Two-Message Authenticated Key Exchange from Public-Key Encryption

You Lyu and Shengli Liu(✉)

Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200240, China
{vergil,slliu}@sjtu.edu.cn

Abstract. In two-message authenticated key exchange (AKE), it is necessary for the initiator to keep a round state after sending the first round-message, because he/she has to derive his/her session key after receiving the second round-message. Up to now almost all two-message AKEs constructed from public-key encryption (PKE) only achieve weak security which does not allow the adversary obtaining the round state. How to support state reveal to obtain a better security called IND-AA security has been an open problem proposed by Hövelmann et al. (PKC 2020).

In this paper, we solve the open problem with a generic construction of two-message AKE from any CCA-secure Tagged Key Encapsulation Mechanism (TKEM). Our AKE supports state reveal and achieves IND-AA security. Given the fact that CCA-secure public-key encryption (PKE) implies CCA-secure TKEM, our AKE can be constructed from any CCA-secure PKE with proper message space. The abundant choices for CCA-secure PKE schemes lead to many IND-AA secure AKE schemes in the standard model. Moreover, following the online-extractability technique in recent work by Don et al. (Eurocrypt 2022), we can extend the Fujisaki-Okamoto transformation to transform any CPA-secure PKE into a CCA-secure Tagged KEM in the QROM model. Therefore, we obtain the first generic construction of IND-AA secure two-message AKE from CPA-secure PKE in the QROM model. This construction does not need any signature scheme, and this result is especially helpful in the post-quantum world, since the current quantum-secure PKE schemes are much more efficient than their signature counterparts.

Keywords: Authenticated key exchange · State reveal · PKE.

1 Introduction

Authenticated Key Exchange (AKE) is an important technical tool of establishing a secure channel for two communication parties, and is widely deployed in a variety of information systems for security. Running with an AKE protocol, two parties can compute a shared session key which is used for the later communications. The security of AKE requires pseudo-randomness of the session key in case of passive attacks and (implicit or explicit) authentication in case of active

$$\begin{array}{lcr}
\underline{\text{Party } P_i(pk_i, sk_i)}: & & \underline{\text{Party } P_j(pk_j, sk_j)}: \\
(M_1, st) \leftarrow \mathsf{Init}(sk_i, pk_j) & \xrightarrow{\quad M_1 \quad} & \\
\quad \downarrow st & & \\
K_i \leftarrow \mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st) & \xleftarrow{\quad M_2 \quad} & (M_2, K_j) \leftarrow \mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1)
\end{array}$$

Fig. 1: Two-message AKE protocol.

attacks. AKE is a well-studied topic and many generic AKE constructions are available up to now [20,11,10,13]. Generally, AKE relies on public-key primitives for security and its building blocks include public-key encryption (PKE), digital signature (SIG) and key encapsulation mechanism (KEM).

Security Models for AKE. Bellare and Rogaway [4] introduced the original security model, which was later developed to several different models, like CK model, eCK model, CK+ model, etc. Lately, Hövelmanns et al. [11] proposed the so-called IND-AA/IND-StAA models for two-message AKEs. IND-AA model captures not only the classical security requirement of pseudo-randomness for session keys, but also security against key compromise (KCI) attack, reflection attack, state reveal attack, and weak forward security. IND-StAA model is similar to but weaker than IND-AA model, since it does not consider state reveal attack. As pointed by [11,6], IND-AA model is strictly stronger than the CK model, but incomparable to eCK model.

AKE from PKE. There are two essential factors affecting the efficiency of AKE. One is the number of rounds and the other is the efficiency of its building blocks. Clearly the optimal round number is 2 for AKE, so two-message AKE has optimal round efficiency. Among the public key primitives, SIG is often used to achieve authentication for AKE. However, generally SIG is not as efficient as PKE, and this is especially true for PKE/SIG schemes with security against quantum computers. For example, in the NIST post-quantum competition, CRYSTALS-Dilithium (SIG) has key size two times larger than CRYSTALS-Kyber (KEM), its signature size is three times larger than the ciphertext size of CRYSTALS-Kyber (KEM), and its signing time is 10 times slower than the encapsulation algorithm of CRYSTALS-Kyber. This motivates the research [11,12,18] on designing AKE solely from PKE. The AKE schemes proposed in [12,18] are constructed from KEM, but have at least three rounds.

The question of designing two-message AKE from PKE was partially solved by Hövelmanns et al.[11]. Recall that a two-message AKE protocol for parties P_i and P_j is captured by three PPT algorithms as shown in Fig. 1. Let (pk_i, sk_i) (resp.(pk_j, sk_j)) be the public/secret key pairs for P_i (resp.P_j).

(1) $\mathsf{Init}(sk_i, pk_j)$. Initiator P_i invokes $\mathsf{Init}(sk_i, pk_j)$ to generate the first-round message M_1 and a round state st.

(2) $\mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1)$. After receiving M_1, responder P_j invokes $\mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1)$ to generate the second-round message M_2 and the session key K_j.

(3) $\mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st)$. Upon receiving M_2, P_i invokes $\mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st)$ to generate its session key K_i.

Compared with IND-StAA security, IND-AA security allows the adversary to implement a so-called "state reveal attack", which is an active attack with initiator P_i's state st_i. So IND-AA security is strictly stronger than IND-StAA security. In [11], Hövelmanns et al. presented a generic construction of two-message AKE from passively (i.e., CPA) secure PKE in the quantum random oracle model (QROM). However, their AKE construction only achieves weak IND-StAA security, so they left an open problem (section 1.1.5 in [11]):

How to design a generic and efficient two-message
AKE protocol with IND-AA security?

Our Contribution. We solve the open problem in this paper. Our contribution has two folds.

1. We propose a generic construction of IND-AA secure two-message AKE from CCA-secure Tagged-KEM [1], CPA-secure PKE, target collision resistant (TCR) hash function, and pseudo-random function (PRF). The IND-AA security of AKE is proven in the standard model.
   - The existence of one-way function implies PRF and TCR-Hash function, and CCA-secure Tagged-KEM can be constructed by CCA-secure PKE. So our AKE can essentially be constructed from CCA-secure PKE.
   - Given many choices for the standard-model instantiations of the building blocks, we obtain the first generic two-message AKE schemes from PKE with IND-AA security in the standard model.
2. Following the online-extractability technique in [7], we extend the Fujisaki-Okamoto transformation to transform any passively (i.e., CPA) secure PKE into a CCA-secure Tagged KEM in the QROM model. As a result, we obtain the first generic construction of two-message AKE from passively secure PKE with IND-AA security proven in the QROM model.

Comparison. We compare our two AKE constructions, AKE_1 in standard model and AKE_2 in the QROM model, with other AKEs constructed from PKE. Comparing the FSXY scheme [8] in the standard model, our AKE_1 has similar efficiency as FSXY, but shorter secret key and better security of IND-AA. Comparing the $\mathsf{AKE}_{\mathsf{FO}}$ scheme [11] in the QROM model, our AKE_2 has similar efficiency as $\mathsf{AKE}_{\mathsf{FO}}$, but enjoys shorter secret key and better security of IND-AA.
Technique Overview. First we review some security requirements for AKE. Plain security means pseudo-randomness of session key but the adversary $\mathcal{A}$ is neither allowed to corrupt users' secret key nor reveal the initiator's round state. Weak forward security (wFS) asks pseudo-randomness of session key in case of passive attacks but $\mathcal{A}$ may corrupt secret keys of both initiator and responder (in this case $\mathcal{A}$ cannot reveal the initiator's round state to avoid trivial attack). State-reveal security requires that $\mathcal{A}$ is not able to implement successful active attack to learn party's session key even if it obtains the initiator's round state.

Table 1: Comparison of our AKE_1 (in the standard model) and AKE_2 (in QROM) with AKEs constructed from PKE/KEM. **Comm** denotes the communication overhead of the protocols, where "$|\mathsf{C}|$" and "$|\mathsf{pk}|$" denote the size of ciphertext and public key of IND-CCA secure KEM. "$|\mathsf{c}|$" denotes the size of ciphertext of IND-CPA secure PKE/KEM. "λ" denotes the security parameter. $(|\mathsf{c}|+|\mathsf{C}|)$ (w.r.t. $(|\mathsf{c}|+|\mathsf{c}|)$) in AKE_1 (w.r.t. AKE_2) denotes the size of ciphertext of IND-CPA secure PKE, because the ciphertext is an (KEM + DEM) encryption of the ciphertext of IND-CCA (w.r.t. IND-CPA) secure KEM. **CompI** and **CompR** denote the computational complexity of initiator and responder. "E" and "D" denote one encapsulation and one decapsulation of an IND-CCA secure KEM, and "e" and "d" denote one encapsulation and one decapsulation of IND-CPA secure KEM. **KeySize** denotes the size of long-term secret key per user. "$|\mathsf{sk}_{\mathsf{cca}}|, |\mathsf{sk}_{\mathsf{cpa}}|, |\mathsf{sk}_{\mathsf{prf}}|, |\mathsf{sk}_{\mathsf{se}}|$" denote the secret key sizes of IND-CCA secure KEM, IND-CPA secure PKE/KEM, PRF and symmetric encryption, respectively.

AKE schemes	Comm	CompI	CompR	KeySize	Security	Model
FSXY[8]	$\lvert\mathsf{pk}\rvert + 2\lvert\mathsf{C}\rvert + \lvert\mathsf{c}\rvert$	$\mathsf{E}+\mathsf{D}+\mathsf{d}$	$\mathsf{E}+\mathsf{D}+\mathsf{e}$	$\lvert\mathsf{sk}_{\mathsf{cca}}\rvert + \lvert\mathsf{sk}_{\mathsf{prf}}\rvert$	IND-stAA	Standard
Our AKE_1	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{C}\rvert + (\lvert\mathsf{c}\rvert + \lvert\mathsf{C}\rvert) + \lambda$	$\mathsf{E}+\mathsf{D}+\mathsf{d}$	$\mathsf{E}+\mathsf{D}+\mathsf{e}$	$\lvert\mathsf{sk}_{\mathsf{cca}}\rvert$	IND-AA	Standard
JKRS[13]	$\lvert\mathsf{pk}\rvert + 2\lvert\mathsf{C}\rvert + \lvert\mathsf{c}\rvert$	$\mathsf{E}+\mathsf{D}+\mathsf{d}$	$\mathsf{E}+\mathsf{D}+\mathsf{e}$	$\lvert\mathsf{sk}_{\mathsf{cca}}\rvert + \lvert\mathsf{sk}_{\mathsf{se}}\rvert$	IND-AA	ROM
$\mathsf{AKE}_{\mathsf{FO}}$[11]	$\lvert\mathsf{pk}\rvert + 3\lvert\mathsf{c}\rvert$	$2\mathsf{e}+2\mathsf{d}$	$3\mathsf{e}+\mathsf{d}$	$\lvert\mathsf{sk}_{\mathsf{cpa}}\rvert + \lvert\mathsf{sk}_{\mathsf{prf}}\rvert$	IND-stAA	QROM
Our AKE_2	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{c}\rvert + (\lvert\mathsf{c}\rvert + \lvert\mathsf{c}\rvert) + \lambda$	$2\mathsf{e}+2\mathsf{d}$	$3\mathsf{e}+\mathsf{d}$	$\lvert\mathsf{sk}_{\mathsf{cpa}}\rvert$	IND-AA	QROM

Party $P_i(pk_i, sk_i)$:

$m_1 \leftarrow_\$ \mathcal{M}, c_1 \leftarrow \mathsf{Enc}(pk_j, m_1)$

$\boxed{(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{Gen}}$

$st = (m_1, \boxed{\tilde{sk}})$

$m_2 \leftarrow \mathsf{Dec}(sk_i, c_2)$

$\boxed{\tilde{m} \leftarrow \mathsf{Dec}(\tilde{sk}, \tilde{c})}$

$K_i := H(m_1|m_2|M_1|M_2|\boxed{\tilde{m}})$

$$\xrightarrow{M_1 = (\boxed{\tilde{pk}}, c_1)}$$

$$\xleftarrow{M_2 = (\boxed{\tilde{c}}, c_2)}$$

Party $P_j(pk_j, sk_j)$:

$m_1 \leftarrow \mathsf{Dec}(sk_j, c_1)$

$m_2 \leftarrow_\$ \mathcal{M}, c_2 \leftarrow \mathsf{Enc}(pk_i, m_2)$

$\boxed{\tilde{m} \leftarrow_\$ \mathcal{M}, \tilde{c} \leftarrow \mathsf{Enc}(\tilde{pk}, \tilde{c})}$

$K_j := H(m_1|m_2|M_1|M_2|\boxed{\tilde{m}})$

Fig. 2: Plain AKE (without gray box) and $\mathsf{AKE}_{\mathsf{FO}}$[11] (with gray box).

We start with a plain construction of AKE which has plain security but has neither forward security nor state-reveal security. Then we show why the $\mathsf{AKE}_{\mathsf{FO}}$ scheme in [11] achieves wFS security but suffers from state-reveal attack. Lastly, we describe how to design our AKE to resist the state-reveal attack while keeping the wFS security, so that IND-AA security is achieved.

Plain AKE. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. There is a plain construction of AKE. P_i and P_j just use its peer's public key to encrypt a random message. Let $c_1 \leftarrow \mathsf{Enc}(pk_j, m_1)$ and $c_2 \leftarrow \mathsf{Enc}(pk_i, m_2)$. P_i has state $st_i := m_1$. After exchanging the ciphertexts c_1 and c_2, they can decrypt the ciphertexts to recover m_1 and m_2 respectively. The final session key is computed by $K_i = K_j = H(m_1|m_2|c_1|c_2)$. See Fig. 2.

