} \left(g_{j+1}^{\tilde{r}_{\lambda,j}} T^{\sum_{i=1}^n b_{i,j}}\right)^{-y_{i_\lambda,j}} \cdot T^{m_1} \\ &= \tilde{u}_{\lambda,1}^{-x_{i_\lambda,1}} \tilde{u}_{\lambda,2}^{-y_{i_\lambda,1}} \cdots \tilde{u}_{\lambda,7}^{-x_{i_\lambda,4}} \tilde{u}_{\lambda,8}^{-y_{i_\lambda,4}} T^{m_1} \bmod N^s, \end{aligned}$$

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

  Therefore, $\tilde{e}_\lambda$ can be computed from $(\tilde{u}_{\lambda,1}, \cdots, \tilde{u}_{\lambda,8})$ in the same way as in G_3 and G_4. Hence the only difference between G_3 and G_4 is the distribution of $(\tilde{u}_{\lambda,1}, \cdots, \tilde{u}_{\lambda,8})$ themselves. We analyze the difference via the following lemma, and the proof is presented in the full version [HLL16].

Lemma 4. *There exists a PPT adversary $\mathcal{B}_1$ against the IV_5 assumption w.r.t.* GenN *and* $\mathbb{QR}_{N^s}$*, such that* $\left|\Pr_3[\mathsf{Win}] - \Pr_4[\mathsf{Win}]\right| \leq \mathsf{Adv}^{iv_5}_{\mathsf{GenN},\mathcal{B}_1}(\ell)$.

– Game G_5: This game is the same as game G_4, except that, the challenger chooses random $r^* \in [\lfloor N/4 \rfloor]$ and $\alpha_1, \cdots, \alpha_5 \in \mathbb{Z}_N$ beforehand (in Initialize). In addition, to respond to the λ-th ($\lambda \in [Q_e]$) Enc query (f_λ, i_λ), the challenger computes $(u_{\lambda,1}, \cdots, u_{\lambda,5})$ as follows:
  - $(u_{\lambda,1}, \cdots, u_{\lambda,5}) := ((g_1^{r^*} T^{\alpha_1})^{r_\lambda}, \cdots, (g_5^{r^*} T^{\alpha_5})^{r_\lambda}) \bmod N^2$.

The only difference between G_4 and G_5 is the distribution of $(u_{\lambda,1},\cdots,u_{\lambda,5})$. In game G_4, it equals $(g_1^{r_\lambda},\cdots,g_5^{r_\lambda}) \bmod N^2$, while in game G_5, it equals $((g_1^{r^*}T^{\alpha_1})^{r_\lambda},\cdots,(g_5^{r^*}T^{\alpha_5})^{r_\lambda}) \bmod N^2$. Similar to the previous lemma, it is straightforward to construct a PPT adversary to solve IV_5 problem by employing the power of adversary $\mathcal{A}$. Thus $\big|\Pr_4[\mathsf{Win}]-\Pr_5[\mathsf{Win}]\big| \le \mathsf{Adv}_{\mathsf{GenN}}^{iv_5}(\ell)$.

– Game G_6: This game is the same as game G_5, except that, the challenger chooses a random tuple $\mathsf{k}^* = (k_1^*,k_2^*,k_3^*,k_4^*)$ beforehand (in Initialize). In addition, to respond to the λ-th ($\lambda \in [Q_e]$) Enc query (f_λ, i_λ), the challenger uses a different way to generate $\mathsf{k}_\lambda = (k_{\lambda,1},k_{\lambda,2},k_{\lambda,3},k_{\lambda,4})$ and $(e_{\lambda,1},\cdots,e_{\lambda,4})$:
  - pick $\mathsf{s}_\lambda = (s_{\lambda,1},s_{\lambda,2},s_{\lambda,3},s_{\lambda,4}) \leftarrow_\$ \mathbb{Z}_N^4$ and $r_\lambda \leftarrow_\$ \big[\lfloor N/4\rfloor\big]$ uniformly, and compute $\mathsf{k}_\lambda = (k_{\lambda,1},k_{\lambda,2},k_{\lambda,3},k_{\lambda,4}) := (r_\lambda k_1^* + s_{\lambda,1},\cdots,r_\lambda k_4^* + s_{\lambda,4})$.
  - $(e_{\lambda,1},\cdots,e_{\lambda,4}) :=$
    $(h_{i_\lambda,1}^{r^* r_\lambda}T^{r_\lambda(k_1^*-\alpha_1 x_{i_\lambda,1}-\alpha_2 y_{i_\lambda,1})+s_{\lambda,1}},\cdots,h_{i_\lambda,4}^{r^* r_\lambda}T^{r_\lambda(k_4^*-\alpha_4 x_{i_\lambda,4}-\alpha_5 y_{i_\lambda,4})+s_{\lambda,4}})$.

  Clearly k_λ is uniformly distributed over $\mathbb{Z}_N^4$, as in game G_5. At the same time, observe that for $j \in \{1,2,3,4\}$,

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

  Thus G_6 is identical to G_5, and $\Pr_5[\mathsf{Win}] = \Pr_6[\mathsf{Win}]$.

– Game G_7: This game is the same as game G_6, except for a modification to answering the Dec queries $(\langle\mathsf{aux},\mathsf{aiae.ct}\rangle, i \in [n])$. The challenger uses the i-th user's secret key $\mathsf{sk}_i = (x_{i,1},y_{i,1},\cdots,x_{i,4},y_{i,4})$ together with $\phi(N)$ to compute the decryption of ciphertext $\langle\mathsf{aux},\mathsf{aiae.ct}\rangle$, where $\mathsf{aux} = (u_1,\cdots,u_5,e_1,\cdots,e_4)$. More precisely, it computes $\mathsf{k} = (k_1,\cdots,k_4)$ and m as follows:
  - $(\alpha'_1,\cdots,\alpha'_5) := \big(\mathrm{dlog}_T(u_1^{\phi(N)})/\phi(N),\cdots,\mathrm{dlog}_T(u_5^{\phi(N)})/\phi(N)\big) \bmod N$,
    $(\gamma'_1,\cdots,\gamma'_4) := \big(\mathrm{dlog}_T(e_1^{\phi(N)})/\phi(N),\cdots,\mathrm{dlog}_T(e_4^{\phi(N)})/\phi(N)\big) \bmod N$,
    $\mathsf{k} = (k_1,\cdots,k_4) := (\alpha'_1 x_{i,1}+\alpha'_2 y_{i,1}+\gamma'_1,\cdots,\alpha'_4 x_{i,4}+\alpha'_5 y_{i,4}+\gamma'_4) \bmod N$,
  - $\mathcal{E}.\mathsf{ct} = (\tilde u_1,\cdots,\tilde u_8,\tilde e,t)/\perp \leftarrow \mathsf{AIAE.Dec}\big(\mathsf{k},\mathsf{aiae.ct},\mathsf{aux}\big)$,
  - $(\tilde\alpha_1,\cdots,\tilde\alpha_8) := \big(\mathrm{dlog}_T(\tilde u_1^{\phi(N)})/\phi(N),\cdots,\mathrm{dlog}_T(\tilde u_8^{\phi(N)})/\phi(N)\big) \bmod N^{s-1}$,
    $\tilde\gamma := \mathrm{dlog}_T(\tilde e^{\phi(N)})/\phi(N) \bmod N^{s-1}$, and $m := \tilde\alpha_1 x_{i,1}+\tilde\alpha_2 y_{i,1}+\tilde\alpha_3 x_{i,2}+\tilde\alpha_4 y_{i,2}+\tilde\alpha_5 x_{i,3}+\tilde\alpha_6 y_{i,3}+\tilde\alpha_7 x_{i,4}+\tilde\alpha_8 y_{i,4}+\tilde\gamma \bmod N^{s-1}$.

