

# Understanding and Constructing AKE via Double-key Key Encapsulation Mechanism

Haiyang Xue<sup>1,2,3</sup>, Xianhui Lu<sup>1,2,3</sup>, Bao Li<sup>1,2,3</sup>, Bei Liang<sup>4</sup>, and Jingnan He<sup>1,2,3</sup>

<sup>1</sup> State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.

<sup>2</sup> Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China.

<sup>3</sup> School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.

<sup>4</sup> Chalmers University of Technology, Gothenburg, Sweden  
{hyxue12, xhlu}@is.ac.cn

**Abstract.** Motivated by abstracting the common idea behind several implicitly authenticated key exchange (AKE) protocols, we introduce a primitive that we call double-key key encapsulation mechanism (2-key KEM). It is a special type of KEM involving two pairs of secret-public keys and satisfying some function and security property. Such 2-key KEM serves as the core building block and provides alternative approaches to simplify the constructions of AKE. To see the usefulness of 2-key KEM, we show how several existing constructions of AKE can be captured as 2-key KEM and understood in a unified framework, including widely used HMQV, NAXOS, Okamoto-AKE, and FSXY12-13 schemes. Then, we show 1) how to construct 2-key KEM from concrete assumptions, 2) how to adapt the classical Fujisaki-Okamoto transformation and KEM combiner to achieve the security requirement of 2-key KEM, 3) an elegant Kyber-AKE over lattice using the improved Fujisaki-Okamoto technique.

**Keywords:** Authenticated Key Exchange, CK Model, Key Encapsulation Mechanism

## 1 Introduction

Key exchange (KE), which enables two parties to securely establish a common session key while communicating over an insecure channel, is one of the most important and fundamental primitives in cryptography. After the introduction of Diffie-Hellman key exchange in [12], cryptographers have devised a wide selection of the KE with various use-cases. One important direction is authenticated key exchange (AKE). The main problems that the following works focus on are specified as security models [5, 7, 25, 6, 15], efficient and provably-secure realizations [26, 27, 7, 23, 25, 28, 1, 15, 16, 35, 30, 2, 36, 3].

In an AKE protocol, each party has a pair of secret-public keys, a *static/long-term public key* and the corresponding *static/long-term secret key*. The static public key is interrelated with a party's identity, which enables the other parties to verify the authentic binding between them. A party who wants to share information with another party generates ephemeral one-time randomness which is known as *ephemeral secret keys*, computes *session state* (which is originally not explicitly defined [7], but nowadays it is generally agreed [25, 15] that the session state should at least contain ephemeral secret keys) from ephemeral and static secret keys and incoming message, then outputs corresponding *ephemeral public outgoing message*. Then each party uses their static secret keys and the ephemeral secret keys along with the transcripts of the session to compute a shared *session key*.

Many studies have investigated the security notion of AKE including BR model and Canetti-Krawczyk (CK) model [7]. Fujioka *et al.* [15] re-formulated the *desirable* security notion of AKE in [24], including resistance to KCI (key compromise impersonation attack), wPFS (weak perfect forward attack) and MEX (maximal exposure attack), as well as provable security in the CK model, and called it the CK<sup>+</sup> security model. LaMacchia *et al.* [25] also proposed a very strong security model, called the eCK model. The CK model and the eCK model are incomparable [6], and the eCK model is not stronger than the CK model while the CK<sup>+</sup> model is [15]. However, each of these two models, eCK and CK<sup>+</sup> can be theoretically seen as a strong version of the AKE security model.

To achieve a secure AKE in one of the above security models (CK, CK<sup>+</sup>, eCK), the solutions are divided into two classes: explicit AKE and implicit AKE. The solution of explicit AKE is to explicitly authenticate the exchanged messages between the involved parties by generally using additional primitives *i.e.*, signature or MAC to combine with the underlying KE, such as IKE [8], SIGMA [23], TLS [22, 2] etc.; while the solution of implicit AKE initiated by [26], is to implicitly authenticate each party by its unique ability so as to compute the resulted session key. These kinds of implicit AKE schemes include (H)MQV [27, 24], Okamoto [28, 29], NAXOS [25], OAKE [35], FSXY variants [1, 15, 16, 34], and AKE from lattice assumptions [36, 3].

**Motivation.** In this paper, we focus on the second class, *i.e.*, constructions of *implicit AKE*. Based on different techniques and assumptions, many implicit AKE protocols have been proposed in recent years [24, 25, 15, 16, 35, 28, 29].

However, the constructing techniques and methods of the existing implicit AKE protocols are somewhat separate and the study on the highly accurate analysis of AKE's requirement for the building block is critically in a shortage, especially for the exact underlying primitives that serve as fundamental building blocks and capture the common idea and technique behind the constructions and security proofs of AKE. On the contrary, with respect to explicit AKE Canetti and Krawczyk [23, 8] gave the frame of "SIGin-and-Mac" (later extended by [30]) which provides a good guideline for designing explicit AKE.

In fact, Boyd *et al.* [1] and Fujioka *et al.* [15, 16] initiated the research on studying frameworks of implicit AKE. Boyd *et al.* firstly noticed the connection between AKE and key encapsulation mechanism (KEM), then Fujioka *et al.* provided CK<sup>+</sup> secure AKE protocols from chosen ciphertext (CCA) secure KEM in the random oracle and standard models. Although the paradigm of connecting the AKE with KEM is of great significance, it can not be applied to explain many widely-used and well-known constructions of AKE such as HMQV and its variant [24, 35] which are built on the challenge-respond signature; AKE protocol in [28] which results from universal hash proof [10]; as well as NAXOS [25].

Hence, one of the important problems on AKE is to give an even more general framework for constructing AKE that is able to not only unify and encompass the existing structures of AKE protocol as much as possible, but also to systemize and simplify the construction and analysis methods of AKE protocol. It will be useful and helpful for understanding the existing works and future studying on formalization of the AKE construction structure under a unified framework, not only with some well-studied cryptographic primitive as building block but also with simple formal functionality and security requirements rather than heuristic ideas and techniques.

**Main Observation.** In order to find out what kind of the fundamental/essential building block is exactly needed for CK<sup>+</sup> secure AKE, let's go back to the original KE, and show insight on how to augment the requirements or capability of adversary so as to achieve CK<sup>+</sup> secure AKE from KE step by step.

In fact, KEM is a KE naturally. The initiator  $U_A$  sends ephemeral public key  $pk$  to responder  $U_B$ .  $U_B$  computes encapsulated key and ciphertext under  $pk$  and returns ciphertext to  $U_A$ . By decapsulating ciphertext using  $sk$ ,  $U_A$  obtains the agreed key encapsulated by  $U_B$ .

*Step 1. AUTHENTICATION.* To make a KE be authenticated, we take unilaterally authenticating  $U_A$  for example. It is required that  $U_A$  has static secret-public keys ( $ssk_A, spk_A$ ), ephemeral secret key  $esk_A$  and ephemeral public outgoing message  $epm_A$ . In light of using KEM with one pair of secret-public key to realize KE naturally, one simple and natural approach to authenticate  $U_A$  with one pair of static key as well as one pair of ephemeral secret key and ephemeral public message is to extend the KEM with one pair of key to a KEM with two pairs of secret-public key. More specifically, for example, to authenticate  $U_A$ ,  $U_A$  sends ephemeral public key  $epk_A$  to  $U_B$ , and  $U_B$  computes encapsulated key and ciphertext under two public keys  $spk_A$  and  $epk_A$ . Only with both secret keys  $ssk_A$  and  $esk_A$ , can  $U_A$  extract encapsulated key. Equipped with the 2-key KEM, the authentication property of AKE comes down to some proper security notions of such 2-key KEM. We analyze its security notion in step 2.

*Step 2. SECURITY.* One security consideration in AKE is to maintain the secrecy of shared session key even if the adversary is allowed to *query session state and key* of non-target session and *send message* by controlling the communications. The capability imparted to adversary with permission of *querying session state and key* of non-target session directly corresponds to the adversary's capability of having

access to strong<sup>1</sup> CCA decryption queries of 2-key KEM. The adversary’s capability of *sending message* corresponds to the power of adversary to adaptively choose the ephemeral public keys  $epk_A$  (under which the challenge ciphertext is computed). Another security consideration in AKE is the *forward security*, in which case the adversary has the static secret key  $ssk_A$ . This forward security comes down to the (chosen plaintext attack) CPA security of such 2-key KEM if  $ssk_A$  is leaked to adversary.

## 1.1 Our Contributions

- Based on the above motivations and observations, we introduce *double-key key encapsulation mechanism* (2-key KEM) and its secure notions, *i.e.*, [IND/OW-CCA, IND/OW-CPA] security. We also show its distinction with previous similar notions.
- Based on the [IND/OW-CCA, IND/OW-CPA] secure 2-key KEM, we present unified frames of CK<sup>+</sup> secure AKE, which in turn conceptually capture the common pattern for the existing constructions and security proof of AKE, including well-known HMQV[24], NAXOS [25], Okamoto-AKE[28, 29], and FSXY12[15], FSXY13[16].
- We investigate the constructions of 2-key KEM based on concrete assumptions. We also show the failure of implying [IND/OW-CCA, IND/OW-CPA] secure 2-key KEM from KEM combiner and the classical Fujisaki-Okamoto (FO) transformation. Hence, with a slight but vital modification by taking public key as input to the hash step we provide improved KEM combiner and improved FO to adapt them in our 2-key KEM setting.
- Equipped with 2-key KEM and our frame above, we propose a post-quantum AKE based on Module-LWE assumption, which consumes less communications than Kyber [3] using frame of FSXY13 [16].

**2-key Key Encapsulation Mechanism.** Generally, the 2-key KEM scheme is a public key encapsulation with two pairs of public and secret keys, but the main distinctions are the functionality and security.

The encapsulation and decapsulation algorithms: instead of taking as input single public key to generate a random key  $K$  and a ciphertext  $C$  and single secret key to decapsulate ciphertext  $C$ , each algorithm takes two public keys  $(pk_1, pk_0)$  to generate  $(C, K)$  and only with both two secret keys  $(sk_1, sk_0)$  the decapsulation algorithm can decapsulate  $C$ .

We define the security notion of 2-key KEM/PKE in the attacking model [IND/OW-CCA, IND/OW-CPA] which captures the idea that the 2-key KEM is secure under one secret-public key pair even if another pair of secret-public key is generated by the adversary. Informally, the [IND/OW-CCA, ·] denotes the security model where adversary  $\mathcal{A}$  aims to attack the ciphertext under  $pk_1$  and  $pk_0^*$  (with its control over the generation of  $pk_0^*$ ), and it is allowed to query a strong decapsulation oracle that will decapsulate the ciphertext under  $pk_1$  and arbitrary  $pk_0'$  (generated by challenger); while [·, IND/OW-CPA] denotes the security model where adversary  $\mathcal{B}$  aims to attack the ciphertext under  $pk_0$  and  $pk_1^*$  (with its control over the generation of  $pk_1^*$ ). We say a 2-key KEM is [IND/OW-CCA, IND/OW-CPA] secure if it is both [IND/OW-CCA, ·] and [·, IND/OW-CPA] secure.

Compared with classical definition of CCA security, the [CCA, ·] adversary of 2-key KEM has two main enhancements: 1) one of the challenge public keys  $pk_0^*$ , under which the challenge ciphertext is computed, is generated by the adversary; 2) the adversary is allowed to query a strong decryption oracle, and get decapsulation of the ciphertext under arbitrary public keys  $(pk_1^*, pk_0')$  where  $pk_0'$  is generated by the challenger.

**AKE from 2-key KEM.** Equipped with [IND/OW-CCA, IND/OW-CPA] 2-key KEM, by taking  $pk_1$  as static public key and  $pk_0$  as ephemeral public key, we give several general frames of CK<sup>+</sup> secure AKE, AKE, AKE<sub>ro-pkic-lr</sub> and AKE<sub>std</sub>, depending on different tricks. The CK<sup>+</sup> security of our AKE is decomposed to the [IND/OW-CCA, ·] security (corresponding to KCI and MEX security) and [·, IND/OW-CPA] security (corresponding to wPFS) of 2-key KEM. Furthermore, to resist the leakage of partial randomness,

<sup>1</sup> Compare with classical decryption queries of CCA security, “strong” means adversary could query decryption oracle with ciphertext under several other public keys.

a function  $f(ssk_B, esk_B)$  is required so that if one of  $ssk_B$  and  $esk_B$  is leaked  $f(ssk_B, esk_B)$  is still computationally indistinguishable with a random string.

In Fig. 1 we summarize which one of our general frames is used to explain which one of the existing AKE protocols by employing the specific tricks and assumptions. Our general protocols capture the common idea of constructing CK<sup>+</sup> secure AKE. And depending on 2-key KEM and different tricks, it facilitates a number of instantiations, including HMQV [24], NAXOS [25], Okamoto [28], FSXY12[15], and FSXY13[16].

Frameworks	Models	Concrete AKEs	Assumptions	Tricks
AKE	RO	FSXY13 [16],Kyber[3]	OW-CCA	Modified KEM Comb.
	RO	AKE-2Kyber(Sec.7)	M-LWE	Modified FO
AKE <sub>ro-pkic-lr</sub>	RO	HMQV [24] OAKE [35]	GDH, KEA1	Remark 1, 2
	RO	NAXOS [25]	GDH	Remark 1, 2
AKE <sub>std</sub>	Std	FSXY12 [15]	IND-CCA	Modified KEM Comb.
	Std	Okamoto [29]	DDH, $\pi$ PRF	Twisted PRF

**Table 1.** The unification of AKEs. Comb. is the abbreviation for combiner. GDH is the Gap-DH assumption. RO denotes the notion of random oracle. Std is the shortened form of standard model.  $\pi$ PRF means the pairwise-independent random source PRF [29].

By considering an AKE protocol in such a framework based on 2-key KEM, the complicated security proofs of existing AKE is decomposed into several smaller cases each of which is easier to work with. Moreover, this general scheme not only explains previous constructions, but also yields efficient AKE from lattice problems. After giving [IND-CPA, IND-CPA] twin-kyber under Module-LWE assumption, we obtain a *post-quantum AKE* with less communications.

**Constructions of 2-key KEM.** In addition to show that existing AKEs imply [CCA, CPA] secure 2-key KEM, we investigate the general constructions.

*Putting Public Key in the Hashing or PRF step.* The Fujisaki-Okamoto (FO) [14, 18] transformation and KEM combiner are general techniques of classical CCA security for one-key KEM. We show the failure of implying [IND/OW-CCA, IND/OW-CPA] secure 2-key KEM from KEM combiner and the classical FO transformation by giving particular attacks on concrete schemes. Hence, we show that with a slight but vital modification, when extracting encapsulated key, by taking public key as input to the hash or PRF step, the modified KEM combiner and FO transformation work for 2-key KEM.

## 1.2 Strong Point of the AKE via 2-key KEM

The main advantage of our contributions is that we use a non-interactive primitive to handle the complex requirement of interactive protocols. The functionality and security requirements of [CCA, CPA] secure 2-key KEM are relatively easier to work with and understand. As it is known, in AKE we have to consider complex and diverse adversaries. However, when considering the AKE under our unified framework based on 2-key KEM, all the attacking strategies in CK<sup>+</sup> model can be simplified to the singular security of 2-key KEM.

The non-interactive 2-key KEM helps us to highly simplify the constructions for AKE as well as to understand the essential working mechanism. In fact, KEM is relatively well-studied and intensively analyzed. Following the first practical CCA secure PKE [9], there have been a number of CCA secure PKE/KEM schemes based on both concrete assumptions [9, 20, 33, 31, 3] and general cryptographic primitives [11, 19, 31]. Therefore, it is possible for us to employ the established and nature technique of classical KEM to construct 2-key KEM, and further AKE.

## 2 Preliminary

For a variable  $x$ , if  $x$  is a bit string, denote  $[x]_i$  as the  $i$ -th bit of  $x$ ; if  $x$  is a polynomial, denote  $[x]_i$  as the  $i$ -th coefficient of  $x$ ; if  $x$  is a sets of vectors (with string or number) denote  $[x]_i$  as the sets of all  $i$ -th element of vectors in  $x$ ;

### 2.1 CK<sup>+</sup> Security Model

We recall the CK<sup>+</sup> model introduced by [24] and later refined by [15, 16], which is a CK [7] model integrated with the weak PFS, resistance to KCI and MEX properties. Since we focus on **two-pass protocols** in this paper, for simplicity, we show the model specified to two pass protocols.

In AKE protocol,  $U_i$  denotes a party indexed by  $i$ , who is modeled as probabilistic polynomial time (PPT) interactive Turing machines. We assume that each party  $U_i$  owns a static pair of secret-public keys  $(ssk_i, spk_i)$ , where the static public key is linked to  $U_i$ 's identity, using some systems i.e. PKI, such that the other parties can verify the authentic binding between them. We do not require the well-formness of static public key, in particular, a corrupted party can adaptively register any static public key of its choice.

**Session.** Each party can be activated to run an instance called a *session*. A party can be activated to initiate the session by an incoming message of the forms  $(\Pi, \mathcal{I}, U_A, U_B)$  or respond to an incoming message of the forms  $(\Pi, \mathcal{R}, U_B, U_A, X_A)$ , where  $\Pi$  is a protocol identifier,  $\mathcal{I}$  and  $\mathcal{R}$  are role identifiers corresponding to *initiator* and *responder*. Activated with  $(\Pi, \mathcal{I}, U_A, U_B)$ ,  $U_A$  is called the session *initiator*. Activated with  $(\Pi, \mathcal{R}, U_B, U_A, X_A)$ ,  $U_B$  is called the session *responder*.

According to the specification of AKE, the party creates randomness which is generally called *ephemeral secret key*, computes and maintains a *session state*, generates outgoing messages, and completes the session by outputting a session key and erasing the session state. Note that Canetti-Krawczyk [7] defines session state as session-specific secret information but leaves it up to a protocol to specify which information is included in session state; LaMacchia et al. [25] explicitly set all random coins used by a party in a session as session-specific secret information and call it *ephemeral secret key*. Here we require that the session state at least contains the ephemeral secret key.

A session may also be aborted without generating a session key. The initiator  $U_A$  creates a session state and outputs  $X_A$ , then may receive an incoming message of the forms  $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$  from the responder  $U_B$ , then may computes the session key  $SK$ . On the contrary, the responder  $U_B$  outputs  $X_B$ , and may compute the session key  $SK$ . We say that a session is *completed* if its owner computes the session key.

A session is associated with its owner, a peer, and a session identifier. If  $U_A$  is the initiator, the session identifier is  $sid = (\Pi, \mathcal{I}, U_A, U_B, X_A)$  or  $sid = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ , which denotes  $U_A$  as an owner and  $U_B$  as a peer. If  $U_B$  is the responder, the session is identified by  $sid = (\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$ , which denotes  $U_B$  as an owner and  $U_A$  as a peer. The *matching session* of  $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$  is  $(\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$  and vice versa.

**Adversary.** The adversary  $\mathcal{A}$  is modeled in the following to capture real attacks in open networks, including the control of communication and the access to some of the secret information.

- **Send(message):**  $\mathcal{A}$  could send message in one of the forms:  $(\Pi, \mathcal{I}, U_A, U_B)$ ,  $(\Pi, \mathcal{R}, U_B, U_A, X_A)$ , or  $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ , and obtains the response.
- **SessionKeyReveal(sid):** if the session  $sid$  is completed,  $\mathcal{A}$  obtains the session key  $SK$  for  $sid$ .
- **SessionStateReveal(sid):** The adversary  $\mathcal{A}$  obtains the session state of the owner of  $sid$  if the session is not completed. The session state includes all ephemeral secret keys and intermediate computation results except for immediately erased information but does not include the static secret key.
- **Corrupt( $U_i$ ):** By this query,  $\mathcal{A}$  learns all information of  $U_A$  (including the static secret, session states and session keys stored at  $U_A$ ); in addition, from the moment  $U_A$  is corrupted all its actions may be controlled by  $\mathcal{A}$ .

**Freshness.** Let  $\text{sid}^* = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$  or  $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$  be a completed session between honest users  $U_A$  and  $U_B$ . If the matching session of  $\text{sid}^*$  exists, denote it by  $\overline{\text{sid}^*}$ . We say session  $\text{sid}^*$  is fresh if  $\mathcal{A}$  does not queries: 1)  $\text{SessionStateReveal}(\text{sid}^*)$ ,  $\text{SessionKeyReveal}(\text{sid}^*)$ , and  $\text{SessionStateReveal}(\overline{\text{sid}^*})$ ,  $\text{SessionKeyReveal}(\overline{\text{sid}^*})$  if  $\overline{\text{sid}^*}$  exists; 2)  $\text{SessionStateReveal}(\text{sid}^*)$  and  $\text{SessionKeyReveal}(\text{sid}^*)$  if  $\overline{\text{sid}^*}$  does not exist.

**Security Experiment.** The adversary  $\mathcal{A}$  could make a sequence of the queries described above. During the experiment,  $\mathcal{A}$  makes the query of  $\text{Test}(\text{sid}^*)$ , where  $\text{sid}^*$  must be a fresh session.  $\text{Test}(\text{sid}^*)$  select random bit  $b \in_U \{0, 1\}$ , and return the session key held by  $\text{sid}^*$  if  $b = 0$ ; and return a random key if  $b = 1$ .

The experiment continues until  $\mathcal{A}$  returns  $b'$  as a guess of  $b$ . The adversary  $\mathcal{A}$  wins the game if the test session  $\text{sid}^*$  is still fresh and  $b' = b$ . The advantage of the adversary  $\mathcal{A}$  is defined as  $\text{Adv}_{\Pi}^{\text{ck}^+}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$ .

**Definition 1.** We say that a AKE protocol  $\Pi$  is secure in the  $\text{CK}^+$  model if the following conditions hold:

(Correctness:) if two honest parties complete matching sessions, then they both compute the same session key except with negligible probability.

(Soundness:) for any PPT adversary  $\mathcal{A}$ ,  $\text{Adv}_{\Pi}^{\text{ck}^+}(\mathcal{A})$  is negligible for the test session  $\text{sid}^*$ ,

1. the static secret key of the owner of  $\text{sid}^*$  is given to  $\mathcal{A}$ , if  $\overline{\text{sid}^*}$  does not exist.
2. the ephemeral secret key of the owner of  $\text{sid}^*$  is given to  $\mathcal{A}$ , if  $\overline{\text{sid}^*}$  does not exist.
3. the static secret key of the owner of  $\text{sid}^*$  and the ephemeral secret key of  $\text{sid}^*$  are given to  $\mathcal{A}$ , if  $\overline{\text{sid}^*}$  exists.
4. the ephemeral secret key of  $\text{sid}^*$  and the ephemeral secret key of  $\overline{\text{sid}^*}$  are given to  $\mathcal{A}$ , if  $\overline{\text{sid}^*}$  exists.
5. the static secret key of the owner of  $\text{sid}^*$  and the static secret key of the peer of  $\text{sid}^*$  are given to  $\mathcal{A}$ , if  $\overline{\text{sid}^*}$  exists.
6. the ephemeral secret key of  $\text{sid}^*$  and the static secret key of the peer of  $\text{sid}^*$  are given to  $\mathcal{A}$ , if  $\overline{\text{sid}^*}$  exists.

As indicated in Table 2, the  $\text{CK}^+$  model captures all non-trivial patterns of exposure of static and ephemeral secret keys listed in Definition 1, and these ten cases cover wPFS, resistance to KCI, and MEX as follows:  $E_1, E_4, E_{7-1}, E_{7-2}, E_{8-1}$  and  $E_{8-2}$  capture KCI, since the adversary obtains either only the static secret key of one party or both the static secret key of one party and the ephemeral secret key of the other party of the test session.  $E_5$  captures wPFS.  $E_2, E_3$  and  $E_6$  capture MEX, since the adversary obtains the ephemeral secret key of one party of the test session at least.

Event	Case	$\text{sid}^*$	$\overline{\text{sid}^*}$	$ssk_A$	$esk_A$	$esk_B$	$ssk_B$	Security
$E_1$	1	A	No	✓	×	-	×	KCI
$E_2$	2	A	No	×	✓	-	×	MEX
$E_3$	2	B	No	×	-	✓	×	MEX
$E_4$	1	B	No	×	-	×	✓	KCI
$E_5$	5	A or B	Yes	✓	×	×	✓	wPFS
$E_6$	4	A or B	Yes	×	✓	✓	×	MEX
$E_{7-1}$	3	A	Yes	✓	×	✓	×	KCI
$E_{7-2}$	3	B	Yes	×	✓	×	✓	KCI
$E_{8-1}$	6	A	Yes	×	✓	×	✓	KCI
$E_{8-2}$	6	B	Yes	✓	×	✓	×	KCI

**Table 2.** The behavior of AKE adversary in  $\text{CK}^+$  model.  $\overline{\text{sid}^*}$  is the matching session of  $\text{sid}^*$ , if it exists. “Yes” means that there exists  $\overline{\text{sid}^*}$ , “No” means do not.  $ssk_A(ssk_B)$  means the static secret key of A(B).  $esk_A(esk_B)$  is the ephemeral secret key of A(B) in  $\text{sid}^*$  or  $\overline{\text{sid}^*}$  if there exists. “✓” means the secret key may be revealed to adversary, “×” means the secret key is not revealed. “-” means the secret key does not exist.

### 3 2-key Key Encapsulation Mechanism

In this section, we introduce the notions of double-key encapsulation and define the security of KEM in double-key setting. We also give some analysis and show differences with previous similar definitions.

#### 3.1 2-key Key Encapsulation Mechanism

Generally, a double-key (2-key) KEM is a public key encapsulation with two pairs of public and secret keys. Formally, a 2-key KEM  $2\text{KEM}=(\text{KeyGen1}, \text{KeyGen0}, \text{Encaps}, \text{Decaps})$  is a quadruple of PPT algorithms together with a key space  $\mathcal{K}$ .

- $\text{KeyGen1}(\lambda, pp)$  : on inputs security parameter  $\lambda$ , and public parameters  $pp$ , output a pair of public-secret keys  $(pk_1, sk_1)$ . In order to show the randomness that is used, we denote key generation algorithm as  $\text{KeyGen1}(\lambda, pp; r)$ . For simplicity, sometimes we omit the input security parameter  $\lambda$  and public parameter  $pp$  and denote it as  $\text{KeyGen1}(r)$  directly.
- $\text{KeyGen0}(\lambda)$  : on inputs security parameter  $\lambda$  output a pair of public and secret keys  $(pk_0, sk_0)$ .
- $\text{Encaps}(pk_1, pk_0; \text{aux}_e)$  : on input public keys  $pk_1, pk_0$  and auxiliary input  $\text{aux}_e$  (if there is), output ciphertext  $c$  and encapsulated key  $k$  in key space  $\mathcal{K}$ . Sometimes, we explicitly add the randomness  $r$  and denote it as  $\text{Encaps}(pk_1, pk_0, r; \text{aux}_e)$ .
- $\text{Decaps}(sk_1, sk_0, c; \text{aux}_d)$  : on input secret keys  $sk_0, sk_1$ , auxiliary input  $\text{aux}_d$  (if there is) and  $c$ , output key  $k$ .

**CORRECTNESS.** For  $(pk_1, sk_1) \leftarrow \text{KeyGen1}(\lambda, pp)$ ,  $(pk_0, sk_0) \leftarrow \text{KeyGen0}(\lambda, pp)$  and  $(c, k) \leftarrow \text{Encaps}(pk_1, pk_0)$ , we require that  $\text{Decaps}(sk_1, sk_0, c) = k$  holds with all but negligible probability.

**SECURITY.** We consider two kinds of security *i.e.*, indistinguishability and one-wayness in the attacking model  $[\text{ATK}_1, \text{ATK}_0]$ . More precisely, in our  $[\text{ATK}_1, \text{ATK}_0]$  security model for  $2\text{KEM}$ , we consider two adversaries, *i.e.*,  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  attacking  $pk_1$  (controlling the generation of  $pk_0^*$ ) and  $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$  attacking  $pk_0$  (controlling the generation of  $pk_1^*$ ). In Figure 1 below we show the security games of one-wayness and indistinguishable security corresponding to  $[\text{IND}/\text{OW-ATK}_1, \cdot]$  and  $[\cdot, \text{IND}/\text{OW-ATK}_0]$  respectively.

To be clear, the auxiliary inputs  $\text{aux}_e$  and  $\text{aux}_d$  may contain public part, called public auxiliary input, and secret part, called secret auxiliary input. In the security games, both the challenger and adversary have public auxiliary input, while only the challenger has the secret auxiliary input. For simplicity, we do not explicitly show  $\text{aux}_e$  and  $\text{aux}_d$  in the security games.

On the  $i$ -th query of  $\mathcal{O}_{\text{leak}_0}$ , the challenger generates  $(pk_0^i, sk_0^i) \leftarrow \text{KeyGen0}(r_0^i)$ , sets  $L_0 = L_0 \cup \{(pk_0^i, sk_0^i, r_0^i)\}$  and returns  $(pk_0^i, sk_0^i, r_0^i)$  to adversary  $\mathcal{A}$ . On the  $i$ -th query of  $\mathcal{O}_{\text{leak}_1}$ , the challenger generates  $(pk_1^i, sk_1^i) \leftarrow \text{KeyGen1}(r_1^i)$ , sets  $L_1 = L_1 \cup \{(pk_1^i, sk_1^i, r_1^i)\}$  and returns  $(pk_1^i, sk_1^i, r_1^i)$  to adversary  $\mathcal{B}$ .

Depending on the definition of oracle  $\mathcal{O}_{\text{ATK}_1}$  the adversary  $\mathcal{A}$  accesses, and  $\mathcal{O}_{\text{ATK}_0}$  that the adversary  $\mathcal{B}$  accesses, we get CPA and CCA notions respectively.

- if  $\mathcal{O}_{\text{ATK}_1}(pk_0^i, c') = -$ , it implies CPA notion;
- if  $\mathcal{O}_{\text{ATK}_1}(pk_0^i, c') \neq -$ , it works as following: If  $pk_0^i \in [L_0]_1 \wedge (c' \neq c^* \vee pk_0^i \neq pk_0^*)$ , compute  $k' \leftarrow \text{Decaps}(sk_1, sk_0^i, c')$ , and return the corresponding  $k'$ , otherwise return  $\perp$ . This case implies CCA notion.
- if  $\mathcal{O}_{\text{ATK}_0}(pk_1^i, c') = -$ , it implies CPA notion;
- if  $\mathcal{O}_{\text{ATK}_0}(pk_1^i, c') \neq -$ , it works as following: If  $pk_1^i \in [L_1]_1 \wedge (c' \neq c^* \vee pk_1^i \neq pk_1^*)$ , compute  $k' \leftarrow \text{Decaps}(sk_1^i, sk_0, c')$ , and return the corresponding  $k'$ , otherwise return  $\perp$ . This case implies CCA notion.

Let  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  be an adversary against  $pk_1$  of  $2\text{KEM}$ . We define the advantage of  $\mathcal{A}$  winning in the game  $\text{IND-ATK1}$  and  $\text{OW-ATK1}$  respectively as:  $\text{Adv}_{2\text{KEM}}^{[\text{IND-ATK1}, \cdot]}(\mathcal{A}) = \left| \Pr[\text{IND-ATK1}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2} \right|$ , and  $\text{Adv}_{2\text{KEM}}^{[\text{OW-ATK1}, \cdot]}(\mathcal{A}) = \Pr[\text{OW-ATK1}^{\mathcal{A}} \Rightarrow 1]$ , where game  $[\text{IND-ATK1}, \cdot]$  and  $[\text{OW-ATK1}, \cdot]$  are described in Figure 1.