– Without the knowledge of sk_i and sk_j, the session key $H(m_1|m_2|c_1|c_2)$ is pseudo-random (assuming by now H is a random oracle). Therefore, this

Party $P_i(pk_i, sk_i)$		Party $P_j(pk_j, sk_j)$
$m_{11}, m_{12} \leftarrow_{\$} \mathcal{M}, \sigma := \mathsf{H}(m_{12})$		
$c_1 \leftarrow \mathsf{Enc}(pk_j, m_{11}\|m_{12}\|i)$		$m_{11}\|m_{12}\|i \leftarrow \mathsf{Dec}(sk_j, c_1)$
$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{Gen}$	$\xrightarrow{M_1 = (\tilde{pk}, c_1)}$	$m_{21}, m_{22} \leftarrow_{\$} \mathcal{M}$
$\downarrow st = (m_{11}, \tilde{sk}, \sigma)$		$c_2 \leftarrow \mathsf{Enc}(pk_i, m_{21}\|m_{22})$
$c_2 \leftarrow \mathsf{Dec}(\tilde{sk}, \tilde{c})$		$\tilde{c} \leftarrow \mathsf{Enc}(\tilde{pk}, c_2)$
$m_{21}\|m_{22} \leftarrow \mathsf{Dec}(sk_i, c_2)$		$C = m_{12} \oplus m_{22}$
If $\mathsf{H}(m_{22} \oplus C) \neq \sigma$: abort	$\xleftarrow{M_2 = (\tilde{c}, C)}$	$K_j := \mathsf{PRF}(m_{11}, M_1\|M_2)$
$K_i := \mathsf{PRF}(m_{11}, M_1\|M_2)$		$\oplus\mathsf{PRF}(m_{21}, M_1\|M_2)$
$\oplus\mathsf{PRF}(m_{21}, M_1\|M_2)$		

Fig. 3: Our generic construction of AKE.

plain AKE has plain security if the underlying PKE has CCA security. The CCA security is required for PKE so that the security reduction algorithm is able to compute session keys for other session instance of the same user.

– If P_i and P_j are corrupted, adversary $\mathcal{A}$ obtains sk_i and sk_j, then $\mathcal{A}$ is also able to decrypt c_1, c_2 to obtain m_1, m_2. Obviously $\mathcal{A}$ also gets the session key $H(m_1|m_2|c_1|c_2)$. Therefore, this plain AKE has no wFS security.
– If state $st_i = m_1$ is exposed to $\mathcal{A}$, then $\mathcal{A}$ can impersonate P_j to send $\hat{c}_2 \leftarrow \mathsf{Enc}(pk_i, \hat{m}_2)$ to P_i. Obviously $\mathcal{A}$ can compute P_i's session key $K_i := H(m_1|\hat{m}_2|c_1|\hat{c}_2)$. Therefore, this plain AKE cannot resist state reveal attack.

$\mathsf{AKE}_{\mathsf{FO}}$[11] with wFS security. To obtain wFS security, an ephemeral public/secret key pair $(\tilde{pk}, \tilde{sk})$ is augmented to the plain AKE, resulting in $\mathsf{AKE}_{\mathsf{FO}}$[11]. P_i also sends $\tilde{pk}$ to P_j and P_j provides P_i a ciphertext $\tilde{c}$ encrypting another random message $\tilde{m}$ under $\tilde{pk}$. The state of P_i is $st_i = (m_1, \tilde{sk})$. Then P_i and P_j can share the ephemeral random $\tilde{m}$, and embed it in the input of the hash function so that $K_i = K_j = H(m_1|m_2|M_1|M_2|\tilde{m})$, where $M_1 = (\tilde{pk}, c_1)$ and $M_2 = (\tilde{c}, c_2)$. See also Fig. 2.

– Even if $\mathcal{A}$ obtains sk_i and sk_j by corruption, $\mathcal{A}$ cannot determine $\tilde{m}$ without the knowledge of $\tilde{sk}$. Therefore, $K_i = K_j = H(m_1|m_2|M_1|M_2|\tilde{m})$ is still random to $\mathcal{A}$. So $\mathsf{AKE}_{\mathsf{FO}}$ achieves wFS security.
– If state $st_i = (m_1, \tilde{sk})$ is exposed to $\mathcal{A}$, then $\mathcal{A}$ can impersonate P_j in the protocol and share a session key with P_i, since it knows m_1 and can choose $\tilde{m}$ and $\hat{m}_2$ so as to derive P_i's session key $K_i = H(m_1|\hat{m}_2|M_1|M_2|\tilde{m})$. Therefore, $\mathsf{AKE}_{\mathsf{FO}}$ cannot resist state reveal attack.

Our Approach to IND-AA security. In plain AKE and $\mathsf{AKE}_{\mathsf{FO}}$, m_1 has two roles. One is used to derive the session key, and the other is used as a token to authenticate P_j since only P_j is able to decrypt c_1 to obtain m_1 (when sk_j is not corrupted). However, with state reveal, $\mathcal{A}$ obtains token m_1 from st_i, so it can always impersonate P_j in plain AKE and $\mathsf{AKE}_{\mathsf{FO}}$. That is why they suffer from the state-reveal attack and only achieve IND-StAA security.

To achieve IND-AA security, we have to deal with the above impersonation attack due to the leakage of m_1 from state reveal. Intuitively, we have to find a way of authenticating P_j even if st_i is leaked to $\mathcal{A}$.

Now we briefly show how to construct our AKE from the plain AKE step by step. Steps (1)-(5) show how to support state reveal to avoid the impersonation attack, and (6) shows how to achieve wFS security.

(1) Partition m_1 by functionality. In algorithm Init, m_1 is divided into two parts $m_{11}|m_{12}$, where m_{11} is used to derive the session key and m_{12} is used as P_j's authenticating token.

(2) Limit information leakage of token m_{12} in state st_i. We do not put token m_{12} in st_i. Instead, only the hash value $\sigma := \mathsf{H}(m_{12})$ (rather than m_{12}) is stored in state st_i (where m_{11} is stored as well). Now even if $\mathcal{A}$ obtains σ from st_i, $\mathcal{A}$ can hardly recover the token m_{12}.

(3) Protect token m_{12} in the second round-message M_2. For explicit authentication, P_j has to transmit the token m_{12} via M_2. Thus we have to protect m_{12} in M_2 to avoid leakage. To this end, in $\mathsf{Der}_{\mathsf{resp}}$, m_2 is further divided into two parts $m_{21}|m_{22}$, where m_{21} is used to derive the session key and m_{22} is used to encrypt m_{22} via one-time pad. Now $M_2 = c_2$ (in the plain AKE) is changed to $M_2 = (c_2, C := m_{12} \oplus m_{22})$.

(4) Authenticate P_j with $\sigma = \mathsf{H}(m_{12})$. P_i can decrypt c_2 to obtain m_{22} and recover $m_{12} := C \oplus m_{22}$. By retrieving σ from st_i, P_i can authenticte P_j by checking whether m_{12} is the hash pre-image of σ.

(5) Avoid leakage m_{12} from man-in-the-middle (MITM) attack. Now that both $M_1 = c_1 = \mathsf{Enc}(pk_j, m_{11}|m_{12})$ and $M_2 = (c_2, m_{12} \oplus m_{22})$ contain the information of m_{12}. But P_j is not able to authenticate P_i by M_1. Then it is possible for $\mathcal{A}$ to implement a MITM attack: copy c_1 from M_1 as its own first round-message; P_j will output $M_2 = (\hat{c}_2 = \mathsf{Enc}(\hat{pk}, \hat{m}_{21}|\hat{m}_{22}), C = m_{12} \oplus \hat{m}_{22})$; $\mathcal{A}$ decrypts $\hat{m}_{21}|\hat{m}_{22} \leftarrow \mathsf{Dec}(\hat{sk}, \hat{c}_2)$ with its own secret key $\hat{sk}$. Then $\mathcal{A}$ can recover the token $m_{12} := C \oplus \hat{m}_{22}$ and then impersonate P_j with the token. This MITM attack can be easily avoided by attaching P_i's identity i to $m_{11}|m_{12}$[1]. So $c_1 \leftarrow \mathsf{Enc}(pk_j, m_{11}|m_{12}|i)$. The CCA security of PKE will guarantee that $\mathcal{A}$'s MITM attack either results decryption failure or a totally different decryption result.

(6) Encryption of c_2 with ephemeral key for the wFS security. P_i puts the ephemeral public key $\tilde{pk}$ in $M_1 = (\tilde{pk}, c_1)$ and the ephemeral secret key $\tilde{sk}$ in $st_i = (m_{11}, \tilde{sk}, \sigma)$. P_j uses $\tilde{pk}$ to encrypt c_2 to obtain $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{pk}, c_2)$. So $M_2 = (\tilde{c}, C)$. Now we arrive at our final AKE construction.

With the protection of ephemeral key, even sk_i and sk_j are corrupted, c_2 is still well-protected from $\mathcal{A}$ as long as $\mathcal{A}$ does not reveal state to obtain $\tilde{sk}$. Consequently, $\mathcal{A}$ knows nothing about m_{21} and the final session key $K_i = K_j = H(m_{11}|m_{21}|M_1|M_2)$ is still random to $\mathcal{A}$. In fact, as long as $\mathcal{A}$ does not obtain both the initiator's the long-term key and its round state

[1] In our final generic construction of AKE, we use tagged KEM to generate c_1 with identity as the tag. Here PKE is only specific construction of tagged KEM.

(to avoid trivial attack), the session key from a non-tampered session is pseudo-random to the adversary. So our AKE achieves the wFS security.

In the session key generation, we can always change the hash function with PRF function so that $K_i = K_j = \mathsf{PRF}(m_{11}|M_1|M_2) \oplus \mathsf{PRF}(m_{21}|M_1|M_2)$. In this way, the IND-AA security is proven in the standard model. Our AKE construction is shown in Fig. 3.

Moreover, we can also change PKE to tagged TKEM and KEM for the generation of c_1 and c_2. Since PKE can be considered a specific instantiation of KEM (or TKEM), this change only makes our AKE construction more general.

Related Works. The FSXY scheme in [8] is a two-message AKE constructed from KEM in the standard model. As noted in [11], its security is essentially the IND-StAA security. As far as we know, the $\mathsf{AKE}_{\mathsf{FO}}$ [11] is the only generic two-message AKE construction from PKE in the QROM model, achieving IND-StAA security. The performances of FSXY and $\mathsf{AKE}_{\mathsf{FO}}$ are shown in Table 1.

There are also other AKE schemes [13,9,10] supporting state reveal (i.e., resisting state reveal attack). In [13,9], a symmetric encryption (SE) is employed to encrypt the round state to support state reveal. With this method, the secret key of SE has to be included into the long-term secret key. Besides, the AKE scheme in [13] is based on the random oracle (RO) model and those in [9,10] rely on SIG to provide authentication.

The HMQV protocol [14] also supports state reveal, but it is Diffie-Hellman type AKE scheme in the RO model, rather than a generic construction. Its solution to state reveal is specific to the Diffie-Hellman algebraic structure.

2 Preliminary

Let $\emptyset$ denote the empty set. If x is defined by y or the value of y is assigned to x, we write $x := y$. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, \ldots, \mu\}$. Denote by $x \leftarrow_{\$} \mathcal{X}$ the procedure of sampling x from set $\mathcal{X}$ uniformly at random. Let $|\mathcal{X}|$ denote the number of elements in $\mathcal{X}$. All our algorithms are probabilistic unless states otherwise. PPT abbreviates probabilistic polynomial time. We use $y \leftarrow \mathcal{A}(x)$ to define the random variable y obtained by executing algorithm $\mathcal{A}$ on input x. We use $y \in \mathcal{A}(x)$ to indicate that y lies in the support of $\mathcal{A}(x)$. We also use $y \leftarrow \mathcal{A}(x; r)$ to make explicit the random coins r used in the probabilistic computation. Let λ denote the security parameter. We assume all algorithms take 1^λ as an implicit input.

2.1 Public Key Encryption

A public key encryption scheme consists of three algorithms $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where $(pk, sk) \leftarrow \mathsf{Gen}$ generates public/secret key pair, $c \leftarrow \mathsf{Enc}(pk, m)$ encrypts plaintext m to ciphertext c and $m/\perp \leftarrow \mathsf{Dec}(sk, c)$ decrypts ciphertext c to recover the plaintext m. The $(1-\delta)$ correctness of PKE requires decryption error is bounded by δ, where the probability is over $(pk, sk) \leftarrow \mathsf{Gen}$ and $c \leftarrow \mathsf{Enc}(pk, m)$.