  According to Eq. (1), for $j \in \{1,2,3,4\}$, we have that

$$\begin{aligned} k_j &\overset{\mathsf{G}_6}{=} \mathrm{dlog}_T(e_j u_j^{x_{i,j}} u_{j+1}^{y_{i,j}}) = \mathrm{dlog}_T\big((e_j u_j^{x_{i,j}} u_{j+1}^{y_{i,j}})^{\phi(N)}\big)/\phi(N) \bmod N \\ &= \mathrm{dlog}_T(u_j^{\phi(N)\cdot x_{i,j}})/\phi(N) + \mathrm{dlog}_T(u_{j+1}^{\phi(N)\cdot y_{i,j}})/\phi(N) + \mathrm{dlog}_T(e_j^{\phi(N)})/\phi(N) \\ &\overset{\mathsf{G}_7}{=} \underbrace{\mathrm{dlog}_T(u_j^{\phi(N)})/\phi(N)}_{\alpha'_j}\cdot x_{i,j} + \underbrace{\mathrm{dlog}_T(u_{j+1}^{\phi(N)})/\phi(N)}_{\alpha'_{j+1}}\cdot y_{i,j} + \underbrace{\mathrm{dlog}_T(e_j^{\phi(N)})/\phi(N)}_{\gamma'_j}, \end{aligned}$$

$$\begin{aligned} m &\overset{\mathsf{G}_6}{=} \mathrm{dlog}_T(\tilde e \tilde u_1^{x_{i,1}} \tilde u_2^{y_{i,1}} \tilde u_3^{x_{i,2}} \tilde u_4^{y_{i,2}} \tilde u_5^{x_{i,3}} \tilde u_6^{y_{i,3}} \tilde u_7^{x_{i,4}} \tilde u_8^{y_{i,4}}) \bmod N^{s-1} \\ &\overset{\mathsf{G}_7}{=} \underbrace{\mathrm{dlog}_T(\tilde u_1^{\phi(N)})/\phi(N)}_{\tilde\alpha_1}\cdot x_{i,1} + \cdots + \underbrace{\mathrm{dlog}_T(\tilde u_8^{\phi(N)})/\phi(N)}_{\tilde\alpha_8}\cdot y_{i,4} + \underbrace{\mathrm{dlog}_T(\tilde e^{\phi(N)})/\phi(N)}_{\tilde\gamma}. \end{aligned}$$

These changes are conceptual. So G_7 is identical to G_6, $\Pr_6[\mathsf{Win}] = \Pr_7[\mathsf{Win}]$.

– Game G_8: This game is the same as game G_7, except that, the challenger adds an additional rejection rule when answering Dec queries as follows:
  - if $\alpha'_1 \neq 0 \vee \cdots \vee \alpha'_5 \neq 0 \ \vee \ \tilde{\alpha}_1 \neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0$, return $\perp$.

  That is, the challenger will not output m in Dec unless $\alpha'_1 = \cdots = \alpha'_5 = 0$ and $\tilde{\alpha}_1 = \cdots = \tilde{\alpha}_8 = 0$ holds. Thus the values of $(x_{i,j}, y_{i,j})_{j=1}^4 \bmod N$, in particular $(x_j, y_j)_{j=1}^4 \bmod N$, are not used any more in Dec.

  Let Bad denote the event that $\mathcal{A}$ makes a Dec query $\big(\langle \mathsf{aux}, \mathsf{aiae.ct}\rangle, i \in [n]\big)$, such that

$$e_1 u_1^{x_{i,1}} u_2^{y_{i,1}}, \cdots, e_4 u_4^{x_{i,4}} u_5^{y_{i,4}} \in \mathbb{RU}_{N^2} \ \wedge \ \mathsf{AIAE.Dec}\big(\mathsf{k}, \mathsf{aiae.ct}, \mathsf{aux}\big) \neq \perp \quad (2)$$
$$\wedge \ \tilde{e}\tilde{u}_1^{x_{i,1}} \tilde{u}_2^{y_{i,1}} \tilde{u}_3^{x_{i,2}} \tilde{u}_4^{y_{i,2}} \tilde{u}_5^{x_{i,3}} \tilde{u}_6^{y_{i,3}} \tilde{u}_7^{x_{i,4}} \tilde{u}_8^{y_{i,4}} \in \mathbb{RU}_{N^s} \ \wedge \ t = g_1^m \bmod N \quad (3)$$
$$\wedge \ \big(\alpha'_1 \neq 0 \vee \cdots \vee \alpha'_5 \neq 0 \ \vee \ \tilde{\alpha}_1 \neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0\big).$$

  Clearly, games G_7 and G_8 are the same until Bad happens. Therefore, we have that $\big|\Pr_7[\mathsf{Win}] - \Pr_8[\mathsf{Win}]\big| \leq \Pr_8[\mathsf{Bad}]$.

  To prove that G_7 and G_8 are indistinguishable, we have to show that $\Pr_8[\mathsf{Bad}]$ is negligible. This is not an easy task, and we further divide Bad to two disjoint sub-events:
  * Bad' denotes the event that $\mathcal{A}$ makes a Dec query such that
$$\text{Conditions (2), (3) hold} \ \wedge \ \big(\alpha'_1 \neq 0 \vee \cdots \vee \alpha'_5 \neq 0\big).$$
  * $\widetilde{\mathsf{Bad}}$ denotes the event that $\mathcal{A}$ makes a Dec query such that
$$\text{Con. (2), (3) hold} \ \wedge \ \big(\alpha'_1 = \cdots = \alpha'_5 = 0\big) \ \wedge \ \big(\tilde{\alpha}_1 \neq 0 \vee \cdots \vee \tilde{\alpha}_8 \neq 0\big).$$

  Then $\Pr_8[\mathsf{Bad}] \leq \Pr_8[\mathsf{Bad}'] + \Pr_8[\widetilde{\mathsf{Bad}}]$. We give an upper bound for $\Pr_8[\mathsf{Bad}']$ via the following lemma. See the full version [HLL16] for the proof. The analysis of $\Pr_8[\widetilde{\mathsf{Bad}}]$ is deferred to subsequent games.

  Lemma 5. $\Pr_8[\mathsf{Bad}'] \leq 2Q_d \cdot \mathsf{Adv}_{\mathsf{AIAE}}^{\textit{weak-int-rka}}(\ell) + Q_d \cdot 2^{-\Omega(\ell)}$.

– Game G_9: This game is the same as game G_8, except that, the challenger chooses another random tuple $\bar{\mathsf{k}}^* = (\bar{k}_1^*, \bar{k}_2^*, \bar{k}_3^*, \bar{k}_4^*)$ besides $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ in Initialize. In addition, to answer the λ-th ($\lambda \in [Q_e]$) Enc query (f_λ, i_λ), the challenger uses a different key for AIAE to compute $\mathsf{aiae.ct}_\lambda$:
  - set $\bar{\mathsf{k}}_\lambda = (\bar{k}_{\lambda,1}, \bar{k}_{\lambda,2}, \bar{k}_{\lambda,3}, \bar{k}_{\lambda,4}) := (r_\lambda \bar{k}_1^* + s_{\lambda,1}, \cdots, r_\lambda \bar{k}_4^* + s_{\lambda,4})$;
  - invoke $\mathsf{aiae.ct}_\lambda \leftarrow_\$ \mathsf{AIAE.Enc}\big(\bar{\mathsf{k}}_\lambda, \mathcal{E}.\mathsf{ct}_\lambda, \mathsf{aux}_\lambda\big)$.