Game [IND-ATK1, ·] on $pk_1$	Game [·, IND-ATK0] on $pk_0$
01 $(pk_1, sk_1) \leftarrow \text{KeyGen1}(pp)$ ;	14 $(pk_0, sk_0) \leftarrow \text{KeyGen0}(pp)$
02 $L_0 = \{(-, -, -)\}$	15 $L_1 = \{(-, -, -)\}$
03 $(state, pk_0^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{ATK1}}, \mathcal{O}_{\text{leak0}}}(pk_1)$	16 $(state, pk_1^*) \leftarrow \mathcal{B}_1^{\mathcal{O}_{\text{ATK0}}, \mathcal{O}_{\text{leak1}}}(pk_0)$ ;
04 $b \leftarrow \{0, 1\}$ ;	17 $b \leftarrow \{0, 1\}$ ;
05 $(c^*, k_0^*) \leftarrow \text{Encaps}(pk_1, pk_0^*), k_1^* \leftarrow \mathcal{K}$ ;	18 $(c^*, k_0^*) \leftarrow \text{Encaps}(pk_1^*, pk_0), k_1^* \leftarrow \mathcal{K}$ ;
06 $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{ATK1}}, \mathcal{O}_{\text{leak0}}}(state, c^*, k_b^*)$ ;	19 $b' \leftarrow \mathcal{B}_2^{\mathcal{O}_{\text{ATK0}}, \mathcal{O}_{\text{leak1}}}(state, c^*, k_b^*)$ ;
07 return $b' \stackrel{?}{=} b$	20 return $b' \stackrel{?}{=} b$
Game [OW-ATK1, ·] on $pk_1$	Game [·, OW-ATK0] on $pk_0$
08 $(pk_1, sk_1) \leftarrow \text{KeyGen1}(pp)$ ;	21 $(pk_0, sk_0) \leftarrow \text{KeyGen0}(pp)$
09 $L_0 = \{(-, -, -)\}$	22 $L_1 = \{(-, -, -)\}$
10 $(state, pk_0^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{ATK1}}, \mathcal{O}_{\text{leak0}}}(pk_1)$	23 $(state, pk_1^*) \leftarrow \mathcal{B}_1^{\mathcal{O}_{\text{ATK0}}, \mathcal{O}_{\text{leak1}}}(pk_0)$ ;
11 $(c^*, k^*) \leftarrow \text{Encaps}(pk_1, pk_0^*)$ ;	24 $(c^*, k^*) \leftarrow \text{Encaps}(pk_1^*, pk_0)$ ;
12 $k' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{ATK1}}, \mathcal{O}_{\text{leak0}}}(state, c^*)$ ;	25 $k' \leftarrow \mathcal{B}_2^{\mathcal{O}_{\text{ATK0}}, \mathcal{O}_{\text{leak1}}}(state, c^*)$ ;
13 return $k' \stackrel{?}{=} k^*$	26 return $k' \stackrel{?}{=} k^*$

**Fig. 1.** The [ATK1, ·], and [·, ATK0] games of 2KEM for adversaries  $\mathcal{A}$  and  $\mathcal{B}$ . The oracles  $\mathcal{O}_{\text{leak0}}$ ,  $\mathcal{O}_{\text{ATK1}}$ ,  $\mathcal{O}_{\text{leak1}}$ , and  $\mathcal{O}_{\text{ATK0}}$  are defined in the following.

We say that 2KEM is [IND-ATK1, ·] secure, if  $\text{Adv}_{2\text{KEM}}^{[\text{IND-ATK1}, \cdot]}(\mathcal{A})$  is negligible; that 2KEM is [OW-ATK1, ·] secure, if  $\text{Adv}_{2\text{KEM}}^{[\text{OW-ATK1}, \cdot]}(\mathcal{A})$  is negligible, for any PPT adversary  $\mathcal{A}$ . The [·, IND-ATK0] and [·, OW-ATK0] security can be defined in the same way. Here to avoid repetition we omit their description.

**[ATK1, ATK0] security.** The scheme 2KEM is called [ATK1, ATK0] secure if it is both [ATK1, ·] and [·, ATK0] secure for any PPT algorithms  $\mathcal{A}$  and  $\mathcal{B}$ . By the combination of adversaries  $\mathcal{A}$  and  $\mathcal{B}$  attacking different security (*i.e.*, indistinguishability and one-wayness), we could get 16 different definitions of security for 2-key KEM.

What we concern in this paper is the [CCA, CPA] security in both indistinguishability and one-wayness setting. For simplicity in the following parts we abbreviate the security model as [IND/OW-CCA, IND/OW-CPA].

### 3.2 Differences between [CCA, ·] Security and Previous Definitions

In order to avoid confusion, we re-clarify the definition of [IND/OW-CCA, ·] security and analyze its difference with previous similar notions, including classical CCA security, KEM combiner [17], and completely non-malleable scheme[13].

Compared with classical CCA adversary, the [CCA, ·] adversary of 2-key KEM 1) has the capability of choosing one of the challenge public key  $pk_0^*$ ; 2) could query a strong decryption oracle, which decapsulates the ciphertext under several public keys  $(pk_1^*, pk_0')$  where  $pk_0'$  is generated by the challenger. While in the classical definition of decapsulation oracle the adversary could only query decapsulation oracle with ciphertext under the challenge public keys  $(pk_1^*, pk_0^*)$ .

Very recently, Giacon *et. al* [17] study combiners for KEMs. That is, given a set of KEMs, an unknown subset of which might be arbitrarily insecure, Giacon *et. al* investigate how they can be combined to form a single KEM that is secure if at least one ingredient KEM is. The KEM combiners treated by Giacon *et. al* have a parallel structure: If the number of KEMs to be combined is  $n$ , a public key of the resulting KEM consists of a vector of  $n$  public keys; likewise for secret keys. The encapsulation procedure performs  $n$  independent encapsulations, one for each combined KEM. The ciphertext of the resulting KEM is simply the concatenation of all generated ciphertexts. The session key is obtained as a function of keys and ciphertexts. Although from the literature our 2-key KEM looks like the two KEM combiner, the security requirement and concrete constructions between them are completely different. Since the two KEM combiner considers the problem that if one of two KEMs is insecure and the other one is



CCA secure, how to combine them to obtain a CCA secure single KEM. In fact, the adversary of KEM combiner security model is the classical CCA adversary (it can only query the decryption oracle under certain public keys). Actually, in Section 6.1, we show there exists [CCA, ·] adversary to attack a CCA secure two KEM combiner.

Aiming to construct non-malleable commitments, Fischlin [13] considered completely non-malleable (NM) schemes. The complete NM scheme is later extended to indistinguishability setting by Barbosa and Farshim [4] with a strong decryption oracle, which allows the adversary to queries with ciphertext under *arbitrary public key of its choice*. Note that our [CCA, ·] is also modeled to allow the adversary to query a strong (but weaker than complete NM) decapsulation oracle with ciphertext under several public keys that are chosen by challenger instead of by adversary. On the other hand, the complete NM adversary is not allowed to choose part of challenge public key, while [CCA, ·] is.

Based on the above observations, we give comparison among these different definitions by considering two public keys in Table 3. For convenience, we consider classical CCA and complete NM schemes in which public keys are expressed as two public keys  $(pk_1, pk_0)$  and let KEM combiner be two combiner of KEM. The differences among security requirements are the capability of adversary, namely, whether the adversary is allowed to choose part of the challenge public keys, or under which public keys the ciphertexts that adversary is allowed to query decryption oracle with are computed.

Definitions	Cha. PK $(pk_1^*, pk_0^*)$	Cha. ciphertext $c^*$	$\mathcal{O}_{\text{Dec}}((pk_1, pk_0), c')$
Classical CCA	$(pk_1^*, pk_0^*) \leftarrow \mathcal{C}$	$c^*$ under $(pk_1^*, pk_0^*)$	$(pk_1, pk_0) = (pk_1^*, pk_0^*)$
KEM Combiner [17]	$(pk_1^*, pk_0^*) \leftarrow \mathcal{C}, \mathcal{A}(sk_0^*)$	$c_1^*    c_0^*, c_i^*$ under $pk_i^*$	$(pk_1, pk_0) = (pk_1^*, pk_0^*)$
Complete NM [13]	$(pk_1^*, pk_0^*) \leftarrow \mathcal{C}$	$c^*$ under $(pk_1^*, pk_0^*)$	$(pk_1, pk_0) \leftarrow \mathcal{A}$
[CCA, ·]	$pk_1^* \leftarrow \mathcal{C}, pk_0^* \leftarrow \mathcal{A}$	$c^*$ under $(pk_1^*, pk_0^*)$	$pk_1 = pk_1^*, pk_0 \leftarrow \mathcal{C}$

**Table 3.** The differences of related definitions. “Cha.” is the abbreviation of “challenge”.  $\mathcal{C}$  denote the challenger and  $\mathcal{A}$  denote the adversary. We use  $\mathcal{A}(sk_0^*)$  to denote that  $\mathcal{A}$  breaks the KEM under  $pk_0^*$ . In both Classical CCA and KEM combiner the decapsulation oracle only returns when  $(pk_1, pk_0) = (pk_1^*, pk_0^*)$ , while in Complete NM  $(pk_1, pk_0)$  could be arbitrary public keys chosen by adversary, and in [CCA, ·],  $pk_0$  could be arbitrary public key chosen by challenger.

### 3.3 Basic Definitions and Results related to 2-key KEM

**[CCA, ·] security with non-adaptive adversary** We can define a weak [CCA, ·] adversary, who is not able to adaptively choose the challenge public key. In this case, taking the adversary  $\mathcal{A}$  attacking  $pk_1$  as an example, the challenge public key  $pk_0^*$  is generated by challenger instead of  $\mathcal{A}$ , which means  $pk_0^* \in [L_0]_1$ .

**Public Key Independent Ciphertext.** The concept of public-key-independent-ciphertext (PKIC) was first proposed in [34]. We extend it to 2-key KEM setting. The PKIC 2-key KEM allows a ciphertext to be generated independently from one of two public keys, while the encapsulated key underlay in such ciphertext to be generated with the randomness and both two public keys. More precisely, algorithm  $(c, k) \leftarrow \text{Encaps}(pk_1, pk_0, r)$  can be realized in two steps: in step 1, ciphertext  $c$  is generated from  $pk_1$  and randomness  $r$ . We precisely denote it as  $c \leftarrow \text{Encaps0}(pk_1, -, r)$ ; in step 2, the encapsulated key  $k$  in  $c$  is generated from  $r, pk_1$ , and  $pk_0$ . We precisely denote it as  $k \leftarrow \text{Encaps1}(pk_1, pk_0, r)$ .

**Classical one-key KEM and 2-key KEM.** Note that given a concrete 2-key KEM, usually it is not obvious and natural to regress to one-key KEM by setting  $pk_0 = -$ . However given any classical one-key KEM, it can be seen as a 2-key KEM with KeyGen0 not in use and  $pk_0 = -$ . At that time, the [OW/IND-CCA, ·] security of this 2-key KEM return to the classical OW/IND-CCA security of the underlying KEM.

**Min-Entropy.** In case of 2-key KEM with PPT adversary  $\mathcal{A}$ , for  $(pk_1, sk_1) \leftarrow \text{KeyGen1}$  and  $pk_0 \leftarrow \mathcal{A}$  or  $(pk_0, sk_0) \leftarrow \text{KeyGen0}$  and  $pk_1 \leftarrow \mathcal{A}$ , we define the *min-entropy* of  $\text{Encaps}(pk_1, pk_0)$  by  $\gamma(pk_1, pk_0, \mathcal{A}) =$

$-\log \max_{c \in \mathcal{C}} \Pr[c = \text{Encaps}(pk_1, pk_0)]$ . We say that KEM is  $\gamma$ -spread if for every  $(pk_1, sk_1) \leftarrow \text{KeyGen1}$  and  $pk_0 \leftarrow \mathcal{A}$  or  $(pk_0, sk_0) \leftarrow \text{KeyGen0}$  and  $pk_1 \leftarrow \mathcal{A}$ ,  $\gamma(pk_1, pk_0, \mathcal{A}) \geq \gamma$ , which means for every ciphertext  $c \in \mathcal{C}$ , it has  $\Pr[c = \text{Encaps}(pk_1, pk_0)] \leq 2^{-\gamma}$ .

## 4 Authenticated Key Exchange from 2-key KEM

In this section, we propose  $\text{CK}^+$  secure AKEs from [CCA, CPA] secure 2-key KEM in both random oracle and standard models. Before showing our AKEs, we need a primitive of random function with half of leakage, that is used by several existing AKEs.

**Definition 2 (Random Function with half of leakage (hl-RF)).** Let  $f : D_{sk} \times D_b \rightarrow R$  be a function from domain  $D_{sk} \times D_b$  to  $R$ . Denote  $\text{KeyGen} \rightarrow D_{sk} \times D_{pk}$  as key generation algorithm for some KEM. Let  $\mathcal{D}_b, \mathcal{R}$  be the uniform distributions over  $D_b, R$ . It is called  $(\varepsilon_1, \varepsilon_2)$  hl-RF with respect to  $\text{KeyGen}$ , if for  $(pk, sk) \leftarrow \text{KeyGen}$ , the following distributions are computational indistinguishable with advantage  $\varepsilon_1, \varepsilon_2$ .

$$\begin{aligned} \{(pk, sk, f(sk, b)) | b \leftarrow \mathcal{D}_b\} &=_{\varepsilon_1} \{(pk, sk, U) | U \leftarrow \mathcal{R}\}; \\ \{(pk, b, f(sk, b)) | b \leftarrow \mathcal{D}_b\} &=_{\varepsilon_2} \{(pk, b, U) | b \leftarrow \mathcal{D}_b, U \leftarrow \mathcal{R}\}. \end{aligned}$$

The hk-RF can be achieved in both random oracle model and standard model.

- In the random oracle model, if  $f$  is a hash function, without the knowledge of  $b$ , the output of  $f$  is totally random; if KEM with respect to  $\text{KeyGen}$  is secure, without the knowledge of  $sk$  the output of  $f$  is computational indistinguishable with a random string (otherwise the adversary must query random oracle with  $sk$  which against the security of KEM) given  $pk$ . Then equation 2 holds. This structure is known as NAXOS trick [25].
- Let  $F' : D_b \times \{0, 1\}^\lambda \rightarrow R$  and  $F'' : D_{sk} \times D_b \rightarrow R$  be two pseudo random functions (PRFs). If assume  $\text{KeyGen}$  outputs an additional string  $s \leftarrow \{0, 1\}^\lambda$ , after obtaining  $(pk, sk)$ , set  $sk = (sk || s)$ . If  $f(sk, b) = F'_b(1^\lambda) \oplus F''_s(b)$ , then even given  $pk$ , without the knowledge of  $s$  or  $b$ ,  $f(sk, b)$  is computational indistinguishable with random distribution over  $R$ . This is known as twisted PRF trick [15][28].

### 4.1 AKE from 2-key KEM in Random Oracle Model

*Roadmap:* We first give a basic AKE from two [OW-CCA, OW-CPA] secure 2-key KEMs. Utilizing extra properties of 2-key KEM, like PKIC or resistance of leakage of partial randomness, we then present two elegant AKEs based on 2-key KEM with different property.

Let  $2\text{KEM} = (\text{KeyGen1}, \text{KeyGen0}, \text{Encaps}, \text{Decaps})$  be a [OW-CCA, OW-CPA] secure 2-key KEM with secret key space  $D_{sk_1} \times D_{sk_0}$ , random space  $R$ . Let  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$  be hash function,  $f_A : D_{sk_1} \times \{0, 1\}^* \rightarrow R$  and  $f_B : D_{sk_1} \times \{0, 1\}^* \rightarrow R$  be hl-RFs. The  $\text{CK}^+$  secure AKE is presented in Figure 2.

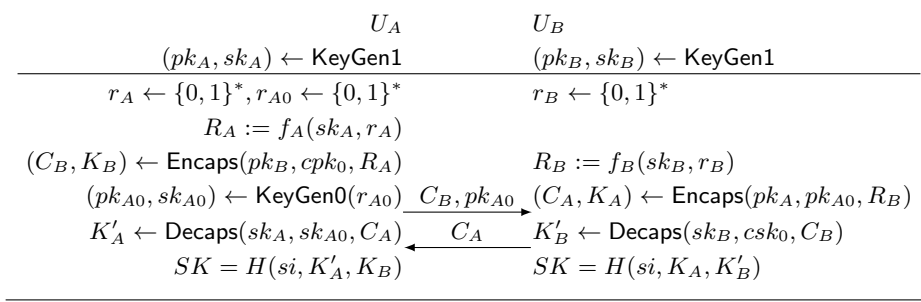
**Stage 0: static secret-public key pair and public parameters.** Each user's static secret-public key pair is generated using  $\text{KeyGen1}$ . Sample one pair of key  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  (which need not to be randomly generated). Set  $cpk_0$  as the predetermined ephemeral public key which will be used by initiator afterwards and  $csk_0$  as the predetermined ephemeral secret key that will be used by responder. Let  $(cpk_0, csk_0)$  be parts of public parameters.

**Stage 1:** Initiator  $U_A$  generates two randomness  $r_A, r_{A0}$ ; it computes  $(C_B, K_B)$  under public key  $pk_B$  and predetermined  $cpk_0$  with randomness  $f_A(sk_A, r_A)$ , and generates ephemeral secret-public key  $(pk_{A0}, sk_{A0}) \leftarrow \text{KeyGen0}(r_{A0})$ . Then it sends  $C_B, pk_{A0}$  to  $U_B$ .

**Stage 2:** Responder  $U_B$  generates randomness  $r_B$ ; it computes  $(C_A, K_A)$  under public keys  $pk_A$  and  $pk_{A0}$  with randomness  $f_B(sk_B, r_B)$ ;  $U_B$  sends  $C_A$  to  $U_A$  and de-encapsulates  $C_B$  using  $sk_B$  and predetermined  $csk_0$  to obtain  $K'_B$ ; it then computes  $SK = H(U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A, K_A, K'_B)$ , and erases  $K'_B$ .

**Stage 3:**  $U_A$  de-encapsulates  $C_A$  using  $sk_A$  and  $sk_{A0}$  to obtain  $K'_A$  and computes  $SK = H(U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A, K'_A, K_B)$ .

The session state of  $sid$  owned by  $U_A$  consists of ephemeral secret key  $r_{A0}, r_A$ , decapsulated key  $K'_A$  and encapsulated key  $K_B$ ; The session state of  $sid$  owned by  $U_B$  consists of ephemeral secret key  $r_B$  and encapsulated key  $K_A$ .



**Fig. 2.** AKE from [OW-CCA, OW-CPA] secure 2KEM in random oracle model.  $cpk_0, csk_0$  are predetermined and default ephemeral keys and they are part of the public parameters.  $si$  here is  $(U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A)$ .

**Theorem 1.** *If the underlying 2KEM is [OW-CCA, OW-CPA] secure and  $\gamma$ -spread,  $f_A, f_B$  are  $(\varepsilon_1, \varepsilon_2)$  hl-RFs, and there are  $N$  users in the AKE protocol and the upbound of sessions between two users is  $l$ , for any PPT adversary  $\mathcal{A}$  against AKE with totally  $q$  times of  $CK^+$  queries, there exists  $\mathcal{S}$  s.t.,*

$$Adv_{AKE}^{ck+}(\mathcal{A}) \leq \frac{1}{2} + \min \left\{ \begin{array}{l} N^2 l \cdot Adv_{2KEM}^{[OW-CCA, \cdot]}(\mathcal{S}) + N^2 l q \cdot (\varepsilon_1 + \varepsilon_2 + 2^{-\gamma}), \\ N^2 l \cdot Adv_{2KEM}^{[\cdot, OW-CPA]}(\mathcal{S}) + N^2 l q \cdot \varepsilon_2 \end{array} \right\}.$$

**Proof of Theorem 1.** Let  $\text{Succ}$  be the event that the guess of  $\mathcal{A}$  against fresh test session is correct. Let  $\text{AskH}$  be the event that  $\mathcal{A}$  poses  $(U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A, K_A, K_B)$  to  $H$ , where  $C_B, pk_{A0}, C_A$  are the views of the test session and  $K_A, K_B$  are the keys encapsulated in the test session. Let  $\overline{\text{AskH}}$  be the complement of  $\text{AskH}$ . Then,

$$\Pr[\text{Succ}] = \Pr[\text{Succ} \wedge \overline{\text{AskH}}] + \Pr[\text{Succ} \wedge \text{AskH}] \leq \Pr[\text{Succ} \wedge \overline{\text{AskH}}] + \Pr[\text{AskH}],$$

where the probability is taken over the randomness used in  $CK^+$  experiment.

We then show that  $\Pr[\text{Succ} \wedge \overline{\text{AskH}}] \leq 1/2$  (as in Lemma 1) and  $\Pr[\text{AskH}]$  is negligible (as in Lemma 2) in all the events (listed in Table 2) of  $CK^+$  model. Followed by Lemma 1 and Lemma 2, we achieve the security of AKE in  $CK^+$  model. Thus, we only need to prove Lemma 1 and Lemma 2.

**Lemma 1.** *If  $H$  is modeled as a random oracle, we have  $\Pr[\text{Succ} \wedge \overline{\text{AskH}}] \leq 1/2$ .*

**Proof of Lemma 1:** If  $\Pr[\overline{\text{AskH}}] = 0$  then the claim is straightforward, otherwise we have  $\Pr[\text{Succ} \wedge \overline{\text{AskH}}] = \Pr[\text{Succ} | \overline{\text{AskH}}] \Pr[\overline{\text{AskH}}] \leq \Pr[\text{Succ} | \text{AskH}] \Pr[\overline{\text{AskH}}]$ . Let  $sid$  be any completed session owned by an honest party such that  $sid \neq sid^*$  and  $sid$  is not matching  $sid^*$ . The inputs to  $sid$  are different from those to  $sid^*$  and  $\overline{sid^*}$  (if there exists the matching session of  $sid^*$ ). If  $\mathcal{A}$  does not explicitly query the view and keys to oracle, then  $H(U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A, K_A, K_B)$  is completely random from  $\mathcal{A}$ 's point of view. Therefore, the probability that  $\mathcal{A}$  wins when  $\text{AskH}$  does not occur is exactly  $1/2$ .

**Lemma 2.** *If the underlying 2KEM is [OW-CCA, OW-CPA] secure, the probability of event  $\text{AskH}$  defined above is negligible. Precisely,*

$$\Pr[\text{AskH}] \leq \min \left\{ \begin{array}{l} N^2 l \cdot Adv_{2KEM}^{[OW-CCA, \cdot]}(\mathcal{S}) + N^2 l q \cdot (\varepsilon_1 + \varepsilon_2 + 2^{-\gamma}), \\ N^2 l \cdot Adv_{2KEM}^{[\cdot, OW-CPA]}(\mathcal{S}) + N^2 l q \cdot \varepsilon_2 \end{array} \right\}.$$

Events	sid	sid*	ssk <sub>A</sub>	esk <sub>A</sub>	esk <sub>B</sub>	ssk <sub>B</sub>	Bounds
AskH $\wedge$ E <sub>1</sub>	A	No	√	×	-	×	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>B</sub> , pk <sub>0</sub> <sup>*</sup> = cpk <sub>0</sub>
AskH $\wedge$ E <sub>2</sub>	A	No	×	√	-	×	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>B</sub> , pk <sub>0</sub> <sup>*</sup> = cpk <sub>0</sub>
AskH $\wedge$ E <sub>3</sub>	B	No	×	-	√	×	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>A</sub> , pk <sub>0</sub> <sup>*</sup> $\leftarrow$ $\mathcal{A}$
AskH $\wedge$ E <sub>4</sub>	B	No	×	-	×	√	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>A</sub> , pk <sub>0</sub> <sup>*</sup> $\leftarrow$ $\mathcal{A}$
AskH $\wedge$ E <sub>5</sub>	A/B	Yes	√	×	×	√	Adv <sub>2KEM</sub> <sup>[·, OW-CPA]</sup> , pk <sub>0</sub> = pk <sub>0</sub> (sid*) pk <sub>1</sub> <sup>*</sup> $\in$ [L <sub>1</sub> ] <sub>1</sub>
AskH $\wedge$ E <sub>6</sub>	A/B	Yes	×	√	√	×	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>A</sub> , pk <sub>0</sub> <sup>*</sup> $\in$ [L <sub>0</sub> ] <sub>1</sub>
AskH $\wedge$ E <sub>7-1</sub>	A	Yes	√	×	√	×	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>B</sub> , pk <sub>0</sub> <sup>*</sup> = cpk <sub>0</sub>
AskH $\wedge$ E <sub>7-2</sub>	B	Yes	×	√	×	√	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>A</sub> , pk <sub>0</sub> <sup>*</sup> $\in$ [L <sub>0</sub> ] <sub>1</sub>
AskH $\wedge$ E <sub>8-1</sub>	A	Yes	×	√	×	√	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>A</sub> , pk <sub>0</sub> <sup>*</sup> $\in$ [L <sub>0</sub> ] <sub>1</sub>
AskH $\wedge$ E <sub>8-2</sub>	B	Yes	√	×	√	×	Adv <sub>2KEM</sub> <sup>[OW-CCA, ·]</sup> , pk <sub>1</sub> = pk <sub>B</sub> , pk <sub>0</sub> <sup>*</sup> = cpk <sub>0</sub>

**Table 4.** The bounds of AskH  $\wedge$  Askh in the proof of Lemma 2. Refer Table 2 for the meanings of notions.

Please refer Appendix A for the formal proof. we give a sketch of proof here. In the following, to bound  $\Pr[\text{AskH}]$ , we work with the events listed in Table 4.

Due to the [OW-CCA, ·] security of 2KEM with  $pk_1 = pk_A$  and  $pk_0^*$  generated by  $\mathcal{A}$ , the probability of events AskH  $\wedge$  E<sub>3</sub> and AskH  $\wedge$  E<sub>4</sub> is negligible; Due to the [OW-CCA, ·] security of KEM with  $pk_1 = pk_B$  and  $pk_0^* = cpk_0$ , the probability of events AskH  $\wedge$  E<sub>1</sub>, AskH  $\wedge$  E<sub>2</sub>, AskH  $\wedge$  E<sub>7-1</sub> and AskH  $\wedge$  E<sub>8-2</sub> is negligible; Due to the [OW-CCA, ·] security of 2KEM with  $pk_1 = pk_A$  and  $pk_0^* \in [L_0]_1$ , the probability of events AskH  $\wedge$  E<sub>6</sub>, AskH  $\wedge$  E<sub>7-2</sub> and AskH  $\wedge$  E<sub>8-1</sub> is negligible. Due to the [·, OW-CPA] security with  $pk_1^* \in [L_1]_1$ , the probability of event AskH  $\wedge$  E<sub>5</sub> is negligible.

Here, we only take AskH  $\wedge$  E<sub>3</sub> as an example to explain in detail. For the other cases we can deal with them in a similar way. In the event E<sub>3</sub>, the test session sid\* has no matching session, and the ephemeral secret keys  $r_B$  of  $U_B$  is given to  $\mathcal{A}$ . In case of AskH  $\wedge$  E<sub>3</sub>, the [OW-CCA, ·] adversary  $\mathcal{S}$  performs as follows. It simulates the CK<sup>+</sup> games, and transfers the probability that the event AskH performed by  $\mathcal{A}$  occurs to the advantage of attacking [OW-CCA, ·] security.

In order to simulate the random oracles,  $\mathcal{S}$  maintains two lists for  $H$  oracle and SessionKeyReveal respectively.  $H$ -oracle and SessionKeyReveal are related, which means the adversary may ask SessionKeyReveal without the encapsulated keys at first, and then may ask  $H$ -oracle with the encapsulated keys. Thus, the reduction must ensure consistency with the random oracle queries to  $H$  and SessionKeyReveal. The decryption oracle for [OW-CCA, ·] game would help to maintain the consistency of  $H$ -oracle and SessionKeyReveal.

On receiving the public key  $pk_1$  from the [OW-CCA, ·] challenger, to simulate the CK<sup>+</sup> game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and the  $i$ -th session as a guess of the test session with success probability  $1/N^2l$ .  $\mathcal{S}$ , picks one preset  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  as public parameters, runs KeyGen1 to set all the static secret and public key pairs  $(pk_P, sk_P)$  for all  $N$  users  $U_P$  except for  $U_A$ . Specially,  $\mathcal{S}$  sets the static secret and public key pairs  $(pk_B, sk_B)$  for  $U_B$ , and sets  $pk_A = pk_1$ .

Without knowing the secret key of  $U_A$ ,  $\mathcal{S}$  chooses totally random  $r_A$  as part of ephemeral secret key and totally random  $R_A$  for Encaps. Since  $f_A$  is  $(\varepsilon_1, \varepsilon_2)$  hl-RF, the difference between simulation with modification of  $r_A$  and real game is bounded by  $\varepsilon_1$ . When a ephemeral public key  $pk_{P0}$  is needed,  $\mathcal{S}$  queries  $(pk_0^i, sk_0^i, r_0^i) \leftarrow \mathcal{O}_{\text{leak}_0}$  and sets  $pk_{P0} = pk_0^i$ . When a session state revealed to a session owned by  $U_A$ , is queried,  $\mathcal{S}$  returns  $r_A$  and  $r_0^i$  of this session as part of ephemeral secret key.

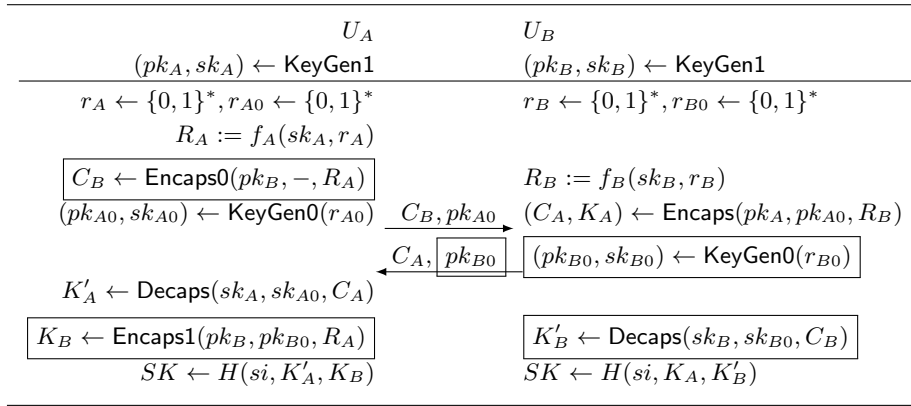
On receiving the  $i$ -th session  $(C'_B, pk_0^*)$  from  $U_A$  (that is sent by  $\mathcal{A}$  in the CK<sup>+</sup> games),  $\mathcal{S}$  returns  $pk_0^*$  to the [OW-CCA, ·] challenger and receives the challenge ciphertext  $C^*$  under public key  $pk_1$  and  $pk_0^*$ . Then  $\mathcal{S}$  returns  $C^*$  to  $U_A$  as the response of  $i$ -th session from  $U_B$ .  $\mathcal{S}$  chooses a totally independent randomness  $r_B$  as the ephemeral secret key of  $U_B$  for  $C^*$  and leaks it to adversary  $\mathcal{A}$ . Since  $f_B$  is  $(\varepsilon_1, \varepsilon_2)$  hl-RF, the difference between simulation with modification of  $r_B$  and real game is bounded by  $\varepsilon_2$ .