Definition 1 (γ-Spreadness of PKE). *We say that* PKE *is γ-spread if for all key pairs $(pk, sk) \in \mathsf{Gen}(\mathsf{pp}_{\mathsf{PKE}})$ and all messages $m \in \mathcal{M}$, it holds that*

$$\max_{c \in \mathcal{C}} \Pr\left[r \leftarrow_{\$} \mathcal{R} : \mathsf{Enc}(pk, m; r) = c\right] \leq 2^{-\gamma}.$$

Definition 2 (γ-Key Diversity of PKE). *We say that* PKE *is γ-key diverse if*

$$\Pr\left[\begin{array}{c} r_1, r_2 \leftarrow_{\$} \mathcal{R} \\ (pk_1, sk_1) \leftarrow \mathsf{Gen}(\mathsf{pp}_{\mathsf{PKE}}; r_1) \\ (pk_2, sk_2) \leftarrow \mathsf{Gen}(\mathsf{pp}_{\mathsf{PKE}}; r_2) \end{array} : pk_1 = pk_2\right] \leq 2^{-\gamma}.$$

Definition 3 (IND-CPA Security for PKE). *For* PKE*, an adversary $\mathcal{A}$'s advantage function is defined by $\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{CPA\text{-}0}}_{\mathsf{PKE},\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{CPA\text{-}1}}_{\mathsf{PKE},\mathcal{A}} \Rightarrow 1\right]\right|$, where*

$$\Pr\left[\mathsf{Exp}^{\mathsf{CPA\text{-}b}}_{\mathsf{PKE},\mathcal{A}} \Rightarrow 1\right] := \Pr\left[\begin{array}{c} (pk, sk) \leftarrow \mathsf{Gen}(\mathsf{pp}_{\mathsf{PKE}}); (m_0, m_1, st) \leftarrow \mathcal{A}(pk) \\ c_b \leftarrow \mathsf{Enc}(sk, m_b); b' \leftarrow \mathcal{A}(st, pk, c_b) \end{array} : b' = 1\right].$$

The IND-CPA security of PKE *requires $\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) = \mathsf{negl}(\lambda)$ for all PPT $\mathcal{A}$.*

2.2 Tagged Key Encapsulation Mechanism

Definition 4 (TKEM). *A tagged key encapsulation mechanism (TKEM) scheme* $\mathsf{TKEM} = (\mathsf{TKEM.Setup}, \mathsf{TKEM.Gen}, \mathsf{TKEM.Encap}, \mathsf{TKEM.Decap})$ *consists of four algorithms.*

- TKEM.Setup. *The setup algorithm outputs public parameters $\mathsf{pp}_{\mathsf{TKEM}}$, which determine an encapsulation key space $\mathcal{K}$, public key space $\mathcal{PK}$, secret key space $\mathcal{SK}$, tag space $\mathcal{T}$ and a ciphertext space $\mathcal{CT}$.*
- $\mathsf{TKEM.Gen}(\mathsf{pp}_{\mathsf{TKEM}})$. *Taking $\mathsf{pp}_{\mathsf{TKEM}}$ as input, the key generation algorithm outputs a pair of public key and secret key $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$.*
- $\mathsf{TKEM.Encap}(pk, \tau)$. *Taking pk and a tag τ as input, the encapsulation algorithm outputs a pair of ciphertext $c \in \mathcal{CT}$ and encapsulated key $K \in \mathcal{K}$.*
- $\mathsf{TKEM.Decap}(sk, c, \tau)$. *Taking as input sk and c and a tag τ, the deterministic decapsulation algorithm outputs $K \in \mathcal{K} \cup \{\perp\}$.*

The $(1-\delta)$-correctness of TKEM *requires that for all tag $\tau \in \mathcal{T}$,*

$$\Pr\left[\begin{array}{c} (pk, sk) \leftarrow \mathsf{TKEM.Gen}(\mathsf{pp}_{\mathsf{KEM}}) \\ (c, K) \leftarrow \mathsf{TKEM.Encap}(pk, \tau) \end{array} : \mathsf{TKEM.Decap}(sk, c, \tau) \neq K\right] \leq \delta.$$

We recall the IND-CCA security of TKEM.

Definition 5 (IND-CCA Security for TKEM[1]). *To a tag key encapsulation mechanism* TKEM*, the advantage functions of an adversary $\mathcal{A}$ is defined by $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{CCA\text{-}0}}_{\mathsf{TKEM},\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{CCA\text{-}1}}_{\mathsf{TKEM},\mathcal{A}} \Rightarrow 1\right]\right|$, where the experiments $\mathsf{Exp}^{\mathsf{CCA\text{-}b}}_{\mathsf{TKEM},\mathcal{A}}$ for $b \in \{0,1\}$ are defined in Fig. 4. The IND-CCA security of tag* KEM *requires $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}) = \mathsf{negl}(\lambda)$ for all PPT algorithm $\mathcal{A}$.*

$\underline{\mathsf{Exp}^{\mathsf{CCA\text{-}b}}_{\mathsf{KEM},\mathcal{A}}:}$ $(\tau^*, st) \leftarrow \mathcal{A}$; $\mathsf{pp}_{\mathsf{TKEM}} \leftarrow \mathsf{TKEM.Setup}$ $(pk, sk) \leftarrow \mathsf{TKEM.Gen}(\mathsf{pp}_{\mathsf{TKEM}})$ $(c^*, K_0^*) \leftarrow \mathsf{TKEM.Encap}(pk, \tau^*)$ $K_1^* \leftarrow \mathcal{K}$; $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{DEC}}(\cdot,\cdot)}(st, pk, c^*, K_b^*)$ Return b'	$\underline{\mathcal{O}_{\mathrm{DEC}}(c,\tau):}$ If $(c,\tau) = (c^*, \tau^*)$: Return $\perp$ $K \leftarrow \mathsf{TKEM.Decap}(sk, c, \tau)$ Return K

Fig. 4: The IND-CCA security experiment $\mathsf{Exp}^{\mathsf{CCA\text{-}b}}_{\mathsf{KEM},\mathcal{A}}$ of Tagged-KEM.

When τ is null, TKEM becomes canonical KEM, and IND-CCA security can be similarly defined for KEM. Now we define the output pseudo-randomness of KEM w.r.t. its input randomness. Roughly speaking, output pseudo-randomness requires the encapsulation key K is indistinguishable from a random key even if $\mathcal{A}$ gets both pk and sk but has no information about ciphertext c .

Definition 6 (Output Pseudo-Randomness of KEM). *A key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{KEM.Setup}, \mathsf{KEM.Gen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ *has output pseudo-randomness if for any PPT adversary* $\mathcal{A}$*,*
$\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{A}) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ps\text{-}0}}_{\mathsf{KEM}} \Rightarrow 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ps\text{-}1}}_{\mathsf{KEM}} \Rightarrow 1\right]\right| = \mathsf{negl}(\lambda)$*, where*

$$\Pr\left[\mathsf{Exp}^{\mathsf{ps\text{-}b}}_{\mathsf{KEM}} \Rightarrow 1\right] := \Pr\left[\begin{array}{c} \mathsf{pp}_{\mathsf{KEM}} \leftarrow \mathsf{KEM.Setup} \\ (pk, sk) \leftarrow \mathsf{KEM.Gen}(\mathsf{pp}_{\mathsf{KEM}}) \\ (c, K_0) \leftarrow \mathsf{KEM.Encap}(pk); K_1 \leftarrow_{\$} \mathcal{K} \\ b' \leftarrow \mathcal{A}(pk, sk, K_b) \end{array} : b' = 1\right].$$

2.3 PRG and PRF

Definition 7 (PRG). *Pseudo-Random Generator (PRG) is a polynomially computable deterministic function* $\mathsf{PRG} : \mathcal{K} \to \mathcal{K}'$*, where* $\mathcal{K}$ *is seed space and* $\mathcal{K}'$ *is output space with* $|\mathcal{K}| < |\mathcal{K}'|$*. The pseudo-randomness of* PRG *requires* $\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{A}) = \mathsf{negl}(\lambda)$ *for all PPT* $\mathcal{A}$*, where*

$$\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{A}) := |\Pr[s \leftarrow_{\$} \mathcal{K}; y \leftarrow \mathsf{PRG}(x) : \mathcal{A}(y) \Rightarrow 1] - \Pr[y \leftarrow_{\$} \mathcal{K}' : \mathcal{A}(y) \Rightarrow 1]|.$$

Definition 8 (PRF). *Pseudo-Random Function (PRF) is a polynomially computable deterministic function* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$*, where* $\mathcal{K}$ *is key space,* $\mathcal{X}$ *is input space and* $\mathcal{Y}$ *is output space. the advantage function of an adversary* $\mathcal{A}$ *is defined by*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{A}) := \Big|&\Pr\left[k \leftarrow_{\$} \mathcal{K}, x^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}(\cdot)}; y \leftarrow \mathsf{PRF}(k, x^*)\ : \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}(\cdot)}(x^*, y) \Rightarrow 1\right] \\ &- \Pr\left[k \leftarrow_{\$} \mathcal{K}, x^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}(\cdot)}; y \leftarrow_{\$} \mathcal{Y} : \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}(\cdot)}(x^*, y) \Rightarrow 1\right]\Big|,\end{aligned}$$

where $\mathcal{O}_{\mathsf{PRF}}(x)$ *returns* $\mathsf{PRF}(k, x)$ *and* x^* *is never queried to* $\mathcal{O}_{\mathsf{PRF}}(\cdot)$*. The pseudo-randomness of* PRF *requires* $\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{A}) = \mathsf{negl}(\lambda)$ *for all PPT* $\mathcal{A}$*.*

2.4 Hash Function: TCR and One-Wayness

Definition 9 (One-Wayness of Hash). *A hash family* $\mathcal{H} = \{\mathsf{H} : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$ *has One-Wayness if the advantage functions of an adversary* $\mathcal{A}$ *defined by* $\mathsf{Adv}_{\mathsf{H}}^{\mathsf{owf}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}_{\mathsf{H}}^{\mathsf{owf}} \Rightarrow 1\right]$ *is negligible for all PPT* $\mathcal{A}$*, where the experiments* $\mathsf{Exp}_{\mathsf{H}}^{\mathsf{owf}}$ *are defined in Fig. 5 (left).*

Definition 10 (TCR of Hash). *A hash family* $\mathcal{H} = \{\mathsf{H} : \{0,1\}^n \to \{0,1\}^{\ell(n)}\}$ *is Target Collision Resistant (TCR), if the advantage function of adversary* $\mathcal{A}$ *defined by* $\mathsf{Adv}_{\mathsf{H}}^{\mathsf{tcr}}(\mathcal{A}) := \Pr\left[\mathsf{Exp}_{\mathsf{H}}^{\mathsf{tcr}} \Rightarrow 1\right]$ *is negligible for all PPT* $\mathcal{A}$*, where the experiments* $\mathsf{Exp}_{\mathsf{H}}^{\mathsf{tcr}}$ *are defined in Fig. 5 (right).*

When $n - \ell(n) \geq \lambda$, TCR property of $\mathcal{H}$ implies one-wayness.

$\underline{\mathsf{Exp}_{\mathsf{H}}^{\mathsf{owf}}:}$	$\underline{\mathsf{Exp}_{\mathsf{H}}^{\mathsf{tcr}}:}$
$\mathsf{H} \leftarrow_{\$} \mathcal{H}$; $m \leftarrow \{0,1\}^n$ $\sigma \leftarrow \mathsf{H}(m)$; $m' \leftarrow \mathcal{A}(\mathsf{H}, \sigma)$ If $\mathsf{H}(m') = \sigma$: Return 1 Else: Return 0	$\mathsf{H} \leftarrow_{\$} \mathcal{H}$; $m \leftarrow \{0,1\}^n$ $m' \leftarrow \mathcal{A}(\mathsf{H}, m)$ If $m \neq m' \wedge \mathsf{H}(m) = \mathsf{H}(m')$: Return 1 Else: Return 0

Fig. 5: $\mathsf{Exp}_{\mathsf{H}}^{\mathsf{owf}}$ (left) and $\mathsf{Exp}_{\mathsf{H}}^{\mathsf{tcr}}$ (right) for $\mathcal{H}$.

3 Two-Message AKE and Its IND-AA Security

A two-message AKE (see Fig. 1) is characterized by four algorithms. Each party, say P_i, will invoke the key generation algorithm $\mathsf{Gen}(i)$ to generate its own public/secret key pair (pk_i, sk_i). An initiator P_i then invokes the initialization algorithm $\mathsf{Init}(sk_i, pk_j)$ to generate the first round-message M_1 and its state st. P_i sends M_1 to its responder P_j and stores the state st locally. Upon receiving M_1, P_j invokes the responder-derivatation algorithm $\mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1)$ to generate the second round-message M_2 and its session key K_j. P_j sends M_2 to P_i. Upon receiving M_2, P_i invokes the initiator-derivatation algorithm $\mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st)$ to derive its session key K_i. The formal definition for two-message AKE is given below.

Definition 11 (Two-Message AKE). *A two-message AKE scheme* $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Der}_{\mathsf{init}}, \mathsf{Der}_{\mathsf{resp}})$ *consists of the following four algorithms.*

- $\mathsf{Gen}(i)$*. Taking a party identity* i *as input, the key generation algorithm outputs a key pair* (pk_i, sk_i)*.*
- $\mathsf{Init}(sk_i, pk_j)$*. Taking as input a secret key* sk_i *and a public key* pk_j*, the initialisation algorithm outputs a message* M_1 *and a state* st*.*
- $\mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1)$*. Taking as input a secret key* sk_j*, a public key* pk_i *and a message* M_1*, the responder derivation algorithm outputs a message* M_2 *and a session key* K_j*.*

– $\mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st)$. *Taking as input a secret key* sk_i*, a public key* pk_j*, a message* M_2 *and a state* st*, the initiator derivation algorithm outputs a session key* K_i*.*

$(1-\delta)$-*Correctness of AKE.* For any distinct and honest parties P_i and P_j with $(pk_i, sk_i) \leftarrow \mathsf{Gen}(i)$ and $(pk_j, sk_j) \leftarrow \mathsf{Gen}(j)$, after their protocol execution of $(M_1, st) \leftarrow \mathsf{Init}(sk_i, pk_j)$, $(M_2, K_j) \leftarrow \mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1)$ and $K_i \leftarrow \mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st)$, the probability that $K_i = K_j \neq \emptyset$ is at least $1-\delta$.