  But the challenger still uses $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ to compute $(e_{\lambda,1}, \cdots, e_{\lambda,4})$.

  In game G_8, the only place that needs the value of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ is the computation of $(e_{\lambda,1}, \cdots, e_{\lambda,4})$ in Enc. More precisely, for $j \in \{1, 2, 3, 4\}$,

$$e_{\lambda,j} = h_{i_\lambda,j}^{r^* r_\lambda} T^{r_\lambda \cdot (k_j^* - \alpha_j x_{i_\lambda,j} - \alpha_{j+1} y_{i_\lambda,j}) + s_{\lambda,j}} \bmod N^2$$
$$= h_{i_\lambda,j}^{r^* r_\lambda} T^{r_\lambda \cdot (k_j^* - \alpha_j x_j - \alpha_{j+1} y_j - \alpha_j \bar{x}_{i_\lambda,j} - \alpha_{j+1} \bar{y}_{i_\lambda,j}) + s_{\lambda,j}} \bmod N^2.$$

  We stress that the computation of $t_\lambda = g_1^{m_\beta} \bmod N$ in Enc only uses the values of $(x_1, y_1, \cdots, x_4, y_4) \bmod \phi(N)/4$, since the order of $g_1 \in \mathbb{SCR}_{N^s}$ is $\phi(N)/4$. We also note that neither $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ nor $(x_j, y_j)_{j=1}^4 \bmod N$ is involved in DEC since DEC rejects the ciphertext unless $\alpha_1' = \cdots = \alpha_5' = 0$ and $\tilde{\alpha}_1 = \cdots = \tilde{\alpha}_8 = 0$. As a result, $\mathsf{k}^* = (k_1^*, k_2^*, k_3^*, k_4^*)$ is totally hidden by the entropy of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ and is uniformly random to $\mathcal{A}$.

Thus the challenger can use an independent $\overline{\mathsf{k}}^* = (\bar{k}_1^*, \cdots, \bar{k}_4^*)$ to compute $\overline{\mathsf{k}}_\lambda$, and use $\overline{\mathsf{k}}_\lambda$ to do the encryption of the AIAE scheme in ENC, as in G_9.

Then games G_8 and G_9 are identically distributed from the point of view of $\mathcal{A}$, thus we have $\Pr_8[\mathsf{Win}] = \Pr_9[\mathsf{Win}]$ and $\Pr_8[\widetilde{\mathsf{Bad}}] = \Pr_9[\widetilde{\mathsf{Bad}}]$.

– Game G_{10}: This game is the same as game G_9, except that, to answer the λ-th ($\lambda \in [Q_e]$) ENC query (f_λ, i_λ), the challenger computes $\mathsf{aiae.ct}_\lambda$ as follows:
  - invoke $\mathsf{aiae.ct}_\lambda \leftarrow_\$ \mathsf{AIAE.Enc}\big(\overline{\mathsf{k}}_\lambda, 0^{\ell_\mathcal{M}}, \mathsf{aux}_\lambda\big)$.

  That is, the challenger computes the AIAE encryption of a constant $0^{\ell_\mathcal{M}}$ instead of $\mathcal{E}.\mathsf{ct}_\lambda$ in ENC. Note that in games G_9 and G_{10}, the key $\overline{\mathsf{k}}^* = (\bar{k}_1^*, \bar{k}_2^*, \bar{k}_3^*, \bar{k}_4^*)$ is used only in the computation of the AIAE encryption, where it uses $\overline{\mathsf{k}}_\lambda = r_\lambda \cdot \overline{\mathsf{k}}^* + \mathsf{s}_\lambda$, $\mathsf{s}_\lambda = (s_{\lambda,1}, \cdots, s_{\lambda,4})$, as the encryption key. The difference between G_9 and G_{10} can be reduced to the IND-$\mathcal{F}_{\text{raff}}$-RKA security of the AIAE scheme directly. Thus we have that both $\big|\Pr_9[\mathsf{Win}] - \Pr_{10}[\mathsf{Win}]\big|$, $\big|\Pr_9[\widetilde{\mathsf{Bad}}] - \Pr_{10}[\widetilde{\mathsf{Bad}}]\big| \leq \mathsf{Adv}_{\mathsf{AIAE}}^{ind\text{-}rka}(\ell)$.

  Now in G_{10}, the challenger computes the AIAE encryption of a constant $0^{\ell_\mathcal{M}}$ in ENC, thus the challenge bit β is completely hidden. Then $\Pr_{10}[\mathsf{Win}] = \frac{1}{2}$.

  We give an upper bound for $\Pr_{10}[\widetilde{\mathsf{Bad}}]$ via the following lemma, and present its proof in the full version [HLL16].

  Lemma 6. $\Pr_{10}[\widetilde{\mathsf{Bad}}] \leq (Q_d + 1) \cdot 2^{-\Omega(\ell)} + \mathsf{Adv}_{\mathsf{GenN}}^{dl}(\ell)$.

Taking all things together, the n-KDM[$\mathcal{F}_{\text{aff}}$]-CCA security of PKE follows. ▌

6 PKE with n-KDM[$\mathcal{F}_{\text{poly}}^d$]-CCA Security

6.1 The Basic Idea

We consider how to construct a PKE which is n-KDM-CCA secure w.r.t. the set of polynomial functions of bounded degree d, denoted by $\mathcal{F}_{\text{poly}}^d$, where d can be polynomial in security parameter ℓ. We will consider adversaries submitting f in the format of *Modular Arithmetic Circuit* (MAC) [MTY11], i.e., a polynomial-size circuit which computes f. In particular, we do not require a prior bound on the size of circuits, but only require a prior bound d on the degree of the polynomials. Our construction still follows the approach in Fig. 1. In fact, our n-KDM[$\mathcal{F}_{\text{poly}}^d$]-CCA secure PKE shares the same building blocks KEM and AIAE with the previous PKE in Fig. 7 which has n-KDM[$\mathcal{F}_{\text{aff}}$]-CCA security. What we should do is to design a new building block $\mathcal{E}$, which can function as an entropy filter for polynomial functions. Our new $\mathcal{E}$ still share the same secret/public key pair with KEM. Hence for $i \in [n]$, we have $\mathsf{sk}_i = (x_{i,1}, y_{i,1}, \cdots, x_{i,4}, y_{i,4})$ and $\mathsf{pk}_i = (h_{i,1}, \cdots, h_{i,4})$ with $h_{i,1} = g_1^{-x_{i,1}} g_2^{-y_{i,1}}$, $\cdots$, $h_{i,4} = g_4^{-x_{i,4}} g_5^{-y_{i,4}} \bmod N^s$.

6.2 Reducing Polynomials of $8n$ Variables to Polynomials of 8 Variables

How to Reduce $8n$-Variable Polynomial f_λ in $\textsc{Enc}(f_\lambda, i_\lambda \in [n])$**.** In the n-$\mathsf{KDM}[\mathcal{F}^d_{\text{poly}}]$-$\mathsf{CCA}$ game, the adversary will submit $(f_\lambda, i_\lambda \in [n])$ to $\textsc{Enc}$ as its λ-th KDM encryption query. Here f_λ is a degree-d polynomial $f_\lambda\big((x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}\big)$ of the n secret keys, which has $8n$ variables. Note that f_λ will contain at most $\binom{8n+d}{8n} = \Theta(d^{8n})$ monomials, which is exponentially large.