$\mathcal{S}$  simulates the oracle queries of  $\mathcal{A}$  and maintains the hash lists. Specially, when AskH happens, which means  $\mathcal{A}$  poses  $(U_A, U_B, pk_A, pk_B, C'_B, pk_0^*, C^*, K_A, K_B)$  to  $H$ , where  $C'_B, pk_0^*, C^*$  are the views of the

test session and  $K_B$  is the key encapsulated in  $C'_B$ ,  $\mathcal{S}$  returns  $K_A$  as the guess of  $K^*$  encapsulated in  $C^*$ , which contradicts with the [OW-CCA, ·] security for  $pk_1 = pk_A, pk_0^* \leftarrow \mathcal{A}$ .  $\square$

**4.1.1 If 2-key KEM is PKIC.** As we notice in AKE, the session state of  $sid$  owned by  $U_B$  does not contain decapsulated key  $K'_B$ . If the underlying 2-key KEM is PKIC (which is defined in Sec. 3.3), and  $U_B$  also sends ephemeral public key  $pk_{B0}$  out in every session,  $K'_B$  is encapsulated under two public keys  $pk_B$  and  $pk_{B0}$ , then  $K'_B$  could be included in session state, and the predetermined ephemeral public key  $cpk_0$  can be omitted. Let  $2KEM_{pkic} = (\text{KeyGen1}, \text{KeyGen0}, \text{Encaps0}, \text{Encaps1}, \text{Decaps})$  be PKIC and [OW-CCA, OW-CPA] secure 2-key KEM. The AKE can be modified to include  $K'_B$  as session state by 1) replacing  $2KEM$  with  $2KEM_{pkic}$ ; 2) requiring  $U_B$  to generate a fresh  $(pk_{B0}, sk_{B0}) \leftarrow \text{KeyGen0}$  and send out ephemeral public key  $pk_{B0}$ ; 2) encapsulating and separating  $(C_B, K_B) \leftarrow \text{Encaps}(pk_B, pk_{B0}, R_A)$  in two steps and computing  $C_B \leftarrow \text{Encaps0}(pk_B, -, R_A)$  and  $K_B \leftarrow \text{Encaps1}(pk_B, pk_{B0}, R_A)$ . The modified protocol  $\text{AKE}_{ro-pkic}$  is shown in Figure 3.

Note that the encapsulation algorithm of PKIC 2-key KEM can be split into two steps. Since the generation of ciphertext  $C_B$  does not require  $pk_{B0}$ , we denote it as  $C_B \leftarrow \text{Encaps0}(pk_B, -, R_A)$ . The computation of encapsulated key  $K_B$  requires  $pk_{B0}$ , and we denote it as  $K_B \leftarrow \text{Encaps1}(pk_B, pk_{B0}, R_A)$ .



**Fig. 3.**  $\text{AKE}_{ro-pkic}$  from PKIC [OW-CCA, OW-CPA] secure 2KEM. Here  $si = (U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A, pk_{B0})$ . The boxed argument is the difference with AKE

Since the proof mainly follows that of Theorem 1, we only show the difference here. The main difference is the analysis of  $\Pr[\text{AskH}]$  in Lemma 2. Now, the probability of events  $\text{AskH} \wedge E_1, \text{AskH} \wedge E_2, \text{AskH} \wedge E_{7-1}, \text{AskH} \wedge E_{8-2}$  is bounded by the [OW-CCA, ·] security of  $2KEM_{pkic}$  with  $pk_0^*$  chosen by  $\mathcal{A}$  rather than the predetermined  $cpk_0$ . Precisely, in those events, when the adversary queries the session state of  $U_B$  whose secret key is unknown to simulator  $\mathcal{S}$ , in AKE,  $\mathcal{S}$  queries the decryption oracle of 2KEM with  $cpk_0$  and  $C_B$  (when adversary queries  $\text{Send}(\Pi, R, U_B, U_P, C_B, pk_{A0})$ ), while in  $\text{AKE}_{pkic}$ ,  $\mathcal{S}$  queries the decryption oracle of  $2KEM_{pkic}$  with  $(pk_{B0}, C_B)$  chosen by  $\mathcal{A}$ . This modification does not affect the proof of security.

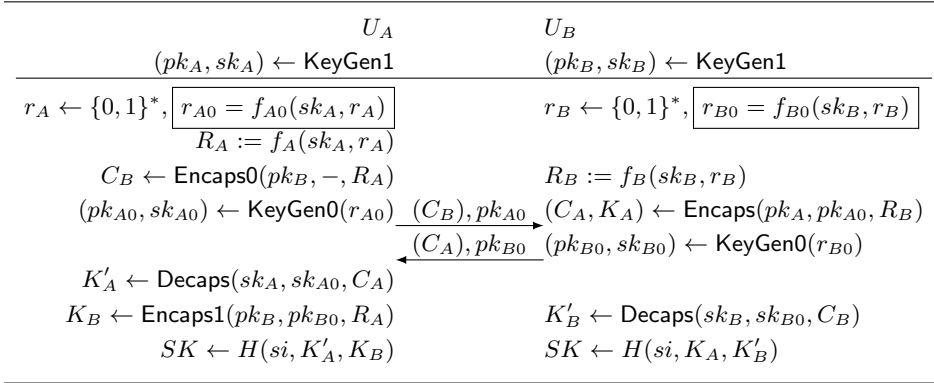
**4.1.2 If PKIC 2-key KEM is even Secure with Leakage of Partial Randomness** We can further refine the framework  $\text{AKE}_{ro-pkic}$  based on two observations: 1) From the proof of Theorem 1 (especially Lemma 2), we can see that the only purpose of  $f_A$  and  $f_B$  is to preserve the [OW-CCA, ·] security with  $pk_1 = pk_A$  and the [·, OW-CPA] security with  $pk_0 = pk_{A0}$  even if part of randomness,  $r_B$  or  $sk_B$  is leaked to the adversary. If the underlying 2-key KEM itself is strong enough to preserve the [OW-CCA, OW-CPA] security with respect to *some* function  $f_A(sk_A, r_A)$  (resp.  $f_B(sk_B, r_A)$ ), and leakage of  $sk_A$  or  $r_A$  for fixed  $pk_A$  (resp.  $sk_B$  or  $r_B$  for fixed  $pk_B$ ), the functions  $f_A$  and  $f_B$  don't have to be

hl-RFs. 2) if the 2-key KEM is strong enough to preserve security even when the randomness  $r_{B0}$  used to generate  $pk_{B0}$  is generated from  $f_{B0}(sk_B, r_B)$  for some function  $f_{B0}$ , then we could regard  $f_{B0}(sk_B, r_B)$  as a random string using to compute  $pk_{B0}$ . The same holds when  $(pk_{A0}, sk_{A0}) \leftarrow \text{KeyGen0}(r_{A0})$  where  $r_{A0} = f_{A0}(sk_A, r_A)$  for some function  $f_{A0}$ .

Therefore, the problem comes down to study the security of 2-key KEM when  $C_A$  (under public keys  $pk_A$  and  $pk_{A0}$ ) shares the randomness of  $pk_B$  and  $pk_{B0}$ .

**Definition 3.** We say 2-key KEM is leakage resistant of partial randomness with respect to  $f_B$  and  $f_{B0}$  (they need not to be hl-RFs), if the following property holds. Under public key  $pk_A$  and  $pk_{A0}$ , the [OW-CCA, OW-CPA] security still holds where the ciphertext is computed as  $\text{Encaps}(pk_A, pk_{A0}, f_B(sk_B, r_B))$  for some fixed  $pk_B$  (where  $(pk_B, sk_B) \leftarrow \text{KeyGen1}$ ), when either  $r_B$  and  $pk_{B0}$  or  $sk_B$  and  $pk_{B0}$  are given to adversary, where  $(pk_{B0}, sk_{B0}) \leftarrow \text{KeyGen0}(f_{B0}(sk_B, r_B))$ .

Equipped with PKIC 2-key KEM that resists to the leakage of partial randomness with respect to  $f_B$  and  $f_{B0}$ , we set  $f_{A0}(sk_A, r_A)$  and  $f_{B0}(sk_B, r_B)$  as the randomness for  $\text{KeyGen0}$ , and denote the result AKE as  $\text{AKE}_{\text{ro-pkic-lr}}$  in Figure 4. The session state of  $\text{sid}$  owned by  $U_A$  consists of  $r_A$ ,  $K'_A$  and  $K_B$ , the session state of  $\text{sid}$  owned by  $U_B$  consists of  $r_B$ ,  $K_A$  and  $K'_B$ .



**Fig. 4.**  $\text{AKE}_{\text{ro-pkic-lr}}$ . Here  $\text{sid} = (U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A, pk_{B0})$ . The boxed argument is the main difference with  $\text{AKE}_{\text{ro-pkic}}$

**Remark 1:** As in the definition of 2-key KEM, both  $\text{Encaps}$  and  $\text{Decaps}$  allow to have auxiliary input  $\text{aux}_e$  or  $\text{aux}_d$ . In  $\text{AKE}_{\text{ro-pkic-lr}}$  (AKE and  $\text{AKE}_{\text{ro-pkic}}$ ), the static public keys are generated by  $\text{KeyGen1}$  during the registration phase (i.e., Stage 0) and publicly available. Thus, in the protocol, it makes sense that  $\text{Encaps}$  and  $\text{Decaps}$  algorithms take the static public keys as public auxiliary input. And for user  $U_A$  (resp.  $U_B$ ), it is also reasonable that  $\text{Encaps}$  executed by  $U_A$  (resp.  $U_B$ ) takes his static secret key  $sk_A$  (resp.  $sk_B$ ) as auxiliary input. In this sense, one couple of 2KEM is really “coupled” with each other.

**Remark 2:** Since  $C_A$  share the randomness of  $pk_{B0}$  and secret key of  $pk_B$ , if the 2-key KEM and function  $f_B/f_{B0}$  further satisfy that  $C_A$  is publicly computable from  $pk_B$  and  $pk_{B0}$ , we can omit  $C_A$  in the communications. The same holds for  $C_B$ , if it is publicly computable from  $pk_A$  and  $pk_{A0}$ , we can omit  $C_B$ .

**Remark 3:** Note that the computation of  $f_B$  is part of  $\text{Encaps}(pk_A, pk_{A0}, R_B)$  algorithm.  $f_B$  may take  $pk_A$  as input. At that time, to be clear, we denote  $f_B(sk_B, r_B)$  as  $f_B(sk_B, r_B, pk_A)$ . It is similar in the case of  $f_A$ .

With these modifications, we should handle the proof more carefully. The main challenge is that the ciphertext  $C_A$ , static public key  $pk_B$ , ephemeral public key  $pk_{B0}$  are correlated (the same holds for  $C_B$ ,  $pk_A$ , and  $pk_{A0}$ ). We should handle the problem that, since  $C_A$  shares the randomness with  $pk_{B0}$  and secret key of  $pk_B$ , when applying the [OW-CCA, ·] security of 2-key KEM with  $pk_1 = pk_A$  in event

$\text{AskH} \wedge E_3, \text{AskH} \wedge E_6$ , not only  $sk_A$  but also  $sk_B$  is unknown to simulator  $\mathcal{S}$ . (The same situation occurs when applying  $[\text{OW-CCA}, \cdot]$  security of 2-key KEM with  $pk_1 = pk_B$  in event  $\text{AskH} \wedge E_2$ ).

The way to solving this problem is to bring in another  $[\text{OW-CCA}, \cdot]$  challenge. As an example, we sketch the proof of event  $\text{AskH} \wedge E_3$  to show how this resolves the above problem. The main modification is for the proof of Lemma 2. In case of  $\text{AskH} \wedge E_3$ , the  $[\text{OW-CCA}, \cdot]$  adversary  $\mathcal{S}$  performs as follows. On receiving the public key  $pk_1$  from the  $[\text{OW-CCA}, \cdot]$  challenger, to simulate the  $\text{CK}^+$  game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and the  $i$ -th session as a guess of the test session.  $\mathcal{S}$  runs  $\text{KeyGen1}$  to generate all static public keys except  $U_A$  and  $U_B$ .  $\mathcal{S}$  queries the first  $[\text{OW-CCA}, \cdot]$  challenger to get  $pk_1$ , and sets  $pk_A = pk_1$ .  $\mathcal{S}$  queries the second  $[\text{OW-CCA}, \cdot]$  challenger again to get another  $pk'_1$  and sets  $pk_B = pk'_1$ .

Note that now  $\mathcal{S}$  does not know the secret key of both  $pk_A$  and  $pk_B$ . Here  $\mathcal{S}$  generates  $(pk_{B0}^*, sk_{B0}^*)$  by itself.  $\mathcal{S}$  sends  $pk_{B0}^*$  to the second challenge to get challenge ciphertext  $C_B^*$  and keeps both  $pk_{B0}^*$  and  $C_B^*$  secret to  $\text{CK}^+$  adversary  $\mathcal{A}$ . On receiving the  $i$ -th session  $(C'_B, pk_{A0}^*)$  from  $U_A$  (that is sent by  $\mathcal{A}$  in the  $\text{CK}^+$  games),  $\mathcal{S}$  queries the first  $[\text{OW-CCA}, \cdot]$  challenger with  $pk_{A0}^*$  and obtains  $C_A^*, pk_{B0}$  and its randomness  $r_{B0}$ .  $\mathcal{S}$  returns  $C_A^*$  and  $pk_{B0}$  to  $U_A$  as the response of  $i$ -th session from  $U_B$ , and sets  $pk_{A0}^*$  as the public key under which  $C_A^*$  is encrypted.  $\mathcal{S}$  also leaks  $r_{B0}$  to adversary as the ephemeral secret key.

With the first  $[\text{OW-CCA}, \cdot]$  challenge,  $\mathcal{S}$  could partially maintain the hash list and  $\text{SessionStateReveal}$  and  $\text{SessionKeyReveal}$  with strong decapsulation oracle when  $U_B$  is not involved. When  $U_B$  is involved, the second  $[\text{OW-CCA}, \cdot]$  challenge is needed. Note that since 2-key KEM is  $\gamma$ -spread, the probability that  $\mathcal{A}$  queries a message with  $C_B = C_B^*$  is bounded by  $q \times 2^{-\gamma}$ . The simulation is perfect and the other part of proof proceeds the same with Lemma 2.

## 4.2 AKE from 2-key KEM in Standard Model

The protocol  $\text{AKE}/\text{AKE}_{\text{ro-pkic}}$  in random oracle model can be easily extended to one that is secure in the standard model, denoted by  $\text{AKE}_{\text{std}}/\text{AKE}_{\text{std-pkic}}$ , via the following steps:

1. replacing the  $[\text{OW-CCA}, \text{OW-CPA}]$  secure 2-key KEM in random oracle model with the  $[\text{IND-CCA}, \text{IND-CPA}]$  secure 2-key KEM in standard model;
2. instantiating the hl-RF functions  $f_A, f_B$  in standard model instead of the random oracle model. As noted after the definition, the instantiation of hl-RF in standard model require PRF and extra randomness. Thus every user holds extra random secret  $s_P \leftarrow \{0, 1\}^\lambda$  as part of the static secret key and  $R_A = f_A(sk_A || s_A, r_A)$ ,  $R_B = f_B(sk_B || s_B, r_B)$ .
3. replacing the random oracle  $H(si, K_A, K_B)$  with  $F_{K_A}(si) \oplus \hat{F}_{K_B}(si)$ , to extract session key, where  $F$  and  $\hat{F}$  are PRFs.

Actually, converting a scheme in the random oracle model into that in the standard model is generally not trivial, and there are many negative results. However, without taking advantage of strong property of random oracle, our step 2 and 3 just use the property that if the input is unknown then the output is totally random. The difficult part is step 1. Once the 2-key KEM in random oracle model is replaced by  $[\text{IND-CCA}, \text{IND-CPA}]$  secure 2-key KEM in standard model, the proof of security for AKE in standard model is straightforward.

## 5 Unification of Prior Works

In this section, we show that existing AKEs, including  $\text{HMQV}$ [24],  $\text{NAXOS}$  [25],  $\text{Okamoto}$  [28], and  $\text{FSXY}$  framework [15, 16], can be explained in our unified frameworks.

### 5.1 $\text{HMQV-AKE}$ .

In  $\text{HMQV}$ [24], the 2-key KEM is initiated by  $2\text{KEM}_{\text{HMQV}}$  in Figure 5. Let  $h$  and  $\hat{H}$  be hash functions. Let  $G$  be a group of prime order  $p$  with  $g$  as a generator. Both  $\text{Encaps}$  and  $\text{Decaps}$  algorithms have auxiliary input  $\text{aux}_e = (B, b)$  where  $B = g^b$  and  $\text{aux}_d = B$ . Note that, here,  $B$  is the public auxiliary input and  $b$  is the secret auxiliary input. By applying  $\text{AKE}_{\text{ro-pkic-lr}}$ , Remark 1-3, we present how the  $\text{HMQV}$  scheme is integrated in our unified framework of AKE and how it is built from the view of 2-key KEM in Figure 6.

KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )	Encaps( $pk_1, pk_0; \text{aux}_e(B, b)$ )	Decaps( $sk_1, sk_0, c; \text{aux}_d(B)$ )
$a \leftarrow \mathbb{Z}_p;$ $A = g^a$ $pk_1 = A$ $sk_1 = a$	$x \leftarrow \mathbb{Z}_p$ $X = g^x$ $pk_0 = X;$ $sk_0 = x.$	$y \leftarrow \mathbb{Z}_p, Y = g^y,$ $e = h(Y, A), d = h(X, B)$ $k = \hat{H}((XA^d)^{y+eb})$ Return $k, c = YB^e.$	$YB^e \leftarrow c;$ $e = h(Y, A), d = h(X, B)$ $k' = \hat{H}((YB^e)^{x+da})$ Return $k'$

**Fig. 5.** The [OW-CCA, OW-CCA] secure  $2\text{KEM}_{\text{HMQV}}$  implied by  $\text{HMQV}$ .

**Theorem 2.** *Under the Gap-DH and KEA1 assumptions<sup>2</sup>,  $2\text{KEM}_{\text{HMQV}}$  in Figure 5 is [OW-CCA, OW-CCA] secure with the resistance to the leakage of partial randomness with respect to  $f_B(b, y, A) = y + b \cdot h(g^y, A)$  and  $f_{B_0}(b, y) = y$  in the random oracle model.*

Please refer Appendix B for the proof of Theorem 2.

As said in Remark 3,  $f_B$  takes  $A$  as input and  $f_B(b, y, A) = y + b \cdot h(g^y, A)$ . By theorem 2,  $2\text{KEM}_{\text{HMQV}}$  is [OW-CCA, OW-CCA] secure even if partial randomness ( $b$  or  $y$ ) is leaked with respect to  $f_B(b, y, A) = y + b \cdot h(g^y, A)$  and  $f_{B_0}(b, y) = y$ . By changing the role of  $A$  and  $B$ ,  $X$  and  $Y$ , we also get a dual scheme of  $2\text{KEM}_{\text{HMQV}}$ , with respect to  $f_A(a, x, B) = x + a \cdot h(g^x, B)$  and  $f_{A_0}(a, x) = x$ . Obviously,  $2\text{KEM}_{\text{HMQV}}$  is PKIC, which means that the ciphertext is independent of the public key  $pk_0$ . Thus the Encaps algorithm can be split into two steps Encaps0 and Encaps1. However, when integrating  $2\text{KEM}_{\text{HMQV}}$  into  $\text{AKE}_{\text{ro-pkic-lr}}$  to reproduce  $\text{HMQV}$ , one may doubt that whether  $\text{aux}_e = (B, b)$  or  $(A, a)$  required by Encaps and  $\text{aux}_d = B$  or  $A$  required by Decaps influence the reconstruction. As explained in Remark 2, since  $B$  and  $A$  are the static public keys and generated during the registration phase, they can be used as the public auxiliary input by any user during the execution phase. As a static secret key,  $b$  can be used by  $U_B$  as secret auxiliary input during the execution phase. Based on the above analysis, applying  $\text{AKE}_{\text{ro-pkic-lr}}$  and Remark 1-3 to  $2\text{KEM}_{\text{HMQV}}$ ,  $\text{HMQV}$  is reconstructed in Fig. 6.

Moreover,  $A, B$  are static public keys, and  $d, e$  are publicly computable,  $C_A, C_B$  can be publicly computed from  $pk_{B_0} = Y$  and  $pk_{A_0} = X$ . Thus, we can apply Remark 1 to omit  $C_B = XA^d$  and  $C_A = YB^e$  in the communications.

$U_A : A = g^a, a$	$U_B : B = g^b, b$
$x \leftarrow \mathbb{Z}_p, X = g^x$	$y \leftarrow \mathbb{Z}_p, Y = g^y$
$d = h(X, B), C_B = XA^d$	$e = h(Y, A), C_A = YB^e$
$e = h(Y, A)$	$d = h(X, B)$
$K_B = K'_A = \hat{H}((YB^e)^{x+ad})$	$K_A = K'_B = \hat{H}((XA^d)^{y+be})$
$SK \leftarrow H(si, K_B)$	$SK \leftarrow H(si, K_A)$

**Fig. 6.** Understanding  $\text{HMQV}$  with  $2\text{KEM}_{\text{HMQV}}$  in the frame  $\text{AKE}_{\text{ro-pkic-lr}}$  where  $si = (U_A, U_B, A, B, C_B, X, C_A, Y)$ .

## 5.2 NAXOS-AKE.

In [25], the 2-key KEM is initiated by  $2\text{KEM}_{\text{NAXOS}}$  in Figure 7. Let  $G$  be a group of prime order  $p$  with  $g$  as a generator. Let  $h : \mathbb{Z}_p \times \mathbb{Z}_p \rightarrow \mathbb{Z}_p$  and  $\hat{H} : \mathbb{Z}_p \times \mathbb{Z}_p \rightarrow \{0, 1\}^\lambda$  be hash functions. By applying  $\text{AKE}_{\text{ro-pkic-lr}}$  and Remark 1-2, in Figure 8, we present how the NAXOS scheme is integrated in our unified framework of AKE and how it is built from the view of 2-key KEM.

**Theorem 3.** *Under the Gap-DH assumption,  $2\text{KEM}_{\text{NAXOS}}$  is [OW-CCA, OW-CCA] secure even with the leakage of one of  $y_0$  and  $b$  where  $f_B(b, y_0) = h(b, y_0)$  and  $f_{B_0}(b, y_0) = h(b, y_0)$  in the random oracle model.*

Please refer Appendix C for the formal proof of Theorem 3.

<sup>2</sup> For formal definitions of Gap-DH and KEA1 assumptions, please refer  $\text{HMQV}$ .



KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )	Encaps( $pk_1, pk_0; \text{aux}_e(B, b)$ );	Decaps( $sk_1, sk_0, c$ )
$a \leftarrow \mathbb{Z}_p$ ; $A = g^a$	$x \leftarrow \mathbb{Z}_p$ $X = g^x$	$y_0 \leftarrow \mathbb{Z}_p, y = h(y_0, b)$ $Y = g^y$	$Y \leftarrow c$ ; $x = h(x_0, a)$
$pk_1 = A$ $sk_1 = a$	$pk_0 = X$ ; $sk_0 = x$	$k = \hat{H}(A^y, X^y)$ Return $k, c = Y$ .	$k' = \hat{H}(Y^a, Y^x)$ Return $k'$

**Fig. 7.** The [OW-CCA, OW-CCA] secure  $2\text{KEM}_{\text{NAXOS}}$  implied by NAXOS.

By theorem 3,  $2\text{KEM}_{\text{NAXOS}}$  is [OW-CCA, OW-CCA] secure even if partial randomness ( $b$  or  $y_0$ ) is leaked with respect to  $f_B(b, y_0) = h(b, y_0)$  and  $f_{B0}(b, y_0) = h(b, y_0)$ . Obviously,  $2\text{KEM}_{\text{NAXOS}}$  is PKIC. We split Encaps algorithm into two steps Encaps0 and Encaps1. As explained in Remark 2, since  $b$  is static secret key and generated by  $U_B$ , in the execution phase  $U_B$  takes it as secret auxiliary input. Based on the above analysis, applying  $\text{AKE}_{\text{ro-pkic-lr}}$  and Remark 1-2 to  $2\text{KEM}_{\text{NAXOS}}$ , NAXOS is reconstructed in Fig. 8.

Moreover,  $C_A$  is equal to  $pk_{B0} = Y$  and  $C_B$  is equal to  $pk_{A0} = X$ . Thus we can apply Remark 2 to omit  $C_B = X$  and  $C_A = Y$  in the communications.

$U_A : A = g^a, a$	$U_B : B = g^b, b$
$x_0 \leftarrow \mathbb{Z}_p, x = h(x_0, a)$	$y_0 \leftarrow \mathbb{Z}_p, y = h(y_0, b)$
$C_B = pk_{A0} = X = g^x$	$C_A = pk_{B0} = Y = g^y$
$K_B = \hat{H}(B^x, Y^x)$	$K_A = \hat{H}(A^y, X^y)$
$K'_A = \hat{H}(Y^a, Y^x)$	$K'_B = \hat{H}(Y^b, X^y)$
$SK \leftarrow H(si, K'_A, K_B)$	$SK \leftarrow H(si, K_A, K'_B)$

**Fig. 8.** Understanding NAXOS with  $2\text{KEM}_{\text{naxos}}$  in the frame  $\text{AKE}_{\text{ro-pkic-lr}}$  where  $si = (U_A, U_B, A, B, X, Y)$ .

### 5.3 Okamoto-AKE.

In Okamoto-AKE [28], the 2-key KEM is initiated by  $2\text{KEM}_{\text{Oka}}$  in Figure 9. In  $2\text{KEM}_{\text{Oka}}$ , the computation is proceeded over group  $G$  of prime order  $p$  with generator  $g$ ,  $h_{\text{tcr}}$  is a target-collision resistant (TCR) hash function and  $\bar{F}$  is a pairwise-independent random source PRF. (Please refer [28] for the formal definition of pairwise-independent random source PRFs.)

$2\text{KEM}_{\text{Oka}}.\text{KeyGen1}(\lambda)$	$2\text{KEM}_{\text{Oka}}.\text{KeyGen0}(\lambda)$
$a_1, a_2, a_3, a_4 \leftarrow \mathbb{Z}_p^A, A_1 = g_1^{a_1} g_2^{a_2}, A_2 = g_1^{a_3} g_2^{a_4}$	$x_3 \leftarrow \mathbb{Z}_p, X_3 = g_1^{x_3}$
$pk_1 = (A_1, A_2), sk_1 = (a_1, a_2, a_3, a_4)$	$pk_0 = X_3, sk_0 = x_3$
$2\text{KEM}_{\text{Oka}}.\text{Encaps}(pk_0, pk_1)$ ;	$2\text{KEM}_{\text{Oka}}.\text{Decaps}(sk_0, sk_1, C)$
$y, y_3 \leftarrow \mathbb{Z}_p^2, Y_1 = g_1^y, Y_2 = g_2^y, Y_3 = g_1^{y_3}$	$C \in G^3, (Y_1, Y_2, Y_3) \leftarrow C$ ;
$C = (Y_1, Y_2, Y_3), c = h_{\text{tcr}}(A_1, A_2, C)$	$c = h_{\text{tcr}}(A_1, A_2, C)$
$\sigma = X_3^{y_3} (A_1 A_2^2)^y$	$\sigma' = Y_3^{x_3} Y_1^{a_1 + ca_3} Y_2^{a_2 + ca_4}$
$K = \bar{F}_\sigma(pk_0, C)$	$K' = \bar{F}_{\sigma'}(pk_0, C)$

**Fig. 9.** The [IND-CCA, IND-CPA] secure  $2\text{KEM}_{\text{Oka}}$  implied by Okamoto-AKE.

Let  $G$  be a group of order  $p$  with the generator  $g$ . Let  $1_G = g^p$  be the identity element. The DDH assumption states that  $\{(G, g^a, g^b, g^{ab})\}_\lambda$  is computationally indistinguishable from  $\{(G, g^a, g^b, g^c)\}_\lambda$ , where  $a, b, c$  are randomly and independently chosen in  $\mathbb{Z}_p$ . If  $c = ab$ ,  $(g, g^a, g^b, g^c)$  is called a DDH tuple, otherwise it's called a non-DDH tuple. Denote the advantage of any PPT algorithm  $\mathcal{B}$  solving DDH problem as  $\text{Adv}_{\mathcal{B}}^{\text{ddh}} = |\Pr[\mathcal{B}(g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{B}(g^a, g^b, g^c) = 1]|$ .

**Theorem 4.** *Under the DDH assumption, if  $h_{\text{tcr}}$  is a TCR hash function and  $\bar{F}$  is a pairwise-independent random source PRF, then  $2\text{KEM}_{\text{Okamoto}}$  in Figure 9 is [IND-CCA, IND-CPA] secure in the standard model.*

Please refer Appendix D for the formal proof of Theorem 4.

By applying  $\text{AKE}_{\text{std}}$ , in Figure 10, we present how the Okamoto scheme is integrated in our unified framework of AKE and how it is built from the view of 2-key KEM. Let  $F' : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \mathbb{Z}_p$  and  $F'' : \mathbb{Z}_p \times \{0, 1\}^\lambda \rightarrow \mathbb{Z}_p$  be PRFs. In the frame of  $\text{AKE}_{\text{std}}$ , by setting  $s_A = a_0, s_B = b_0, r_A = x'_1 || x'_2, r_{A0} = x_3, r_B = y'_1 || y'_2$ , choosing  $cpk_0 = 1_G, csk_0 = p$ , initiating  $f_A$  and  $f_B$  as  $F'_{x'_1}(1^k) \oplus F''_{\sum_0^4 a_i}(x'_2)$  and  $F'_{y'_1}(1^k) \oplus F''_{\sum_0^4 b_i}(y'_2)$ , and applying  $2\text{KEM}_{\text{Okamoto}}$  as 2-key KEM, we will get Okamoto AKE in Fig. 10.

$U_A : A_1, A_2, a_1, a_2, a_3, a_4, a_0 \leftarrow \mathbb{Z}_p$	$U_B : B_1, B_2, b_1, b_2, b_3, b_4, b_0 \leftarrow \mathbb{Z}_p$
$x'_1, x'_2 \leftarrow \{0, 1\}^\lambda$	$y'_1, y'_2 \leftarrow \{0, 1\}^\lambda$
$(x, x_3) = F'_{x'_1}(1^k) + F''_{\sum_0^4 a_i}(x'_2)$	$(y, y_3) = F'_{y'_1}(1^k) + F''_{\sum_0^4 b_i}(y'_2)$
$X_1 = g_1^x, X_2 = g_2^x, X_3 = g_1^{x_3}$	$Y_1 = g_1^y, Y_2 = g_2^y, Y_3 = g_1^{y_3}$
$C_B = (X_1, X_2, 1_G), pk_{A0} = X_3$	$C_A = (Y_1, Y_2, Y_3)$
$d = h_{\text{tcr}}(U_B, X_1, X_2)$	$c = h_{\text{tcr}}(U_A, Y_1, Y_2, Y_3)$
$\sigma_B = (B_1 B_2^d)^x, K_B = \bar{F}_{\sigma_B}(1_G, C_B)$	$\sigma_A = X_3^{y_3} (A_1 A_2^c)^y, K_A = \bar{F}_{\sigma_A}(X_3, C_A)$
$c = h_{\text{tcr}}(U_A, C_A)$	$d = h_{\text{tcr}}(U_B, C_B)$
$\sigma'_A = X_3^{y_3} Y_1^{a_1 + ca_3} Y_2^{a_2 + ca_4}$	$\sigma'_B = X_1^{b_1 + db_3} X_2^{b_2 + db_4}$
$K'_A = \bar{F}_{\sigma'_A}(X_3, C_A)$	$K'_B = \bar{F}_{\sigma'_B}(1_G, C_B)$
$SK \leftarrow F_{K_B}(si) \oplus \hat{F}_{K'_A}(si)$	$SK \leftarrow F_{K'_B}(si) \oplus \hat{F}_{K_A}(si)$

**Fig. 10.** Understanding Okamoto-AKE from  $2\text{KEM}_{\text{Okamoto}}$  where  $si = (U_A, U_B, C_B, X_3, C_A)$  in frame  $\text{AKE}_{\text{std}}$ . Some notions are borrowed from  $2\text{KEM}_{\text{Okamoto}}$

#### 5.4 FSXY12-AKE and FSXY13-AKE.

Fujioka *et al.* in PKC 12 (called FSXY12 [15]) proposed a construction of AKE from IND-CCA secure KEM and IND-CPA secure KEM in the standard model. In FSXY12 [15],  $U_B$  sends a ciphertext of IND-CCA secure KEM and a ciphertext of IND-CPA secure KEM, and the session key is computed from these two encapsulated keys, public key of IND-CPA secure KEM, and ciphertext in the PRF functions. As we point out in section 6.1, the FSXY12 scheme implies a trivial [IND-CCA, IND-CPA] secure 2-key KEM from the improved KEM combiner in the standard model. More precisely, in  $\text{AKE}_{\text{std}}$ ,  $cpk_0$  and  $csk_0$  is set to be empty;  $C_B$  is just  $c_{B1} || -$ , where  $c_{B1}$  is the ciphertext of IND-CCA secure one-key KEM under  $pk_B$ ;  $C_A$  is replaced by the concatenation of  $c_{A1} || c_{A0}$ , where  $c_{A1}$  is the ciphertext of IND-CCA secure one-key KEM under  $pk_A$  with encapsulated key  $k_{A1}$  and  $c_{A0}$  is the ciphertext of IND-CPA secure one-key KEM under  $pk_{A0}$  with encapsulated key  $k_{A0}$ ; and  $K_A$  is replaced by  $F_{k_{A1}}(pk_{A0}, c_{A1} || c_{A0}) \oplus F_{k_{A0}}(pk_{A0}, c_{A1} || c_{A0})$ . To make it clearer, in section 6.1 we explain why we should put public key in PRFs when combining two KEMs. Note that FSXY12 implicitly did it in the same way by putting  $sid$  in PRF. Thus, due to this observation, our frame of  $\text{AKE}_{\text{std}}$  with improved KEM combiner can be used to explain the FSXY12 scheme.