Remark 1. Note that in a two-message AKE, the initiator P_i has to invoke two algorithms. Therefore, P_i has to transmit a round state st_i from Init to $\mathsf{Der}_{\mathsf{init}}$. However, responder P_j does not have to store any (secret) state, since P_j only invokes one algorithm for session key.

We will use the IND-AA security model proposed in [11]. This model formalizes the adversary's passive attack, active attack, state reveals of session instances. Suppose there are at most μ users $P_1, P_2, \ldots, P_\mu$, and each user will involve at most ℓ sessions. The sessions run the protocol algorithms with access to the party's long-term key material, and also have their own local variables. The local variables of each session, indexed by the integer sID, are shown below.

– $\mathsf{holder}[\mathsf{sID}]$: the party running the session sID.
– $\mathsf{peer}[\mathsf{sID}]$: the intended communication peer of $\mathsf{holder}[\mathsf{sID}]$.
– $\mathsf{sent}[\mathsf{sID}]$: the message sent by the session sID.
– $\mathsf{recv}[\mathsf{sID}]$: the message received by the session sID.
– $\mathsf{role}[\mathsf{sID}] \in \{\mathsf{initiator}, \mathsf{responder}\}$: it indicates holder plays the role of $\mathsf{initiator}$ or $\mathsf{responder}$.
– $st[\mathsf{sID}]$: round state in sID. If $\mathsf{role}[\mathsf{sID}] = \mathsf{initiator}$, then st is output by Init, otherwise, $st = \perp$.
– $\mathsf{sKey}[\mathsf{sID}]$: the session key of sID.

Definition 12 (Matching Sessions). *We say two sessions* sID *and* sID' *are matching if the following requirements hold:*

1. $(\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}]) = (\mathsf{peer}[\mathsf{sID}'], \mathsf{holder}[\mathsf{sID}'])$
2. $(\mathsf{sent}[\mathsf{sID}], \mathsf{recv}[\mathsf{sID}]) = (\mathsf{recv}[\mathsf{sID}'], \mathsf{sent}[\mathsf{sID}'])$
3. $\mathsf{role}[\mathsf{sID}] \neq \mathsf{role}[\mathsf{sID}']$

Let $\mathfrak{M}(\mathsf{sID})$ denote the set of session identities which match sID.

Definition 13 (Partner Sessions). *We say two sessions* sID *and* sID' *are partner if the following requirements hold:*

1. $(\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}]) = (\mathsf{peer}[\mathsf{sID}'], \mathsf{holder}[\mathsf{sID}'])$
2. $\mathsf{role}[\mathsf{sID}] \neq \mathsf{role}[\mathsf{sID}']$

Let $\mathfrak{P}(\mathsf{sID})$ *denote the set of session identities which are partnered to* sID*.*

Next, we formalize the oracles that deal with $\mathcal{A}$'s queries as follows.

$\mathsf{EST}(i,j)$: The query means that $\mathcal{A}$ wants to establish a new session sID for holder i and its peer j. Upon such a query, oracle EST assigns a new session identity $\mathsf{sID} := cnt$ and sets $\mathsf{holder}[\mathsf{sID}] := i$ and $\mathsf{peer}[\mathsf{sID}] := j$ for $\mathcal{A}$.

$\mathsf{INIT}(\mathsf{sID})$: The query means that $\mathcal{A}$ asks the oracle to initiate session sID. Then the oracle generates the first round message $M \leftarrow \mathsf{Init}(sk_i, pk_j)$ and replies M to $\mathcal{A}$. Here sk_i is the secret key of $\mathsf{holder}[\mathsf{sID}]$ and pk_j is the public key of $\mathsf{peer}[\mathsf{sID}]$.

$\mathsf{DER}_{\mathsf{resp}}(\mathsf{sID}, \mathbf{M})$: This query means that $\mathcal{A}$ asks session sID to respond the first-round message M (so $\mathsf{role}[\mathsf{sID}] = \mathsf{responder}$). The oracle will invoke $M' \leftarrow \mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M)$ and return M' as the second round message to $\mathcal{A}$. Here sk_j is the secret key of $\mathsf{holder}[\mathsf{sID}]$ and pk_i is the public key of $\mathsf{peer}[\mathsf{sID}]$.

$\mathsf{DER}_{\mathsf{init}}(\mathsf{sID}, \mathbf{M'})$: This query means that $\mathcal{A}$ asks session sID to respond the second-round message M' (so $\mathsf{role}[\mathsf{sID}] = \mathsf{initiator}$). The oracle will invoke $K_i \leftarrow \mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M, st[\mathsf{sID}])$ to generate the session key $\mathsf{sKey}[\mathsf{sID}] := K_i$. Here sk_i is the secret key of $\mathsf{holder}[\mathsf{sID}]$ and pk_j is the public key of $\mathsf{peer}[\mathsf{sID}]$.

$\mathsf{REVEAL}(\mathsf{sID})$: It means that $\mathcal{A}$ reveals the session key of session sID. The oracle will return $\mathsf{sKey}[\mathsf{sID}]$ to $\mathcal{A}$.

$\mathsf{REV\text{-}STATE}(\mathsf{sID})$: It means that $\mathcal{A}$ reveals the state of session sID. The oracle will return $st[\mathsf{sID}]$ to $\mathcal{A}$

$\mathsf{CORRUPT}(i)$: It means that $\mathcal{A}$ reveals the long-term key of party P_i. The oracle will return sk_i to $\mathcal{A}$.

$\mathsf{TEST}(\mathsf{sID})$: It means that $\mathcal{A}$ chooses sID as the target session and the session key of sID for challenge (test). The oracle will set $K_0 := \mathsf{sKey}[\mathsf{sID}]$, sample $K_1 \leftarrow_{\$} \mathcal{K}$, and return K_b to $\mathcal{A}$.

$\mathsf{Trivial}(\mathsf{sID}^*)$: It identifies whether $\mathcal{A}$'s behavior leads to a trivial attack for the target (test) session sID^*. The oracle will first create a list of all matching sessions for sID^*. The list is denoted by $\mathfrak{M}(\mathsf{sID}^*)$. Then the oracle outputs 1 in case of the following trivial attacks.

- session sID^* is tested but $\mathsf{sKey}[\mathsf{sID}^*]$ is revealed to $\mathcal{A}$.
- session sID^* is tested and both long-term key sk_i of $\mathsf{holder}[\mathsf{sID}^*]$ and secret state $st[\mathsf{sID}^*]$ are revealed to $\mathcal{A}$.
- session sID^* is tested, there is only one matching session ptr (i.e., $\mathfrak{M}(\mathsf{sID}^*) = \{ptr\}$), and the session key $\mathsf{sKey}[ptr]$ of matching session ptr is revealed.
- session sID^* is tested, there is only one matching session ptr (i.e., $\mathfrak{M}(\mathsf{sID}^*) = \{ptr\}$), and both long-term key sk_j of $\mathsf{peer}[\mathsf{sID}] = \mathsf{holder}[ptr]$ and secret state $st[ptr]$ of session ptr are revealed to $\mathcal{A}$.
- session sID^* is tested, there is no matching session with sID^* (i.e., $\mathfrak{M}(\mathsf{sID}^*) = \emptyset$), and the long-term key sk_j of $j := \mathsf{peer}[\mathsf{sID}^*]$ is revealed to $\mathcal{A}$.

Recall that μ is the number of users and ℓ is the maximum number of sessions per user. The security experiment $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ with $b \in \{0,1\}$ is played between challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

1. For each party P_i, $\mathcal{C}$ runs $\mathsf{Gen}(i)$ to get the long-term key pair (pk_i, sk_i). Then $\mathcal{C}$ provides $\mathcal{A}$ with the list of public keys $(pk_1, \ldots, pk_\mu)$.

$\underline{\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}}$ $\quad$ // $b \in \{0,1\}$
$cnt := 0$ $\quad$ //session counter
$\mathsf{sID}^* := 0$ $\quad$ //test session's id
for $i \in [\mu]$:
  $(pk_i, sk_i) \leftarrow \mathsf{Gen}(i)$
  $crp[i] :=$ **false** $\quad$ corruption variables
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AKE}}}(pk_1, \ldots, pk_\mu)$
If $\mathsf{Trivial}(\mathsf{sID}^*)$:
  Return 0
Return b'

$\underline{\mathsf{EST}((i,j) \in [\mu]^2)}$:
$cnt := cnt + 1$
$\mathsf{sID} := cnt$
$\mathsf{holder}[\mathsf{sID}] := i$
$\mathsf{peer}[\mathsf{sID}] := j$
$stRev[\mathsf{sID}] :=$ **false** $\quad$ //state reveal variables
$rev[\mathsf{sID}] :=$ **false** //session key reveal variables
Return sID

$\underline{\mathsf{INIT}(\mathsf{sID})}$:
If $\mathsf{holder}[\mathsf{sID}] = \perp$:
  Return $\perp$ $\quad$ //session not established
If $\mathsf{sent}[\mathsf{sID}] \neq \perp$: Return $\perp$ $\quad$ //no re-use
$\mathsf{role}[\mathsf{sID}] := \mathsf{initiator}$
$(i,j) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
$(M, st) := \mathsf{Init}(sk_i, pk_j)$
$(\mathsf{sent}[\mathsf{sID}], st[\mathsf{sID}]) := (M, st)$
Return M

$\underline{\mathsf{DER}_{\mathsf{resp}}(\mathsf{sID}, M)}$:
If $\mathsf{holder}[\mathsf{sID}] = \perp$:
  Return $\perp$
If $\mathsf{sent}[\mathsf{sID}] \neq \perp$:
  Return $\perp$ $\quad$ //no re-use
If $\mathsf{role}[\mathsf{sID}] = \mathsf{initiator}$: Return $\perp$
$\mathsf{role}[\mathsf{sID}] := \mathsf{responder}$
$(j,i) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
$(M', K') \leftarrow \mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M)$
$\mathsf{sKey}[\mathsf{sID}] := K'$
$(\mathsf{recv}[\mathsf{sID}], \mathsf{sent}[\mathsf{sID}]) := (M, M')$
Return M'

$\underline{\mathsf{DER}_{\mathsf{init}}(\mathsf{sID}, M)}$:
If $\mathsf{holder}[\mathsf{sID}] = \perp \vee st[\mathsf{sID}] = \perp$:
  Return $\perp$
If $\mathsf{sKey}[\mathsf{sID}] \neq \perp$: Return $\perp$ $\quad$ //no re-use
$(i,j) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
$\mathsf{sKey}[\mathsf{sID}] := \mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M, st[\mathsf{sID}])$
$\mathsf{recv}[\mathsf{sID}] := M$
Return $\emptyset$

$\underline{\mathsf{REVEAL}(\mathsf{sID})}$:
If $\mathsf{sKey}[\mathsf{sID}] = \perp$: Return $\perp$
$rev[\mathsf{sID}] :=$ **true**
Return $\mathsf{sKey}[\mathsf{sID}]$

$\underline{\mathsf{REV\text{-}STATE}(\mathsf{sID})}$:
If $st[\mathsf{sID}] = \perp$: Return $\perp$
$stRev[\mathsf{sID}] :=$ **true**
Return $st[\mathsf{sID}]$

$\underline{\mathsf{CORRUPT}(i \in [\mu])}$:
$crp[i] :=$ **true**
Return sk_i

$\underline{\mathsf{TEST}(\mathsf{sID})}$: $\quad$ //only one query
$\mathsf{sID}^* := \mathsf{sID}$
If $\mathsf{sKey}[\mathsf{sID}^*] = \perp$:
  Return $\perp$
$K_0^* := \mathsf{sKey}[\mathsf{sID}^*]$
$K_1^* \leftarrow \mathcal{K}$
Return K_b^*

$\underline{\mathsf{Trivial}(\mathsf{sID}^*)}$:
$(i,j) := (\mathsf{holder}[\mathsf{sID}^*], \mathsf{peer}[\mathsf{sID}^*])$
If $rev[\mathsf{sID}^*] =$ **true**: Return **true**
If $crp[i] =$ **true** $\wedge\ stRev[\mathsf{sID}^*] =$ **true** :
  Return **true**
$\mathfrak{M}(\mathsf{sID}^*) := \emptyset$;
For $1 \leq ptr \leq cnt$:
  If $(\mathsf{sent}[ptr], \mathsf{recv}[ptr]) = (\mathsf{recv}[\mathsf{sID}^*], \mathsf{sent}[\mathsf{sID}^*])$
    $\wedge(\mathsf{holder}[ptr], \mathsf{peer}[ptr]) = (j,i) \wedge \mathsf{role}[\mathsf{sID}^*] \neq \mathsf{role}[ptr]$:
    $\mathfrak{M}(\mathsf{sID}^*) := \mathfrak{M}(\mathsf{sID}^*) \cup \{ptr\}$ $\quad$ //session matches
If $|\mathfrak{M}(\mathsf{sID}^*)| = 0$: $\quad$ // active attack
  If $crp[j] =$ **true** : Return **true**
  Else: Return **false**
If $|\mathfrak{M}(\mathsf{sID}^*)| > 1$: $\quad$ //multiple matching sessions
  Return **false** $\quad$ //This is not a trivial attack.
If $|\mathfrak{M}(\mathsf{sID}^*)| = 1$:
  Let $\mathfrak{M}(\mathsf{sID}^*) = \{ptr\}$
  If $rev[ptr] =$ **true** : Return **true**
  If $crp[j] =$ **true** $\wedge\ stRev[ptr] =$ **true** : Return **true**
  Return **false**

Fig. 6: The security experiments $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ where $b \in \{0,1\}$, where $\mathcal{O}_{\mathsf{AKE}} := \{\mathsf{EST}, \mathsf{INIT}, \mathsf{DER}_{\mathsf{resp}}, \mathsf{DER}_{\mathsf{init}}, \mathsf{REVEAL}, \mathsf{REV\text{-}STATE}, \mathsf{CORRUPT}, \mathsf{TEST}\}$.