To reduce the number of monomials, we can always change the polynomial $f_\lambda\big((x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}\big)$ of $8n$ variables to a polynomial $f'_\lambda\big((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}\big)$ of 8 variables as follows. Then f'_λ will contain at most $\binom{8+d}{8} = \Theta(d^8)$ monomials, which is polynomial in ℓ.

In $\textsc{Initialize}$, the secret keys can be generated with $x_{i,j} := x_j + \bar{x}_{i,j}$ and $y_{i,j} := y_j + \bar{y}_{i,j} \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$ and $j \in [4]$. Then with the values of $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n],j\in[4]}$, we can represent $(x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}$ as shifts of $(x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}$:

$$x_{i,j} = x_{i_\lambda,j} + \bar{x}_{i,j} - \bar{x}_{i_\lambda,j}, \qquad y_{i,j} = y_{i_\lambda,j} + \bar{y}_{i,j} - \bar{y}_{i_\lambda,j},$$

and reduce the polynomial f_λ in $8n$ variables $(x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}$ to a polynomial f'_λ in 8 variables $(x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}$:

$$f_\lambda\big((x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}\big) = f_\lambda\big((\underbrace{x_{i_\lambda,j} + \bar{x}_{i,j} - \bar{x}_{i_\lambda,j}}_{x_{i,j}}, \underbrace{y_{i_\lambda,j} + \bar{y}_{i,j} - \bar{y}_{i_\lambda,j}}_{y_{i,j}})_{i\in[n],j\in[4]}\big)$$
$$= f'_\lambda\big((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}\big) = \sum_{0\le c_1+\cdots+c_8\le d} a_{(c_1,\cdots,c_8)} \cdot x^{c_1}_{i_\lambda,1} y^{c_2}_{i_\lambda,1} x^{c_3}_{i_\lambda,2} y^{c_4}_{i_\lambda,2} x^{c_5}_{i_\lambda,3} y^{c_6}_{i_\lambda,3} x^{c_7}_{i_\lambda,4} y^{c_8}_{i_\lambda,4}.$$

The resulting polynomial f'_λ is also of degree at most d, and the coefficients $a_{(c_1,\cdots,c_8)}$ are determined by $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n],j\in[4]}$ completely.

How to Determine Coefficients $a_{(c_1,\cdots,c_8)}$ for f'_λ Efficiently with Only $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n],j\in[4]}$. Repeat choosing values of $(x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}$ randomly, feeding MAC (which functions as f_λ) with input of $(x_{i_\lambda,j} + \bar{x}_{i,j} - \bar{x}_{i_\lambda,j}, y_{i_\lambda,j} + \bar{y}_{i,j} - \bar{y}_{i_\lambda,j})_{i\in[n],j\in[4]}$, where $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n],j\in[4]}$ always takes the values chosen in $\textsc{Initialize}$, and recording the output of MAC. After about $\binom{8+d}{8} = \Theta(d^8)$ times, we can extract all $a_{(c_1,\cdots,c_8)}$ by simply solving a system of linear equations (with $a_{(c_1,\cdots,c_8)}$ unknowns):

$$f_\lambda\big((x_{i_\lambda,j} + \bar{x}_{i,j} - \bar{x}_{i_\lambda,j}, y_{i_\lambda,j} + \bar{y}_{i,j} - \bar{y}_{i_\lambda,j})_{i\in[n],j\in[4]}\big)$$
$$= \sum_{0\le c_1+\cdots+c_8\le d} a_{(c_1,\cdots,c_8)} \cdot x^{c_1}_{i_\lambda,1} y^{c_2}_{i_\lambda,1} x^{c_3}_{i_\lambda,2} y^{c_4}_{i_\lambda,2} x^{c_5}_{i_\lambda,3} y^{c_6}_{i_\lambda,3} x^{c_7}_{i_\lambda,4} y^{c_8}_{i_\lambda,4}.$$

This can be done in time polynomial in ℓ.

6.3 How to Design $\mathcal{E}$: A Warmup

Let us first consider a simple case: design $\mathcal{E}$ w.r.t. a specific type of monomials

$$f'_\lambda\big((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}\big) = a \cdot x_{i_\lambda,1} y_{i_\lambda,1} x_{i_\lambda,2} y_{i_\lambda,2} x_{i_\lambda,3} y_{i_\lambda,3} x_{i_\lambda,4} y_{i_\lambda,4}.$$

We describe the encryption and decryption algorithms $\mathcal{E}.\mathsf{Enc}$, $\mathcal{E}.\mathsf{Dec}$ in Fig. 8.

```
$\mathcal{E}.\mathsf{ct} \leftarrow_\$ \mathcal{E}.\mathsf{Enc}\big(\mathsf{pk} = (h_1, h_2, h_3, h_4), m\big)$:
For $l \in [0, 8]$,
    $\tilde{r}_{l,1}, \tilde{r}_{l,2}, \tilde{r}_{l,3}, \tilde{r}_{l,4} \leftarrow_\$ \big[\lfloor \frac{N}{4} \rfloor\big]$.
    $(\tilde{u}_{l,1}, \cdots, \tilde{u}_{l,8}) := (g_1^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,4}}, g_5^{\tilde{r}_{l,4}})$.
    $\tilde{v}_l := h_1^{\tilde{r}_{l,1}} h_2^{\tilde{r}_{l,2}} h_3^{\tilde{r}_{l,3}} h_4^{\tilde{r}_{l,4}}$.
$\mathsf{table} := \begin{array}{|c|c|c|c|}\hline \tilde{u}_{0,1} & \tilde{u}_{0,2} & \cdots & \tilde{u}_{0,8} \\ \hline \tilde{u}_{1,1} \cdot \tilde{v}_0 & \tilde{u}_{1,2} & \cdots & \tilde{u}_{1,8} \\ \hline \tilde{u}_{2,1} & \tilde{u}_{2,2} \cdot \tilde{v}_1 & \cdots & \tilde{u}_{2,8} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \tilde{u}_{8,1} & \tilde{u}_{8,2} & \cdots & \tilde{u}_{8,8} \cdot \tilde{v}_7 \\ \hline \end{array}$
$\tilde{e} := \tilde{v}_8 \cdot T^m \bmod N^s$.
$t := g_1^m \bmod N \in \mathbb{Z}_N$.
Return $\mathcal{E}.\mathsf{ct} := (\mathsf{table}, \tilde{e}, t)$.
```

```
$m/\perp \leftarrow \mathcal{E}.\mathsf{Dec}\big(\mathsf{sk} = (x_1, y_1, \cdots, x_4, y_4), \mathcal{E}.\mathsf{ct}\big)$:
Parse $\mathcal{E}.\mathsf{ct} = (\mathsf{table}, \tilde{e}, t)$.
Parse $\mathsf{table} = \begin{array}{|c|c|c|c|}\hline \hat{u}_{0,1} & \hat{u}_{0,2} & \cdots & \hat{u}_{0,8} \\ \hline \hat{u}_{1,1} & \hat{u}_{1,2} & \cdots & \hat{u}_{1,8} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \hat{u}_{8,1} & \hat{u}_{8,2} & \cdots & \hat{u}_{8,8} \\ \hline \end{array}$
$\hat{v}_0 := \hat{u}_{0,1}^{-x_1} \hat{u}_{0,2}^{-y_1} \hat{u}_{0,3}^{-x_2} \hat{u}_{0,4}^{-y_2} \cdots \hat{u}_{0,7}^{-x_4} \hat{u}_{0,8}^{-y_4}$.
$\hat{v}_1 := (\hat{u}_{1,1}/\hat{v}_0)^{-x_1} \hat{u}_{1,2}^{-y_1} \hat{u}_{1,3}^{-x_2} \hat{u}_{1,4}^{-y_2} \cdots \hat{u}_{1,7}^{-x_4} \hat{u}_{1,8}^{-y_4}$.
$\hat{v}_2 := \hat{u}_{2,1}^{-x_1} (\hat{u}_{2,2}/\hat{v}_1)^{-y_1} \hat{u}_{2,3}^{-x_2} \hat{u}_{2,4}^{-y_2} \cdots \hat{u}_{2,7}^{-x_4} \hat{u}_{2,8}^{-y_4}$.
$\vdots$
$\hat{v}_8 := \hat{u}_{8,1}^{-x_1} \hat{u}_{8,2}^{-y_1} \hat{u}_{8,3}^{-x_2} \hat{u}_{8,4}^{-y_2} \cdots \hat{u}_{8,7}^{-x_4} (\hat{u}_{8,8}/\hat{v}_7)^{-y_4}$.
If $\tilde{e}/\hat{v}_8 \in \mathbb{RU}_{N^s}$, $m := \mathrm{dlog}_T(\tilde{e}/\hat{v}_8) \bmod N^{s-1}$.
    If $t = g_1^m \bmod N$, Return $m$.
Otherwise, Return $\perp$.
```