Considering efficiency, Fujioka *et al.* in AsiaCCS 13 (called FSXY13 [16]) proposed AKE from OW-CCA secure KEM and OW-CPA secure KEM in the random oracle model. In FSXY13 [16],  $U_B$  sends a ciphertext of OW-CCA secure KEM and a ciphertext of OW-CPA secure KEM. The session key is computed from these two encapsulated keys, public key of CPA secure KEM, and ciphertext in the hashing step. As we point out in section 6.1, the FSXY13 scheme implies a trivial [OW-CCA, OW-CPA] secure 2-key KEM from the improved KEM combiner in the random oracle model. Precisely, in AKE,  $cpk_0$

and  $csk_0$  is set to be empty;  $C_B$  is just  $c_{B1}||-$ , where  $c_{B1}$  is the ciphertext of OW-CCA secure one-key KEM under  $pk_B$ ;  $C_A$  is replaced by the concatenation of  $c_{A1}||c_{A0}$ , where  $c_{A1}$  is the ciphertext of OW-CCA secure one-key KEM under  $pk_A$  with encapsulated key  $k_{A1}$  and  $c_{A0}$  is the ciphertext of OW-CPA secure one-key KEM under  $pk_{A0}$  with encapsulated key  $k_{A0}$ ; and  $K_A$  is replaced by  $\hat{H}(pk_{A0}, k_1||k_{A0}, c_{A1}||c_{A0})$ . In section 6.1 we explain why we should put public key in hashing step when combining two KEMs. Note that FSXY13 implicitly did it in the same way by putting  $sid$  in hashing step. Thus, our frame of AKE with improved KEM combiner works for explaining the FSXY13 scheme.

## 6 More General Constructions for 2-key KEM

As shown in Section 5, many widely-used AKEs are able to imply 2-key KEM. And over cyclic group, HMQV and NAXOS consume the least amount of communication. However, their techniques are not compatible with lattice assumptions. Although Zhang *et al.* [36] extend HMQV-type AKE to that based on Ring LWE, the resulted AKE only achieves BR security and costs more communications. In this section we investigate how to improve the KEM combiner [17] and Fujisaki-Okamoto transformation [14, 18] so as to yield more general constructions of 2-key KEM, which are much more well-suited for lattice assumptions.

### 6.1 Improved Combiner of Two KEMs

Giacon *et. al* [17] propose two KEM combiner and yield a new single KEM that is classical CCA secure as long as one of the ingredient KEMs is. We show that the simple KEM combiner does not work for our 2-key KEM. Furthermore, we show that with a slight but vital modification the combiner could work.

**6.1.1 The failure to imply [OW-CCA, ·] secure 2key KEM from KEM combiner** We give a scheme that is a CCA secure two KEM combiner but is not [OW-CCA, ·] secure.

Let  $h$  and  $H$  be hash functions. Let  $G = \langle g \rangle$  be a group with prime order  $p$ . Let  $pk_1 = (g_1, g_2 = g_1^a)$ ,  $sk_1 = a$ , the ciphertext be  $c_1 = (g_1^r, g_2^r \cdot m)$  where  $r = h(m)$  and the encapsulated key be  $k_1 = H(m)$ . By the FO transformation [14] and DDH assumption, the first KEM is one-way-CCA secure. Let  $pk_0 = (h_1, h_2 = h_1^b)$ ,  $sk_0 = b$ , the ciphertext be  $c_0 = h_1^x$  and the encapsulated key be  $k_0 = H(h_2^x)$ ; and obviously the second KEM is IND-CPA secure.

Let the combined ciphertext be  $(c_1||c_0)$  and combined encapsulated key be  $K = \hat{H}(k_1||k_0, c_1||c_0)$ , by the KEM combiner [17] (Lemma 6 and example 3 in [17]), the combined KEM is CCA secure. However, such combined KEM is not [OW-CCA, ·] secure which means there exists an adversary  $\mathcal{A}$  that can break [OW-CCA, ·] game.

Note that  $c_0 = h_1^x$  encapsulates the key  $k_0^* = H(h_2^x)$  under public key  $pk_0 = (h_1, h_2)$  while it encapsulates the same key  $k_0^* = H(h_2^x)$  under public key  $pk_0 = (h_1^c, h_2^c)$  for some  $c \in \mathbb{Z}_p$ . The [OW-CCA, ·] adversary  $\mathcal{A}$  first queries the  $\mathcal{O}_{\text{leak}}$  oracle and gets  $pk_0 = (h_1, h_2)$ . Then it randomly chooses  $c \in \mathbb{Z}_p$  and sets  $pk_0^* = (h_1^c, h_2^c)$ . After receiving  $c_1^*||c_0^*$  under public keys  $pk_1$  and  $pk_0^*$ ,  $\mathcal{A}$  queries the decryption oracle with  $(pk_1, pk_0, c_1^*||c_0^*)$ , and would receive exactly  $K^* = \hat{H}(k_1^*||k_0^*, c_1^*||c_0^*)$ .

**6.1.2 Improvement on KEM combiner to achieve [CCA, CPA] secure 2-key KEM** Inspired by the attacks above, we propose a improved combiner of CCA secure and CPA secure KEMs to achieve [CCA, CPA] secure 2-key KEM. Let  $\text{KEM}_{\text{cca}} = (\text{KeyGen}_{\text{cca}}, \text{Encaps}_{\text{cca}}, \text{Decaps}_{\text{cca}})$  be IND-CCA secure KEM,  $\text{KEM}_{\text{cpa}} = (\text{KeyGen}_{\text{cpa}}, \text{Encaps}_{\text{cpa}}, \text{Decaps}_{\text{cpa}})$  be IND-CPA secure KEM. Let  $\hat{H}$  be a hash function and  $F$  be a PRF. The improved combiner is shown in Figure 11, where function  $f(pk_0, k_1||k_0, c)$  can be initiated by  $\hat{H}(pk_0, k_1||k_0, c)$  or  $F_{k_1}(pk_0, c) \oplus F_{k_0}(pk_0, c)$ . Our main modification is to take public key as input to the hash function or PRF when generating encapsulated key.

**Theorem 5.** *Let the underlying two KEMs be IND-CCA and IND-CPA secure. If  $f(pk_0, k_1||k_0, c) = \hat{H}(pk_0, k_1||k_0, c)$  for a hash function  $\hat{H}$ ,  $2\text{KEM}_f$  in Figure 11 is [OW-CCA, OW-CCA] secure in random oracle model; if  $f(pk_0, k_1||k_0, c) = F_{k_1}(pk_0, c) \oplus F_{k_0}(pk_0, c)$  for PRF  $F$ ,  $2\text{KEM}_f$  in Figure 11 is [IND-CCA, IND-CPA] secure in standard model.*

KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )	Enc( $pk_1, pk_0$ );	Dec( $sk_1, sk_0, c_1    c_0$ )
$(pk_1, sk_1) \leftarrow$ KeyGen <sub>cca</sub>	$(pk_0, sk_0) \leftarrow$ KeyGen <sub>cpa</sub>	$(c_1, k_1) \leftarrow$ Encaps <sub>cca</sub> ( $pk_1$ ) $(c_0, k_0) \leftarrow$ Encaps <sub>cpa</sub> ( $pk_0$ ) $c = c_1    c_0, k = f(pk_0, k_1    k_0, c)$	$k_1 \leftarrow$ Decaps <sub>cca</sub> ( $sk_1, c_1$ ) $k_0 \leftarrow$ Decaps <sub>cpa</sub> ( $sk_0, c_0$ ) $k = f(pk_0, k_1    k_0, c)$

**Fig. 11.** The [CCA, CPA] secure 2KEM<sub>f</sub> in random oracle or standard model depending on the instantiation of  $f(pk_0, k_1 || k_0, c)$ .

Please refer Appendix E for the full proof.

## 6.2 Modified FO Transformation

In this section, we investigate the constructions of passively 2-key PKE and give a modified FO transformation which can be used to transform a passively secure 2-key PKE to an adaptively secure 2-key KEM.

**6.2.1 Passively Secure 2-key PKE** As the preparation for realizing adaptively secure 2-key KEM and the modified FO transformation, similar to the notion of 2-key KEM, we can also provide the notion of 2-key (public key encryption) PKE.

Informally, a 2-key PKE 2PKE=(KeyGen0, KeyGen1, Enc, Dec) is a quadruple of PPT algorithms together with a plaintext space  $\mathcal{M}$  and a ciphertext space  $\mathcal{C}$ , where KeyGen1 outputs a pair of public and secret keys  $(pk_1, sk_1)$ , KeyGen0 outputs a pair of keys  $(pk_0, sk_0)$ , Enc( $pk_1, pk_0, m$ ) outputs the ciphertext  $C \in \mathcal{C}$ , and Dec( $sk_1, sk_0, C$ ) outputs a plaintext  $m$ . Sometimes, we explicitly add the randomness  $r$  to Enc and denote it as Enc( $pk_1, pk_0, m, r$ ). Here we only describe the [IND-CPA, IND-CPA] security game. For more concrete and full definition of 2-key PKE please refer Appendix F.

<b>Game IND-CPA on <math>pk_1</math></b>	<b>Game IND-CPA on <math>pk_0</math></b>
01 $(pk_1, sk_1) \leftarrow$ KeyGen1( $pp$ )	15 $(pk_0, sk_0) \leftarrow$ KeyGen0( $pp$ )
02 $L_0 = \{(-, -, -)\}$	16 $L_1 = \{(-, -, -)\}$
03 $(state, pk_1^*, m_1, m_1) \leftarrow \mathcal{A}_1^{\text{leak}_0}(pk_1)$	17 $(state, pk_1^*, m_0, m_1) \leftarrow \mathcal{B}_1^{\text{leak}_1}(pk_0)$
04 $b \leftarrow \{0, 1\}$ ;	18 $b \leftarrow \{0, 1\}$
05 $c^* \leftarrow$ Enc( $pk_1, pk_0^*, m_b$ );	19 $c^* \leftarrow$ Enc( $pk_1^*, pk_0, m_b$ );
06 $b' \leftarrow \mathcal{A}_2^{\text{leak}_0}(state, c^*)$	20 $b' \leftarrow \mathcal{B}_2^{\text{leak}_1}(state, c^*)$
07 return $b' \stackrel{?}{=} b$	21 return $b' \stackrel{?}{=} b$

**Fig. 12.** The [IND-CPA, ·], and [·, IND-CPA] games of 2PKE for adversaries  $\mathcal{A}$  and  $\mathcal{B}$ .

**Passively Secure twin-ElGamal from DDH assumption.** Our construction is actually a conjoined ElGamal encryption. Let's call it twin-ElGamal. The [IND-CPA, IND-CPA] secure twin-ElGamal 2PKE<sub>cpaddh</sub>=(KeyGen1, KeyGen0, Enc, Dec) is presented in detail in Figure 13.

KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )	Enc( $pk_1, pk_0, m$ );	Dec( $sk_0, sk_1, C$ )
$a_1 \leftarrow \mathbb{Z}_p, h_1 = g^{a_1};$ $pk_1 = (g, h_1), sk_1 = a_1$	$a_0 \leftarrow \mathbb{Z}_p, h_0 = g^{a_0};$ $pk_0 = (g, h_0), sk_0 = a_0$	$r_1, r_0 \leftarrow \mathbb{Z}_p$ $c = g^{r_1}, g^{r_0}, h_1^{r_1} h_0^{r_0} \cdot m$	$(c_1, c_2, c_3) \leftarrow C$ $m' = c_3 / c_1^{a_1} c_2^{a_0}$

**Fig. 13.** The [IND-CPA, IND-CPA] secure 2PKE<sub>cpaddh</sub> under DDH assumption.

**Theorem 6.** Under the DDH assumption, the twin-ElGamal 2PKE<sub>cpaddh</sub> scheme shown in Figure 13 is [IND-CPA, IND-CPA] secure.

Please refer Appendix G for the formal proof.

**6.2.2 Modified FO Transformation from Passive to Adaptive Security** In the random oracle model, the FO [14, 18] technique is able to transform a passively secure one-key encryption scheme to an adaptively secure scheme. We show that the classical FO transformation does not work for our 2-key encryption scheme. Then we show that with a slight but vital modification the FO transformation could work.

**The failure of Classical FO Transform on 2-key KEM** We give a novel twin-ElGamal scheme by injecting redundant public keys, and show that such twin-ElGamal scheme after FO transformation is still OW-CCA secure, but not [OW-CCA, ·] secure.

The  $\text{KeyGen0}$  algorithm of  $2\text{PKE}_{\text{cpaddh}}$  chooses a random  $z \leftarrow \mathbb{Z}_p$ , and sets  $pk_0 = (g, h_0, g_0 = g^z)$ ,  $sk_0 = (a_0, z)$ . The algorithm  $\text{KeyGen1}, \text{Enc}, \text{Dec}$  are the same as in  $2\text{PKE}_{\text{cpaddh}}$ . Obviously this novel twin-ElGamal scheme is IND-CPA secure under DDH assumption. Let  $2\text{PKE}_{\text{cpaddh}}^{\text{fo}}$  be the scheme by applying classical FO transform on the novel twin-Elgamal. It is OW-CCA secure. Note that the encapsulated key is  $K = H(m, c)$  where  $H$  is a hash function.

However, there exists an [IND-CCA, ·] attacker  $\mathcal{A}$  of  $2\text{PKE}_{\text{cpaddh}}^{\text{fo}}$  that works as follows:  $\mathcal{A}$  first queries the  $\mathcal{O}_{\text{leak}_0}$  and gets  $pk_0^1 = (g, h_0, g_0 = g^z)$ ,  $sk_0^1 = (a_0, z)$ . Then  $\mathcal{A}$  chooses  $g'_0 \neq g_0 \in \mathbb{G}$ , and sets  $pk_0^* = (g, h_0, g'_0)$  as challenge public key. On receiving challenge ciphertext  $c^*$  under  $(pk_1, pk_0^*)$ ,  $\mathcal{A}$  queries  $\mathcal{O}_{\text{ow-cca}}$  with  $(pk_0^1, c^*)$ . Since  $pk_0^1 \neq pk_0^*$ ,  $\mathcal{O}_{\text{ow-cca}}$  would return  $K'$ .  $\mathcal{A}$  just outputs  $K'$ . Since  $c^*$  encapsulated the same key  $K^* = H(m, c^*)$  under both public keys  $(pk_1, pk_0^1)$  and  $(pk_1, pk_0^*)$ .  $\mathcal{A}$  will succeed with probability 1.

**Modification on FO Transform to achieve [IND-CCA, IND-CCA] secure 2-key KEM from 2-key PKE** Motivated by the above attacks, we give a modified FO transform by a slight but vital modification from “Hashing” in [18] to “Hashing with public key as input”. Actually, taking the public keys as input to hash function is also motivated by the fact that: from the perspective of proof, “Hashing with public key as input” would help to preserve the consistency of strong decryption oracle and hashing list.

Since we take the decryption failure into account, let’s firstly recall and adapt the definition of correctness for decryption in [18] to our 2-key setting. When  $2\text{PKE} = 2\text{PKE}^G$  is defined with respect to a random oracle  $G$ , it is said to be  $\delta_{q_G}$ -correct if for adversary  $\mathcal{A}$  making at most  $q_G$  queries to random oracle  $G$ , it holds that  $\Pr[\text{COR-RO}^{\mathcal{A}_{2\text{PKE}}} \Rightarrow 1] \leq \delta_{q_G}$ , where the correctness game COR-RO is defined as following:  $(pk_1, sk_1) \leftarrow \text{KeyGen1}(pp)$ ,  $(pk_0, sk_0) \leftarrow \text{KeyGen0}(pp)$ ,  $m \leftarrow \mathcal{A}^{G(\cdot)}(pk_1, sk_1, pk_0, sk_0)$ ,  $c \leftarrow \text{Enc}(pk_1, pk_0, m)$ . Return  $\text{Dec}(sk_1, sk_0, c) \stackrel{?}{=} m$ .

Let  $2\text{PKE} = (\text{KeyGen1}', \text{KeyGen0}', \text{Enc}, \text{Dec})$  be a [IND-CPA, IND-CPA] secure 2-key PKE with message space  $\mathcal{M}$ . The [IND-CCA, IND-CCA] secure 2KEM =  $(\text{KeyGen1}, \text{KeyGen0}, \text{Encaps}, \text{Decaps})$  are described as in Figure 14.

KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )
$(pk'_1, sk'_1) \leftarrow \text{KeyGen1}'$ , $s_1 \leftarrow \{0, 1\}^l$ ; $sk_1 = (sk'_1, s_1)$ , $pk_1 = pk'_1$	$(pk'_0, sk'_0) \leftarrow \text{KeyGen0}'$ , $s_0 \leftarrow \{0, 1\}^l$ ; $sk_0 = (sk'_0, s_0)$ , $pk_0 = pk'_0$ ;
<b>Encaps</b> ( $pk_1, pk_0$ ); $m \leftarrow \mathcal{M}$ $c \leftarrow \text{Enc}(pk_1, pk_0, m; G(m))$ $K = H(pk_1, pk_0, m, c)$ ; return $(K, c)$	<b>Decaps</b> ( $sk_1, sk_0, c$ ) $sk_1 = (sk'_1, s_1)$ , $sk_0 = (sk'_0, s_0)$ $m' = \text{Dec}(sk'_1, sk'_0, c)$ $c' = \text{Enc}(pk_1, pk_0, m'; G(m'))$ if $m' = \perp$ or $c \neq c'$ , let $m' = s_1    s_0$ return $K = H(pk_1, pk_0, m', c)$

Fig. 14. The [IND-CCA, IND-CCA] secure 2-key KEM 2KEM by modified FO.

**Theorem 7.** For any  $[\text{IND-CCA}, \cdot]$  adversary  $\mathcal{C}$ , or  $[\cdot, \text{IND-CCA}]$  adversary  $\mathcal{D}$  against 2KEM with at most  $q_D$  queries to decapsulation oracle  $\text{DECAPS}$ ,  $q_H$  (resp.  $q_G$ ) queries to random oracle  $H$  (resp.  $G$ ), there are  $[\text{IND-CPA}, \cdot]$  adversary  $\mathcal{A}$ , or  $[\cdot, \text{IND-CPA}]$  adversary  $\mathcal{B}$  against 2PKE, that make at most  $q_H$  (resp.  $q_G$ ) queries to random oracle  $H$  (resp.  $G$ ) s.t.

$$\text{Adv}_{2\text{KEM}}^{[\text{IND-CCA}, \cdot]}(\mathcal{C}) \leq \frac{q_H}{2^l} + \frac{q_H + 1}{|M|} + q_G \cdot \delta + 4\text{Adv}_{2\text{PKE}}^{[\text{IND-CPA}, \cdot]}(\mathcal{A}).$$

Please refer Appendix H for the formal proof.

## 7 Efficient Post-quantum AKE from Module-LWE

With the above analysis and tools, we give a more compact AKE from Module-LWE assumption with less communications than Kyber [3]. The roadmap is that we first give a  $[\text{IND-CPA}, \text{IND-CPA}]$  secure 2-key PKE from Module-LWE, by applying the modified FO transform in section 6.2.2 and the AKE in section 4.1 step by step, and we finally obtain a AKE scheme.

Let  $q$  be a prime and  $R_q$  denote the ring  $\mathbb{Z}_q[x]/(x^n + 1)$ . Define the centered binomial distribution  $B_\eta$  for positive integer  $\eta$  as: sample  $(a_1, \dots, a_\eta, b_1, \dots, b_\eta)$  uniformly from  $\{0, 1\}$ , and output  $\sum_{i=1}^\eta (a_i - b_i)$ . Denote  $\mathbf{s} \leftarrow \beta_\eta$  as that each of  $\mathbf{s}$ 's coefficient is generated according to  $B_\eta$ . Let  $k, m$  be a positive integer parameter. For PPT adversary  $\mathcal{A}$ , the advantage  $\text{Adv}_{m, k, \eta}^{\text{mlwe}}(\mathcal{A})$  of solving Module-LWE problem is the advantage of distinguishing two distributions  $\{(\mathbf{A} \leftarrow R_q^{m \times k}, \mathbf{A}\mathbf{s} + \mathbf{e}) | (\mathbf{s}, \mathbf{e}) \leftarrow \beta_\eta^k \times \beta_\eta\}$  and  $\{(\mathbf{A} \leftarrow R_q^{m \times k}, \mathbf{b} \leftarrow R_q^m)\}$ .

Let  $d_{t_1}, d_{t_0}, d_{u_1}, d_{u_0}, d_v$  be positive numbers, depending on the special choice of the parameters settings, and  $n = 256$ . Every message in  $\mathcal{M} = \{0, 1\}^n$  can be seen as a polynomial in  $R_q$  with coefficients in  $\{0, 1\}$ . Let  $\mathbf{A}$  be a random  $k \times k$  matrix in  $R_q$ . Let  $\lceil x \rceil$  be the rounding of  $x$  to the closest integer. For distribution  $X$ , let  $\sim X = \text{Samp}(r)$  be sample algorithm with randomness  $r$  according to distribution  $X$ .

For an even (resp. odd) positive integer  $\alpha$ , we define  $r' = r \bmod^\pm \alpha$  to be the unique element  $r'$  in the range  $-\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$  (resp.  $-\frac{\alpha-1}{2} \leq r' \leq \frac{\alpha-1}{2}$ ) such that  $r' = r \bmod \alpha$ . For any positive integer  $\alpha$ , define  $r' = r \bmod^+ \alpha$  to be the unique element  $r'$  in the range  $0 < r' < \alpha$  such that  $r' = r \bmod \alpha$ . When the exact representation is not important, we simplify it as  $r \bmod \alpha$ . For  $x \in \mathbb{Q}$ ,  $d \leq \log_2 q$ , define the compress function as  $\text{Comp}_q(x, d) = \lceil (2^d)/q \cdot x \rceil \bmod^+ 2^d$ , and the decompress function as  $\text{Decomp}_q(x, d) = \lceil q/(2^d) \cdot x \rceil$ . And when applying the  $\text{Comp}$  and  $\text{Decomp}$  function to  $\mathbf{x}$ , the procedure is applied to coefficient.

**Twin-Kyber** Our construction, called twin-kyber, is an extension of kyber scheme [3] in the same conjoined way for our twin-ElGamal scheme. With the parameters above, twin-kyber  $2\text{PKE}_{\text{mlwe}} = (\text{KeyGen1}, \text{KeyGen0}, \text{Enc}, \text{Dec})$  is shown in Figure 15.

KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )
01 $\sigma_1 \leftarrow \{0, 1\}^{256}$	05 $\sigma_0 \leftarrow \{0, 1\}^{256}$
02 $(\mathbf{s}_1, \mathbf{e}_1) \sim \beta_\eta^k \times \beta_\eta^k = \text{Sam}(\sigma_1)$	06 $(\mathbf{s}_0, \mathbf{e}_0) \sim \beta_\eta^k \times \beta_\eta^k = \text{Sam}(\sigma_0)$
03 $\mathbf{t}_1 = \text{Comp}_q(\mathbf{A}\mathbf{s}_1 + \mathbf{e}_1, d_{t_1} = \lceil \log q \rceil)$	07 $\mathbf{t}_0 = \text{Comp}_q(\mathbf{A}\mathbf{s}_0 + \mathbf{e}_0, d_{t_0} = \lceil \log q \rceil)$
04 $(pk_1 = \mathbf{t}_1, sk_1 = \mathbf{s}_1)$	08 $(pk_0 = \mathbf{t}_0, sk_0 = \mathbf{s}_0)$
<hr/>	
$\text{Enc}(pk_1 = \mathbf{t}_1, pk_0 = \mathbf{t}_0, m \in \mathcal{M})$	$\text{Dec}(sk_1 = \mathbf{s}_1, sk_0 = \mathbf{s}_0, c = (\mathbf{u}_1, \mathbf{u}_0, v))$
09 $r', r \leftarrow \{0, 1\}^{256}$	15 $\mathbf{u}_1 = \text{Decomp}_q(\mathbf{u}_1, d_{u_1})$
10 $(\mathbf{r}_1, \mathbf{r}_0, \mathbf{e}_3, \mathbf{e}_4, e) \sim (\beta_\eta^k)^4 \times \beta_\eta = \text{Sam}(r)$	16 $\mathbf{u}_0 = \text{Decomp}_q(\mathbf{u}_0, d_{u_0})$
11 $\mathbf{u}_1 = \text{Comp}_q(\mathbf{A}^T \mathbf{r}_1 + \mathbf{e}_3, d_{u_1})$	17 $v = \text{Decomp}_q(v, d_v)$
12 $\mathbf{u}_0 = \text{Comp}_q(\mathbf{A}^T \mathbf{r}_0 + \mathbf{e}_4, d_{u_0})$	18 $m' = \text{Comp}_q(v - \mathbf{s}_1^T \mathbf{u}_1 - \mathbf{s}_0^T \mathbf{u}_0, 1)$
13 $v = \text{Comp}_q(\mathbf{t}_1^T \mathbf{r}_1 + \mathbf{t}_0^T \mathbf{r}_0 + e + \lceil \frac{q}{2} \rceil m, d_v)$	
14 $c = (\mathbf{u}_1, \mathbf{u}_0, v)$	

**Fig. 15.** The  $[\text{IND-CPA}, \text{IND-CPA}]$  secure  $2\text{PKE}_{\text{mlwe}}$  under Module-LWE assumption.

**Theorem 8.** *If there is a PPT adversary  $\mathcal{A}$  against  $[IND-CPA, IND-CPA]$  security of  $2PKE_{mlwe}$ , there exists  $\mathcal{B}$  such that,  $\mathbf{Adv}_{2PKE_{mlwe}}^{[IND-CPA, IND-CPA]}(\mathcal{A}) \leq 2\mathbf{Adv}_{k+1, k, \eta}^{mlwe}(\mathcal{B})$ .*

Please refer Appendix I for the analysis of decryption failure and formal proof.

By applying the modified FO transformation to  $2PKE_{mlwe}$ , we obtain a  $[OW-CCA, OW-CCA]$  secure  $2KEM_{mlwe}$ . Then by setting  $cpk_0 = (0)^k$  and  $csk_0 = (0)^k$ , and integrating  $2KEM_{mlwe}$  to AKE in section 4, a novel and efficient post-quantum AKE from Module-LWE assumption is constructed.

The parameter setting and comparison are given in Table 5 and 6. Note that by setting  $d_{t_1} = d_{t_0} = \lceil \log q \rceil$  we actually do not apply compress on public keys. (which fix one bug of the security proof in [3]). One may doubt that with  $q = 3329$  we can not apply NTT technique to accelerate the multiplications of two polynomials  $f(x) \times g(x)$  over  $R_q$ , since  $512 \nmid 3328$ . Actually, we can fix this gap. Separate  $f(x) = f_B(x^2) + xf_A(x^2)$ ,  $g(x) = g_2(x^2) + xg_1(x^2)$  into a series of odd power and a series of even power, then  $f(x) \times g(x) = f_B(x^2)g_2(x^2) + (f_A(x^2)g_2(x^2) + f_B(x^2)g_1(x^2))x + f_A(x^2)g_1(x^2)x^2$ . Then we can apply NTT to  $f_i(y)g_j(y)$  over  $Z_q[y]/(y^{128} + 1)$  by setting  $y = x^2$  since  $256 \mid 3328$ .

Scheme	$n$	$k$	$q$	$\eta$	$(d_{t_1}, d_{t_0}, d_{u_1}, d_{u_0}, d_v)$	$\delta$	Security Level
$2KEM_{mlwe}$	256	4	3329	1	(12, 12, 9, 9, 5)	$2^{-174.3}$	256

**Table 5.** The parameters for  $2KEM_{mlwe}$ .  $\delta$  is the decryption failure.

AKEs	Assumptions	Sec	$U_A \rightarrow U_B$ (Bytes)	$U_B \rightarrow U_A$ (Bytes)
Kyber.AKE	$\mathbf{Adv}_{5,4,5}^{mlwe}$	256	2912	3008
AKE from $2KEM_{mlwe}$	$\mathbf{Adv}_{5,4,5}^{mlwe}$	256	2838	2464

**Table 6.** The message size for Kyber in frame of FSXY13 and ours in frame of AKE.

## Acknowledgment

Haiyang Xue was supported by the National Natural Science Foundation of China 61602473, 61672019, 61772522, and the National Cryptography Development Fund MMJJ20170116. Xianhui Lu was supported by the National Natural Science Foundation of China 61572495. Bao Li was supported by the National Natural Science Foundation of China 61772515. Jingnan He was supported by the National Natural Science Foundation of China 61672030. Bei Liang was partially supported by the STINT grant (no 3720596). This work was supported by the Fundamental theory and cutting edge technology Research Program of Institute of Information Engineering, CAS (Grant No. Y7Z0291103). This work was supported by the National 973 Program of China under Grant 2014CB340603.