2. $\mathcal{A}$ has access to oracles EST, INIT, $\mathsf{DER}_{\mathsf{resp}}$, $\mathsf{DER}_{\mathsf{init}}$, REVEAL, $\mathsf{REV\text{-}STATE}$, $\mathsf{CORRUPT}$, and TEST. Note that $\mathcal{A}$ can issue only one query to TEST. The oracles will reply the corresponding answers to $\mathcal{A}$.
3. At the end of the experiment, $\mathcal{A}$ terminates with an output b'.
4. If $\mathsf{Trivial}(\mathsf{sID}^*) = \mathbf{true}$, the experiment returns 0. Otherwise, return b'.

Details of experiment $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ are given in Fig. 6. IND-AA Security of AKE requires key indistinguishability in $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}0}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ and $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}1}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$.

Definition 14 (IND-AA Security of AKE). *In experiment* $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$ *with* $b \in \{0,1\}$, *the IND-AA advantage function of an adversary* $\mathcal{A}$ *against AKE is defined as*

$$\mathsf{Adv}^{\mathsf{IND\text{-}AA}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} := \left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}0}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}1}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \Rightarrow 1\right]\right|.$$

The IND-AA Security of AKE asks $\mathsf{Adv}^{\mathsf{IND\text{-}AA}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} \le \mathsf{negl}(\lambda)$ *for all PPT* $\mathcal{A}$.

4 Generic Construction of Two-Message AKE and Its Security Proof

We propose a generic construction of $\mathsf{AKE} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Der}_{\mathsf{init}}, \mathsf{Der}_{\mathsf{resp}})$ with session key space $\mathcal{K}$ from the following building blocks.

– A tagged key encapsulation mechanism scheme $\mathsf{TKEM} = (\mathsf{TKEM.Gen}, \mathsf{TKEM.Encap}, \mathsf{TKEM.Decap})$, where the encapsulation key space is $\mathcal{K}$.
– A key encapsulation mechanism scheme $\mathsf{KEM} = (\mathsf{KEM.Gen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ with encapsulation key space is $\mathcal{K}$ and ciphertext space $\mathcal{E}$.
– A public key encryption scheme $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with message space $\mathcal{E}$.
– A pseudo-random generator $\mathsf{PRG} : \mathcal{K} \to \mathcal{K} \times \mathcal{K}$.
– A pseudo-random function $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^* \to \mathcal{K}$.
– A target collision resistant hash function $\mathsf{H} : \mathcal{K} \to \Sigma$, which is randomly chosen from hash family $\mathcal{H}$. Suppose $\mathcal{K} = \Sigma \times \Sigma$.

Our generic construction is given in Fig. 7.

Correctness. Suppose the $\mathsf{KEM}, \mathsf{PKE}, \mathsf{TKEM}$ are all $(1-\delta)$-correct, then the AKE construction is $(1-3\delta)$-correct.

Next we consider the security of our generic AKE construction.

Theorem 1 (Key Indistinguishablity of AKE). *Suppose that* KEM, TKEM, PKE *are* $(1-\delta)$*-correct,* TKEM *is an IND-CCA tagged-KEM scheme,* KEM *is an IND-CCA secure KEM scheme with output pseudo-randomness,* PKE *is an IND-CPA secure PKE scheme satisfying* γ*-spreadness and* γ*-key diverse,* H *is a target collision resistant hash function (and also a one way function),* PRG *is a pseudo-random generator, and* PRF *is a pseudo-random function. Then for any*

$\underline{\mathsf{Init}(sk_i, pk_j):}$	$\underline{\mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1):}$	$\underline{\mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st):}$

$(c_1, seed_i) \leftarrow \mathsf{TKEM.Encap}(pk_j, i)$
$m_{11} \vert m_{12} \leftarrow \mathsf{PRG}(seed_i)$
$\sigma := \mathsf{H}(m_{12})$
$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{PKE.Gen}$
$M_1 := (\tilde{pk}, c_1)$
$st := (m_{11}, \tilde{sk}, \sigma, M_1)$
Return (M_1, st)
	Parse $M_1 = (\tilde{pk}, c_1)$
	If $\mathsf{TKEM.Decap}(sk_j, c_1, i) = \perp$:
	  Return $\perp$
	$seed'_i \leftarrow \mathsf{TKEM.Decap}(sk_j, c_1, i)$
	$m'_{11} \vert m'_{12} \leftarrow \mathsf{PRG}(seed'_i)$
	$(c_2, seed_j) \leftarrow \mathsf{KEM.Encap}(pk_i)$
	$m_{21} \vert m_{22} \leftarrow \mathsf{PRG}(seed_j)$
	$\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, c_2)$
	$C := m'_{12} \oplus m_{22}$
	$M_2 := (\tilde{c}, C)$
	$K := \mathsf{PRF}(m'_{11}, M_1 \vert M_2) \oplus \mathsf{PRF}(m_{21}, M_1 \vert M_2)$
	Return (M_2, K)
		Parse $M_2 = (\tilde{c}, C)$
		Parse $st = (m_{11}, \tilde{sk}, \sigma, M_1 = (\tilde{pk}, c_1))$
		If $\mathsf{PKE.Dec}(\tilde{sk}, \tilde{c}) = \perp$:
		  Return $\perp$
		$c'_2 \leftarrow \mathsf{PKE.Dec}(\tilde{sk}, \tilde{c})$
		If $\mathsf{KEM.Decap}(sk_i, c'_2) = \perp$:
		  Return $\perp$
		$seed'_j \leftarrow \mathsf{KEM.Decap}(sk_i, c'_2)$
		$m'_{21} \vert m'_{22} \leftarrow \mathsf{PRG}(seed_j)$
		If $\mathsf{H}(C \oplus m'_{22}) \neq \sigma$:
		  Return $\perp$
		$K := \mathsf{PRF}(m_{11}, M_1 \vert M_2) \oplus \mathsf{PRF}(m'_{21}, M_1 \vert M_2)$
		Return K

Fig. 7: Generic construction of two-message AKE.

PPT adversary $\mathcal{A}$ against AKE that establishes sessions among at most μ users and at most ℓ sessions per user, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}AA}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} =& 2\mu^2\ell \cdot \Big((\ell+2)\cdot \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + (\ell+1)\cdot \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{B}_{\mathsf{TKEM}}) + \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}})\\ &+\ell\cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{owf}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}) + (3\ell+2)\cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}})\\ &+\ell\cdot \mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}}) + (3\ell+3)\cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) + (4\ell^2+\ell+5)\cdot\delta + 2^{-\gamma+1}\Big).\end{aligned}$$

Proof. We now consider a sequence of games and analyze $\mathcal{A}$'s advantages in these games. Let $\mathsf{G}_{i,b}$ denote the i-th game w.r.t. $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$. Let $adv_i := |\Pr[\mathsf{G}_{i,0} \Rightarrow 1] - \Pr[\mathsf{G}_{i,1} \Rightarrow 1]|$. Then $|adv_i - adv_{i+1}| \le 2|\Pr[\mathsf{G}_{i,b} \Rightarrow 1] - \Pr[\mathsf{G}_{i+1,b} \Rightarrow 1]|$.

Game $\mathsf{G}_{0,b}$. $\mathsf{G}_{0,b}$ is the original experiment $\mathsf{Exp}^{\mathsf{IND\text{-}AA\text{-}}b}_{\mathsf{AKE},\mu,\ell,\mathcal{A}}$. We have

$$\mathsf{Adv}^{\mathsf{IND\text{-}AA}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} := |\Pr[\mathsf{G}_{0,0} \Rightarrow 1] - \Pr[\mathsf{G}_{0,1} \Rightarrow 1]| = adv_0. \tag{1}$$

Game $\mathsf{G}_{1,b}$. In $\mathsf{G}_{1,b}$, challenger $\mathcal{C}$ first chooses $(i^*, j^*, \mathsf{sID}^*) \leftarrow_{\$} [\mu]^2 \times [\ell]$. At the end of $\mathsf{G}_{1,b}$, if $\mathcal{A}$ does not query $\mathsf{Test}(\mathsf{sID}^*)$ or $\mathsf{holder}[\mathsf{sID}^*] \neq i^*$ or $\mathsf{peer}[\mathsf{sID}^*] \neq j^*$, $\mathcal{C}$ will return 0. Obviously, we have

$$\begin{aligned}\Pr[\mathsf{G}_{0,b} \Rightarrow 1] &= \mu^2\ell \cdot \Pr[\mathsf{G}_{1,b} \Rightarrow 1],\\ adv_0 &= \mu^2\ell \cdot adv_1.\end{aligned} \tag{2}$$

Let $\mathfrak{M}(\mathsf{sID}^*)$ be the set of all session identities matching with sID^*. Obviously,

$$\begin{aligned}\Pr[\mathsf{G}_{1,b} \Rightarrow 1] =& \Pr[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator}] && (\textbf{Case 1})\\ &+\Pr[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{responder}] && (\textbf{Case 2})\\ &+\Pr[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) \neq \emptyset]. && (\textbf{Case 3})\end{aligned}$$

Define $\mathsf{G}^y_{x,b}$ as $\mathsf{G}_{x,b}$ in case y and $adv^y_x := \left|\Pr\left[\mathsf{G}^y_{x,0} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^y_{x,1} \Rightarrow 1\right]\right|$. Then

$$\begin{aligned}\left|adv^y_x - adv^y_{x+1}\right| \le 2\left|\Pr\left[\mathsf{G}^y_{x,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^y_{x+1,b} \Rightarrow 1\right]\right|,\\ adv_1 \le adv^1_1 + adv^2_1 + adv^3_1.\end{aligned} \tag{3}$$

Now we consider $\Pr[\mathsf{G}_{1,b} \Rightarrow 1]$ the above three cases.

Case 1: $\mathsf{G}_{1,b} \Rightarrow 1 \ \wedge\ \mathfrak{M}(\mathsf{sID}^*) = \emptyset \ \wedge\ \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator}$.

In this case, $\mathfrak{M}(\mathsf{sID}^*) = \emptyset$ means that no matching session exists for sID^*, and $\mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator}$ implies that test session is held by an initiator. Then, in session sID^*, initiator i^* must suffer from an active attack from adversary $\mathcal{A}$. If j^* is further corrupted, then this is a trivial attack leading to $\mathsf{G}_{1,b} \Rightarrow 0$. Therefore, for $b \in \{0,1\}$,

$$\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge crp[j^*]\right] = 0. \tag{4}$$

On the other hand, if $\mathcal{A}$ both corrupts i^* and obtains its states by $\mathsf{StateReveal}(\mathsf{sID}^*)$, then this is also a trivial attack leading to $\mathsf{G}_{1,b} \Rightarrow 0$. Therefore,

$$\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge crp[i^*] \wedge stRev[\mathsf{sID}^*]\right] = 0. \tag{5}$$

Consequently,

$$\begin{aligned}
&\Pr\left[\mathsf{G}^{1}_{1,b} \Rightarrow 1\right] := \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator}\right] \\
=&\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge \neg crp[j^*]\right] \\
\leq&\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge \neg crp[i^*] \wedge \neg crp[j^*]\right] + \qquad \textbf{(Case 1.1)} \\
&\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge \neg stRev[\mathsf{sID}^*] \wedge \neg crp[j^*]\right] \ \textbf{(Case 1.2)} \\
=&\Pr\left[\mathsf{G}^{1.1}_{1,b} \Rightarrow 1\right] + \Pr\left[\mathsf{G}^{1.2}_{1,b} \Rightarrow 1\right]
\end{aligned}$$

and

$$adv_1^1 \leq adv_1^{1.1} + adv_1^{1.2}. \tag{6}$$

That means Case 1 can be further divided into the following two subcases:

Case 1.1: $\neg crp[i^*] \wedge \neg crp[j^*]$ in Case 1. In this case, neither i^* nor j^* are corrupted. Now we consider $\mathsf{G}_{2,b}$ - $\mathsf{G}_{12,b}$ in Case 1.1, and denote it by $\mathsf{G}^{1.1}_{2,b}$ - $\mathsf{G}^{1.1}_{12,b}$. The full codes of $\mathsf{G}^{1.1}_{2,b}$ - $\mathsf{G}^{1.1}_{12,b}$ are presented in Fig. 8. Brief description of $\mathsf{G}_{1,b}$ and $\mathsf{G}^{1.1}_{2,b}$ - $\mathsf{G}^{1.1}_{12,b}$ games for Case 1.1 are given in Table 2. Define $adv_i^{1.1} := \left|\Pr\left[\mathsf{G}^{1.1}_{i,0} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{i,1} \Rightarrow 1\right]\right|$.