Fig. 8. $\mathcal{E}$ designed for specific monomials $a \cdot x_{i_\lambda,1} y_{i_\lambda,1} x_{i_\lambda,2} y_{i_\lambda,2} x_{i_\lambda,3} y_{i_\lambda,3} x_{i_\lambda,4} y_{i_\lambda,4}$.

Security proof. We can prove KDM-CCA security w.r.t. the specific type of monomials, i.e., $a \cdot x_{i_\lambda,1} y_{i_\lambda,1}\ x_{i_\lambda,2} y_{i_\lambda,2} x_{i_\lambda,3} y_{i_\lambda,3} x_{i_\lambda,4} y_{i_\lambda,4}$, in a similar way as the proof of Theorem 2. The only difference lies in games G_3-G_4, which are related to $\mathcal{E}$. We replace G_3-G_4 with the following three steps (Step 1 - Step 3). More precisely, we change the $\mathcal{E}.\mathsf{Enc}$ part of ENC so that it can reserve the entropy of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$, behaving like an entropy filter w.r.t. this specific kind of monomials.

Suppose that the adversary submits $(f_\lambda, i_\lambda \in [n])$ to ENC. Our aim is to reserve the entropy of $(x_j, y_j)_{j=1}^4 \bmod N$ from $\mathcal{E}.\mathsf{Enc}\big(\mathsf{pk}_{i_\lambda}, f_\lambda\big((x_{i,j}, y_{i,j})_{i \in [n], j \in [4]}\big)\big)$.

Step 0: In INITIALIZE, the secret keys are generated with $x_{i,j} := x_j + \bar{x}_{i,j}$ and $y_{i,j} := y_j + \bar{y}_{i,j} \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$, $j \in [4]$. This is the same as G_2 in the proof of Theorem 2.

Step 1: Use $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i \in [n], j \in [4]}$ to re-explain $(f_\lambda, i_\lambda \in [n])$ as $(f'_\lambda, i_\lambda \in [n])$, and determine the coefficient a of the monomial

$$f'_\lambda\big((x_{i_\lambda,j}, y_{i_\lambda,j})_{j \in [4]}\big) = a \cdot x_{i_\lambda,1} y_{i_\lambda,1} x_{i_\lambda,2} y_{i_\lambda,2} x_{i_\lambda,3} y_{i_\lambda,3} x_{i_\lambda,4} y_{i_\lambda,4}.$$

Step 2: Use secret key $\mathsf{sk}_{i_\lambda} = (x_{i_\lambda,j}, y_{i_\lambda,j})_{j \in [4]}$ (together with public key $\mathsf{pk}_{i_\lambda} = (h_{i_\lambda,j})_{j \in [4]}$) to implement $\mathcal{E}.\mathsf{Enc}$ (This corresponds to G_3 in the proof of Theorem 2).

- Setup table, just like $\mathcal{E}.\mathsf{Enc}$.
- Compute $\hat{v}_0, \cdots, \hat{v}_8$ from table, just like $\mathcal{E}.\mathsf{Dec}$.
- Use $\hat{v}_8$ instead of $\tilde{v}_8$ to compute $\tilde{e}$ with $\tilde{e} := \hat{v}_8 \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j \in [4]})} \bmod N^s$, and $t := g_1^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j \in [4]})} \bmod N$.

It is easy to check that $\hat{v}_0, \cdots, \hat{v}_8$ computed from table (via $\mathcal{E}.\mathsf{Dec}$) are identical to $\tilde{v}_0, \cdots, \tilde{v}_8$ that are used to generate table (via $\mathcal{E}.\mathsf{Enc}$). Thus this change is conceptual.

Step 3: This corresponds to G_4 in the proof of Theorem 2.

- table is set up in a similar way as in $\mathcal{E}.\mathsf{Enc}$, but with the following difference. The item of row 1 and column 1 in table now is computed as $\hat{u}_{1,1} = (\tilde{u}_{1,1}T^a) \cdot \tilde{v}_0$ instead of $\hat{u}_{1,1} = \tilde{u}_{1,1} \cdot \tilde{v}_0$. This change is computationally indistinguishable, due to the IV$_5$ assumption. (We refer to a detailed analysis in the full version [HLL16].)
- Compute $\hat{v}_0, \cdots, \hat{v}_8$ from table, just like $\mathcal{E}.\mathsf{Dec}$.
- $\tilde{e} := \hat{v}_8 \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})} \bmod N^s$, and $t := g_1^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})} \bmod N$.

It is easy to check that $\hat{v}_0 = \tilde{v}_0, \hat{v}_1 = \tilde{v}_1 \cdot T^{-ax_{i_\lambda,1}}, \hat{v}_2 = \tilde{v}_2 \cdot T^{-ax_{i_\lambda,1}y_{i_\lambda,1}}$, $\cdots$, $\hat{v}_8 = \tilde{v}_8 \cdot T^{-ax_{i_\lambda,1}y_{i_\lambda,1}\cdots x_{i_\lambda,4}y_{i_\lambda,4}} = \tilde{v}_8 \cdot T^{-f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})}$, thus $\tilde{e} = \hat{v}_8 \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})} = \tilde{v}_8$. Therefore we can also implement Step 3 equivalently as follows.

Step 3 (Equivalent Form):

- table is set up in a similar way as in $\mathcal{E}.\mathsf{Enc}$, but with the following difference. The item of row 1 and column 1 in table is computed as $\hat{u}_{1,1} = (\tilde{u}_{1,1}T^a) \cdot \tilde{v}_0$ instead of $\hat{u}_{1,1} = \tilde{u}_{1,1} \cdot \tilde{v}_0$.
- $\tilde{e} := \tilde{v}_8 \bmod N^s$, and $t := g_1^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}) \bmod \phi(N)/4} \bmod N$.