## References

1. Boyd, C., Cliff, Y., Gonzalez Nieto, J.M., Paterson, K.G.: Efficient One-Round Key Exchange in the Standard Model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 69-83. Springer, Heidelberg (2008)
2. Bos, J. W., Costello, C., Naehrig, M., and Stebila, D.: Post-quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In 2015 IEEE Symposium on Security and Privacy, pp. 553-570.
3. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., and Stehlé D.: CRYSTALS - Kyber: a CCA-secure Module-lattice-based KEM. In 2018 IEEE Symposium on Security and Privacy, pp. 353-367. code is available in <https://github.com/pq-crystals/kyber>.
4. Barbosa, M., Farshim, P.: Relations among Notions of Complete Non-malleability: Indistinguishability Characterisation and Efficient Construction without Random Oracles. In: R. Steinfeld and P. Hawkes (eds.) ACISP 2010, LNCS 6168, pp. 145-163. Springer, Heidelberg (2010)
5. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS 773, pp. 232-249. Springer, Heidelberg (1994)

6. Cremers, C.J.F.: Session-state Reveal Is Stronger Than Ephemeral Key Reveal: Attacking the NAXOS Authenticated Key Exchange Protocol. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 20-33. Springer, Heidelberg (2009)
7. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453-474. Springer, Heidelberg (2001)
8. Canetti, R., and Krawczyk H.: Security analysis of IKEs signature-based key-exchange protocol. In: Yung M. (eds) CRYPTO 2002, pp. 143-161. Springer, Heidelberg (2002)
9. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13-25. Springer, Heidelberg (1998)
10. Cramer R., Shoup V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Public Key Encryption. In: Knudsen L.R. (eds.) EUROCRYPT02. LNCS 2332, pp. 45-64. Springer, Heidelberg (1998)
11. Dolev, D., Dwork, C., and Naor, M.: Non-malleable cryptography. SIAM Journal on Computing, 30:391-437, 2000.
12. Diffie, W., and Hellman, M.: New directions in cryptography, IEEE Transactions on Information Theory, Vol. 22, Issue 6, pp.644-654.
13. Fischlin, M.: Completely Non-Malleable Schemes, In In: Caires L., Italiano G.F., Monteiro L., Palamidessi C., Yung M. (eds) ICALP 2005, LCNS 3580, pp. 779-790. Springer, Heidelberg (2005)
14. Fujisaki, E., and Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener M. (eds) CRYPTO 1999, LNCS 1666, pp. 537-554. Springer, Heidelberg (1999)
15. Fujioka A., Suzuki K., Xagawa K., Yoneyama K.: Strongly Secure Authenticated Key Exchange from Factoring Codes and Lattices. In: Fischlin M., Buchmann J., Manulis M. (eds) PKC 2012, pp. 467-484. Springer, Heidelberg (2012)
16. Fujioka A., Suzuki K., Xagawa K., Yoneyama K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In AsiaCCS 2013, pp. 83-94.
17. Giacon F., Heuer F., Poettering B.: KEM Combiners. In: Abdalla M., Dahab R. (eds) PKC 2018, LCNS 10769. pp. 190-281. Springer, Heidelberg (2018)
18. Hofheinz, D., Hövelmanns, K., and Kiltz, E.: A Modular Analysis of the Fujisaki-Okamoto Transformation. In Y. Kalai and L. Reyzin (eds) TCC 2017, LNCS 10677, pp 341-371. Springer, Heidelberg (2017)
19. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi S., Rabin T. (eds) TCC 2006, LNCS 3876, pp. 581-600. Springer, Heidelberg (2006)
20. Kiltz, E.: Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. In Okamoto, T., and Wang, X. (eds) PKC 2007, LNCS 4450, pp. 282-297. Springer, Heidelberg (2007)
21. Kiltz E., Pietrzak K., Stam M., Yung M.: A New Randomness Extraction Paradigm for Hybrid Encryption, In: Joux A. (eds) EUROCRYPT 2009, LCNS 5479, pp. 590-609. Springer, Heidelberg (2009)
22. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian J. (eds) CRYPTO 2001, LCNS 2139, pp. 310-331. Springer, Heidelberg (2001)
23. Krawczyk, H.: SIGMA: The SiGn-and-MAC Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols, In: Boneh D. (eds) CRYPTO 2003, LNCS 2729. pp. 400-425. Springer, Heidelberg (2003)
24. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (eds) CRYPTO 2005. LNCS, vol. 3621, pp. 546-566. Springer, Heidelberg (2005)
25. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1-16. Springer, Heidelberg (2007)
26. T. Matsumoto, Y. Takashima, and H. Imai: On seeking smart public-key distribution systems, Trans. IECE of Japan, 1986, E69(2), pp. 99-106.
27. A. Menezes, M. Qu, and S. Vanstone: Some new key agreement protocols providing mutual implicit authentication, In SAC 1995, pp. 22-32.
28. Okamoto, T.: Authenticated Key Exchange and Key Encapsulation Without Random Oracles. eprint archive: report 2007/473, full version of [29].
29. Okamoto, T.: Authenticated Key Exchange and Key Encapsulation in the Standard Model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474-484. Springer, Heidelberg (2007)
30. Peikert, C.: Lattice Cryptography for the Internet. In Mosca, M. (eds) PQCrypto 2014, LNCS 8772, pp. 197-219. Springer, Heidelberg (2014)
31. Peikert, C., Waters, B.: Lossy Trapdoor Functions and Their Applications. In: STOC 2008, pp. 187-196 (2008)
32. M. O. Saarinen. Hila5. Technical report, available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions>.



33. Wee, H.: Efficient Chosen-Ciphertext Security via Extractable Hash Proofs. In: Rabin T. (eds) CRYPTO 2010. LNCS 6223, pp. 314-332. Springer, Heidelberg (2010)
34. Yoneyama, K.: One-Round Authenticated Key Exchange with Strong Forward Secrecy in the Standard Model against Constrained Adversary. In: Hanaoka G., Yamauchi T. (eds) IWSEC 2012. LNCS 7631, pp. 69-86 Springer, Heidelberg (2012)
35. Yao, A.C.C., Zhao, Y.: OAKE: A new family of implicitly authenticated Diffie-Hellman protocols. In CCS 2013, pp. 1113-1128.
36. Zhang J., Zhang Z., Ding J., Snook M., Dagdelen O.: Authenticated key exchange from ideal lattices. In: Oswald E., Fischlin M. (eds) EUROCRYPT 2015, pp 719-751. Springer, Heidelberg (2015)

## Appendix A: Proof of Lemma 2

In order to bound the probability of AskH, we investigate the events  $\text{AskH} \wedge E_i$  for  $1 \leq i \leq 8$  listed in Table 4 one by one.

### Event $\text{AskH} \wedge E_1$

In the event  $E_1$ , the test session  $\text{sid}^*$  has no matching session, and the static secret key of  $U_A$  is given to  $\mathcal{A}$ . In case of  $\text{AskH} \wedge E_1$ , the [OW-CCA,  $\cdot$ ] adversary  $\mathcal{S}$  with  $pk_0^* = cpk_0$  performs as follows. It simulates the  $\text{CK}^+$  games, and transforms the probability of the occurrence of event AskH performed by  $\mathcal{A}$  to the advantage of attacking [OW-CCA,  $\cdot$ ] security with  $pk_0^* = cpk_0$ .

In order to simulate the random oracles,  $\mathcal{S}$  maintains hash list  $L_H$  and  $L_{sk}$ , corresponding to the queries and answers of the  $H$  oracle,  $\text{SessionStateReveal}$  and  $\text{SessionKeyReveal}$ .  $L_H$  and  $L_{sk}$  are interrelated with each other since the adversary may ask  $L_{sk}$  without the encapsulated keys firstly, then ask  $L_H$  with the encapsulated keys. Thus, the reduction must ensure consistency of the random oracle queries to  $L_H$  and  $L_{sk}$ . The decryption oracle of [OW-CCA,  $\cdot$ ] game could help to maintain the consistency as done in  $H$ -oracle and  $\text{SessionKeyReveal}$  in the following.

In the [OW-CCA,  $\cdot$ ] game, on receiving the public key  $pk_1$ ,  $\mathcal{S}$  returns an empty  $pk_0^* = cpk_0$  to the challenger. Then on receiving the challenge ciphertext  $C^*$  with public key  $pk_1$  and  $pk_0^*$  for encapsulated key  $K^*$ , to simulate the  $\text{CK}^+$  game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and guesses a random  $i$ -th session as a guess of the test session with probability of success  $1/N^2l$ .  $\mathcal{S}$  samples a key pair  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  as public parameters. By computing  $(pk_1, sk_1) \leftarrow \text{KeyGen1}$ ,  $\mathcal{S}$  sets all the static secret and public key pairs  $(pk_P, sk_P)$  for all  $N$  users  $U_P$  except  $U_B$ .  $\mathcal{S}$  sets  $pk_B = pk_1$ .

Without knowing the secret key of  $U_B$ ,  $\mathcal{S}$  chooses a random  $r_B$  as part of ephemeral secret keys and a random  $R_B$  as the randomness for  $\text{Encaps}$ . Since  $f_B$  is  $(\varepsilon_1, \varepsilon_2)$  hl-RF, the difference between the simulated game with modification of  $r_B$  and real game is bounded by  $\varepsilon_2$ . If it is in need of an ephemeral public key  $pk_{P0}$  sent out by  $U_P$ ,  $\mathcal{S}$  queries  $(pk_0^i, sk_0^i, r_0^i) \leftarrow \mathcal{O}_{\text{leak}_0}$  and sets  $pk_{P0} = pk_0^i$ . When a session state revealed to a session owned by  $U_B$ , is queried,  $\mathcal{S}$  returns  $r_B$  and  $r_0^i$  as the ephemeral secret key part.

Specially, by computing  $(pk_A, sk_A) \leftarrow \text{KeyGen1}$ , randomly choosing  $s_A \leftarrow \{0, 1\}^\lambda$  and querying  $(pk_{A0}, sk_{A0}, r_{A0}) \leftarrow \mathcal{O}_{\text{leak}_1}$ ,  $\mathcal{S}$  can set the static secret and public key pairs  $(pk_A, sk_A)$  for  $U_A$ , as well as the ephemeral secret and public key pairs  $(pk_{A0}, sk_{A0})$  for the  $i$ -th session of  $U_A$ .  $\mathcal{S}$  sets  $C^*, pk_{A0}$  as the messages sent out by  $U_A$  in  $i$ -th session. Meanwhile,  $\mathcal{S}$  leaks the static secret key  $sk_A$  of  $U_A$  to the adversary  $\mathcal{A}$ .

$\mathcal{S}$  simulates the oracle queries of  $\mathcal{A}$  as the following. Specially, if AskH happens, which means that  $\mathcal{A}$  submits  $(U_A, U_B, pk_A, pk_B, C^*, pk_{A0}, C_A, K_A, K_B)$  to  $H$  where  $C^*, pk_{A0}, C_A$  is the view of the test session and  $K_A$  is the key encapsulated in  $C_A$ , then return  $K_B$  as the guess of  $K^*$ .

$\mathcal{S}$  simulates the oracle queries of  $\mathcal{A}$  and maintains the hash lists  $L_H, L_{sk}$  as follows.

- Querying  $H$ -oracle with  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P0}, C_P, K_P, K_Q)$ 
  - 1: If  $P = A, Q = B, C_B = C^*$ , and  $(\Pi, I, U_A, U_B, pk_A, pk_B, C_B, pk_{A0}, C_A)$  is the  $i$ -th session of  $U_A$ , then  $\mathcal{S}$  outputs the  $K_B$  as the answer of [OW-CCA,  $\cdot$ ] challenge, namely  $K^*$ , and sets  $\text{flag} = \text{ture}$ .
  - 2: Else if  $\exists (U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P0}, C_P, K_P, K_Q, h) \in L_H$ , returns  $h$ ,
  - 3: Else if  $P = B$  and  $\exists (U_B, U_Q, pk_B, pk_Q, C_Q, pk_{B0}, C_B, h) \in L_{sk}$ :

1. if  $(C_Q, pk_{B0})$  is sent by  $\mathcal{A}$ : with the knowledge of  $sk_Q$ ,  $\mathcal{S}$  extracts  $K'_Q = \text{Decaps}(sk_Q, csk_0, C_Q)$ ; As  $C_B$  is generated by  $\mathcal{S}$  itself,  $\mathcal{S}$  knows the encapsulated key  $K'_B$  in  $C_B$ .
2. if  $C_B$  is sent by  $\mathcal{A}$ : As  $C_Q$  is generated by  $\mathcal{S}$ , it has the knowledge of encapsulated key  $K'_Q$  in  $C_Q$ .  $\mathcal{S}$  queries the decryption oracle of [OW-CCA,  $\cdot$ ] with  $pk_{B0}$  (which is the output of  $\mathcal{O}_{\text{leak}_1}$ ) and  $C_B$  to extract  $K'_B$ .
3. if both  $(C_Q, pk_{B0})$  and  $C_B$  are sent out by  $\mathcal{S}$ :  $\mathcal{S}$  knows both  $K'_B$  and  $K'_Q$  encapsulated in  $C_B$  and  $C_Q$  respectively.

If  $(K_B, K_Q) = (K'_B, K'_Q)$ , then return  $h$  and add the tuple  $(U_B, U_Q, pk_B, pk_Q, C_Q, pk_{B0}, C_B, K_B, K_Q, h)$  to the list  $L_H$ ;

4: Else if  $Q = B$  and  $\exists (U_P, U_B, pk_P, pk_B, C_B, pk_{P0}, C_P, h) \in L_{sk}$ :

1. if  $C_B \neq C^*$ 
  - (a) if  $(C_B, pk_{P0})$  is sent by  $\mathcal{A}$ :  $\mathcal{S}$  queries the decryption oracle of [OW-CCA,  $\cdot$ ] with  $pk' = cpk_0$  and  $C_B$  to extract encapsulated key  $K'_B$ ; As  $C_P$  is generated by  $\mathcal{S}$ , it has the knowledge of  $K'_P$  encapsulated in  $C$ .
  - (b) if  $C_P$  is sent by  $\mathcal{A}$ : with the knowledge of  $sk_P$  and  $sk_{P0}$ ,  $\mathcal{S}$  extracts  $K'_P = \text{Decaps}(sk_P, sk_{P0}, C_P)$ ; As  $C_B$  is generated by  $\mathcal{S}$  itself,  $\mathcal{S}$  has the knowledge of encapsulated key  $K'_B$  in  $C_B$ .
  - (c) if both  $(C_B, pk_{P0})$  and  $C_P$  are sent out by  $\mathcal{S}$ :  $\mathcal{S}$  has the knowledge of  $K'_P$  and  $K'_B$  encapsulated in  $C_P$  and  $C_B$  respectively.

If  $(K_P, K_B) = (K'_P, K'_B)$ , then return  $h$  and add  $(U_P, U_B, pk_P, pk_B, C_B, pk_{P0}, C_P, K_P, K_B, h)$  to the list  $L_H$ ;

2. if  $C_B = C^*$

- (a) if  $(C_B, pk_{P0})$  is sent by  $\mathcal{S}$ : As  $C_B = C^*$  is generated by  $\mathcal{S}$ ,  $\mathcal{S}$  just outputs the corresponding encapsulated key  $K'_B$  as the answer of [OW-CCA,  $\cdot$ ] game, namely  $K^*$ .
- (b) if  $(C_B, pk_{P0})$  is sent by  $\mathcal{A}$ : if  $K_P = K'_P$  ( $C_P$  and encapsulated key  $K'_P$  are generated by  $\mathcal{S}$ ), then  $\mathcal{S}$  outputs  $h$  and adds  $(U_P, U_B, pk_P, pk_B, C_B, pk_{P0}, C_P, K_P, K_Q, h)$  to the list  $L_H$ , else  $\mathcal{S}$  returns a random value  $h$  and adds  $(U_P, U_Q, pk_P, pk_Q, C_B, pk_{P0}, C_P, K_P, K_B, h)$  to the list  $L_H$ .

If  $(K_P, K_B) = (K'_P, K'_B)$ , then return  $h$  and add  $(U_P, U_B, pk_P, pk_B, C_B, pk_{P0}, C_P, K_P, K_B, h)$  to the list  $L_H$ ;

5: Else if  $B \neq P$  or  $Q$  and  $\exists (U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P0}, C_P, h) \in L_{sk}$ :

1. if  $(C_Q, pk_{P0})$  is sent by  $\mathcal{A}$ : with the knowledge of  $sk_Q$ ,  $\mathcal{S}$  extracts  $K'_Q = \text{Decaps}(sk_Q, csk_0, C_Q)$ ; As  $C_P$  is generated by  $\mathcal{S}$  itself,  $\mathcal{S}$  has the knowledge of encapsulated key  $K'_P$  in  $C_P$ .
2. if  $C_P$  is sent by  $\mathcal{A}$ : As  $C_Q$  and  $pk_{P0}$  are generated by  $\mathcal{S}$ ,  $\mathcal{S}$  has both the knowledge of encapsulated key  $K'_Q$  in  $C_Q$  and ephemeral secret key  $sk_{P0}$ ; With the knowledge of  $sk_P$  and  $sk_{P0}$ ,  $\mathcal{S}$  extracts  $K'_P$  from  $C_P$ .
3. if both  $(C_Q, pk_{P0})$  and  $C_P$  are sent by  $\mathcal{S}$ :  $\mathcal{S}$  has the knowledge of  $K'_P$  and  $K'_Q$  encapsulated in  $C_P$  and  $C_Q$  respectively.

If  $(K_P, K_Q) = (K'_P, K'_Q)$ , then return  $h$  and add  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P0}, C_P, K_P, K_Q, h)$  to the list  $L_H$ ;

6: otherwise,  $\mathcal{S}$  returns a random value  $h$  and adds  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P0}, C_P, K_P, K_Q, h)$  to the list  $L_H$ .

– Send( $\Pi, I, U_P, U_Q$ ):

1. If  $P = A$  and this session is the  $i$ -th session of  $U_A$ , then  $\mathcal{S}$  queries  $\mathcal{O}_{\text{leak}_1}$  to get a public and secret key pair. As in the setup,  $\mathcal{S}$  sets them as ephemeral public and secret key pair  $(pk_{A0}, sk_{A0})$  of  $U_A$  in  $i$ -th session. Then  $\mathcal{S}$  returns  $C_Q^*, pk_{A0}$ .
2. If  $P = B$ ,  $\mathcal{S}$  queries  $\mathcal{O}_{\text{leak}_1}$  to get  $(pk_{B0}, sk_{B0}, r)$  and generates two independent randomness  $(r_B, R_B)$  (to pretend that  $R_B$  is computed as  $f_B(sk_B, r_B)$ ). It will not be detected by  $\mathcal{A}$  since  $f_B$  is a  $(\epsilon_1, \epsilon_2)$  hl-RF).  $\mathcal{S}$  generates  $(C_Q, K_Q) \leftarrow \text{Encaps}(pk_Q, cpk_0, h_B)$  and sends out  $(C_Q, pk_{P0})$ .

3. otherwise,  $\mathcal{S}$  generates randomness  $r_P$  and  $R_P$  honestly, and computes  $(C_Q, K_Q) \leftarrow \text{Encaps}(pk_Q, cpk_0, h_P)$ .  
 $\mathcal{S}$  queries  $\mathcal{O}_{\text{leak}_1}$  to get  $(pk_{P_0}, sk_{P_0}, r)$  and returns  $(C_Q, pk_{P_0})$ .
- **Send**( $\Pi, R, U_Q, U_P, C_Q, pk_{P_0}$ ):  $\mathcal{S}$  computes the messages and session key, and maintains the session key list  $L_{sk}$  as follows.
1. If  $Q = B$ :  $\mathcal{S}$  generates two independent randomness  $(r_B, h_B)$  (to pretend that  $h_B$  is computed as  $f_B(sk_B, r_B)$ , which will not be detected by  $\mathcal{A}$ ).  $\mathcal{S}$  computes  $(C_P, K'_P) \leftarrow \text{Encaps}(pk_P, pk_{P_0}, h_B)$  and returns  $C_P$ . If  $\exists (U_P, U_B, pk_P, pk_B, C_B, pk_{P_0}, C_P, K_P, K_B, h) \in L_H$  and  $K'_P = K_P$ ,  $\mathcal{S}$  proceeds as following: if  $C_B = C^*$  then set  $SK = h$ , else (as  $C_B \neq C^*$ ) send the query  $(cpk_0, C_B)$  to the decryption oracle to get  $K'_B$ ; if  $K'_B = K_B$ , set  $SK = h$ .
  2. If  $Q \neq B$ :  $\mathcal{S}$  generates randomness  $r_{Q2}$  and queries  $h$ -oracle to get  $h_Q$ .  $\mathcal{S}$  generates  $(C_P, K'_P) \leftarrow \text{Encaps}(pk_P, pk_{P_0}, h_Q)$ . With the knowledge of  $sk_Q$ ,  $\mathcal{S}$  extracts  $K'_Q = \text{Decaps}(sk_Q, cpk_0, C_Q)$ . If there exists  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P_0}, C_P, K_P, K_Q, h) \in L_H$  and  $K_P = K'_P, K_Q = K'_Q$ , set  $SK = h$ .
  3. otherwise,  $\mathcal{S}$  chooses  $SK$  randomly.
- $\mathcal{S}$  keeps record of it as a completed session and adds  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P_0}, C_P, SK)$  to the session key list  $L_{sk}$ .
- **Send**( $\Pi, I, U_P, U_Q, C_Q, pk_{P_0}, C_P$ ):  $\mathcal{S}$  has the knowledge of  $K'_Q$  encapsulated in  $C_Q$ .  $(pk_{P_0}, sk_{P_0}, r)$  is received from  $\mathcal{O}_{\text{leak}_1}$ .
1. If  $P = B$ , then  $\mathcal{S}$  queries the decryption oracle of [OW-CCA,  $\cdot$ ] with  $(pk_0, C_B)$  to get  $K'_B$ . If there exists  $(U_B, U_Q, pk_B, pk_Q, C_Q, pk_{B_0}, C_B, K_B, K_Q, h) \in L_H$  and  $K'_B = K_B, K'_Q = K_Q$ , set  $SK = h$ .
  2. If  $Q = B$ , with the knowledge of  $sk_P$  and  $sk_{P_0}$ ,  $\mathcal{S}$  computes  $K'_P = \text{Decaps}(sk_P, sk_{P_0}, C)$ . If  $\exists (U_P, U_B, pk_P, pk_B, C_B, pk_{P_0}, C_P, K_P, K_B, h) \in L_H$  and  $K'_P = K_P, K'_Q = K_B$ , set  $SK = h$ .
  3. If  $P \neq B$  and  $Q \neq B$ , with the knowledge of  $sk_P$  and  $sk_{P_0}$ ,  $\mathcal{S}$  computes  $K'_P = \text{Decaps}(sk_P, sk_{P_0}, C)$ . If there exists  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P_0}, C_P, K_P, K_Q, h) \in L_H$  and  $K'_P = K_P, K'_Q = K_Q$ , set  $SK = h$ .
  4. otherwise,  $\mathcal{S}$  chooses  $SK$  randomly.
- $\mathcal{S}$  keeps record of it as a completed session and adds  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P_0}, C_P, SK)$  to the session key list  $L_{sk}$ .
- **Querying SessionKeyReveal**(sid): The session key list  $L_{sk}$  is maintained as in the Send queries.
1. If the session sid is not completed,  $\mathcal{S}$  aborts.
  2. Else if sid is in the list  $L_{sk}$ ,  $(U_P, U_Q, pk_P, pk_Q, C_Q, pk_{P_0}, C_P, SK) \in L_{sk}$ , then return  $SK$ .
  3. otherwise,  $\mathcal{S}$  returns a random value  $SK$  and adds it in  $L_{sk}$ .
- **Querying SessionStateReveal**(sid): As the definition of freshness, sid is not the test session.
1. If the owner of sid is  $B$ , and  $B$  is a responder. The session state is generated by himself, or received from  $\mathcal{O}_{\text{leak}_1}$ .  $\mathcal{S}$  just returns them.
  2. If the owner of sid is  $B$ , and  $B$  is a initiator. The session state is generated by himself, or received from  $\mathcal{O}_{\text{leak}_1}$ , or extractable from the decryption oracle.  $\mathcal{S}$  just returns them.
  3. otherwise,  $\mathcal{S}$  holds the secret key of other users and could return the session state as the definition.
- **Querying Corrupt**( $U_P$ ):  
 $\mathcal{S}$  returns the static secret key of  $U_P$ .
- **Test**(sid :)  
If sid is not the  $i$ -th session of  $U_A$ ,  $\mathcal{S}$  aborts; otherwise,  $\mathcal{S}$  responds to the query as the definition above.
- If  $\mathcal{A}$  outputs a guess  $b'$ ,  $\mathcal{S}$  aborts.

The simulator  $\mathcal{S}$  maintains all the consistencies of  $H$ -oracle,  $h$ -oracle, **SessionStateReveal**, and **SessionKeyReveal**, with the decryption oracle of 2KEM. Note that in the first case of the  $H$ -oracle, if  $\text{flag} = \text{ture}$ , then  $\mathcal{S}$  would succeed in the [OW-CCA,  $\cdot$ ] game. Thus,  $\Pr[\text{AskH} \wedge E_1] \leq N^2 l \cdot \text{Adv}_{2\text{KEM}}^{[\text{OW-CCA}, \cdot]}(\mathcal{S}) + N^2 l q \cdot \varepsilon_2$ .

**Event AskH  $\wedge$   $E_2$**

In the event  $E_2$ , the test session  $\text{sid}^*$  (with owner as initiator) has no matching session, and the ephemeral secret key of  $U_A$  is given to  $\mathcal{A}$ . In case of  $\text{AskH} \wedge E_2$ , the [OW-CCA,  $\cdot$ ] adversary  $\mathcal{S}$  performs as follows.

It simulates the  $\text{CK}^+$  games and transforms the probability of occurrence of event  $\text{AskH}$  performed by  $\mathcal{A}$  to the advantage of attacking  $[\text{OW-CCA}, \cdot]$  security with  $pk_0^* = cpk_0$ .

In order to simulate the random oracles,  $\mathcal{S}$  maintains hash list  $L_H$  and  $L_{sk}$ , corresponding to the queries and answers of the  $H$  oracle,  $\text{SessionStateReveal}$  and  $\text{SessionKeyReveal}$ .  $L_H$  and  $L_{sk}$  are interrelated with each other since the adversary may ask  $L_{sk}$  without the encapsulated keys firstly, then ask  $L_H$  with the encapsulated keys. Thus, the reduction must ensure consistency of the random oracle queries to  $L_H$  and  $L_{sk}$ . The decryption oracle of  $[\text{OW-CCA}, \cdot]$  game could help to maintain the consistency as done in  $H$ -oracle and  $\text{SessionKeyReveal}$  in the following.

In the  $[\text{OW-CCA}, \cdot]$  game, on receiving the public key  $pk_1$ ,  $\mathcal{S}$  returns an empty  $pk_0^* = cpk_0$  to the challenger. Then on receiving the challenge ciphertext  $C^*$  with public key  $pk_1$  and  $pk_0^*$  for encapsulated key  $K^*$ , to simulate the  $\text{CK}^+$  game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and guesses a random  $i$ -th session as a guess of the test session with probability of success  $1/N^2l$ .  $\mathcal{S}$  samples a key pair  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  as public parameters. By computing  $(pk_1, sk_1) \leftarrow \text{KeyGen1}$ ,  $\mathcal{S}$  sets all the static secret and public key pairs  $(pk_P, sk_P)$  for all  $N$  users  $U_P$  except  $U_B$ .  $\mathcal{S}$  sets  $pk_B = pk_1$ .

Without the knowledge of the secret key of  $U_B$ ,  $\mathcal{S}$  chooses totally random  $r_B$  as part of ephemeral secret keys and  $R_B$  as the randomness for  $\text{Encaps}$ . Since  $f_B$  is  $(\varepsilon_1, \varepsilon_2)$  hl-RF, the difference between the simulated game with modification of  $r_B$  and real game is bounded by  $\varepsilon_1$ . If it is in need of an ephemeral public key  $pk_{P0}$  sent out by  $U_P$ ,  $\mathcal{S}$  queries  $(pk_0^i, sk_0^i, r_0^i) \leftarrow \mathcal{O}_{\text{leak}_0}$  and sets  $pk_{P0} = pk_0^i$ . When a session state revealed to a session owned by  $U_B$ , is queried,  $\mathcal{S}$  returns  $r_B$  and  $r_0^i$  of this session.

Specially, by computing  $(pk_A, sk_A) \leftarrow \text{KeyGen1}$  and querying  $(pk_{A0}, sk_{A0}, r_{A0}) \leftarrow \mathcal{O}_{\text{leak}_1}$ ,  $\mathcal{S}$  sets the static secret and public key pairs  $(pk_A, sk_A)$  for  $U_A$ , as well as the ephemeral secret and public key pairs  $(pk_{A0}, sk_{A0})$  for the  $i$ -th session of  $U_A$ .  $\mathcal{S}$  sets  $C^*, pk_{A0}$  as the message sent out by  $U_A$  in  $i$ -th session.  $\mathcal{S}$  chooses an independent randomness  $r_{A2}$  and leaks the ephemeral secret keys  $r_{A0}, r_{A2}$  in the  $i$ -th session of  $U_A$  to adversary  $\mathcal{A}$ .

$\mathcal{S}$  simulates the oracle queries of  $\mathcal{A}$  as what it does in the case above and maintains the hash lists as in the event  $\text{AskH} \wedge E_1$ . Specially, when  $\text{AskH}$  happens, which means that  $\mathcal{A}$  submits  $(U_A, U_B, pk_A, pk_B, C^*, pk_{A0}, C_A, K_A, K_B)$  to  $H$ , where  $C^*, pk_{A0}, C_A$  is the view of the test session and  $K_A$  is the key encapsulated in  $C_A$ , then return  $K_B$  as the guess of  $K^*$ .

As in the event  $\text{AskH} \wedge E_1$ , we have  $\Pr[\text{AskH} \wedge E_2] \leq N^2l \cdot \text{Adv}_{2\text{KEM}}^{[\text{OW-CCA}, \cdot]}(\mathcal{S}) + N^2lq \cdot \varepsilon_2$ .

### Event $\text{AskH} \wedge E_3$

In the event  $E_3$ , the test session  $\text{sid}^*$  (with owner as responder) has no matching session, and the ephemeral secret keys of  $U_B$  are given to  $\mathcal{A}$ . In the case of  $\text{AskH} \wedge E_3$ , the  $[\text{OW-CCA}, \cdot]$  adversary  $\mathcal{S}$  with  $pk_0^* \leftarrow \mathcal{A}$  performs as follows. It simulates the  $\text{CK}^+$  games and transforms probability of the occurrence of event  $\text{AskH}$  performed by  $\mathcal{A}$  to the advantage of attacking  $[\text{OW-CCA}, \cdot]$  security.

In order to simulate the random oracles,  $\mathcal{S}$  maintains hash list  $L_H$  and  $L_{sk}$ , corresponding to the queries and answers of the  $H$  oracle  $\text{SessionStateReveal}$  and  $\text{SessionKeyReveal}$ .  $L_H$  and  $L_{sk}$  are interrelated with each other since the adversary may ask  $L_{sk}$  without the encapsulated keys firstly, then ask  $L_H$  with the encapsulated keys. Thus, the reduction must ensure consistency of the random oracle queries to  $L_H$  and  $L_{sk}$ . The decryption oracle of  $[\text{OW-CCA}, \cdot]$  game could help to maintain the consistency as done in  $H$ -oracle and  $\text{SessionKeyReveal}$  in the following.

On receiving the public key  $pk_1$  from the  $[\text{OW-CCA}, \cdot]$  challenger, to simulate the  $\text{CK}^+$  game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and guesses a random  $i$ -th session as a guess of the test session with probability of success  $1/N^2l$ .  $\mathcal{S}$  samples a key pair  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  as public parameters. By computing  $(pk_1, sk_1) \leftarrow \text{KeyGen1}$ ,  $\mathcal{S}$  sets all the static secret and public key pairs  $(pk_P, sk_P)$  for all  $N$  users  $U_P$  except  $U_A$ . Specially,  $\mathcal{S}$  sets the static secret and public key pairs  $(pk_B, sk_B)$  for  $U_B$ .  $\mathcal{S}$  sets  $pk_A = pk_1$ .