Game $\mathsf{G}^{1.1}_{2,b}$. $\mathsf{G}^{1.1}_{2,b}$ is the same as $\mathsf{G}_{1,b}$, except that $\mathcal{C}$ will return 0 directly if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$ or $\mathsf{role}[\mathsf{sID}^*] \neq \mathsf{initiator}$ or $crp[i^*] = \textbf{true}$ or $crp[j^*] = \textbf{true}$. Hence, we have

$$\Pr\left[\mathsf{G}^{1.1}_{2,b} \Rightarrow 1\right] = \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge \neg crp[i^*] \wedge \neg crp[j^*]\right].$$

Game $\mathsf{G}^{1.1}_{3,b}$. It is the same as $\mathsf{G}^{1.1}_{2,b}$ except for the behavior of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\widetilde{pk}, c_1))$ and $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\widetilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, it will additionally record the intermediate values $(c_2, seed_{j^*}, m_{21}, m_{22})$ with $\mathsf{record}[c_2] := (seed_{j^*}, m_{21}, m_{22})$, where $(c_2, seed_{j^*}) \leftarrow \mathsf{KEM.Encap}$ and $(m_{21}, m_{22}) \leftarrow \mathsf{PRG}(seed_{j^*})$.
- $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = i^*$, and the output c_2' of $\mathsf{PKE.Dec}$ has ever been recorded with $\mathsf{record}[c_2'] := (seed_{j^*}, m_{21}, m_{22})$ by $\mathsf{Der}_{\mathsf{resp}}$, it will directly use the recorded values of $(seed_{j^*}, m_{21}, m_{22})$ for the generation of session key, rather than computing $seed_{j^*}$ with $\mathsf{KEM.Decap}$ and (m_{21}, m_{22}) with PRG as did in $\mathsf{G}^{1.1}_{2,b}$.

Game	$\mathsf{Init}(\mathsf{sID}^*)$	$\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}^*))$	$\mathsf{Der}_{\mathsf{init}}(\mathsf{sID})$	Remark
$\mathsf{G}_{1,b}$	Abort if $\neg\mathsf{TEST}(\mathsf{sID}^*) \vee (\mathsf{holder}[\mathsf{sID}^*], \mathsf{peer}[\mathsf{sID}^*]) \neq (i^*, j^*)$			with security loss $\mu^2\ell$
$\mathsf{G}_{2,b}^{1.1}$	Abort if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset \vee \mathsf{role}[\mathsf{sID}^*] \neq \mathsf{initiator} \vee crp[i^*] \vee crp[j^*]$			$\mathsf{G}_{1,b}$ in Case 1.1
$\mathsf{G}_{3,b}^{1.1}$		$\mathsf{record}[c_2] := (seed_{j^*}, m_{21}, m_{22})$	if $\exists\mathsf{record}[c_2]$: use $\mathsf{record}[c_2]$ for $\mathsf{holder}[\mathsf{sID}] = i^*$	Correctness of KEM
$\mathsf{G}_{4,b}^{1.1}$		$seed_{j^*} \leftarrow_\$ \mathcal{K}$		CCA-security of KEM
$\mathsf{G}_{5,b}^{1.1}$		$m_{21}, m_{22} \leftarrow_\$ \mathcal{K}$		pseudo-randomness of PRG
$\mathsf{G}_{6,b}^{1.1}$		$C \leftarrow_\$ \mathcal{K}$, $m_{22} := C \oplus m'_{12}$		identical (concept change)
$\mathsf{G}_{7,b}^{1.1}$	$\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$	if $\exists\mathsf{record}[i^*, c_1]$: use $\mathsf{record}[i^*, c_1]$		Correctness of tagged TKEM
$\mathsf{G}_{8,b}^{1.1}$	$seed_{i^*} \leftarrow_\$ \mathcal{K}$			CCA-security of tagged TKEM
$\mathsf{G}_{9,b}^{1.1}$	$m_{11}, m_{12} \leftarrow_\$ \mathcal{K}$			pseudo-randomness of PRG
$\mathsf{G}_{10,b}^{1.1}$			Rejection Rule 1 for sID^*: reject if $m'_{12} \neq m_{12}$	target collision resistance of TCR
$\mathsf{G}_{11,b}^{1.1}$		$\mathsf{record}[c_2, C] := \mathsf{sID}$	Rejection Rule 2 for sID^*: reject if $\nexists\mathsf{record}[c_2, C]$	one-wayness of TCR
$\mathsf{G}_{12,b}^{1.1}$			$\mathsf{sKey}[\mathsf{sID}^*] \leftarrow_\$ \mathcal{K}$ for sID^*	pseudo-randomness of PRF

Table 2: Brief description of $\mathsf{G}_{1,b}$ and $\mathsf{G}_{2,b}^{1.1}$ - $\mathsf{G}_{12,b}^{1.1}$ for Case 1.1

Due to the (1-δ)-correctness of KEM and the fact that user j^* has at most ℓ sessions, we have

$$\left|\Pr\left[\mathsf{G}_{2,b}^{1.1} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{3,b}^{1.1} \Rightarrow 1\right]\right| \leq \ell\delta,$$
$$\left|adv_2^{1.1} - adv_3^{1.1}\right| \leq 2\ell\delta. \tag{7}$$

Game $\mathsf{G}_{4,b}^{1.1}$. It is the same as $\mathsf{G}_{3,b}^{1.1}$ except for $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, the value of $seed_{j^*}$ is randomly chosen in $\mathsf{G}_{4,b}^{1.1}$, instead of being generated by $\mathsf{KEM.Encap}$ in $\mathsf{G}_{3,b}^{1.1}$. The values of c_2, m_{21}, m_{22} are still the outputs of $\mathsf{KEM.Encap}$ and PRG, and $(c_2, seed_{j^*}, m_{21}, m_{22})$ is recorded in the same way as $\mathsf{G}_{3,b}^{1.1}$.

Due to the CCA security of KEM and the fact user j^* has at most ℓ sessions, we have

$$\left|\Pr\left[\mathsf{G}_{3,b}^{1.1} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{4,b}^{1.1} \Rightarrow 1\right]\right| \leq \ell \cdot \mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{KEM}}),$$
$$\left|adv_3^{1.1} - adv_4^{1.1}\right| \leq 2\ell \cdot \mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{KEM}}). \tag{8}$$

Game $\mathsf{G}_{5,b}^{1.1}$. It is the same as $\mathsf{G}_{4,b}^{1.1}$ except for $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, the value of m_{21}, m_{22} are randomly chosen in $\mathsf{G}_{5,b}^{1.1}$, instead of being generated by PRG as in $\mathsf{G}_{4,b}^{1.1}$. The values of $c_2, seed_{j^*}$ are generated the same way as in $\mathsf{G}_{4,b}^{1.1}$. And $(c_2, seed_{j^*}, m_{21}, m_{22})$ is recorded in the same way as $\mathsf{G}_{4,b}^{1.1}$.

Given random $seed_{j^*}$, the output of $\mathsf{PRG}(seed_{j^*})$ is pseudo-random. Together with the fact that user j^* has at most ℓ sessions, we have

$$\left|\Pr\left[\mathsf{G}_{4,b}^{1.1} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{5,b}^{1.1} \Rightarrow 1\right]\right| \leq \ell \cdot \mathsf{Adv}_{\mathsf{PRG}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRG}}),$$
$$\left|adv_4^{1.1} - adv_5^{1.1}\right| \leq 2\ell \cdot \mathsf{Adv}_{\mathsf{PRG}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRG}}). \tag{9}$$

Game $\mathsf{G}_{6,b}^{1.1}$. It is the same as $\mathsf{G}_{5,b}^{1.1}$ except the computations of C and m_{22} in Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, then $C \leftarrow_\$ \mathcal{K}$ and set $m_{22} := C \oplus m'_{12}$.

Recall that in $\mathsf{G}_{5,b}^{1.1}$, $m_{22} \leftarrow_\$ \mathcal{K}$ and $C := m_{22} \oplus m'_{12}$. It is easy to see that (m_{22}, C) has identical distribution in the two games. Hence,

$$\Pr\left[\mathsf{G}_{5,b}^{1.1} \Rightarrow 1\right] = \Pr\left[\mathsf{G}_{6,b}^{1.1} \Rightarrow 1\right],$$
$$adv_5^{1.1} = adv_6^{1.1}. \tag{10}$$

Game $\mathsf{G}_{7,b}^{1.1}$. It is the same as $\mathsf{G}_{6,b}^{1.1}$ except the behavior of Oracles $\mathsf{Init}(\mathsf{sID}^*)$ and $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Init}(\mathsf{sID}^*)$. It will additionally record the intermediate values $(i^*, c_1, seed_{i^*}, m_{11}, m_{12})$ for sID^* with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, where $(c_1, seed_{i^*})$ is the output of $\mathsf{TKEM.Encap}$ and (m_{11}, m_{12}) is the output of $\mathsf{PRG}(seed_{i^*})$.
– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, and (i^*, c_1) has already been recorded by $\mathsf{Init}(\mathsf{sID}^*)$ with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, it will directly use the recorded value of $(seed_{i^*}, m_{11}, m_{12})$ for the generation of session key and the second round-message M_2, rather than computing them with $(c_1, seed_{i^*}) \leftarrow \mathsf{TKEM.Decap}$ and $(m_{11}, m_{12}) \leftarrow \mathsf{PRG}$ as did in $\mathsf{G}_{6,b}^{1.1}$.

Due to the (1-δ)-correctness of TKEM, we have

$$\left|\Pr\left[\mathsf{G}_{6,b}^{1.1} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{7,b}^{1.1} \Rightarrow 1\right]\right| \le \delta,$$
$$\left|adv_6^{1.1} - adv_7^{1.1}\right| \le 2\delta. \tag{11}$$

Game $\mathsf{G}_{8,b}^{1.1}$. It is the same as $\mathsf{G}_{7,b}^{1.1}$ except the behavior of Oracles $\mathsf{Init}(\mathsf{sID}^*)$.

– $\mathsf{Init}(\mathsf{sID}^*)$. Now $seed_{i^*}$ is randomly sampled by $seed_{i^*} \leftarrow_\$ \mathcal{K}$ instead of being generated by $(c_1, seed_{i^*}) \leftarrow \mathsf{TKEM.Encap}(pk_{j^*}, i^*)$ in $\mathsf{G}_{7,b}^{1.1}$. The oracle records the intermediate values $(c_1, seed_{i^*}, m_{11}, m_{12})$ for sID^* with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, where c_1, m_{11}, m_{12} are computed in the same way as in $\mathsf{G}_{7,b}^{1.1}$.

By the CCA security of TKEM, we know that the encapsulated key $seed_{i^*}$ is pseudo-random. Hence we have

$$\left|\Pr\left[\mathsf{G}_{7,b}^{1.1} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{8,b}^{1.1} \Rightarrow 1\right]\right| \le \mathsf{Adv}_{\mathsf{TKEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{TKEM}}),$$
$$\left|adv_7^{1.1} - adv_8^{1.1}\right| \le 2\mathsf{Adv}_{\mathsf{TKEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{TKEM}}). \tag{12}$$

Game $\mathsf{G}_{9,b}^{1.1}$. It is the same as $\mathsf{G}_{8,b}^{1.1}$ except the behavior of Oracles $\mathsf{Init}(\mathsf{sID}^*)$.

– $\mathsf{Init}(\mathsf{sID}^*)$. Now m_{11}, m_{12} are randomly sampled by $m_{11}, m_{12} \leftarrow_\$ \mathcal{K}$, instead of $m_{11}|m_{12} \leftarrow_\$ \mathsf{PRG}(seed_{i^*})$ as did in $\mathsf{G}_{8,b}^{1.1}$. The oracle records the intermediate value $(c_1, seed_{i^*}, m_{11}, m_{12})$ for sID^* with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, where $c_1, seed_{i^*}$ are computed in the same way as in $\mathsf{G}_{8,b}^{1.1}$.

Given random $seed_{i^*}$, the output of $\mathsf{PRG}(seed_{i^*})$ is pseudo-random. So

$$\left|\Pr\left[\mathsf{G}^{1.1}_{8,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{9,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}),$$
$$\left|adv^{1.1}_{8} - adv^{1.1}_{9}\right| \leq 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}). \tag{13}$$

Game $\mathsf{G}^{1.1}_{10,b}$. It is the same as $\mathsf{G}^{1.1}_{9,b}$ except that the rejection rule is changed in Oracle $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$.

- $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$. During the process, we apply the following rejection rule.
  Rejection Rule 1: If $m'_{12} \neq m_{12}$, reject the query.
  Here m'_{12} is the intermediate value and m_{12} is from $\mathsf{record}[i^*, c_1] = (seed_{i^*}, m_{11}, m_{12})$.

Recall that in $\mathsf{G}^{1.1}_{9,b}$, $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$ will reject the query if $\mathsf{H}(m'_{12}) \neq \sigma$, where $\sigma = \mathsf{H}(m_{12})$ is recorded in state $st[\mathsf{sID}^*]$. Therefore, $\mathsf{G}^{1.1}_{9,b}$ is identical to $\mathsf{G}^{1.1}_{10,b}$, unless a hash collision $\mathsf{H}(m'_{12}) = \mathsf{H}(m_{12})$ but $m'_{12} \neq m_{12}$ happens. Recall that m_{12} is randomly distributed. By the TCR property of H, we have

$$\left|\Pr\left[\mathsf{G}^{1.1}_{9,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{10,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}),$$
$$\left|adv^{1.1}_{9} - adv^{1.1}_{10}\right| \leq 2\mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}). \tag{14}$$

Game $\mathsf{G}^{1.1}_{11,b}$. It is the same as $\mathsf{G}^{1.1}_{10,b}$ except for the behaviors of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$ and $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, it will record (sID, c_2, C) with $\mathsf{record}[c_2, C] := \mathsf{sID}$, where c_2 is the intermediate encapsulation ciphertext output by $(c_2, seed_{j^*}) \leftarrow \mathsf{KEM.Encap}(pk_{i^*})$ and C is the element in the output message $M_2 = (\tilde{c}, C)$.
- $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$. An additional rejection rule is added. Suppose c'_2 is output by $c'_2 \leftarrow \mathsf{PKE.Dec}(\tilde{c})$.
  Rejection Rule 2: If c'_2 is never recorded with $\mathsf{record}[c_2 = c'_2, C] = \mathsf{sID}$ by oracle $\mathsf{Der}_{\mathsf{resp}}(\cdot, \cdot)$, then reject the query immediately.