In this step, $\mathcal{E}.\mathsf{Enc}$ does not use $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ at all (only uses $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n], j\in[4]}$ and $(x_1, y_1, \cdots, x_4, y_4) \bmod \phi(N)/4$).

Consequently, through the computationally indistinguishable change, the entropy of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ is reserved by the $\mathcal{E}.\mathsf{Enc}$ part of Enc.

Similarly, Dec can be changed to do decryptions without $(x_j, y_j)_{j=1}^4 \bmod N$. This can be done with $\phi(N)$ and the (mod $\phi(N)/4$) part of secret key. (This corresponds to G_7-G_8 in the proof of Theorem 2). Use $\phi(N)$ to make sure that all items in table of $\mathcal{E}.\mathsf{ct}$ belong to $\mathbb{SCR}_{N^s}$. If not, reject immediately. As a result, Dec does not leak any information of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$. This change is computationally indistinguishable, just like the analysis of $\Pr[\mathsf{Bad}]$ as in the proof of Theorem 2.

6.4 The General $\mathcal{E}$ Designed for $\mathcal{F}^d_{\text{poly}}$

The previous subsection showed how to design $\mathcal{E}$ for a specific type of monomials. A general f'_λ of degree d contains at most $\binom{8+d}{8} = \Theta(d^8)$ monomials. To design a general $\mathcal{E}$ for $\mathcal{F}^d_{\text{poly}}$, we have to consider all possible types of monomials. For each type of non-constant monomial, we create a table and each table is associated with a $\tilde{v}$, which is called a *title*, and those $\tilde{v}$'s are used to hide message in $\tilde{e}$. We describe $\mathcal{E}.\mathsf{Enc}$ and $\mathcal{E}.\mathsf{Dec}$ in Fig. 9.

There are totally $\binom{8+d}{8} - 1$ types of non-constant monomials of degree at most d if we neglect the coefficients. Each type of non-constant monomial $x^{c_1}_{i_\lambda,1}y^{c_2}_{i_\lambda,1}x^{c_3}_{i_\lambda,2}$ $y^{c_4}_{i_\lambda,2}x^{c_5}_{i_\lambda,3}y^{c_6}_{i_\lambda,3}x^{c_7}_{i_\lambda,4}y^{c_8}_{i_\lambda,4}$ is associated with a tuple $\mathsf{c} = (c_1, \cdots c_8)$, which determines degrees of each variable. Denote by $\mathcal{S}$ the set containing all such tuples, i.e., $\mathcal{S} := \{\mathsf{c} = (c_1, \cdots c_8) \mid 1 \le c_1 + \cdots + c_8 \le d\}$.

```
$\mathcal{E}.\mathsf{ct} \leftarrow_\$ \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$:

For each $\mathsf{c} = (c_1, \cdots, c_8) \in \mathcal{S}$
    $(\mathsf{table}^{(\mathsf{c})}, \tilde{v}^{(\mathsf{c})}) \leftarrow_\$ \mathsf{TableGen}(\mathsf{pk}, \mathsf{c})$.

$\tilde{e} := \prod_{\mathsf{c} \in \mathcal{S}} \tilde{v}^{(\mathsf{c})} \cdot T^m \bmod N^s$.
$t := g_1^m \bmod N \in \mathbb{Z}_N$.
Return $\mathcal{E}.\mathsf{ct} := \big((\mathsf{table}^{(\mathsf{c})})_{\mathsf{c} \in \mathcal{S}}, \tilde{e}, t\big)$.
```

```
$m/\bot \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, \mathcal{E}.\mathsf{ct})$:
Parse $\mathcal{E}.\mathsf{ct} = \big((\mathsf{table}^{(\mathsf{c})})_{\mathsf{c} \in \mathcal{S}}, \tilde{e}, t\big)$.
For each $\mathsf{c} = (c_1, \cdots, c_8) \in \mathcal{S}$
    $\hat{v}^{(\mathsf{c})} \leftarrow \mathsf{CalculateV}(\mathsf{sk}, \mathsf{table}^{(\mathsf{c})}, \mathsf{c})$.
If $\tilde{e} \cdot \big(\prod_{\mathsf{c} \in \mathcal{S}} \hat{v}^{(\mathsf{c})}\big)^{-1} \in \mathbb{RU}_{N^s}$
    $m := \mathrm{dlog}_T\big(\tilde{e} \cdot \big(\prod_{\mathsf{c} \in \mathcal{S}} \hat{v}^{(\mathsf{c})}\big)^{-1}\big) \bmod N^{s-1}$.
    If $t = g_1^m \bmod N$, Return $m$.
Otherwise, Return $\bot$.
```

```
$\mathsf{TableGen}\big(\mathsf{pk} = (h_1, h_2, h_3, h_4), \mathsf{c} = (c_1, \cdots, c_8)\big)$:
For each $l \in \{0, 1, \cdots, \sum_{j=1}^{8} c_j\}$
    $\tilde{r}_{l,1}, \tilde{r}_{l,2}, \tilde{r}_{l,3}, \tilde{r}_{l,4} \leftarrow_\$ \big[\big\lfloor \frac{N}{4} \big\rfloor\big]$.
    $(\tilde{u}_{l,1}, \cdots, \tilde{u}_{l,8}) := (g_1^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,1}}, g_2^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,2}}, g_3^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,3}}, g_4^{\tilde{r}_{l,4}}, g_5^{\tilde{r}_{l,4}})$.
    $\tilde{v}_l := h_1^{\tilde{r}_{l,1}} h_2^{\tilde{r}_{l,2}} h_3^{\tilde{r}_{l,3}} h_4^{\tilde{r}_{l,4}}$.
```

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

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

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

```
$\mathsf{CalculateV}\big(\mathsf{sk} = (x_1, y_1, \cdots, x_4, y_4), \mathsf{table}^{(\mathsf{c})}, \mathsf{c} = (c_1, \cdots, c_8)\big)$:
Parse $\mathsf{table}^{(\mathsf{c})} = \Big\{\boxed{\hat{u}_{l,1} \,|\, \hat{u}_{l,2} \,|\, \cdots \,|\, \hat{u}_{l,8}}\Big\}_{l \in \{0,1,\cdots,\sum_{j=1}^{8} c_j\}}$.
$\hat{v}_0 := \hat{u}_{0,1}^{-x_1} \hat{u}_{0,2}^{-y_1} \hat{u}_{0,3}^{-x_2} \hat{u}_{0,4}^{-y_2} \hat{u}_{0,5}^{-x_3} \hat{u}_{0,6}^{-y_3} \hat{u}_{0,7}^{-x_4} \hat{u}_{0,8}^{-y_4}$.
For each $l \in \{1, \cdots, c_1\}$
    $\hat{v}_l := (\hat{u}_{l,1}/\hat{v}_{l-1})^{-x_1} \hat{u}_{l,2}^{-y_1} \hat{u}_{l,3}^{-x_2} \hat{u}_{l,4}^{-y_2} \hat{u}_{l,5}^{-x_3} \hat{u}_{l,6}^{-y_3} \hat{u}_{l,7}^{-x_4} \hat{u}_{l,8}^{-y_4}$.
For each $l \in \{c_1 + 1, \cdots, c_1 + c_2\}$
    $\hat{v}_l := \hat{u}_{l,1}^{-x_1} (\hat{u}_{l,2}/\hat{v}_{l-1})^{-y_1} \hat{u}_{l,3}^{-x_2} \hat{u}_{l,4}^{-y_2} \hat{u}_{l,5}^{-x_3} \hat{u}_{l,6}^{-y_3} \hat{u}_{l,7}^{-x_4} \hat{u}_{l,8}^{-y_4}$.
                    $\vdots$
For each $l \in \{\sum_{j=1}^{7} c_j + 1, \cdots, \sum_{j=1}^{8} c_j\}$
    $\hat{v}_l := \hat{u}_{l,1}^{-x_1} \hat{u}_{l,2}^{-y_1} \hat{u}_{l,3}^{-x_2} \hat{u}_{l,4}^{-y_2} \hat{u}_{l,5}^{-x_3} \hat{u}_{l,6}^{-y_3} \hat{u}_{l,7}^{-x_4} (\hat{u}_{l,8}/\hat{v}_{l-1})^{-y_4}$.
Return $\hat{v}^{(\mathsf{c})} := \hat{v}_{\sum_{j=1}^{8} c_j}$.
```