Without knowing the secret key of  $U_A$ ,  $\mathcal{S}$  chooses totally random  $r_A$  as part of ephemeral secret key and  $R_A$  as the randomness for  $\text{Encaps}$ . Since  $f_A$  is  $(\varepsilon_1, \varepsilon_2)$  hl-RF, the difference between the simulated game with modification of  $r_A$  and real game is bounded by  $\varepsilon_1$ . If it is in need of an ephemeral public key  $pk_{P0}$  sent out by  $U_P$ ,  $\mathcal{S}$  queries  $(pk_0^i, sk_0^i, r_0^i) \leftarrow \mathcal{O}_{\text{leak}_0}$  and sets  $pk_{P0} = pk_0^i$ . When a session state revealed to a session owned by  $U_A$ , is queried,  $\mathcal{S}$  returns  $r_A$  and  $r_0^i$  of this session.

On receiving the  $i$ -th session  $(C'_B, pk_0^*)$  from  $U_A$  (which is sent by  $\mathcal{A}$  in the  $\text{CK}^+$  games),  $\mathcal{S}$  returns  $pk_0^*$  to the  $[\text{OW-CCA}, \cdot]$  challenger and receives the challenge ciphertext  $C^*$  under public key  $pk_1$  and  $pk_0^*$  with encapsulated key  $K^*$ . Then  $\mathcal{S}$  sends  $C^*$  to  $U_A$  as the response of  $i$ -th session from  $U_B$ .  $\mathcal{S}$  chooses a totally independent randomness  $r_B$  as the ephemeral secret key of  $U_B$  and leaks it to adversary  $\mathcal{A}$ .

$\mathcal{S}$  simulates the oracle queries of  $\mathcal{A}$  as what it does in the case above and maintains the hash lists as in the event of  $\text{AskH} \wedge E_2$ . Specially, when  $\text{AskH}$  happens, which means  $\mathcal{A}$  submits  $(U_A, U_B, pk_A, pk_B, C'_B, pk_0^*, C^*, K_A, K_B)$  to  $H$ , where  $C'_B, pk_0^*, C^*$  is the view of the test session and  $K_B$  is the key encapsulated in  $C'_B$ , then return  $K_A$  as the guess of  $K^*$ .

As in event  $\text{AskH} \wedge E_2$ ,  $\Pr[\text{AskH} \wedge E_3] \leq N^2 l \cdot \text{Adv}_{2\text{KEM}}^{[\text{OW-CCA}, \cdot]}(\mathcal{S}) + N^2 l q \cdot \varepsilon_1$ .

**Event  $\text{AskH} \wedge E_4$**

In the event  $E_4$ , the test session  $\text{sid}^*$  (with owner as responder) has no matching session, and the static secret keys of  $U_B$  are given to  $\mathcal{A}$ . In the case of  $\text{AskH} \wedge E_4$ , the  $[\text{OW-CCA}, \cdot]$  adversary  $\mathcal{S}$  with  $pk_0^* \leftarrow \mathcal{A}$  performs as follows. It simulates the  $\text{CK}^+$  games and transforms the probability of the occurrence of event  $\text{AskH}$  performed by  $\mathcal{A}$  to the advantage of attacking  $[\text{OW-CCA}, \cdot]$  security.

In order to simulate the random oracles,  $\mathcal{S}$  maintains hash list  $L_H$  and  $L_{sk}$ , corresponding to the queries and answers of the  $H$  oracle,  $\text{SessionStateReveal}$  and  $\text{SessionKeyReveal}$ .  $L_H$  and  $L_{sk}$  are interrelated with each other since the adversary may ask  $L_{sk}$  without the encapsulated keys firstly, then ask  $L_H$  with the encapsulated keys. Thus, the reduction must ensure consistency of the random oracle queries to  $L_H$  and  $L_{sk}$ . The decryption oracle of  $[\text{OW-CCA}, \cdot]$  game could help to maintain the consistency as done in  $H$ -oracle and  $\text{SessionKeyReveal}$  in the following.

On receiving the public key  $pk_1$  from the  $[\text{OW-CCA}, \cdot]$  the challenger, to simulate the  $\text{CK}^+$  game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and guesses a random  $i$ -th session as a guess of the test session with probability of success  $1/N^2 l$ .  $\mathcal{S}$  samples a key pair  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  as public parameters. By computing  $(pk_1, sk_1) \leftarrow \text{KeyGen1}$  and sets all the static secret and public key pairs  $(pk_P, sk_P)$  for all  $N$  users  $U_P$  except  $U_A$ . Specially,  $\mathcal{S}$  sets the static secret and public key pairs  $(pk_B, sk_B)$  for  $U_B$ .  $\mathcal{S}$  sets  $pk_A = pk_1$ . If it is in need of an ephemeral public key  $pk_{P0}$  sent out by  $U_P$ ,  $\mathcal{S}$  queries  $(pk_0^i, sk_0^i, r_0^i) \leftarrow \mathcal{O}_{\text{leak}_0}$  and sets  $pk_{P0} = pk_0^i$ .

On receiving the  $i$ -th session  $(C'_B, pk_0^*)$  from  $U_A$  (which is sent by  $\mathcal{A}$  in the  $\text{CK}^+$  games),  $\mathcal{S}$  returns  $pk_0^*$  to the  $[\text{OW-CCA}, \cdot]$  challenger and receives the challenge ciphertext  $C^*$  under public key  $pk_1$  and  $pk_0^*$  with encapsulated key  $K^*$ . Then  $\mathcal{S}$  returns  $C^*$  to  $U_A$  as the response of  $i$ -th session from  $U_B$ .  $\mathcal{S}$  leaks the static secret key  $sk_B$  of  $U_B$  to the adversary  $\mathcal{A}$ .

$\mathcal{S}$  simulates the oracle queries of  $\mathcal{A}$  as what it does in the case above and maintains the hash lists as in the event of  $\text{AskH} \wedge E_3$ . Specially, when  $\text{AskH}$  happens, which means  $\mathcal{A}$  submits  $(U_A, U_B, pk_A, pk_B, C'_B, pk_0^*, C^*, K_A, K_B)$  to  $H$ , where  $C'_B, pk_0^*, C^*$  is the view of the test session and  $K_B$  is the key encapsulated in  $C'_B$ , then return  $K_A$  as the guess of  $K^*$ .

As in the event  $\text{AskH} \wedge E_3$ , we have  $\Pr[\text{AskH} \wedge E_4] \leq N^2 l \cdot \text{Adv}_{2\text{KEM}}^{[\text{OW-CCA}, \cdot]}(\mathcal{S}) + N^2 l q \cdot \varepsilon_1$ .

**Event  $\text{AskH} \wedge E_5$**

In event  $E_5$ , the test session  $\text{sid}^*$  (with owner as responder or initiator) has matching session  $\overline{\text{sid}^*}$ . Both static secret keys of initiator and responder are leaked to  $\mathcal{A}$ . In this case, the  $[\cdot, \text{OW-CPA}]$  adversary  $\mathcal{S}$  performs as follows. It simulates the  $\text{CK}^+$  games and transforms the probability of the occurrence of event  $\text{AskH}$  performed by  $\mathcal{A}$  to the advantage of attacking  $[\cdot, \text{OW-CPA}]$  security.

To simulate the  $\text{CK}^+$  game,  $\mathcal{S}$  randomly chooses two parties  $U_A, U_B$  and guesses a random  $i$ -th session as a guess of the test session with probability of success  $1/N^2 l$ .  $\mathcal{S}$  queries some  $(pk_1^i, sk_1^i, r_1^i) \leftarrow \mathcal{O}_{\text{leak}_1}$ , and sets all the static secret and public key pairs  $(pk_P, sk_P) = (pk_1^i, sk_1^i)$  for all  $N$  users  $U_P$ .  $\mathcal{S}$  samples a key pair  $(cpk_0, csk_0) \leftarrow \text{KeyGen0}$  as public parameters. When an ephemeral public key is required,  $\mathcal{S}$  generates  $(pk_{P0}, sk_{P0}) \leftarrow \text{KeyGen0}(r_0)$  by himself. In the  $[\cdot, \text{OW-CPA}]$  game,  $\mathcal{S}$  sends  $pk_A$  to the challenger and receives challenge ciphertext  $C^*$ . In the  $i$ -th session of  $U_A$ ,  $\mathcal{S}$  sends  $C^*, pk_0$  to  $U_B$ .  $\mathcal{S}$  leaks  $sk_A$  and  $sk_B$  to adversary  $\mathcal{A}$ .

With all the static secret keys,  $\mathcal{S}$  could perfectly simulate the  $\text{CK}^+$  games. When  $\text{AskH}$  happens, which means  $\mathcal{A}$  submits  $(U_A, U_B, pk_A, pk_B, C_B, pk_0^*, C^*, K_A, K_B)$  to  $H$ , where  $C'_B, pk_0^*, C^*$  is the view of the test session and  $K_B$  is the key encapsulated in  $C_B$ , then return  $K_A$  as the guess of  $K^*$ .

Thus,  $\Pr[\text{AskH} \wedge E_5] \leq N^2 l \cdot \text{Adv}_{2\text{KEM}}^{[\cdot, \text{OW-CPA}]}(\mathcal{S}) + N^2 l q \cdot \varepsilon_2$ .

**Event AskH  $\wedge$   $E_6$** 

In event  $E_6$ , the test session  $\text{sid}^*$  has matching session  $\overline{\text{sid}^*}$ . Both ephemeral secret keys of initiator and responder are leaked to  $\mathcal{A}$ . This is almost the same as Event AskH  $\wedge$   $E_3$ , in which case the ephemeral public key is generated by  $\mathcal{A}$ . In this case, the only difference is that the ephemeral secret key of test session (or the matching session) is leaked to  $\mathcal{A}$  but not generated by  $\mathcal{A}$ , which means  $pk_0^* \in L_0$ .

**Event AskH  $\wedge$   $E_{7-1}$** 

In event  $E_{7-1}$ , the test session  $\text{sid}^*$  has matching session  $\overline{\text{sid}^*}$ . Both ephemeral secret keys of responder and static secret key of initiator are leaked to  $\mathcal{A}$ . This is almost the same with Event AskH  $\wedge$   $E_1$ . In this case, the only difference is that the ephemeral secret key of  $U_B$  is leaked to  $\mathcal{A}$ , which does not affect the proof.

**Event AskH  $\wedge$   $E_{7-2}$** 

In event  $E_{7-2}$ , the test session  $\text{sid}^*$  has matching session  $\overline{\text{sid}^*}$ . Both ephemeral secret keys of initiator and static secret key of responder are leaked to  $\mathcal{A}$ . This is almost the same as Event AskH  $\wedge$   $E_6$ . In this case, the only difference is that the ephemeral secret key of  $U_A$  is leaked to  $\mathcal{A}$ , which does not make any influences to the proof.

**Event AskH  $\wedge$   $E_{8-1}$** 

In event  $E_{8-1}$ , the test session  $\text{sid}^*$  has matching session  $\overline{\text{sid}^*}$ . Both ephemeral secret keys of initiator and the static secret key of responder are leaked to  $\mathcal{A}$ . This is almost the same as Event AskH  $\wedge$   $E_{7-2}$ . In this case, the only difference is the position of initiator and responder, which does not affect the proof.

**Event AskH  $\wedge$   $E_{8-2}$** 

In event  $E_{8-2}$ , the test session  $\text{sid}^*$  has matching session  $\overline{\text{sid}^*}$ . Both static secret keys of initiator and the ephemeral secret key of responder are leaked to  $\mathcal{A}$ . This is almost the same as Event AskH  $\wedge$   $E_{8-2}$ . In this case, the only difference is the position of initiator and responder, which does not affect the proof.

## Appendix B: Proofs of Theorem 2 related to HMQV

We first show the [OW-CCA, OW-CCA] security of  $2\text{KEM}_{\text{HMQV}}$  against the resistance to the leakage of  $b$  by proving the security of  $2\text{KEM}_{\text{HMQV}^0}$  in Figure 16 (in Lemma 4), then show the resistance to the leakage of randomness  $y$  by reducing it to the security of Dual HCR signature (in Lemma 5).

Since after replacing  $Y$  in  $2\text{KEM}_{\text{HMQV}^0}$  by  $YB^e$  and  $y$  in  $2\text{KEM}_{\text{HMQV}^0}$  by  $y+eb$ , we will get  $2\text{KEM}_{\text{HMQV}}$ , and  $B$  is public parameter, the security of  $2\text{KEM}_{\text{HMQV}^0}$  is preserved in  $2\text{KEM}_{\text{HMQV}}$ . Thus, if  $2\text{KEM}_{\text{HMQV}^0}$  is [OW-CCA, OW-CCA] secure, then  $2\text{KEM}_{\text{HMQV}}$  also is [OW-CCA, OW-CCA] secure even if  $b$  is leaked.

KeyGen1( $\lambda$ )	KeyGen0( $\lambda$ )	Encaps( $pk_1, pk_0$ )	Decaps( $sk_1, sk_0, c$ )
$a \leftarrow \mathbb{Z}_p;$	$x \leftarrow \mathbb{Z}_p$	$y \leftarrow \mathbb{Z}_p, Y = g^y$	$Y \leftarrow c$
$A = g^a$	$X = g^x$	$d = h(X, B)$	$d = h(X, B)$
$pk_1 = A$	$pk_0 = X;$	$k' = \hat{H}((XA^d)^y)$	$k = \hat{H}(Y^{x+da})$
$sk_1 = a$	$sk_0 = x.$	Return $k, c = Y$	Return $k'$

**Fig. 16.** The [OW-CCA, OW-CCA] secure  $2\text{KEM}_{\text{HMQV}^0}$ .

Therefore, in the following we focus on the security of  $2\text{KEM}_{\text{HMQV}^0}$ . Furthermore, we reduce its security to the unforgeability of HCR signature. Before going to the lemmas, we depict the HCR signature scheme provided in [24] and its unforgeability game.

**Definition 4 (The (Dual) HCR signature, [24]).** Let  $U_A$  be a signer with public key  $A = g^a$ ,  $U_B$  be a verifier with public key  $B$ . The HCR signature of  $U_A$  is defined as:  $Y = g^y, X = g^x$  and  $\text{HSIG}(m, Y, X) = \hat{H}(Y^{x+da})$ ; The Dual HCR signature of  $U_A$  is defined as:  $Y = g^y, X = g^x$  and  $\text{HSIG}(m, Y, X) = \hat{H}((YB^e)^{x+da})$ ; where  $Y$  is a challenge computed by  $U_B$ ,  $X$  is a respond generated by  $U_A$  ( $x$  is chosen by  $U_A$ ), and  $e = h(Y, A), d = h(X, B)$ .

The forgery game for HCR signature is described as follows. Any PPT forger  $\mathcal{F}$  with the challenged public key  $A$  and  $X_0$  of his choice, interactively queries a sign oracle  $\text{SignO}$ . Finally  $\mathcal{F}$  outputs “fail”

or a forgery  $(X_0, m_0, \sigma)$ . The sign oracle  $SignO$  proceeds as following: build a list  $L = \{-, -, -, -\}$  first. On receiving a message  $m$  from  $\mathcal{F}$  it returns  $X = g^x$  for  $x \leftarrow Z_q$ , and sets  $L = L \cup (x, X, m, \cdot, \cdot)$  where the last two elements are empty. On receiving  $(X', m')$  and challenge  $Y'$ , if  $(\cdot, X', m', \cdot, \cdot) \in L$  and  $Y' \neq 0$ , it returns  $\sigma' = H(Y^{x+h(X', m')^a})$ , then makes up the tuple  $(\cdot, X', m', \cdot, \cdot)$  as  $(\cdot, X', m', Y', \sigma')$ , else it returns  $\perp$ . We say that  $\mathcal{F}$  wins the game successfully if both of the following two conditions hold: *i*)  $(\cdot, X_0, m_0, \cdot, \cdot) \notin L$ , or  $(\cdot, X_0, m_0, \cdot, \cdot) \in L \wedge (\cdot, X_0, m_0, Y_0, \sigma) \notin L$ ; *ii*)  $\sigma = \hat{H}(Y^{x_0+h(X_0, m_0)^a})$ , where  $x_0 = \log_g X_0$ . Note that in HMQV [24], the case of  $(\cdot, X_0, m_0, \cdot, \cdot) \in L \wedge (\cdot, X_0, m_0, Y_0, \sigma) \notin L$  is not considered as a successful forgery in the forgery game defined by the authors [24]. But the proof still works when this type of forgery is also included in the forgery game.

The advantage of  $\mathcal{F}$  is defined as  $\text{Adv}_{\mathcal{F}}^{\text{uf}} = \Pr[\mathcal{F} \text{ wins}]$ . HCR is said to be unforgeable if  $\text{Adv}_{\text{HCR}}^{\text{uf}}(\mathcal{F})$  is negligible for any PPT forger  $\mathcal{F}$ . The unforgeability of Dual HCR can be defined similarly.

**Lemma 3** ([24], **Lemma 27** & **Remark 7.1**). *Under the Gap-DH, KEA1 assumptions, HCR is unforgeable in the random oracle model; and Dual HCR is unforgeable with the leakage of randomness  $y$ .*

**Lemma 4**. *If HCR is unforgeable in the random oracle model,  $2\text{KEM}_{\text{HMQV}^0}$  is [OW-CCA, OW-CCA] secure in the random oracle model.*

*Proof.* We reduce the [OW-CCA,  $\cdot$ ] security to the unforgeability of HCR. It is analogous for the [ $\cdot$ , OW-CCA] security.

We construct a forger  $\mathcal{F}$  that performs as follows.  $\mathcal{F}$  simulates the [OW-CCA,  $\cdot$ ] game for  $\text{KEM}_{\text{HMQV}^0}$ , and transfers the advantage of adversary  $\mathcal{A}$  attacking  $2\text{KEM}_{\text{HMQV}^0}$  to that of forging HCR. As shown in Figure 17,  $\mathcal{F}$  perfectly simulates  $\mathcal{O}_{\text{leak}0}$  and  $\mathcal{O}_{\text{OW-CCA}}$  using  $SignO$ . If  $\mathcal{A}$  succeeds in [OW-CCA,  $\cdot$ ] game, it holds  $k' = H(Y^{x_0+h(X_0, B)^a})$ . Thus, we have  $\text{Adv}_{2\text{KEM}_{\text{HMQV}^0}}^{\text{IND-CCA}, \cdot}(\mathcal{A}) \leq \text{Adv}_{\text{HCR}}^{\text{uf}}(\mathcal{F})$ .

<p><b>Forger <math>\mathcal{F}^{\mathcal{A}}(A, Y_0)</math>:</b></p> <p>01 send <math>pk_1 = A</math> to <math>\mathcal{A}</math>, build <math>L_0 = \{-, -, -\}</math></p> <p>02 on the <math>i</math>-th query of <math>\mathcal{O}_{\text{leak}0}</math> <math>\mathcal{F}</math> performs as following:</p> <p>03 query <math>SignO</math> with <math>m = B</math> and get <math>(x_i, X_i)</math></p> <p>04 set <math>L_0 = L_0 \cup (pk_0^i = X_i, sk_0^i = x_i, r_0^i = x_i)</math></p> <p>05 return <math>(sk_0^i = x_i, pk_0^i = X_i, r_0^i = x_i)</math></p> <p>06 on receiving <math>pk_0^* = X_0</math>, return <math>C^* = Y_0 = g^{y_0}</math> as challenge ciphertext</p> <p>07 on receiving <math>\mathcal{O}_{\text{OW-CCA}}(pk_0' = X', C' = Y')</math>;</p> <p>08 if <math>X' \in [L_0]_1 \wedge X' \neq X_0</math> or <math>X' \in [L_0]_1 \wedge X' = X_0 \wedge Y' \neq Y_0</math></p> <p>09 query <math>SignO</math> with <math>(X', B)</math> and challenge <math>Y'</math></p> <p>10 send what <math>SignO</math> returns to <math>\mathcal{A}</math></p> <p>11 otherwise send <math>\perp</math> to <math>\mathcal{A}</math></p> <p>12 on receiving <math>k'</math> from <math>\mathcal{A}</math> as the guess of <math>k^*</math></p> <p>13 return <math>(X_0, A, k')</math> as signature on challenge <math>Y_0</math>.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Fig. 17.** Forger of HCR using [OW-CCA,  $\cdot$ ] adversary  $\mathcal{A}$

**Lemma 5.** *If Dual HCR is unforgeable with the leakage of randomness  $y$ ,  $2\text{KEM}_{\text{HMQV}}$  is [OW-CCA, OW-CCA] secure with the leakage of randomness  $y$  in the random oracle model.*

## Appendix C: Proofs of Theorem 3 related to NAXOS

*Proof.* We reduce the [OW-CCA,  $\cdot$ ] security to the underlying Gap-DH assumption. It is analogous for the [ $\cdot$ , OW-CCA] security. For convenience, we define a twisted  $2\text{KEM}'_{\text{NAXOS}}$ , in which the encapsulation algorithm  $\text{Encaps}$  chooses  $y \leftarrow \mathbb{Z}_p$  directly, rather than by computing  $y = h(y_0, b)$  where  $y_0 \leftarrow \mathbb{Z}_p$ . Obviously, in the random oracle model, to prove the [OW-CCA,  $\cdot$ ] security of  $2\text{KEM}_{\text{NAXOS}}$  with the leakage of  $x_0$  or  $b$ , we only need to prove the [OW-CCA,  $\cdot$ ] security of  $2\text{KEM}'_{\text{NAXOS}}$  itself.

We construct an algorithm  $\mathcal{B}$  which utilizes the [OW-CCA,  $\cdot$ ] adversary  $\mathcal{A}$  as a sub-routine to solve the Gap-DH problem. Given Gap-DH instance,  $\mathcal{B}$  simulates the [OW-CCA,  $\cdot$ ] games for  $\mathcal{A}$ , and transforms the advantage of  $\mathcal{A}$  attacking [OW-CCA,  $\cdot$ ] security to that of solving Gap-DH instance. To perfectly simulate the [OW-CCA,  $\cdot$ ] game for  $\mathcal{A}$ ,  $\mathcal{B}$  maintains one decapsulation list  $L_{\text{dec}}$  and one hash list  $L_{\hat{H}}$ , and guarantees the consistency of two lists by utilizing the DDH oracle.

$\mathcal{B}$  is given as input  $(X_0, Y_0)$ , where  $X_0 = g^{x_0}$  and  $Y_0 = g^{y_0}$  for random  $x_0, y_0$ , and finally outputs a value *guess*.  $\mathcal{B}$  is also given a DDH oracle  $\mathcal{O}_{\text{DDH}}$ . Firstly  $\mathcal{B}$  sets  $pk_1 = A = X_0$  and sends  $A$  to  $\mathcal{A}$ . On receiving  $pk_0^* = X$  from  $\mathcal{A}$ ,  $\mathcal{B}$  sets  $c^* = Y_0$ , and chooses a random  $k^* \leftarrow \{0, 1\}^\lambda$ . Then  $\mathcal{B}$  sets a default value  $g$  as a *guess*. We expect that  $\mathcal{A}$  at some point makes a queries of the form  $(Z = Y_0^{x_0}, D')$  to  $\hat{H}$ -oracle such that  $\mathcal{O}_{\text{DDH}}(A, Y_0, Z) = 1$ , otherwise  $\mathcal{A}$  would have no advantage to output challenge session key. To find out when  $\mathcal{A}$  queries its  $\hat{H}$ -oracle with  $Y_0^{x_0}$ , we query  $\mathcal{O}_{\text{DDH}}(A, Y_0, Z)$  whenever  $\mathcal{A}$  makes a query  $(Z, D')$  to  $\hat{H}$ -oracle. If  $\mathcal{O}_{\text{DDH}}(A, Y_0, Z) = 1$ , then  $Z = Y_0^{x_0}$ . We then update the value of *guess* to  $Z$ .

- The decapsulation  $\text{DECAPS}(X', Y')$  (where  $pk_0' = X'$  and  $c' = Y'$ ) is simulated as follows: If  $(X', Y') = (X, Y_0)$ , it just aborts. If  $(X', Y')$  has been asked to  $\text{DECAPS}$  (which means  $\exists(X', Y', k') \in L_{\text{dec}}$ ), then just return  $k'$ . If  $(X', Y')$  has not been asked to  $\text{DECAPS}$ , then first check whether  $\exists(Z', D', h') \in L_{\hat{H}}$  (which means  $(Z', D')$  has been asked to  $\hat{H}$ -oracle) where  $\mathcal{O}_{\text{DDH}}(A, Y', Z') = 1 \wedge \mathcal{O}_{\text{DDH}}(X', Y', D') = 1$ . If it is the case, set  $k' = h'$ ; else set  $k'$  as a random value. At last, add  $(X', Y', k')$  to list  $L_{\text{dec}}$ .
- The  $\hat{H}$ -oracle is simulated as follows: If  $(Z', D')$  has been asked to  $\hat{H}$ -oracle (which means  $\exists(Z', D', h') \in L_{\hat{H}}$ ) just return  $h'$ . If  $(Z', D')$  has not been asked to  $\text{DECAPS}$ , then first check whether  $\mathcal{O}_{\text{DDH}}(A, Y_0, Z') = 1$ . If it is the case, update the value of *guess* to  $Z$ . If not, then check whether  $\exists(X', Y', k') \in L_{\text{dec}}$  (which means  $(X', Y')$  has been asked to  $\text{DECAPS}$ ) where  $\mathcal{O}_{\text{DDH}}(A, Y', Z') = 1 \wedge \mathcal{O}_{\text{DDH}}(X', Y', D') = 1$ . If it is true, set  $h' = k'$ ; else set  $h'$  as a random value. At last, add  $(Z', D', h')$  to the hash list  $L_{\hat{H}}$ .

Note that no matter which oracle  $\mathcal{A}$  asked first,  $\hat{H}$  with  $(Z', D')$  or  $\text{DECAPS}$  with  $(X', Y')$ , two lists are consistent.

Consider the game [OW-CCA,  $\cdot$ ] and let  $\text{AskA}$  denote the event that the  $\hat{H}$ -oracle query  $(Y_0^{x_0}, D')$  for some  $D'$  is asked by  $\mathcal{A}$  and  $\overline{\text{AskA}}$  is the complement event of  $\text{AskA}$ . When  $(Y_0^{x_0}, D')$  for some  $D'$  is not asked by  $\mathcal{A}$  to  $\hat{H}$ -oracle, there is no way to output the challenge key  $k^*$  other than guessing with probability  $1/2^\lambda$ . Thus we have that

$$\begin{aligned} \text{Adv}_{2\text{KEM}_{\text{NAXOS}}}^{\text{[OW-CCA, } \cdot \text{]}}(\mathcal{A}) &= \Pr[\text{OW-CCA}^{\mathcal{A}} \Rightarrow 1 \wedge \text{AskA}] + \Pr[\text{OW-CCA}^{\mathcal{A}} \Rightarrow 1 \wedge \overline{\text{AskA}}] \\ &\leq \Pr[\text{OW-CCA}^{\mathcal{A}} \Rightarrow 1 \wedge \text{AskA}] + 1/2^\lambda \\ &\leq \Pr[1 \wedge \text{AskA}] + 1/2^\lambda \\ &\leq \text{Adv}^{\text{Gap-DH}}(\mathcal{B}) + 1/2^\lambda \end{aligned}$$

□

Put them all together, we have that  $2\text{KEM}_{\text{NAXOS}}$  is [OW-CCA, OW-CCA] secure even with the leakage of one of  $x_0$  and  $b$ .

## Appendix D: Proofs of Theorem 4 and optimized AKE related to Okamoto

**The proof of Theorem 4.** The proof of [ $\cdot$ , IND-CPA] security proceeds by a series of games. Let  $\mathcal{A}$  be the adversary that is involved in the [ $\cdot$ , IND-CPA] game. We set it as game  $G_0$ , then  $\text{Adv}_{2\text{KEM}_{\text{Okamoto}}}^{[\cdot, \text{IND-CPA}]}(\mathcal{A}) = |\Pr[b' = b \text{ in } G_0] - 1/2|$ . In game  $G_1$ , when computing challenge encapsulated key  $k_0^*$  corresponding to  $c^* = (Y_1, Y_2, Y_3)$ ,  $X_3^{y_3^*}$  used in challenge encapsulated key is substituted with a uniformly random value in  $G_1$ . There exists an algorithm  $\mathcal{B}$  such that  $\Pr[b' = b \text{ in } G_0] - \Pr[b' = b \text{ in } G_1] \leq \text{Adv}_{\mathcal{B}}^{\text{ddh}}$ .  $\mathcal{B}$  performs as following: on receiving DDH challenge  $(g_1, X_3, Y_3, T)$ , it computes and returns  $pk_0 = X_3$ .



After receiving  $pk_1^* = (A_1, A_2)$  from  $\mathcal{A}$ ,  $\mathcal{B}$  computes challenge ciphertext as  $c^* = (Y_1 = g_1^y, Y_2 = g_2^y, Y_3)$  and  $\sigma^* = T \cdot (A_1 A_2^c)^y$ , then  $k_0^* = \hat{F}_{\sigma^*}(pk_0, c^*)$ . Finally, on receiving  $b'$  and the guess of  $b$ ,  $\mathcal{B}$  returns  $b' \stackrel{?}{=} b$ . If  $(g_1, X_1, Y_1, T)$  is a DDH tuple, this is exactly the game  $G_0$ ; If  $(g_1, X_1, Y_1, T)$  is a non-DDH tuple, this is exactly the game  $G_1$ . Note that in  $G_1$ ,  $k_b^*$  is independent of  $b$ , therefore  $\Pr[b = b' \text{ in } G_1] = 1/2$ .

To prove the [IND-CCA,  $\cdot$ ] security, we are confronted with the problem that the adversary may query the strong decapsulation oracle with ciphertexts under other public keys, thus the inputs of PRF should include public key and the PRF is lifted to pairwise-independent random source PRF, which is still a PRF even if the random key is only pairwise-independent. The proof of [IND-CCA,  $\cdot$ ] security proceeds by a series of games. Let  $\mathcal{A}$  be the adversary that is involved in the [IND-CCA,  $\cdot$ ] game. We set it as game  $G_0$ , then  $\text{Adv}_{2\text{KEM}_{\text{Okamoto}}}^{\text{[IND-CCA, } \cdot \text{]}}(\mathcal{A}) = |\Pr[b' = b \text{ in } G_0] - 1/2|$ .

In game  $G_1$ , the decryption oracle will reject queries with  $(Y_1', Y_2', Y_3') \neq (Y_1, Y_2, Y_3)$ , and  $h_{\text{tcr}}(A_1, A_2, Y_1', Y_2', Y_3') = h_{\text{tcr}}(A_1, A_2, Y_1, Y_2, Y_3)$ . Note that this will happen with negligible probability if  $h_{\text{tcr}}$  is a target collision resistant hash function.

In game  $G_2$ , to generate the challenge encapsulated key,  $\sigma^*$  corresponding to  $c^* = (Y_1, Y_2, Y_3)$ , is computed by using  $X_3^{y_3} \cdot Y_1^{a_1+ca_3} Y_2^{a_2+ca_4}$  instead of  $X_3^{y_3} \cdot (A_1 A_2^c)^{y^*}$  which is the exact value computed in game  $G_1$ .

In game  $G_3$ ,  $Y_1, Y_2$  used in  $c^*$  are substituted with non-DDH tuple. There exists an algorithm  $\mathcal{B}$  such that  $\Pr[b' = b \text{ in } G_2] - \Pr[b' = b \text{ in } G_3] \leq \text{Adv}_{\mathcal{B}}^{\text{ddh}}$ .