Define Z as the event that adversary $\mathcal{A}$ ever issued a query $(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$ to $\mathsf{Der}_{\mathsf{init}}$ such that $(\mathsf{sID}, c'_2 = \mathsf{PKE.Dec}(\tilde{c}), C)$ has never been recorded by $\mathsf{Der}_{\mathsf{resp}}(\cdot, \cdot)$ but $m'_{12} = m_{12}$. Here, m'_{12} is the intermediate value computed by $\mathsf{Der}_{\mathsf{init}}$ and m_{12} is from the tuple $(i^*, c_1, seed_{i^*}, m_{11}, m_{12})$ recorded by $\mathsf{Init}(\mathsf{sID}^*)$. If Z happens, the query will not be rejected in $\mathsf{G}^{1.1}_{10,b}$, but will be rejected in $\mathsf{G}^{1.1}_{11,b}$. Thus $\mathsf{G}^{1.1}_{11,b}$ is the same as $\mathsf{G}^{1.1}_{10,b}$ unless Z happens, i.e., $\left|\Pr\left[\mathsf{G}^{1.1}_{10,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{11,b} \Rightarrow 1\right]\right| \leq \Pr[Z]$.
Next, we show that $\Pr[Z] = \mathsf{negl}(\lambda)$ due to the one-wayness of H. To this end, we construct a PPT algorithm $\mathcal{B}_{\mathsf{H}}$ against the one-wayness of H. Let $\mathcal{C}_{\mathsf{H}}$ be the challenger of $\mathcal{B}_{\mathsf{H}}$. $\mathcal{C}_{\mathsf{H}}$ generates $m^* \leftarrow_{\$} \mathcal{K}$, computes $\sigma \leftarrow \mathsf{H}(m^*)$ and gives σ^* to $\mathcal{B}_{\mathsf{H}}$. Then $\mathcal{B}_{\mathsf{H}}$ will simulate $\mathsf{G}^{1.1}_{11,b}$ for $\mathcal{A}$ as follows.

- **Simulation of** $\mathsf{Init}(\mathsf{sID}^*)$**.** $\mathcal{B}_{\mathsf{H}}$ invokes $(c_1, K) \leftarrow \mathsf{TKEM.Encap}(pk_{j^*}, i^*)$ and $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{PKE.Gen}(\mathsf{pp}_{\mathsf{PKE}})$. Then it randomly samples $m_{11}, seed_{i^*} \leftarrow_{\$} \mathcal{K}$ and implicitly sets $m_{12} := m^*$. Next it records $(i^*, c_1, seed_{i^*}, m_{11}, ?)$ with $\mathsf{record}[i^*, c_1] = (seed_{i^*}, m_{11}, ?)$ and sets $st[\mathsf{sID}^*] = (m_{11}, \tilde{sk}, \sigma^*, M_1 = (\tilde{pk}, c_1))$. Return M_1 to $\mathcal{A}$.
- **Simulation of** $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}^*), M_1 = (\tilde{pk}, c_1))$**.** $\mathcal{B}_{\mathsf{H}}$ samples $m_{21}, C \leftarrow_{\$} \mathcal{K}$. It checks whether (i^*, c_1) appears in $\mathsf{record}[i^*, c_1] = (seed_{i^*}, m_{11}, ?)$ which is recorded by $\mathsf{Init}(\mathsf{sID}^*)$. If yes, it retrieves $(m_{11}, ?)$ and sets $m'_{11} := m_{11}$ and $m_{22} :=?$. Otherwise, it invokes $seed_{i^*} \leftarrow \mathsf{TKEM.Decap}(sk_{j^*}, c_1, i^*)$, $m'_{11}|m'_{12} \leftarrow \mathsf{PRG}(seed_{i^*})$ and computes $m_{22} := C \oplus m'_{12}$. It records $(c_2, seed_{j^*}, m_{21}, m_{22})$ and (c_2, C, j^*) with $\mathsf{record}[c_2] = (seed_{j^*}, m_{21}, m_{22})$ and $\mathsf{record}[c_2, C] = \mathsf{sID}$ respectively. For the session key, it computes $\mathsf{sKey}[\mathsf{sID}] := \mathsf{PRF}(m'_{11}, M_1|M_2) \oplus \mathsf{PRF}(m'_{21}, M_1|M_2)$. Finally, it returns $M_2 := (\tilde{c}, C)$ to $\mathcal{A}$.
- **Simulation of** $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$**.** $\mathcal{B}_{\mathsf{H}}$ retrieves $st[\mathsf{sID}^*] = (m_{11}, \tilde{sk}, \sigma^*, M_1 = (\tilde{pk}, c_1))$ and invokes $c'_2 \leftarrow \mathsf{PKE.Dec}(\tilde{sk}, \tilde{c})$. If c'_2 appears in $\mathsf{record}[c_2] = (seed_{j^*}, m_{21}, m_{22})$ which is recorded by $\mathsf{Der}_{\mathsf{resp}}(\cdot, \cdot)$, then it retrieves m_{21}, m_{22} from $\mathsf{record}[c_2]$. If $m_{22} =?$ then $\mathcal{B}_{\mathsf{H}}$ aborts the game (since event Z never happens which will be explained later). If $m_{22} \neq ?$, then it sets $m'_{22} := m_{22}$. If c'_2 never appears in any record $(c_2 = c'_2, seed_{j^*}, m_{21}, m_{22})$, it computes $seed_{j'} \leftarrow \mathsf{KEM.Decap}(sk_{i^*}, c'_2)$ and $(m'_{21}|m'_{22}) \leftarrow \mathsf{PRG}(seed_{j'})$. Next, it computes $m'_{12} := C \oplus m'_{22}$. If (sID, c'_2, C) has never been recorded by $\mathsf{Der}_{\mathsf{resp}}(\cdot, \cdot)$, then $\mathcal{B}_{\mathsf{H}}$ submits m'_{12} to its own challenger as the answer. Otherwise, $\mathcal{B}_{\mathsf{H}}$ aborts the game. If event Z happens, it must hold $m'_{12} = m^*$, thus $\mathcal{B}_{\mathsf{H}}$ wins.

For other oracle simulations, $\mathcal{B}_{\mathsf{H}}$ does just like $\mathsf{G}^{1.1}_{11,b}$.
Now we explain why Z never happens when $m_{22} =?$ during simulation of $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C'))$. Note that $m_{22} =?$ implies $c_1 = c'_1$ and $c_2 = c'_2$ where sID^* generates c_1 and computes $c'_2 \leftarrow \mathsf{Dec}(\tilde{sk}, \tilde{c})$ while its partner session sID generates c_2, C and receives c'_1. Since sID^* and sID share the same c_1 and c_2, they must share the same m_{12} and m_{22}. Now, if event Z happens, then $(c_2, C) \neq (c'_2, C')$ and $m'_{12} = m_{12}$. Given the fact $c_2 = c'_2$, it must hold that $C \neq C'$, where C is generated by sID and C' is received by sID^*. Thus, $m'_{12} = C' \oplus m_{22} = C' \oplus (C \oplus m_{12}) = m_{12} \oplus (C \oplus C') \neq m_{12}$, leading to a contradiction. So event Z never happens in this case. Note that $\mathcal{B}_{\mathsf{H}}$ wins as long as event Z happens. Consequently, $\Pr\left[Z \text{ in } \mathsf{G}^{1.1}_{11,b}\right] \leq \mathsf{Adv}^{\mathsf{owf}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}})$, so

$$\left|\Pr\left[\mathsf{G}^{1.1}_{10,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{11,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{owf}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}),$$
$$\left|adv^{1.1}_{10} - adv^{1.1}_{11}\right| \leq 2\mathsf{Adv}^{\mathsf{owf}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}). \tag{15}$$

Game $\mathsf{G}^{1.1}_{12,b}$. It is the same as $\mathsf{G}^{1.1}_{11,b}$ except for the generation of session key $\mathsf{sKey}[\mathsf{sID}^*]$ in Oracle $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$.

- $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$. During the process, if the query (sID^*, M_2) passes **Rejection Rule 1 and Rule 2**, oracle $\mathsf{Der}_{\mathsf{init}}$ uniformly samples $\mathsf{sKey}[\mathsf{sID}^*] \leftarrow_{\$} \mathcal{K}$, instead of invoking $\mathsf{sKey}[\mathsf{sID}^*] \leftarrow \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ as did in $\mathsf{G}^{1.1}_{11,b}$.

Query (sID^*, M_2) passes **Rejection Rule 1 and Rule 2** means that sID^* shares the same (c_2, C) with some partner session $\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}^*)$. In other words, oracle $\mathsf{Der_{resp}}(\mathsf{sID}, \overline{M}_1)$ also obtains (c_2, C), hence results in the same m_{21}. Then $\mathsf{Der_{resp}}(\mathsf{sID}, \overline{M}_1)$ may output $\overline{M}_2$ and generate session key $\mathsf{sKey}[\mathsf{sID}] \leftarrow \mathsf{PRF}(\overline{m}_1, \overline{M}_1|\overline{M}_2) \oplus \mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$.
Hence all the information about m_{21} leaked to $\mathcal{A}$ is limited by $\mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$. Recall that sID^* has no matching session, i.e., $\mathfrak{M}[\mathsf{sID}^*] = \emptyset$. Therefore, $\overline{M}_1|\overline{M}_2 \neq M_1|M_2$. Given that m_{21} is randomly distributed, we know that $\mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random, hence $\mathsf{PRF}(\overline{m}_1, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random as well. This yields

$$\left|\Pr\left[\mathsf{G}^{1.1}_{11,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{12,b} \Rightarrow 1\right]\right| \leq \ell \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}),$$
$$\left|adv^{1.1}_{11} - adv^{1.1}_{12}\right| \leq 2\ell \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}), \tag{16}$$

where factor ℓ is resulted from the guessing strategy of reduction to the pseudo-randomness of PRF, since there are at ℓ session instance for j^*. Now $\mathcal{A}$'s view in $\mathsf{G}^{1.1}_{12,b}$ is independent of b. So

$$adv^{1.1}_{12} = \left|\Pr\left[\mathsf{G}^{1.1}_{12,0} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{1.1}_{12,1} \Rightarrow 1\right]\right| = 0. \tag{17}$$

By (7),(8),(9),(10), (11), (12),(13), (14), (15), (16), and (17), we have

$$\begin{aligned} adv^{1.1}_{1} \leq 2 \cdot (&\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \ell \cdot \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{B}_{\mathsf{TKEM}}) + (\ell+1) \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) \\ &+ \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}) + \mathsf{Adv}^{\mathsf{owf}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}) + \ell \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}) + (\ell+1) \cdot \delta). \end{aligned} \tag{18}$$

Case 1.2: $\neg stRev[\mathsf{sID}^*] \wedge \neg crp[j^*]$ in Case 1. In this case, neither $st[\mathsf{sID}^*]$ nor j^* is corrupted. Hence, m_{11} is uniformly distributed which further guarantees the pseudo-randomness of session key $\mathsf{sKey}[\mathsf{sID}^*]$.

Game	$\mathsf{Init}(\mathsf{sID}^*)$	$\mathsf{Der_{resp}}(\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}^*))$	$\mathsf{Der_{init}}(\mathsf{sID})$	Remark
$\mathsf{G}_{1,b}$	Abort if $\neg\mathsf{TEST}(\mathsf{sID}^*) \vee (\mathsf{holder}[\mathsf{sID}^*], \mathsf{peer}[\mathsf{sID}^*]) \neq (i^*, j^*)$			with security loss $\mu^2\ell$
$\mathsf{G}^{1.2}_{2,b}$	Abort if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset \vee \mathsf{role}[\mathsf{sID}^*] \neq \mathsf{initiator} \vee stRev[sid^*] \vee crp[j^*]$			$\mathsf{G}_{1,b}$ in Case 1.2
$\mathsf{G}^{1.2}_{3,b}$	$\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$	if $\exists\mathsf{record}[i^*, c_1]$: use $\mathsf{record}[i^*, c_1]$		Correctness of tagged TKEM
$\mathsf{G}^{1.2}_{4,b}$	$seed_{i^*} \leftarrow_{\$} \mathcal{K}$			CCA-security of tagged TKEM
$\mathsf{G}^{1.2}_{5,b}$	$m_{11}, m_{12} \leftarrow_{\$} \mathcal{K}$			pseudo-randomness of PRG
$\mathsf{G}^{1.2}_{6,b}$			$\mathsf{sKey}[\mathsf{sID}^*] \leftarrow_{\$} \mathcal{K}$ for sID^*	pseudo-randomness of PRF

Table 3: Brief description of $\mathsf{G}_{1,b}$ and hybrid games $\mathsf{G}^{1.2}_{2,b}$ - $\mathsf{G}^{1.2}_{6,b}$ for Case 1.2

Now we consider $\mathsf{G}_{2,b}$ - $\mathsf{G}_{6,b}$ in Case 1.2, and denote it by $\mathsf{G}^{1.2}_{2,b}$ - $\mathsf{G}^{1.2}_{6,b}$.
Game $\mathsf{G}^{1.2}_{2,b}$. In $\mathsf{G}^{1.2}_{2,b}$, if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$ or $\mathsf{role}[\mathsf{sID}^*] \neq \mathsf{initiator}$ or $crp[j^*] = \mathbf{true}$ or $stRev[\mathsf{sID}^*] = \mathbf{true}$, $\mathcal{C}$ will return 0 directly. Hence, we have $\Pr\left[\mathsf{G}^{1.2}_{2,b} \Rightarrow 1\right] =$