Fig. 9. Top: $\mathcal{E}.\mathsf{Enc}$ (left) and $\mathcal{E}.\mathsf{Dec}$ (right) of $\mathcal{E}$ designed for $\mathcal{F}_{\text{poly}}^{d}$; Middle: $\mathsf{TableGen}$, which generates $\mathsf{table}^{(\mathsf{c})}$ together with a title $\tilde{v}^{(\mathsf{c})}$; Bottom: $\mathsf{CalculateV}$, which calculates a title $\hat{v}^{(\mathsf{c})}$ from $\mathsf{table}^{(\mathsf{c})}$ using secret key.

For each $\mathsf{c} = (c_1, \cdots c_8) \in \mathcal{S}$, we generate $\mathsf{table}^{(\mathsf{c})}$ and its title $\tilde{v}^{(\mathsf{c})}$ for monomial $x_{i_\lambda,1}^{c_1} y_{i_\lambda,1}^{c_2} x_{i_\lambda,2}^{c_3} y_{i_\lambda,2}^{c_4}\ x_{i_\lambda,3}^{c_5} y_{i_\lambda,3}^{c_6} x_{i_\lambda,4}^{c_7} y_{i_\lambda,4}^{c_8}$ via the algorithm $\mathsf{TableGen}$ illustrated in Fig. 9. Intuitively, $\mathsf{TableGen}$ generates $\mathsf{table}^{(\mathsf{c})}$ of $1 + c_1 + \cdots + c_8$ rows. The

0-th row of $\mathsf{table}^{(\mathsf{c})}$ is $\tilde{u}_{0,1}, \cdots, \tilde{u}_{0,8}$. The form of other rows are similar to row 0 with a small difference: the next c_1 rows in the 1-st column are multiplied with $\tilde{v}_0, \tilde{v}_1, \cdots, \tilde{v}_{c_1-1}$ respectively; the next c_2 rows in the 2-nd column are multiplied with $\tilde{v}_{c_1}, \tilde{v}_{c_1+1}, \cdots, \tilde{v}_{c_1+c_2-1}$ respectively, and so forth. $\mathsf{TableGen}$ also generates a title $\tilde{v}^{(\mathsf{c})}$ for $\mathsf{table}^{(\mathsf{c})}$. The product of all the titles, i.e., $\prod_{\mathsf{c}\in\mathcal{S}} \tilde{v}^{(\mathsf{c})}$, is used to hide T^m in $\tilde{e}$.

On the other hand, the title $\hat{v}^{(\mathsf{c})} = \tilde{v}^{(\mathsf{c})}$ can be recovered from $\mathsf{table}^{(\mathsf{c})}$ with secret key $\mathsf{sk} = (x_1, y_1, \cdots, x_4, y_4)$ via the $\mathsf{CalculateV}$ algorithm in Fig. 9. Therefore, one can always use the secret key to extract the titles $(\tilde{v}^{(\mathsf{c})})_{\mathsf{c}\in\mathcal{S}}$ from tables $\big(\mathsf{table}^{(\mathsf{c})}\big)_{\mathsf{c}\in\mathcal{S}}$ one by one with $\mathsf{CalculateV}$ and then recover m correctly.

Security proof. The proof of KDM[$\mathcal{F}^d_{\text{poly}}$]-CCA security is similar to that of Theorem 2. But games G_3-G_4 should be replaced with the following three steps (Step 1 - Step 3), so that the $\mathcal{E}.\mathsf{Enc}$ part of ENC can be changed to work as an entropy filter, i.e., reserving the entropy of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$, w.r.t. any polynomial of degree at most d.

Suppose that the adversary submits $(f_\lambda, i_\lambda \in [n])$ to ENC. Our aim is to reserve the entropy of $(x_j, y_j)_{j=1}^4 \bmod N$ from $\mathcal{E}.\mathsf{Enc}\big(\mathsf{pk}_{i_\lambda}, f_\lambda\big((x_{i,j}, y_{i,j})_{i\in[n],j\in[4]}\big)\big)$.

Step 0: In INITIALIZE, the secret keys are generated with $x_{i,j} := x_j + \bar{x}_{i,j}$ and $y_{i,j} := y_j + \bar{y}_{i,j} \bmod \lfloor N^2/4 \rfloor$ for $i \in [n]$, $j \in [4]$. This is the same as G_2 in the proof of Theorem 2.

Step 1: Use $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n],j\in[4]}$ to re-explain $(f_\lambda, i_\lambda \in [n])$ as $(f'_\lambda, i_\lambda \in [n])$, and determine the coefficients $a_{(c_1,\cdots,c_8)}$ of each monomial of f'_λ, as discussed in Subsection 6.2. Note that $a_{(c_1,\cdots,c_8)} = 0$ if the associated monomial does not appear in f'_λ. Then

$$f'_\lambda\big((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}\big) = \sum_{(c_1,\cdots,c_8)\in\mathcal{S}} a_{(c_1,\cdots,c_8)} \cdot x^{c_1}_{i_\lambda,1} y^{c_2}_{i_\lambda,1} x^{c_3}_{i_\lambda,2} y^{c_4}_{i_\lambda,2} x^{c_5}_{i_\lambda,3} y^{c_6}_{i_\lambda,3} x^{c_7}_{i_\lambda,4} y^{c_8}_{i_\lambda,4} + \delta,$$

where $\delta = a_{(0,\cdots,0)}$ denotes the constant term of f'_λ.

Step 2: Use secret key $\mathsf{sk}_{i_\lambda} = (x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]}$ (together with public key $\mathsf{pk}_{i_\lambda} = (h_{i_\lambda,j})_{j\in[4]}$) to implement $\mathcal{E}.\mathsf{Enc}$ (This corresponds to G_3 in the proof of Theorem 2).

- For each $\mathsf{c} = (c_1, \cdots, c_8) \in \mathcal{S}$
  (1) $(\mathsf{table}^{(\mathsf{c})}, \tilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}_{i_\lambda}, \mathsf{c})$,
  (2) $\hat{v}^{(\mathsf{c})} \leftarrow \mathsf{CalculateV}(\mathsf{sk}_{i_\lambda}, \mathsf{table}^{(\mathsf{c})}, \mathsf{c})$.
- Use $\big(\hat{v}^{(\mathsf{c})}\big)_{\mathsf{c}\in\mathcal{S}}$ instead of $\big(\tilde{v}^{(\mathsf{c})}\big)_{\mathsf{c}\in\mathcal{S}}$ to compute $\tilde{e}$ with $\tilde{e} := \prod_{\mathsf{c}\in\mathcal{S}} \hat{v}^{(\mathsf{c})} \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})} \bmod N^s$, and $t := g_1^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})} \bmod N$.