In game  $G_4$ , with the trapdoor  $s = \log_{g_2} g_1$ , on receiving the decryption queries with  $(X_3' = g_1^{x_3}; Y_1', Y_2', Y_3')$ , set  $\sigma'$  as a totally random element, if  $(Y_1, Y_2) \neq (Y_1', Y_2') \wedge Y_2' \neq Y_1'^s$ .  $G_4$  is identical with  $G_3$ , except that when bad happens, namely,  $(Y_1, Y_2) \neq (Y_1', Y_2') \wedge Y_2' \neq Y_1'^s$  but  $(A_1 A_2^c)^y = Y_1^{a_1+ca_3} Y_2^{a_2+ca_4}$ . From [9], we have that  $Y_1^{a_1+ca_3} Y_2^{a_2+ca_4}$  is the universal 2 function and bad happens with probability less than  $1/p$ .

In game  $G_5$ , the encapsulated key  $k_0^*$  is substituted with a random string. Note that in the case  $(Y_1', Y_2') = (Y_1, Y_2)$ , we have  $(pk_0', c') \neq (pk_0^*, c^*)$  (otherwise the decryption oracle aborts); in the case  $(Y_1', Y_2') \neq (Y_1, Y_2)$ ,  $\sigma^*$  is pairwise-independent with  $\sigma^i$  (where  $\sigma^i$  as the internal value is computed by the  $i$ -th decryption oracle). By the definition of pairwise-independent random source PRF, the difference between  $G_4$  and  $G_5$  is bounded by the advantage against pairwise-independent random source PRF. Note that in  $G_5$ ,  $k_b^*$  is independent of  $b$ , therefore  $\Pr[b = b' \text{ in } G_5] = 1/2$ .

### Optimized Okamoto AKE.

What is more, let  $H_4 : \{0, 1\}^* \rightarrow \{0, 1\}^l$  be a 4-wise independent hash function [21]. We employ the technique of optimizing classical KEM [21] to  $2\text{KEM}_{\text{Okamoto}}$  and get optimized  $2\text{KEM}_{\text{Okamoto-opt}}$  scheme shown in Figure 18. Applying  $2\text{KEM}_{\text{Okamoto-opt}}$  to  $\text{AKE}_{\text{std}}$ , we will get an optimized AKE of Okamoto-AKE.

$2\text{KEM}_{\text{Okamoto-opt}}.\text{KeyGen1}(\lambda)$	$2\text{KEM}_{\text{Okamoto-opt}}.\text{KeyGen0}(\lambda)$
$a_1, a_2 \leftarrow \mathbb{Z}_p^3, A = g_1^{a_1} g_2^{a_2};$	$x_3 \leftarrow \mathbb{Z}_p, X_3 = g_1^{x_3}$
$pk_1 = A, sk_1 = (a_1, a_2)$	$pk_0 = X_3, sk_0 = x_3$
$2\text{KEM}_{\text{Okamoto-opt}}.\text{Encaps}(pk_0, pk_1);$	$2\text{KEM}_{\text{Okamoto-opt}}.\text{Decaps}(sk_0, sk_1, C)$
$y, y_3 \leftarrow \mathbb{Z}_p, Y_1 = g_1^y, Y_2 = g_2^y, Y_3 = g_1^{y_3}$	$C \in G^3, (Y_1, Y_2, Y_3) \leftarrow C$
$\sigma = H_4(X_3^{y_3} \cdot A^y)$	$\sigma' = H_4(Y_3^{x_3} \cdot Y_1^{a_1} Y_2^{a_2})$
$C = (Y_1, Y_2, Y_3), K = \bar{F}_\sigma(pk_0, C)$	$K' = \bar{F}_{\sigma'}(pk_0, C)$

Fig. 18. The [IND-CCA, IND-CPA] secure optimized  $2\text{KEM}_{\text{Okamoto-opt}}$ .

**Theorem 9.** *If  $H_4$  is a 4-wise independent hash function, then  $2\text{KEM}_{\text{Okamoto-opt}}$  is [IND-CCA, IND-CPA] secure under DDH assumption.*

**Lemma 6 ([21], Theorem 4.3, Lemma 5.1).** *Let  $G$  be a group with prime order  $p$ , and generators  $g_1, g_2$ . Let  $A = g_1^{a_1} g_2^{a_2}$ . If both  $(g_1, g_2, Y_1, Y_2)$  and  $(g_1, g_2, Y_1', Y_2')$  are non-DDH tuples and  $(Y_1, Y_2) \neq (Y_1', Y_2')$ , then  $\{A, H_4, H_4(Y_1^{a_1} Y_2^{a_2}), H_4(Y_1'^{a_1} Y_2'^{a_2})\}$  is statistically indistinguishable with  $\{A, H_4, U_{2l}\}$ .*

The proof of [ $\cdot$ , IND-CPA] security proceeds by a series of games. Let  $\mathcal{A}$  be the adversary that is involved in the [ $\cdot$ , IND-CPA] game. We set it as game  $G_0$ , then  $\text{Adv}_{2\text{KEM}_{\text{Okamoto-opt}}}^{[\cdot, \text{IND-CPA}]}(\mathcal{A}) = |\Pr[b' = b \text{ in } G_0] - 1/2|$ . In

game  $G_1$ , when computing challenge encapsulated key  $k_0^*$  corresponding to  $c^* = (Y_1, Y_2, Y_3), X_3^{y_3^*}$  used in challenge encapsulated key is substituted with an uniform random values in  $G$ . There exists an algorithm  $\mathcal{B}$  such that  $\Pr[b' = b \text{ in } G_0] - \Pr[b' = b \text{ in } G_1] \leq \text{Adv}_{\mathcal{B}}^{\text{ddh}}$ .  $\mathcal{B}$  performs as following: on receiving DDH challenge  $(g_1, X_3, Y_3, T)$ , it computes and returns  $pk_0 = X_3$ . After receiving  $pk_1^* = A$  from  $\mathcal{A}$ ,  $\mathcal{B}$  computes challenge ciphertext as  $c^* = (Y_1 = g_1^{y_1}, Y_2 = g_2^{y_2}, Y_3)$  and  $k_0^* = \hat{F}_{H_4(T \cdot A^y)}(pk_0, c^*)$ . Finally, on receiving  $b'$  and the guess of  $b$ ,  $\mathcal{B}$  returns  $b' \stackrel{?}{=} b$ . If  $(g_1, X_1, Y_1, T)$  is a DDH tuple, this is exactly the game  $G_0$ ; If  $(g_1, X_1, Y_1, T)$  is a non-DDH tuple, this is exactly the game  $G_1$ . Note that in  $G_1$ ,  $k_b^*$  is independent of  $b$ , therefore  $\Pr[b = b' \text{ in } G_2] = 1/2$ .

The proof of [IND-CCA,  $\cdot$ ] security proceeds by a series of games. Let  $\mathcal{A}$  be the adversary that is involved in the [IND-CCA,  $\cdot$ ] game. We set it as game  $G_0$ , then  $\text{Adv}_{2\text{KEM}_{\text{Oka-opt}}}^{\text{[IND-CCA, \cdot]}}(\mathcal{A}) = |\Pr[b' = b \text{ in } G_0] - 1/2|$ .

In game  $G_1$ , to generate the challenge encapsulated key,  $\sigma^*$  corresponding to  $c^* = (Y_1, Y_2, Y_3)$ , is computed by using  $X_3^{y_3} \cdot Y_1^{a_1} Y_2^{a_2}$  instead of  $X_3^{y_3} \cdot A^{y^*}$  which is the exact value computed in game  $G_0$ .

In game  $G_2$ ,  $Y_1, Y_2$  used in  $c^*$  are substituted with non-DDH tuple. There exists an algorithm  $\mathcal{B}$  such that  $\Pr[b' = b \text{ in } G_1] - \Pr[b' = b \text{ in } G_2] \leq \text{Adv}_{\mathcal{B}}^{\text{ddh}}$ .

In game  $G_3$ , with the trapdoor  $s = \log_{g_2} g_1$ , on receiving the decryption queries with  $(X_3' = g_1^{x_3'}; Y_1', Y_2', Y_3')$ , set  $\sigma'$  as a totally random key, if  $(Y_1, Y_2) \neq (Y_1', Y_2') \wedge Y_2' \neq Y_1'^s$ .  $G_2$  is identical with  $G_3$ , except that when bad happens, namely,  $(Y_1, Y_2) \neq (Y_1', Y_2') \wedge Y_2' \neq Y_1'^s$  but  $\sigma' = H_4(Y_3^{x_3'} \cdot Y_1'^{a_1} Y_2'^{a_2})$ . From Lemma 6, bad happens with probability less than  $1/2^t$ .

In game  $G_4$ , the encapsulated key  $k_0^*$  is substituted with a random string. Note that in case  $(Y_1', Y_2') = (Y_1, Y_2)$ , we have  $(pk_0', c') \neq (pk_0^*, c^*)$  (otherwise the decryption oracle aborts); in the case  $(Y_1', Y_2') \neq (Y_1, Y_2)$ ,  $\sigma^*$  is pairwise -independent with  $\sigma^i$  (where  $\sigma^i$  as the internal value is computed by the  $i$ -th decryption oracle). By the definition of pairwise-independent random source PRF, the difference between  $G_4$  and  $G_3$  is bounded by the advantage against pairwise-independent random source PRF. Note that in  $G_4$ ,  $k_b^*$  is independent of  $b$ , therefore  $\Pr[b = b' \text{ in } G_4] = 1/2$ .

## Appendix E: Proof of Theorem 5 related to improved KEM combiner

**Proof of Theorem 5 in the random oracle model**,  $f(pk_0, k_1 || k_0, c) = \hat{H}(pk_0, k_1 || k_0, c)$ . Since  $[\cdot, \text{OW-CPA}]$  security is straightforward, in the rest we only prove the  $[\text{OW-CCA}, \cdot]$  security. The proof proceeds with a sequence of games. Let  $S_i$  denote the advantage of  $[\text{OW-CCA}, \cdot]$  adversary in Game  $i$ .

In Game 0, it is the original  $[\text{OW-CCA}, \cdot]$  game, namely, on receiving  $(pk_0', c_1' || c_0')$  the decapsulation oracle proceeds as follows: if  $(pk_0', c_1' || c_0') = (pk_0^*, c_1^* || c_0^*)$ , abort; else if  $pk_0' \notin [L_0]_0$ , abort; else compute  $k_1' = \text{Decaps}_{\text{cca}}(sk_1, c_1')$  and  $k_0' = \text{Decaps}_{\text{cpa}}(sk_0, c_0')$ , and return  $k' = \hat{H}(pk_0', k_1' || k_0', c')$ . The challenger also maintains a hash list  $L_{\hat{H}}$ , which works as follows: on receiving  $(pk_0', k_1' || k_0', c_1' || c_0')$ , if  $\exists (pk_0', k_1' || k_0', c_1' || c_0', k') \in L_{\hat{H}}$ , return  $k'$ ; else select  $k \leftarrow \mathcal{K}$ , set  $L_{\hat{H}} = L_{\hat{H}} \cup \{pk_0', k_1' || k_0', c_1' || c_0', k\}$ , and return  $k$ .

In Game 1, we modify the decapsulation oracle and hash list. The decapsulation oracle works as follows: on receiving  $(pk_0', c_1' || c_0')$ , if  $(pk_0', c_1' || c_0') = (pk_0^*, c_1^* || c_0^*)$ , abort; else if  $pk_0' \notin [L_0]_0$ , abort; else if  $\exists (pk_0', c_1' || c_0', k') \in L_D$ , return  $k'$ ; else choose  $k \leftarrow \mathcal{K}$ , set  $L_D = L_D \cup \{pk_0', c_1' || c_0', k\}$ , and return  $k$ .

The challenger also maintains a hash list  $L_{\hat{H}}$ , which works as follows: on receiving  $(pk_0', k_1' || k_0', c_1' || c_0')$ , if  $\exists (pk_0', k_1' || k_0', c_1' || c_0', k') \in L_{\hat{H}}$ , return  $k'$ ; else select  $k \leftarrow \mathcal{K}$ ; if  $pk_0' \in [L_0]_1$ , return  $k$ ; if  $pk_0' \notin [L_0]_1$ ,  $k_1' = \text{Decaps}_{\text{cca}}(sk_1, c_1')$ ,  $k_0' = \text{Decaps}_{\text{cpa}}(sk_0, c_0')$ , and  $\exists k'$  s.t.  $(pk_0', c_1' || c_0', k') \in L_D$  then return  $k'$ ; else, set  $L_D = L_D \cup \{(pk_0', c_1' || c_0', k)\}$ . At last  $L_{\hat{H}} = L_{\hat{H}} \cup \{pk_0', k_1' || k_0', c_1' || c_0', k'\}$ . Then if not abort, return  $k'$ .

To show the equivalence of Game 1 and Game 0 from the point view of  $\mathcal{A}$ , consider the following cases:

- Case 1:  $pk_0' \notin [L_0]_1$ . The decapsulation oracle aborts in both Game 0 and Game 1.
- Case 2:  $pk_0' \in [L_0]_1$ . if  $k_1' = \text{Decaps}_{\text{cca}}(sk_1, c_1')$  and  $k_0' = \text{Decaps}_{\text{cpa}}(sk_0, c_0')$ , the decapsulation oracle returns  $\hat{H}(pk_0', k_1' || k_0', c_1' || c_0')$  in both Game 1 and Game 0. And if  $\mathcal{A}$  queries decapsulation oracle first, it adds  $(pk_0', k_1' || k_0', c_1' || c_0', k \leftarrow \mathcal{K})$  to  $L_D$ . When  $\mathcal{A}$  queries  $H$  on  $(pk_0', k_1' || k_0', c_1' || c_0')$  later on, if the conditions of  $k_1' = \text{Decaps}_{\text{cca}}(sk_1, c_1')$  and  $k_0' = \text{Decaps}_{\text{cpa}}(sk_0, c_0')$  are satisfied, it

adds  $(pk'_0, k'_1 || k'_0, c'_1 || c'_0, k)$  to  $L_H$  and declares  $H(pk'_0, k'_1 || k'_0, c'_1 || c'_0, \cdot) = k$ . if  $\mathcal{A}$  queries  $\hat{H}$  oracle first, if the conditions  $k'_1 = \text{Decaps}_{\text{cca}}(sk_1, c'_1)$  and  $k'_0 = \text{Decaps}_{\text{cpa}}(sk_0, c'_0)$  are satisfied, it adds  $(pk'_0, k'_1 || k'_0, c'_1 || c'_0, k)$  to  $L_H$  to declare  $\hat{H}(pk'_0, k'_1 || k'_0, c'_1 || c'_0, \cdot) = k$  and adds  $(pk'_0, k'_1 || k'_0, c'_1 || c'_0, k)$  to  $L_D$ . Later, if it queries  $(pk'_0, k'_1 || k'_0, c'_1 || c'_0)$  to decapsulation oracle., it returns  $k$ .

We now re-clarify the sub-cases of case 2 for hash list.

- Subcase 2.1: if  $pk'_0 \in [L_0]_0$  and  $pk'_0 = pk_0^*$  and  $k'_0 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$ 
  - Subsubcase 2.1.1  $c_1^* = c'_1$ , compute  $k_1 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$  and check if  $k'_1 \stackrel{?}{=} k_1$ .
  - Subsubcase 2.1.2  $c_1^* \neq c'_1$ , compute  $k_1 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$  and check if  $k'_1 \stackrel{?}{=} k_1$ .
- Subcase 2.2: if  $pk'_0 \in [L_0]_0$  and  $pk'_0 \neq pk_0^*$  and  $k'_0 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$ 
  - Subsubcase 2.2.1  $c_1^* = c'_1$ , compute  $k_1 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$  and check if  $k'_1 \stackrel{?}{=} k_1$ .
  - Subsubcase 2.2.2  $c_1^* \neq c'_1$ , compute  $k_1 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$  and check if  $k'_1 \stackrel{?}{=} k_1$ .

In Game 2, we add flags in the hash list in two cases

- Case 1: if  $pk'_0 \notin [L_0]_0$ , and  $pk'_0 = pk_0^* \wedge c'_1 || c'_0 = c_0^* || c_1^*$ , set flag as true and abort.
- Case 2: if  $pk'_0 \in [L_0]_0$ ,  $k'_0 = \text{Decaps}_{\text{cpa}}(sk_0, c'_0)$  and  $pk'_0 = pk_0^* \wedge c'_1 || c'_0 = c_0^* || c_1^*$ , set flag as true and abort;

While now in Case 2 the subcases is

- Subcase 2.1: if  $pk'_0 \in [L_0]_0$  and  $pk'_0 = pk_0^*$  and  $k'_0 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$ 
  - Subsubcase 2.1.1  $c_1^* = c'_1$ , set flag as true and abort;
  - Subsubcase 2.1.2  $c_1^* \neq c'_1$ , compute  $k_1 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$  and check if  $k'_1 \stackrel{?}{=} k_1$ ;
- Subcase 2.2: if  $pk'_0 \in [L_0]_0$  and  $pk'_0 \neq pk_0^*$  and  $k'_0 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$ 
  - Subsubcase 2.2.1  $c_1^* = c'_1$ , set flag as true and abort;
  - Subsubcase 2.2.2  $c_1^* \neq c'_1$ , compute  $k_1 = \text{Decasp}_{\text{cpa}}(sk'_0, c'_0)$  and check if  $k'_1 \stackrel{?}{=} k_1$ .

In both case 1 and case 2 the event that flag=true is bounded by the OW-CCA security of  $\text{KEM}_{\text{cca}}$ .

By the property of random oracle, the  $[\text{OW-CCA}, \cdot]$  adversary only has advantage when he asks  $\hat{H}$  with  $(pk_0^*, k_1^* || k_0^*, c_1^* || c_0^*)$ , where  $k_1^*$  is the key encapsulated in  $c_1^*$ . We denote this event as  $\text{AskH}$ , and prove that the probability of the occurrence of event  $\text{AskH}$  is negligible if  $\text{KEM}_{\text{cca}}$  is OW-CCA secure.

Given a  $\text{KEM}_{\text{cca}}$  challenge ciphertext  $c^*$ , the CCA adversary  $\mathcal{S}$  simulates the  $[\text{OW-CCA}, \cdot]$  game and transform the probability of the occurrence of event  $\text{AskH}$  to the advantage of solving OW-CCA problem. Given a  $\text{KEM}_{\text{cca}}$  challenge ciphertext  $c^*$ , since  $\mathcal{S}$  does not know the secret key  $sk_1$ , the difficulties for  $\mathcal{S}$  to simulate the  $[\text{OW-CCA}, \cdot]$  game is in the subsubcase 2.1.2 and 2.2.2. But  $\mathcal{S}$  could fix them by querying the decapsulation oracle of  $\text{KEM}_{\text{cca}}$ .

Now in Game 2,  $\hat{H}(pk_0^*, k_1^* || k_0^*, c_1^* || c_0^*)$  will not be given to the  $\hat{H}$  adversary, thus the adversary's view is independent of the challenge encapsulated key. Thus the adversary's advantage in Game 2 is negligible.

To sum up, we have that the  $[\text{OW-CCA}, \cdot]$  security is guaranteed by the OW-CCA security of  $\text{KEM}_{\text{cca}}$ .

**Proof of Theorem 5 in the standard model.**  $f(pk_0, k_1 || k_0, c) = F_{k_1}(pk_0, c) \oplus F_{k_0}(pk_0, c)$ . Since  $[\cdot, \text{IND-CPA}]$  security is straightforward, in the rest we only prove the  $[\text{IND-CCA}, \cdot]$  security and reduce it to the IND-CCA security of  $\text{KEM}_{\text{cca}}$  and the security of PRF. The proof proceeds with a sequence of games. Let  $S_i$  denote the advantage of  $[\text{IND-CCA}, \cdot]$  adversary in Game  $i$ .

In Game 0, it is the original  $[\text{IND-CCA}, \cdot]$  game, namely, on receiving  $(pk'_0, c'_1 || c'_0)$  the decapsulation oracle performs as follows: if  $(pk'_0, c'_1 || c'_0) = (pk_0^*, c_1^* || c_0^*)$ , abort; else if  $pk'_0 \notin [L_0]_0$ , abort; else compute  $k'_1 = \text{Decaps}_{\text{cca}}(sk_1, c'_1)$ ,  $k'_0 = \text{Decaps}_{\text{cpa}}(sk_0, c'_0)$  and return  $k' = F_{k'_1}(pk'_0, c') \oplus F_{k'_0}(pk'_0, c')$ .

In Game 1, we change the decryption oracle when  $c'_1 = c_1^*$ . That is: if  $(pk'_0, c'_1 || c'_0) = (pk_0^*, c_1^* || c_0^*)$ , abort; else if  $pk'_0 \notin [L_0]_0$ , abort; else if  $c'_1 = c_1^*$ , sample  $k'_1 \leftarrow \mathcal{K}$ ; else if  $k'_1 = \text{Decaps}_{\text{cca}}(sk_1, c'_1)$ , then compute  $k'_0 = \text{Decaps}_{\text{cpa}}(sk_0, c'_0)$  and return  $k' = F_{k'_1}(pk'_0, c') \oplus F_{k'_0}(pk'_0, c')$ . There exists a IND-CCA adversary  $\mathcal{B}$  against  $\text{KEM}_{\text{cca}}$  if  $[\text{IND-CCA}, \cdot]$  adversary  $\mathcal{A}$  can distinguish Game 0 and Game 1. After receiving challenge ciphertext  $c_1^*$  and  $K_b^*$ , in the  $[\text{IND-CCA}, \cdot]$  game, if  $c'_1 = c_1^*$ , then the IND-CCA

adversary  $\mathcal{B}$  sets  $k'_1 = k^*$ . Note that if  $K_b^*$  is the key encapsulated in  $c_1^*$ , it corresponds to Game 0. If  $K_b^*$  is a totally random key, it corresponds to Game 1.

In Game 2, the computation of  $(c_1^* || c_0^*, k^*)$  is changed, where the PRF  $F$  is replaced by a random function. Note that the decryption oracle only works when  $(pk'_0, c') \neq (pk_0^*, c_1^* || c_0^*)$ , and  $k_1^*$  is replaced by a totally random string. Since  $F$  is a PRF, this replacement will not be detected by [IND-CCA,  $\cdot$ ] adversary. Note that in Game 2 the challenge ciphertext contains none information about  $b$ , thus  $\Pr[S_2] = 1/2$ .

To sum them up, we have that the [IND-CCA,  $\cdot$ ] security is guaranteed by the IND-CCA security of  $\text{KEM}_{\text{cca}}$  and pseudorandomness of PRF.

## Appendix F: Definitions of 2-key Public Key Encryption

Similar to the notion of 2-key KEM, we can also define the notion of 2-key public key encryption (PKE). Formally, a double-key public key encryption  $2\text{PKE} = (\text{KeyGen0}, \text{KeyGen1}, \text{Enc}, \text{Dec})$  is a quadruple of probabilistic algorithms together with a plaintext space  $\mathcal{M}$  and a ciphertext space  $\mathcal{C}$ .

**SECURITY.** To define  $[\text{ATK}_1, \text{ATK}_0]$  security of 2PKE, we consider two adversaries, *i.e.*,  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_1)$  attacking  $pk_1$  and  $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_1)$  attacking  $pk_0$ . In Figure 19 we show the security games of  $\text{ATK}_1$  and  $\text{ATK}_0$  respectively.

<p><b>Game IND-ATK1 on <math>pk_1</math></b></p> <p>01 <math>(pk_1, sk_1) \leftarrow \text{KeyGen1}(pp)</math>;</p> <p>02 <math>L_0 = \{(-, -, -)\}</math></p> <p>03 <math>(state, pk_0^*, m_0, m_1) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{ATK}_1}, \mathcal{O}_{\text{leak}_0}}(pk_1)</math></p> <p>04 <math>b \leftarrow \{0, 1\}</math>;</p> <p>05 <math>c^* \leftarrow \text{Enc}(pk_1, pk_0^*, m_b)</math>;</p> <p>06 <math>b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{ATK}_1}, \mathcal{O}_{\text{leak}_0}}(state, c^*)</math></p> <p>07 return <math>b' \stackrel{?}{=} b</math></p>	<p><b>Game IND-ATK0 on <math>pk_0</math></b></p> <p>15 <math>(pk_0, sk_0) \leftarrow \text{KeyGen0}(pp)</math></p> <p>16 <math>L_1 = \{(-, -, -)\}</math></p> <p>17 <math>(state, pk_1^*, m_0, m_1) \leftarrow \mathcal{B}_1^{\mathcal{O}_{\text{ATK}_0}, \mathcal{O}_{\text{leak}_1}}(pk_0)</math></p> <p>18 <math>b \leftarrow \{0, 1\}</math></p> <p>19 <math>c^* \leftarrow \text{Enc}(pk_1^*, pk_0, m_b)</math>;</p> <p>20 <math>b' \leftarrow \mathcal{B}_2^{\mathcal{O}_{\text{ATK}_0}, \mathcal{O}_{\text{leak}_1}}(state, c^*)</math></p> <p>21 return <math>b' \stackrel{?}{=} b</math></p>
<p><b>Game OW-ATK1 on <math>pk_1</math></b></p> <p>08 <math>(pk_1, sk_1) \leftarrow \text{KeyGen1}(pp)</math>;</p> <p>09 <math>L_0 = \{(-, -, -)\}</math></p> <p>10 <math>(state, pk_0^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{ATK}_1}, \mathcal{O}_{\text{leak}_0}}(pk_1)</math></p> <p>11 <math>m \leftarrow \mathcal{M}</math>;</p> <p>12 <math>c^* \leftarrow \text{Enc}(pk_1, pk_0^*, m)</math>;</p> <p>13 <math>m' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{ATK}_1}, \mathcal{O}_{\text{leak}_1}}(state, c^*)</math>;</p> <p>14 return <math>m' \stackrel{?}{=} m</math></p>	<p><b>Game OW-ATK0 on <math>pk_0</math></b></p> <p>22 <math>(pk_0, sk_0) \leftarrow \text{KeyGen0}(pp)</math></p> <p>23 <math>L_1 = \{(-, -, -)\}</math></p> <p>24 <math>(state, pk_1^*) \leftarrow \mathcal{B}_1^{\mathcal{O}_{\text{ATK}_0}, \mathcal{O}_{\text{leak}_1}}(pk_0)</math>;</p> <p>25 <math>m \leftarrow \mathcal{M}</math>;</p> <p>26 <math>c^* \leftarrow \text{Enc}(pk_1^*, pk_0, m)</math>;</p> <p>27 <math>m' \leftarrow \mathcal{B}_2^{\mathcal{O}_{\text{ATK}_0}, \mathcal{O}_{\text{leak}_1}}(state, c^*)</math>;</p> <p>28 return <math>m' \stackrel{?}{=} m</math></p>

**Fig. 19.** The  $[\text{ATK}_1, \cdot]$ , and  $[\cdot, \text{ATK}_0]$  games of 2PKE for adversaries  $\mathcal{A}$  and  $\mathcal{B}$ . The oracles  $\mathcal{O}_{\text{leak}_0}$ ,  $\mathcal{O}_{\text{ATK}_1}$ ,  $\mathcal{O}_{\text{leak}_1}$ , and  $\mathcal{O}_{\text{ATK}_0}$  are defined in the following.

On the  $i$ -th query of  $\mathcal{O}_{\text{leak}_0}$  and  $\mathcal{O}_{\text{leak}_1}$ , the challenger performs as what it does in 2-key KEM. Depending on the definition of oracle  $\mathcal{O}_{\text{ATK}_i}$  for  $i = 1, 0$  the adversary gets access to, one gets CPA and CCA notions respectively:

- if  $\mathcal{O}_{\text{ATK}_1}(pk'_0, c') = -$ , it implies CPA notion;
- if  $\mathcal{O}_{\text{ATK}_1}(pk'_0, c') \neq -$  it works as following: If  $pk'_0 \in [L_0]_1 \wedge (c' \neq c^* \vee pk'_0 \neq pk_0^*)$  return the corresponding plaintext, otherwise return  $\perp$ . It implies CCA notion.
- if  $\mathcal{O}_{\text{ATK}_0}(pk'_1, c') = -$ , it implies CPA notion;
- if  $\mathcal{O}_{\text{ATK}_0}(pk'_1, c') \neq -$  it works as following: If  $pk'_1 \in [L_1]_1 \wedge (c' \neq c^* \vee pk'_1 \neq pk_1^*)$  return the corresponding plaintext, otherwise return  $\perp$ . It implies CCA notion.

Let  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  be an adversary against  $pk_1$  of 2PKE. We define its advantage winning in the game IND-ATK1 and OW-ATK1 as:  $\text{Adv}_{2\text{PKE}}^{[\text{IND-ATK1}, \cdot]}(\mathcal{A}) = \left| \Pr[\text{IND-ATK1}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2} \right|$  and  $\text{Adv}_{2\text{PKE}}^{[\text{OW-ATK1}, \cdot]}(\mathcal{A}) = \Pr[\text{OW-ATK1}^{\mathcal{A}} \Rightarrow 1]$ , where game  $[\text{IND-ATK1}, \cdot]$  and  $[\text{OW-ATK1}, \cdot]$  are described in Figure 19.

We say that 2PKE is  $[\text{IND-ATK1}, \cdot]$  secure, if  $\text{Adv}_{2\text{PKE}}^{[\text{IND-ATK1}, \cdot]}(\mathcal{A})$  is negligible; that 2PKE is  $[\text{OW-ATK1}, \cdot]$  secure, if  $\text{Adv}_{2\text{PKE}}^{[\text{OW-ATK1}, \cdot]}(\mathcal{A})$  is negligible, for any PPT adversary  $\mathcal{A}$ . The  $[\cdot, \text{IND-ATK0}]$  and  $[\cdot, \text{OW-ATK0}]$  security can be defined in the same way. Here for avoiding repetition we omit their descriptions.

**[ATK1, ATK0] security.** The  $[\text{ATK1}, \text{ATK0}]$  security is similar with that in 2-key KEM.

**Reduction**  $[\text{IND-CCA}, \cdot] \Rightarrow [\text{OW-CCA}, \cdot]$ .

**Lemma 7.** For any adversary  $\mathcal{A}$  attacks the  $[\text{IND-CPA}, \cdot]$  security on  $pk_1$  with message space  $M$ , there exists an adversary  $\mathcal{B}$  with the same running time as that of  $\mathcal{A}$  such that

$$\text{Adv}_{\text{PKE}}^{[\text{OW-CPA}, \cdot]}(\mathcal{A}) \leq 1/|M| + 2 \cdot \text{Adv}_{\text{PKE}}^{[\text{IND-CPA}, \cdot]}(\mathcal{B}).$$

The lemma still works when attacks on  $pk_0$ .

*Proof.* We process the proof by constructing the adversary  $\mathcal{B}$  using  $\mathcal{A}$  as subroutine shown in Figure 20.

IND-CPA adversary $\mathcal{B}^{\mathcal{A}}$ :
01 $m_1, m_0 \leftarrow M$
02 $(\text{state}, pk_0^*) \leftarrow \mathcal{B}_1^{\text{leak0}}(pk_1)$
03 On receiving $c^*$
04 $m' \leftarrow \mathcal{B}_2^{\text{leak0}}(\text{state}, c^*)$
05 if $m' = m_1$ return 1 else 0.

**Fig. 20.** Reduction between  $[\text{OW-CPA}, \cdot]$ , and  $[\text{IND-CPA}, \cdot]$  security for 2-key PKE.

In the construction of adversary  $\mathcal{B}$ , since  $\Pr[b' = 1 | b = 1] \geq \text{Adv}_{\text{PKE}}^{[\text{OW-CPA}, \cdot]}(\mathcal{A})$ , and if  $b = 0$  the probability that  $\mathcal{B}$  outputs  $m_1$  is less than  $1/|M|$ ,

$$\begin{aligned} \text{Adv}_{\text{PKE}}^{[\text{IND-CPA}, \cdot]}(\mathcal{C}) &= \frac{1}{2}(\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]) \\ &\geq \frac{1}{2}(\text{Adv}_{\text{PKE}}^{[\text{Partial-OW-CPA}, \cdot]}(\mathcal{B}) - 1/|M|). \end{aligned}$$

## Appendix G: Proof of Theorem 6 about Twin-ElGamal

As the proofs of  $[\text{IND-CPA}, \cdot]$  and  $[\cdot, \text{IND-CPA}]$  security are similar, we only show the proof for  $[\text{IND-CPA}, \cdot]$  security here. Let  $\mathcal{A}$  be the adversary executing in the  $[\text{IND-CPA}, \cdot]$  game that is denoted as game  $G_0$ . Then  $\text{Adv}_{2\text{PKE}_{\text{cpaddh}}}^{[\text{IND-CPA}, \cdot]}(\mathcal{A}) = |\Pr[b' = b \text{ in } G_0] - 1/2|$ . In game  $G_1$ ,  $g^{r_1}$  and  $h_1^{r_1}$  used in the challenge ciphertext are substituted with uniform random values  $(g^*, h_1^*)$ . There exists an algorithm  $\mathcal{B}$  such that  $\Pr[b' = b \text{ in } G_0] - \Pr[b' = b \text{ in } G_1] \leq \text{Adv}_{\mathcal{B}}^{\text{ddh}}$ .  $\mathcal{B}$  works as following: on receiving DDH challenge  $(g, h_1, g', h_1')$ , set  $pk_1 = (g, h_1)$ . After receiving  $pk_0 = (g, h_0)$  from  $\mathcal{A}$ , return challenge ciphertext  $(g', g^{r_0}, h_1' h_0^{r_0} \cdot m_b)$ . Then on receiving  $b'$ , to guess the value of  $b$ , return  $b \stackrel{?}{=} b'$ . If  $(g, h_1, g', h_1')$  is a DDH instance, this is exactly  $G_0$ ; If  $(g, h_1, g', h_1')$  is a random instance, this is exactly  $G_1$ . Note that in  $G_1$ ,  $h_1^* h_0^{r_0} \cdot m_b$  is uniformly distributed and independent of  $b$ , therefore  $\Pr[b = b' \text{ in } G_1] = 1/2$ .

## Appendix H: Proof of Theorem 7 related to modified FO transform.

**Sketch of proof:** The main idea of the proof is to simulate the decapsulation oracle without part of the secret keys. This can be achieved by replacing the decryption using secret keys with “re-encryption”. That is to answer the decapsulation oracle the challenger chooses a random key, while to answer the random oracle queries for encapsulated key with public keys together with the plaintext and ciphertext, the challenger “re-encrypts” it so as to maintain the consistency. (The challenger may answer the random oracle firstly and then the decapsulation oracle). In order to simulate the random oracle queries,  $pk'_0$  (chosen by adversary) should be included in the inputs to the random oracle. For the same reason,  $pk'_1$  also should be included in the inputs to the random oracle.

**Formal Proof:** The only decapsulation failure 2KEM may happen only if the decryption of 2PKE fails. Consider the CORR-RO game of 2PKE, where the adversaries send at most  $q_G$  queries to  $G$  which may lead to the decryption failure (namely,  $\text{Dec}(sk_1, sk'_0, \text{Enc}(pk_1, pk'_0, m; G(m))) \neq m$  for  $pk'_0 \in [L_0]_1$  when attacking  $pk_1$ ). Since 2PKE is  $\delta$ -correct and  $G$  outputs independent randomness, each query of  $G$  exhibits a correctness error with probability  $\delta$ . The probability that at least 1 query of  $G$  exhibits a correctness error is  $1 - (1 - \delta)^{q_G} \leq q_G \delta$ . Hence  $\delta_1 = q_G \delta$ .

To show the security, we reduce the IND-CCA security against  $pk_1$  to the IND-CPA security only. The reduction of IND-CCA security against  $pk_0$  to IND-CPA security is almost the same. Consider the sequence of games in Figure 21. Let  $S_i$  be the probability that adversary  $\mathcal{C}$  outputs 1 in Game  $i$ .

<b>Games</b> $G_0$ - $G_5$	$H(pk_1, pk'_0, m, c)$	// $G_0$ - $G_5$
01 $(pk'_1, sk'_1) \leftarrow \text{KeyGen1}'$ , $pk_1 = pk'_1$ ;	28 if $\exists (pk_1, pk'_0, m, c, K) \in \mathcal{L}_H$ return $K$	
02 $s_1    t_1 \leftarrow \mathcal{M}$ , $sk_1 = (sk'_1, s_1)$	29 $K \leftarrow \mathcal{K}$	
03 $L_0 = \{(-, -, -)\}$	30 if $pk'_0 \in [L_0]_1$	
04 $(state, pk_0^*) \leftarrow \mathcal{C}_1^{\text{DECAPS}, \mathcal{O}_{\text{leak}_0}}(pk_1)$	31 if $m = s_1    s_0$ , $\text{flag}_s = \text{ture}$ , <b>abort</b>	// $G_1$ - $G_5$
05 $m^* \leftarrow \mathcal{M}$	32 if $\text{Dec}(sk_1, sk'_0, c) = m$	
06 $c^* \leftarrow \text{Enc}(pk_1, pk_0, m^*, G(m^*))$	$\wedge \text{Enc}(pk_1, pk'_0, m; G(m)) = c$	// $G_2$
07 $K_0^* = H(pk_1, pk_0^*, m^*, c^*)$	33 if $\text{Enc}(pk_1, pk'_0, m; G(m)) = c$	// $G_3$ - $G_5$
08 $K_1^* \leftarrow \{0, 1\}^n$	34 if $pk'_0 = pk_0^* \wedge c = c^*$	// $G_4$ - $G_5$
09 $b \leftarrow \{0, 1\}$	35 $\text{flag}_{\text{in}}^H = \text{ture}$ , <b>abort</b>	// $G_4$ - $G_5$
10 $b' \leftarrow \mathcal{C}_2^{\text{DECAPS}, H, G}(state, c^*, K_b^*)$	36 if $\exists K'$ , s.t. $(pk_1, pk'_0, c, K') \in \mathcal{L}_D$	// $G_2$ - $G_5$
11 return $b \stackrel{?}{=} b'$	37 $K = K'$	// $G_2$ - $G_5$
<b>DECAPS</b> $(pk'_0, c)$	38 else $\mathcal{L}_D = \mathcal{L}_D \cup \{(pk_1, pk'_0, c, K)\}$	// $G_2$ - $G_5$
12 if $pk'_0 \notin [L_0]_1$ , <b>abort</b>	39 if $pk'_0 \notin [L_0]_1 \wedge (pk'_0, m, c) = (pk_0^*, m^*, c^*)$	// $G_5$
13 if $pk'_0 \in [L_0]_1 \wedge (pk'_0, c) = (pk_0^*, c^*)$ , <b>abort</b>	40 $\text{flag}_{\text{out}}^H = \text{ture}$ , <b>abort</b>	// $G_5$
14 $m' = \text{Dec}(sk_1, sk'_0, c)$ ,	41 $\mathcal{L}_H = \mathcal{L}_H \cup \{(pk_1, pk'_0, m, c, K)\}$	
15 $c' = \text{Enc}(pk_1, pk'_0, m'; G(m'))$	42 return $K$	
16 if $(m', c') = (\perp, c)$	$G(m)$	// $G_0$ - $G_5$
17 return $K = H(pk_1, pk'_0, s_1    s_0, c)$	43 if $\exists r$ , s. t. $(m, r) \in \mathcal{L}_G$	
18 if $(m', c') = (\perp, c)$	44 return $r$	
19 return $K = H'(pk_1, pk'_0, c)$	45 $r \leftarrow \mathcal{R}$	
20 if $(m', c') = (s_1    s_0, c)$	46 $\mathcal{L}_G = \mathcal{L}_G \cup \{(m, r)\}$	
21 return $K = H'(pk_1, pk'_0, c)$	47 return $r$	
22 return $K = H(pk_1, pk'_0, m', c)$	$\mathcal{O}_{\text{leak}_0}$	
23 if $\exists K$ s.t. $(pk_1, pk'_0, c, K) \in \mathcal{L}_D$	48 $r_0^i \leftarrow \{0, 1\}^*$	
24 return $K$	49 $(pk_0^{i'}, sk_0^{i'}) \leftarrow \text{KeyGen0}'(r_0^i)$	
25 else $K \leftarrow \mathcal{K}$	50 $s_0^i \leftarrow \{0, 1\}^*$ , $sk_{0,i} = (sk_{0,1}^i, s_0)$	
26 $\mathcal{L}_D = \mathcal{L}_D \cup \{(pk_1, pk'_0, c, K)\}$	51 $pk_0^i = pk_0^{i'}$	
27 return $K$	52 $L_0 = L_0 \cup \{(pk_0^i, sk_0^i, r_0^i    s_0^i)\}$	
	53 Return $(pk_0^i, sk_0^i, r_0^i    s_0^i)$	

**Fig. 21.** Game 1-Game 5 for the proof of Theorem 7.

**Game 0:** This is the original [IND-CCA, ·] game against  $pk_1$  with  $\mathcal{C}$ , and

$$|\Pr[S_0] - 1/2| = \text{Adv}_{2\text{KEM}}^{\text{IND-CCA}, \cdot}(\mathcal{C}).$$

**Game 1:** In this Game, we add a flag  $\text{flag}_s$  and set it to be true and abort when  $H(pk_1, pk'_0, s_1 || s_0, \cdot)$  is queried (line 31). At the same time, in DECAPS, replace the condition that if  $m' = \perp$ ,  $K = H(pk_1, pk'_0, m', c)$  (line 16-17) with that if  $m' = \perp$  or  $s_1 || s_0$ ,  $K = H'(pk_1, pk'_0, c)$  (line 18-21), where  $H'$  is an internal random oracle. This will only be noticed if  $\mathcal{C}$  queries  $H$  with  $(pk_1, pk'_0, s_1 || s_0, \cdot)$ , as it will be abort. Since  $s_1$  is uniformly random over  $\{0, 1\}^l$ , we have  $|\Pr[S_1] - \Pr[S_0]| = \frac{q_H}{2^l}$ .

**Game 2:** In this game, the DECAPS oracle does not use the secret key to decapsulate any longer. We maintain two hash lists  $\mathcal{L}_D$  and  $\mathcal{L}_H$  and guarantee the consistency by testing if  $\text{Dec}(sk_1, sk'_0, c) = m \wedge \text{Enc}(pk_1, pk'_0, m; G(m))$  (line 32) during the  $H$  queries.  $(pk_1, pk'_0, m, c, K) \in \mathcal{L}_H$  implies one of the following two cases happens, one of which is that  $(pk_1, pk'_0, m, c)$  was queried on  $H$  and  $H$  returns a random  $K$ , and another is that for  $pk'_0 \in [L_0]_1$ ,  $(pk_1, pk'_0, c, K) \in \mathcal{L}_D$  and  $\text{Dec}(sk_1, sk'_0, c) = m \wedge \text{Enc}(pk_1, pk'_0, m; G(m))$ .

To show the identity of Game 1 and Game 2 from the point view of  $\mathcal{C}$ , consider the following cases for fixed  $pk'_0, c, m'$  that  $\text{Dec}(sk_1, sk'_0, c) = m \wedge \text{Enc}(pk_1, pk'_0, m; G(m))$ .

- Case 1:  $pk'_0 \notin [L_0]_1$ . The decapsulation oracle DECAPS aborts in Game 1 and 2. The oracle  $H$  outputs a random  $K$  in both two games in this case. For  $pk'_0 \notin [L_0]_1$ , since the  $\text{DECAPS}(pk_1, pk'_0, \cdot)$  results in abort, the queries of  $H(pk_1, pk'_0, m, c)$  will return a uniformly random key, as in Game  $G_1$ .
- Case 2:  $pk'_0 \in [L_0]_1 \wedge m' \in \{\perp, s_1 || s_0\} \wedge c' = c$ . Since  $H(pk_1, pk'_0, \perp, c)$  is not allowed and  $H(pk_1, pk'_0, s_1 || s_0, c)$  results in abort, the random oracle  $H$  would not add  $(pk_1, pk'_0, c, K)$  to  $\mathcal{L}_D$  in this case. The  $\text{DECAPS}(pk_1, pk'_0, c)$  will return a totally random key as in Game 1.
- Case 3:  $pk'_0 \in [L_0]_1 \wedge m' \notin \{\perp, s_1 || s_0\} \wedge c' = c$ . In Game 1, the DECAPS oracle and  $H$  oracle are consistent, as the DECAPS returns the key  $H(pk_1, pk'_0, \text{Dec}(sk_1, sk'_0, c), c)$  by querying  $H$ . In Game 2, the lists  $\mathcal{L}_D$  and  $\mathcal{L}_H$  check each other firstly, and help to maintain the consistency by verifying the condition  $m = \text{Dec}(sk_1, sk'_0, c) \wedge c = \text{Enc}(pk_1, pk'_0, m; G(m))$  (line 32) in two cases:  $\mathcal{C}$  may queries  $H$  on  $(pk_1, pk'_0, m, c)$  first, then DECAPS on  $(pk'_0, c)$ ; or the other way around.
  - If  $\mathcal{C}$  queries  $H$  on  $(pk_1, pk'_0, m, c)$  first, in this case (by checking  $m = \text{Dec}(sk_1, sk'_0, c) \wedge c = \text{Enc}(pk_1, pk'_0, m; G(m))$  (line 32)), there is no entry  $(pk'_0, c, K)$  in  $\mathcal{L}_D$  yet. In addition to add  $(pk_1, pk'_0, m, c, K \leftarrow \mathcal{K})$  to  $\mathcal{L}_H$ ,  $H$  also adds  $(pk'_0, c, K)$  to  $\mathcal{L}_D$ . When  $(pk'_0, c)$  is queried to DECAPS, it returns  $K$  from  $\mathcal{L}_D$ .
  - If  $\mathcal{C}$  queries DECAPS on  $(pk'_0, c)$  first, it adds  $(pk'_0, c, K \leftarrow \mathcal{K})$  to  $\mathcal{L}_D$  to declare  $\text{DECAPS}(pk'_0, c) = K$ . When  $\mathcal{C}$  queries  $H$  on  $(pk_1, pk'_0, m, c)$  later, if the decryption and re-encrypt condition (as in line 32) are true,  $H$  adds  $(pk_1, pk'_0, m, c, K)$  to  $\mathcal{L}_H$  to declare that  $H(pk_1, pk'_0, m, c) = K$ . Thus  $H(pk_1, pk'_0, m, c) = K = \text{DECAPS}(pk'_0, c)$ .

From the analysis in sub-cases, the view of  $\mathcal{C}$  is identical to that in Games 1 and  $\Pr[S_2] = \Pr[S_1]$ .

**Game 3:** In this game, we replace the condition  $m = \text{Dec}(sk_1, sk'_0, c) \wedge c = \text{Enc}(pk_1, pk'_0, m; G(m))$  (line 32) with  $c = \text{Enc}(pk_1, pk'_0, m; G(m))$  (line 33), which does not check  $m = \text{Dec}(sk_1, sk'_0, c)$  any more. The Game 2 and Game 1 are different only when  $\text{Dec}(sk_1, sk'_0, \text{Enc}(pk_1, pk'_0, m; G(m))) \neq m$  happens.  $\mathcal{C}$  makes at most  $q_G$  queries to  $G$ , which may introduce the correctness error, namely,  $\text{Dec}(sk_1, sk'_0, \text{Enc}(pk_1, pk'_0, m; G(m))) \neq m$ , for  $(pk'_0, sk'_0) \in L$ . Since 2PKE is  $\delta$ -correct and  $G$  outputs independent randomness, each query of  $G$  exhibits a correctness error with probability  $\delta$ . The probability that at least 1 query of  $G$  exhibits a correctness error is  $1 - (1 - \delta)^{q_G} \leq q_G \delta$ . Thus  $|\Pr[S_3] - \Pr[S_2]| = q_G \delta$ .

**Game 4:** In this game, we add a flag  $\text{flag}_{\text{in}}^H$  (line 34-35) and abort when it is true. The difference between Game 4 and Game 3 is bounded by the events  $\text{flag}_{\text{in}}^H = \text{ture}$ , thus,  $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[\text{flag}_{\text{in}}^H = \text{ture}]$ .

To bound  $\Pr[\text{flag}_{\text{in}}^H = \text{ture}]$ , we construct an adversary  $\mathcal{A}_{\text{in}}$  against the [OW-CPA, ·] security of 2PKE when  $pk_0^* \in [L_0]_1$ , as in Figure 22. The simulation of  $H$  and ENCAPS is the same with Game 3. And it is perfectly simulated because the decryption key  $sk_1$  and  $sk'_0$  (since  $pk'_0 \in [L_0]_1$ ) are not required. After  $\mathcal{C}$  outputs  $b'$ , the adversary  $\mathcal{A}$  checks the list  $\mathcal{L}_H$ , if  $\exists (pk_1, pk_0^*, m', c, K) \in \mathcal{L}_H$ , outputs  $m'$ ; else abort.  $\text{flag}_{\text{in}}^H = \text{ture}$  means that  $\mathcal{C}$  queries  $H(pk_1, pk_0^*, m', c)$  and  $(pk_1, pk_0^*, m', c, K') \in \mathcal{L}_H$ , thus  $m' = m^*$ .

Hence  $\Pr[\text{flag}_{\text{in}}^H = \text{ture}] = \text{Adv}_{2\text{PKE}}^{[\text{OW-CPA}, \cdot]}(\mathcal{A}_{\text{in}})$ .

**Game 5:** In this game, we add a flag  $\text{flag}_{\text{out}}^H$  (line 39-40) and abort when it is true. The difference between Game 4 and Game 5 is bounded by the events  $\text{flag}_{\text{out}}^H = \text{ture}$ , thus,  $|\Pr[S_4] - \Pr[S_5]| \leq \Pr[\text{flag}_{\text{out}}^H = \text{ture}]$ .

In this game,  $H(pk_1, pk_0^*, m^*, c^*)$  will not be given to  $\mathcal{C}$  in both cases  $pk_0^* \in [L_0]_1$  and  $pk_0^* \notin [L_0]_1$ , which means that  $b$  is independent with  $\mathcal{C}$ 's view. Hence  $\Pr[S_5] = 1/2$ .

To bound  $\Pr[\text{flag}_{\text{out}}^H = \text{ture}]$ , we construct an adversary  $\mathcal{A}_{\text{out}}$  against the  $[\text{OW-CPA}, \cdot]$  security of 2PKE when  $pk_0^* \notin [L_0]_1$  as in Figure 22.

[OW-CPA, ·] adversary $\mathcal{A}_{\text{in}}$ and $\mathcal{A}_{\text{out}}$		[IND-CPA, ·] adversary $\mathcal{A}'$			
01	$K^* \leftarrow \mathcal{K};$	$\mathcal{A}_{\text{in}}, \mathcal{A}_{\text{out}}$	01	$m_1, m_0 \leftarrow \mathcal{M}$	
02	$s_1    t_1, s_0    t_0 \leftarrow \mathcal{M}$	$\mathcal{A}_{\text{in}}, \mathcal{A}_{\text{out}}$	02	$K^* \leftarrow \mathcal{K}, s_1, s_0 \leftarrow \mathcal{M};$	
03	$(state, pk_0^*) \leftarrow \mathcal{C}_1^{\text{DECAPS}, \mathcal{O}_{\text{leak0}}, H, G}(pk_1)$	$\mathcal{A}_{\text{in}}, \mathcal{A}_{\text{out}}$	03	$(state, pk_0^*) \leftarrow \mathcal{C}_1^{\text{DECAPS}, \mathcal{O}_{\text{leak0}}, H, G}(pk_1)$	
04	$b' \leftarrow \mathcal{C}_2^{\text{DECAPS}, \mathcal{O}_{\text{leak0}}, H, G}(state, c^*, K^*)$	$\mathcal{A}_{\text{in}}, \mathcal{A}_{\text{out}}$	04	$b'' \leftarrow \mathcal{C}_2^{\text{DECAPS}, \mathcal{O}_{\text{leak0}}, H, G}(state, c^*, K^*)$	
05	If $\exists (pk_1, pk_0^*, m', c, K') \in \mathcal{L}_H$	$\mathcal{A}_{\text{in}}$	05	$\mathcal{L}'_H \leftarrow \mathcal{L}_H \cap \{(pk_1, pk_0^*, \cdot, c^*, \cdot)\}$	
06	return $m'$	$\mathcal{A}_{\text{in}}$	06	if $ \mathcal{L}'_H(m_1)  >  \mathcal{L}'_H(m_0) $ , $b' = 1$	
07	else return $\perp$ .	$\mathcal{A}_{\text{in}}$	07	if $ \mathcal{L}'_H(m_1)  <  \mathcal{L}'_H(m_0) $ , $b' = 0$	
08	$m' \leftarrow \mathcal{L}_H \cap \{(pk_1, pk_0^*, \cdot, c^*, \cdot)\}$ ,	$\mathcal{A}_{\text{out}}$	08	if $ \mathcal{L}'_H(m_1)  =  \mathcal{L}'_H(m_0) $ , $b' \leftarrow \{0, 1\}$	
09	return $m'$ .	$\mathcal{A}_{\text{out}}$	09	return $b'$	

**Fig. 22.** The  $[\text{OW-CPA}, \cdot]$  adversary  $\mathcal{A}_{\text{in}}$  in Game 4 and  $\mathcal{A}_{\text{out}}$  in Game 5 for the proof Theorem 21; The  $[\text{IND-CPA}, \cdot]$  adversary  $\mathcal{A}'$  for the proof Theorem 21 in Game 5.  $\mathcal{L}'_H(m_1)$  is the set of all  $(pk_1, pk_0^*, m_1 || \cdot, c^*, \cdot) \in \mathcal{L}'_H$ . The DEC PAS,  $H$  and  $G$  oracle are those (in corresponding Game) in Figure 21

If  $pk_0^* \notin [L_0]_1$ , on input  $pk_1$  and  $c^* \leftarrow \text{Enc}(pk_1, pk_0^*, m^*)$ , it is perfectly simulated in Game 5. If  $\text{flag}_{\text{out}}^H = \text{ture}$ , there exists  $(pk_1, pk_0^*, m^*, c^*, \cdot) \in \mathcal{L}_H$ , and  $\mathcal{A}$  returns the correct  $m' = m^*$  with probability at most  $1/q_H$ . Hence  $\Pr[\text{flag}_{\text{out}}^H] \leq q_H \cdot \text{Adv}_{2\text{PKE}}^{[\text{OW-CPA}, \cdot]}(\mathcal{A})$ .

To sum up, we get first claim of the theorem.

If 2PKE is  $[\text{IND-CPA}, \text{IND-CPA}]$ -secure, the reduction is tight. By Lemma 20, in Game 4 it holds  $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[\text{flag}_{\text{in}}^H = \text{ture}] \leq \text{Adv}_{2\text{PKE}}^{[\text{OW-CPA}, \cdot]}(\mathcal{A}) \leq 1/|M| + 2 \cdot \text{Adv}_{2\text{PKE}}^{[\text{IND-CPA}, \cdot]}(\mathcal{A}')$ . In Game 5, to bound  $\Pr[\text{flag}_{\text{out}}^H = \text{ture}]$ , we construct an adversary  $\mathcal{A}'$  against the  $[\text{IND-CPA}, \cdot]$  security of 2PKE when  $pk_0^* \notin [L_0]_1$  as in Figure 22. Consider the  $[\text{IND-CPA}, \cdot]$  game with challenge bit  $b$ , denote Bad as the event that  $\mathcal{A}'$  queries  $H$  with  $(pk_1, pk_0^*, m_{1-b} || \cdot, c^*)$ . Since  $m_{1-b}$  is uniformly distributed over  $M$ , we have that  $\Pr[\text{Bad}] \leq q_H/|M|$ . If  $pk_0^* \notin [L_0]_1 \wedge \text{Bad} \wedge \text{flag}_{\text{out}}^H = \text{ture}$ ,  $\mathcal{C}$  queried on  $(pk_1, pk_0^*, m_b || \cdot, c^*)$  and  $|\mathcal{L}'_H(m_b)| \geq |\mathcal{L}'_H(m_{1-b})|$ . If  $pk_0^* \notin [L_0]_1 \wedge \text{Bad} \wedge \text{flag}_{\text{out}}^H = \text{false}$ ,  $\mathcal{C}$  did not query on  $(pk_1, pk_0^*, m_b || \cdot, c^*)$  and  $\Pr[b = b'] = 1/2$ .

Thus we have  $\text{Adv}_{2\text{PKE}}^{[\text{IND-CPA}, \cdot]}(\mathcal{A}') + q_H/|M| \leq |\Pr[b' = b] - 1/2| = |\Pr[\text{flag}_{\text{out}}^H = \text{ture}] + 1/2\text{flag}_{\text{out}}^H = \text{false} - 1/2| = 1/2\Pr[\text{flag}_{\text{out}}^H = \text{ture}]$ .

## Appendix I: Decryption Failure and proof of security for Twin-Kyber

**Decryption Failure:** To handle the decryption failure, for a uniformly random  $\mathbf{y} \in R_q^k$  with  $d_y$ , define  $\mathbf{c}_y = \mathbf{y} - \text{Decomp}_q(\text{Comp}_q(\mathbf{y}, d_y), d_y) \bmod \pm 2^d$  as the output of distribution  $\Psi_{d_y}^k$ . For  $d_{t_1}, d_{t_0}$ , set distributions  $\Psi_{d_{t_1}}^k$  and  $\Psi_{d_{t_0}}^k$ . Set the above parameters as public parameters. Since under Module-LWE assumption,  $\mathbf{A}\mathbf{s}_i + \mathbf{e}_i$ ,  $\mathbf{A}^T \mathbf{r}_i + \mathbf{e}_{4-i}$  ( $i = 0, 1$ ) and  $\mathbf{t}_1^T \mathbf{r}_1 + \mathbf{t}_0^T \mathbf{r}_0 + e$  are indistinguishable from uniform random values. Then, for algorithm Enc, we have  $\mathbf{t}_1 = \mathbf{A}\mathbf{s}_1 + \mathbf{e}_1$  and  $\mathbf{t}_0 = \mathbf{A}\mathbf{s}_0 + \mathbf{e}_0$ . For algorithm Dec, we have  $\mathbf{u}_1 = \mathbf{A}^T \mathbf{r}_1 + \mathbf{e}_3 + \mathbf{c}_{u_1}$ ,  $\mathbf{u}_0 = \mathbf{A}^T \mathbf{r}_0 + \mathbf{e}_4 + \mathbf{c}_{u_0}$ . Thus from the decryption algorithm,

$$E = v - \mathbf{s}_1^T \mathbf{u}_1 - \mathbf{s}_0^T \mathbf{u}_0 = (\mathbf{e}_1 + \mathbf{e}'_1)^T \mathbf{r}_1 + (\mathbf{e}_0 + \mathbf{e}'_0)^T \mathbf{r}_0 + \mathbf{s}_1^T (\mathbf{e}_3 + \mathbf{c}_{u_1}) + \mathbf{s}_0^T (\mathbf{e}_4 + \mathbf{c}_{u_0}) + e + \mathbf{c}_v.$$

Denote every coefficient of  $E$  as  $[E]_i$ , for  $1 \leq i \leq 256$ . From the computation rule over  $R_q$ , all the variables in computing  $[E]_i$  are independent, but they are reused in other summations for  $[E]_j$  for



$j \neq i$ . Although the average-case distribution of each  $[E]_j$  is the same, they are not fully independent. However, [32] and [3]<sup>3</sup> proposed an independence assumption, which states that for  $\mathbf{x}, \mathbf{y}$  and  $\mathbf{z}$  chosen according to  $\beta_\eta^k, \beta_\eta^k$  and  $\Psi_d^k$  the distributions of each two coefficients of  $\mathbf{x}^T \mathbf{y}$  and  $\mathbf{x}^T \mathbf{z}$  are independent. Their assumptions further implies the independence of each bit of  $E$ , here we continue to use the their assumptions.

Let  $\delta_i = 1 - \Pr[[m]_i = [m']_i]$  be the failure probability of the  $i$ -th single bits. Then  $\delta_i = 1 - \Pr[[E]_i < \lceil \frac{q}{4} \rceil]$ . Under the independence assumption, the failure probability of each bit  $\delta_{1bit} = \delta_i$  for any  $1 \leq i \leq 255$ . Thus, the total failure probability is bounded by  $\delta = n\delta_{1bit}$ .

**Proof of Theorem 8**

*Proof.* Without loss of generality, we only show the  $[\text{IND-CPA}, \cdot]$  security here, and the proof for  $[\cdot, \text{IND-CPA}]$  security is similar. The proof proceeds in a sequence of hybrid games. Let game  $G_0$  be the original  $[\text{IND-CPA}, \cdot]$  game, then  $\text{Adv}_{2\text{PKE}_{\text{mlwe}}}^{[\text{IND-CPA}, \cdot]}(\mathcal{A}) = \Pr[b' = b \text{ in } G_0]$ . In Game  $G_1$ ,  $\mathbf{A}\mathbf{s}_1 + \mathbf{e}_1$  is replaced by a uniform random value in  $R_q^k$ . If  $\mathcal{A}$  is able to distinguish  $\mathbf{A}\mathbf{s}_1 + \mathbf{e}_1$  from a unifor random value, then there exists an algorithm  $\mathcal{B}$  to solve the  $k \times k$  Module-LWE problem. That is  $\Pr[b' = b \text{ in } G_0] - \Pr[b' = b \text{ in } G_1] \leq \text{Adv}_{k,k,\eta}^{\text{mlwe}}(\mathcal{B}) \leq \text{Adv}_{k+1,k,\eta}^{\text{mlwe}}(\mathcal{B})$ . In game  $G_2$ , when generating the challenge ciphertext,  $\mathbf{A}^T \mathbf{r}_1 + \mathbf{e}_3$  of line 14, and  $\mathbf{t}_1^T \mathbf{r}_1 + e$  of line 17 are substituted by uniform random values in  $R_q^k$  or  $R_q$ . If  $\mathcal{A}$  is able to distinguish the challenge ciphertext from a random value, then there exists an algorithm  $\mathcal{B}$  to solve the  $(k+1) \times k$  Module-LWE problem. That is  $\Pr[b' = b \text{ in } G_1] - \Pr[b' = b \text{ in } G_2] \leq \text{Adv}_{k+1,k,\eta}^{\text{mlwe}}(\mathcal{B})$ . Specially, given a  $(k+1) \times k$  instance,  $\mathcal{B}$  parses the first  $k$  rows as  $(\mathbf{A}, \mathbf{b})$  and the last rows as  $(\mathbf{t}_1, \mathbf{b}_1)$ , and sets the public key as  $\mathbf{t}_1 = \text{Comp}_q(\mathbf{A}\mathbf{s}_1 + \mathbf{e}_1, d_{t_1})$ . By choosing  $\mathbf{r}_0, \mathbf{e}_4, e$ ,  $\mathcal{B}$  computes the challenge ciphertext including  $\mathbf{u}_1 = \text{Comp}_q(\mathbf{b}, d_{u_1})$ ,  $\mathbf{u}_0 = \text{Comp}_q(\mathbf{A}^T \mathbf{r}_0 + \mathbf{e}_4, d_{u_0})$ , and  $v = \text{Comp}_q(\mathbf{b}_1 + \mathbf{t}_0^T \mathbf{r}_0 + \lceil \frac{q}{2} \rceil c_m, d_v)$ . Finally,  $\mathcal{B}$  just outputs what the adversary  $\mathcal{A}$  returns. Note that in  $G_2$ , the challenge ciphertext is independent of  $b$ , thus  $\Pr[b' = b \text{ in } G_2] = 1/2$ .  $\square$

<sup>3</sup> the assumption is implicitly given in their code