It is easy to check that for each $\mathsf{c} = (c_1, \cdots, c_8) \in \mathcal{S}$, $\hat{v}^{(\mathsf{c})}$ computed from $\mathsf{table}^{(\mathsf{c})}$ via $\mathsf{CalculateV}$ is identical to $\tilde{v}^{(\mathsf{c})}$ associated with $\mathsf{table}^{(\mathsf{c})}$ via $\mathsf{TableGen}$. Thus this change is conceptual.

Step 3: This corresponds to G_4 in the proof of Theorem 2.

- For each $\mathsf{c} = (c_1, \cdots, c_8) \in \mathcal{S}$
  (1) Compute $\mathsf{table}^{(\mathsf{c})}$ via $(\mathsf{table}^{(\mathsf{c})}, \tilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}_{i_\lambda}, \mathsf{c})$, but with one difference. The item of row 1 and column $j := \min\{i \mid 1 \le i \le 8, c_i \neq 0\}$ in $\mathsf{table}^{(\mathsf{c})}$ now is computed as $\hat{u}_{1,j} = (\tilde{u}_{1,j} T^{a_{(c_1,\cdots,c_8)}}) \cdot \tilde{v}_0$

instead of $\hat{u}_{1,j} = \tilde{u}_{1,j} \cdot \tilde{v}_0$. This change is computationally indistinguishable, due to the IV$_5$ assumption.

(2) Invoke $\hat{v}^{(\mathsf{c})} \leftarrow \mathsf{CalculateV}(\mathsf{sk}_{i_\lambda}, \mathsf{table}^{(\mathsf{c})}, \mathsf{c})$ to extract a title $\hat{v}^{(\mathsf{c})}$ from the modified $\mathsf{table}^{(\mathsf{c})}$.

– $\tilde{e} := \prod_{\mathsf{c}\in\mathcal{S}} \hat{v}^{(\mathsf{c})} \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})}$, and $t := g_1^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})} \bmod N$.

Observe that for each $\mathsf{c} = (c_1, \cdots, c_8) \in \mathcal{S}$,

$$\hat{v}^{(\mathsf{c})} = \tilde{v}^{(\mathsf{c})} \cdot T^{-a_{(c_1,\cdots,c_8)} x_{i_\lambda,1}^{c1} y_{i_\lambda,1}^{c2} x_{i_\lambda,2}^{c3} y_{i_\lambda,2}^{c4} x_{i_\lambda,3}^{c5} y_{i_\lambda,3}^{c6} x_{i_\lambda,4}^{c7} y_{i_\lambda,4}^{c8}}.$$

Then $\tilde{e} = \prod_{\mathsf{c}\in\mathcal{S}} \hat{v}^{(\mathsf{c})} \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})}$

$= \prod_{\mathsf{c}\in\mathcal{S}} \left(\tilde{v}^{(\mathsf{c})} \cdot T^{-a_{(c_1,\cdots,c_8)} x_{i_\lambda,1}^{c1} y_{i_\lambda,1}^{c2} x_{i_\lambda,2}^{c3} \cdots y_{i_\lambda,3}^{c6} x_{i_\lambda,4}^{c7} y_{i_\lambda,4}^{c8}}\right) \cdot T^{f'_\lambda((x_{i_\lambda,j}, y_{i_\lambda,j})_{j\in[4]})}$

$= \prod_{\mathsf{c}\in\mathcal{S}} \tilde{v}^{(\mathsf{c})} \cdot T^{\delta}$,

where δ is the constant term of f'_λ. Therefore we can implement Step 3 equivalently as follows.

Step 3 (Equivalent Form):

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

Compute $\mathsf{table}^{(\mathsf{c})}$ via $(\mathsf{table}^{(\mathsf{c})}, \tilde{v}^{(\mathsf{c})}) \leftarrow_{\$} \mathsf{TableGen}(\mathsf{pk}_{i_\lambda}, \mathsf{c})$, but with one difference. The item of row 1 and column $j := \min\{i \mid 1 \le i \le 8, c_i \neq 0\}$ in $\mathsf{table}^{(\mathsf{c})}$ now is computed as $\hat{u}_{1,j} = (\tilde{u}_{1,j} T^{a_{(c_1,\cdots,c_8)}}) \cdot \tilde{v}_0$ instead of $\hat{u}_{1,j} = \tilde{u}_{1,j} \cdot \tilde{v}_0$.

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

In this step, $\mathcal{E}.\mathsf{Enc}$ does not use $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ at all (only uses $(\bar{x}_{i,j}, \bar{y}_{i,j})_{i\in[n],j\in[4]}$ and $(x_1, y_1, \cdots, x_4, y_4) \bmod \phi(N)/4$).

As a result, through the computationally indistinguishable change, the entropy of $(x_1, y_1, \cdots, x_4, y_4) \bmod N$ is reserved by the $\mathcal{E}.\mathsf{Enc}$ part of Enc.

Similarly, Dec can be changed to do decryptions without $(x_j, y_j)_{j=1}^4 \bmod N$, the same argument as in Subsection 6.3.

Acknowledgments. The authors are supported by the National Natural Science Foundation of China Grant (Nos. 61672346, 61373153 and 61133014). We thank the anonymous reviewers for their comments and suggestions.

References

[ACPS09] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009, LNCS, vol. 5677, pp. 595–618. Springer (2009)

[BG10] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010, LNCS, vol. 6223, pp. 1–20. Springer (2010)

[BGK11] Brakerski, Z., Goldwasser, S., Kalai, Y.T.: Black-box circular-secure encryption beyond affine functions. In: Ishai, Y. (ed.) TCC 2011, LNCS, vol. 6597, pp. 201–218. Springer (2011)

[BHHI10] Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010, LNCS, vol. 6110, pp. 423–444. Springer (2010)

[BHHO08] Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008, LNCS, vol. 5157, pp. 108–125. Springer (2008)

[BRS02] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) Selected Areas in Cryptography 2002, LNCS, vol. 2595, pp. 62–75. Springer (2002)

[CCS09] Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009, LNCS, vol. 5479, pp. 351–368. Springer (2009)

[CL01] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001, LNCS, vol. 2045, pp. 93–118. Springer (2001)

[DJ01] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001, LNCS, vol. 1992, pp. 119–136. Springer (2001)

[GHV12] Galindo, D., Herranz, J., Villar, J.L.: Identity-based encryption with master key-dependent message security and leakage-resilience. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012, LNCS, vol. 7459, pp. 627–642. Springer (2012)

[GM84] Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. vol. 28(2), pp. 270–299 (1984)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008, LNCS, vol. 4965, pp. 415–432. Springer (2008)

[HLL16] Han, S., Liu, S., Lyu, L.: Efficient KDM-CCA secure public-key encryption for polynomial functions. IACR Cryptology ePrint Archive, Report 2016/829 (2016)

[Hof13] Hofheinz, D.: Circular chosen-ciphertext security with compact ciphertexts. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013, LNCS, vol. 7881, pp. 520–536. Springer (2013)

[KD04] Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M.K. (ed.) CRYPTO 2004, LNCS, vol. 3152, pp. 426–442. Springer (2004)

[LLJ15] Lu, X., Li, B., Jia, D.: KDM-CCA security from RKA secure authenticated encryption. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I, LNCS, vol. 9056, pp. 559–583. Springer (2015)

[MTY11] Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011, LNCS, vol. 6632, pp. 507–526. Springer (2011)

[Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) STOC 2005, pp. 84–93. ACM (2005)