$\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator} \wedge \neg stRev[\mathsf{sID}^*] \wedge \neg crp[j^*]\right]$,

$\mathsf{G}^{1.1}_{2,b}$, $\mathsf{G}^{1.1}_{3,b}$, $\mathsf{G}^{1.1}_{4,b}$, $\mathsf{G}^{1.1}_{5,b}$, $\mathsf{G}^{1.1}_{6,b}$, $\mathsf{G}^{1.1}_{7,b}$, $\mathsf{G}^{1.1}_{8,b}$, $\mathsf{G}^{1.1}_{9,b}$, $\mathsf{G}^{1.1}_{10,b}$, $\mathsf{G}^{1.1}_{11,b}$, $\mathsf{G}^{1.1}_{12,b}$

$\mathsf{sID}^* \leftarrow_\$ [\ell]$
$(i^*, j^*) \leftarrow_\$ [\mu] \times [\mu]$
for $i \in [\mu]$: $(pk_i, sk_i) \leftarrow \mathsf{AKE.Gen}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AKE}}(\cdot)}(pk_1, \ldots, pk_\mu)$
If $\mathsf{Trivial}(\mathsf{sID}^*)$: Return 0
Return b'

$\underline{\mathsf{EST}(i,j):}$
$cnt := cnt + 1$, $\mathsf{sID} := cnt$
If $\mathsf{sID} = \mathsf{sID}^* \wedge (i,j) \neq (i^*, j^*)$: abort //Not Case 1.1
$(\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}]) := (i,j)$
$(stRev[\mathsf{sID}], rev[\mathsf{sID}]) := (\mathbf{false}, \mathbf{false})$
Return sID

$\underline{\mathsf{CORRUPT}(i):}$
If $i = i^* \wedge i = j^*$: abort //Not Case 1.1
$crp[i] := \mathbf{true}$
Return sk_i

$\underline{\mathsf{TEST}(\mathsf{sID}):}$ //Only one query
If $\mathsf{sID} \neq \mathsf{sID}^*$: abort //Not Case 1.1
If $\mathsf{sKey}[sid^*] = \perp$: Return $\perp$
$K_0^* \leftarrow_\$ \mathcal{K}, K_1^* := \mathsf{sKey}[\mathsf{sID}^*]$
Return K_b^*

$\underline{\mathsf{Init}(\mathsf{sID}):}$
If $\mathsf{holder}[\mathsf{sID}] = \perp \vee \mathsf{sent}[\mathsf{sID}] \neq \perp$: Return $\perp$
$\mathsf{role}[\mathsf{sID}] = \mathsf{initiator}$
$(i,j) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
$(c_1, seed_i) \leftarrow \mathsf{TKEM.Encap}(pk_j, i)$
If $\mathsf{sID} = \mathsf{sID}^*$: $seed_i \leftarrow_\$ \mathcal{K}$
$m_{11} | m_{12} \leftarrow \mathsf{PRG}(seed_i)$
If $\mathsf{sID} = \mathsf{sID}^*$: $m_{11}, m_{12} \leftarrow_\$ \mathcal{K}$
If $\mathsf{sID} = \mathsf{sID}^*$:
 $\mathsf{record}[i, c_1] := (seed_i, m_{11}, m_{12})$
$\sigma := \mathsf{H}(m_{12})$
$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{PKE.Gen}$
$M_1 := (\tilde{pk}, c_1)$
$st[\mathsf{sID}] := (m_{11}, \tilde{sk}, \sigma, M_1)$; $\mathsf{sent}[\mathsf{sID}] := M_1$
Return M_1

$\underline{\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1)):}$
If $\mathsf{holder}[\mathsf{sID}] = \perp \vee \mathsf{role}[\mathsf{sID}] = \mathsf{initiator} \vee \mathsf{sKey}[\mathsf{sID}] \neq \perp$:
 Return $\perp$
If $\mathsf{sID} = \mathsf{sID}^*$: Return $\perp$ //Not Case 1.1
$\mathsf{role}[\mathsf{sID}] = \mathsf{responder}$; $(j,i) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
If $\mathsf{TKEM.Decap}(sk_j, c_1, i) = \perp$: Return $\perp$
$seed_i' \leftarrow \mathsf{TKEM.Decap}(sk_j, c_1, i)$
$m_{11}' | m_{12}' \leftarrow \mathsf{PRG}(seed_i')$
If $i = i^* \wedge j = j^* \wedge \mathsf{record}[i, c_1] \neq \perp$:
 Parse $\mathsf{record}[i, c_1] = (seed_i, m_{11}, m_{12})$
 $seed_i' := seed_i$
 $(m_{11}', m_{12}') := (m_{11}, m_{12})$
$(c_2, seed_j) \leftarrow \mathsf{KEM.Encap}(pk_i)$
If $(j,i) = (j^*, i^*)$: $seed_j \leftarrow_\$ \mathcal{K}$
$\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, c_2)$
$m_{21} | m_{22} \leftarrow \mathsf{PRG}(seed_j)$
If $(j,i) = (j^*, i^*)$:
 $m_{21}, m_{22} \leftarrow_\$ \mathcal{K}$
 $\mathsf{record}[c_2] := (seed_j, m_{21}, m_{22})$
$C := m_{12}' \oplus m_{22}$
If $(j,i) = (j^*, i^*)$:
 $C \leftarrow_\$ \mathcal{K}$
 $m_{22} := C \oplus m_{12}'$
 $\mathsf{record}[c_2] := (seed_j, m_{21}, m_{22})$
 $\mathsf{record}[c_2, C] := \mathsf{sID}$
$M_2 := (\tilde{c}, C)$
$K_1 \leftarrow \mathsf{PRF}(m_{11}', M_1 | M_2)$
$K_2 \leftarrow \mathsf{PRF}(m_{21}, M_1 | M_2)$
$K := K_1 \oplus K_2$
$\mathsf{sKey}[\mathsf{sID}] := K$
$(\mathsf{recv}[\mathsf{sID}], \mathsf{sent}[\mathsf{sID}]) := (M_1, M_2)$
Return M_2

$\underline{\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C)):}$
If $st[\mathsf{sID}] = \perp \vee \mathsf{sKey}[\mathsf{sID}] \neq \perp$:
 Return $\perp$
$(i,j) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
Parse $st[\mathsf{sID}] = (m_{11}, \tilde{sk}, \sigma, M_1 = (\tilde{pk}, c_1))$
If $\mathsf{PKE.Dec}(\tilde{sk}, \tilde{c}) = \perp$:
 Return $\perp$
$c_2' \leftarrow \mathsf{PKE.Dec}(\tilde{sk}, \tilde{c})$
If $\mathsf{KEM.Decap}(sk_i, c_2') = \perp$:
 Return $\perp$
If $\mathsf{sID} = \mathsf{sID}^* \wedge \mathsf{record}[(c_2', C)] = \perp$:
 $\mathsf{sKey}[\mathsf{sID}] := \mathbf{reject}$ //Reject Rule 2
 Return $\perp$
$seed_j' \leftarrow \mathsf{KEM.Decap}(sk_i, c_2')$
$m_{21}' | m_{22}' \leftarrow \mathsf{PRG}(seed_j')$
If $i = i^* \wedge \mathsf{record}[c_2'] \neq \perp$:
 Parse $\mathsf{record}[c_2'] = (seed_j, m_{21}, m_{22})$
 $seed_j' := seed_j$
 $(m_{21}', m_{22}') := (m_{21}, m_{22})$
$m_{12}' := C \oplus m_{22}'$
If $\mathsf{H}(m_{12}') \neq \sigma$:
 $\mathsf{sKey}[\mathsf{sID}] := \mathbf{reject}$
 Return $\perp$
If $\mathsf{sID} = \mathsf{sID}^*$:
 Retrieve m_{12} from $\mathsf{record}[i^*, c_1]$
 If $m_{12} \neq m_{12}'$:
  $\mathsf{sKey}[\mathsf{sID}] := \mathbf{reject}$ //Reject Rule 1
  Return $\perp$
$K_1 \leftarrow \mathsf{PRF}(m_{11}, M_1 | M_2)$
$K_2 \leftarrow \mathsf{PRF}(m_{21}', M_1 | M_2)$
$K := K_1 \oplus K_2$
If $\mathsf{sID} = \mathsf{sID}^*$: $K \leftarrow_\$ \mathcal{K}$
$\mathsf{sKey}[\mathsf{sID}] := K$
$\mathsf{recv}[\mathsf{sID}] := M_2$

Fig. 8: Game $\mathsf{G}^{1.1}_{2,b}$ - $\mathsf{G}^{1.1}_{12,b}$. Queries to $\{\mathsf{REVEAL}, \mathsf{REV\text{-}STATE}\}$ are defined as in the original game in Fig. 6.

and $adv_2^{1.2} := \left|\Pr\left[\mathsf{G}_{2,0}^{1.2} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{2,1}^{1.2} \Rightarrow 1\right]\right|$.

Game $\mathsf{G}_{3,b}^{1.2}$. It is the same as $\mathsf{G}_{2,b}^{1.2}$ except for the behavior of Oracles $\mathsf{Init}(\mathsf{sID}^*)$ and $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Init}(\mathsf{sID}^*)$. It will additionally record the intermediate values $(i^*, c_1, seed_{i^*}, m_{11}, m_{12})$ for sID^* with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, where $(c_1, seed_{i^*})$ are the outputs of $\mathsf{TKEM.Encap}$ and m_{11}, m_{12} are the outputs of PRG.
– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$ and $\mathsf{peer}[\mathsf{sID}] = i^*$, and the input c_1 is consistent, it will directly use the recorded value of $(seed_{i^*}, m_{11}, m_{12})$ for the generation of session key and the second round-message M_2, rather than computing them with $seed_{i^*} \leftarrow \mathsf{TKEM.Decap}$ and $(m_{11}, m_{12}) \leftarrow \mathsf{PRG}$ as did in $\mathsf{G}_{6,b}^{1.1}$.

Due to the (1-δ)-correctness of TKEM, we have

$$\left|\Pr\left[\mathsf{G}_{2,b}^{1.2} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{3,b}^{1.2} \Rightarrow 1\right]\right| \le \delta,$$
$$\left|adv_2^{1.2} - adv_3^{1.2}\right| \le 2\delta. \tag{19}$$

Game $\mathsf{G}_{4,b}^{1.2}$. It is the same as $\mathsf{G}_{3,b}^{1.2}$ except the behavior of Oracles $\mathsf{Init}(\mathsf{sID}^*)$.

– $\mathsf{Init}(\mathsf{sID}^*)$. Now $seed_{i^*}$ is randomly sampled by $seed_{i^*} \leftarrow_{\$} \mathcal{K}$ instead of being generated by $(c_1, seed_{i^*}) \leftarrow \mathsf{TKEM.Encap}(pk_{j^*})$ in $\mathsf{G}_{3,b}^{1.2}$. The oracle records the intermediate values $(c_1, seed_{i^*}, m_{11}, m_{12})$ for sID^* with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, where c_1, m_{11}, m_{12} are computed in the same way as in $\mathsf{G}_{3,b}^{1.2}$.

By the CCA security of TKEM, we know that the encapsulated key $seed_{i^*}$ is pseudo-random. Hence we have

$$\left|\Pr\left[\mathsf{G}_{3,b}^{1.2} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{4,b}^{1.2} \Rightarrow 1\right]\right| \le \mathsf{Adv}_{\mathsf{TKEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{TKEM}}),$$
$$\left|adv_3^{1.2} - adv_4^{1.2}\right| \le 2\mathsf{Adv}_{\mathsf{TKEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{TKEM}}). \tag{20}$$

Game $\mathsf{G}_{5,b}^{1.2}$ It is the same as $\mathsf{G}_{4,b}^{1.2}$ except the behavior of Oracles $\mathsf{Init}(\mathsf{sID}^*)$.

– $\mathsf{Init}(\mathsf{sID}^*)$. Now m_{11}, m_{12} are randomly sampled by $m_{11}, m_{12} \leftarrow_{\$} \mathcal{K}$, instead of $m_{11}|m_{12} \leftarrow_{\$} \mathsf{PRG}(seed_i)$ as did in $\mathsf{G}_{4,b}^{1.2}$. The oracle records the intermediate value $(c_1, seed_{i^*}, m_{11}, m_{12})$ for sID^* with $\mathsf{record}[i^*, c_1] := (seed_{i^*}, m_{11}, m_{12})$, where $c_1, seed_{i^*}$ are computed in the same way as in $\mathsf{G}_{4,b}^{1.2}$.

Given random $seed_{j^*}$, the output of $\mathsf{PRG}(seed_{j^*})$ is pseudo-random. So we have

$$\left|\Pr\left[\mathsf{G}_{4,b}^{1.2} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{5,b}^{1.2} \Rightarrow 1\right]\right| \le \mathsf{Adv}_{\mathsf{PRG}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRG}}),$$
$$\left|adv_4^{1.2} - adv_5^{1.2}\right| \le 2\mathsf{Adv}_{\mathsf{PRG}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRG}}). \tag{21}$$

Game $\mathsf{G}_{6,b}^{1.2}$. It is the same as $\mathsf{G}_{5,b}^{1.2}$ except for the generation of session key $\mathsf{sKey}[\mathsf{sID}^*]$ in Oracle $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$.

– $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}^*, M_2 = (\tilde{c}, C))$. During the process, if the query (sID^*, M_2) leads to $\mathsf{H}(m'_{12}) = \sigma$, oracle $\mathsf{Der}_{\mathsf{init}}$ uniformly samples $\mathsf{sKey}[\mathsf{sID}^*] \leftarrow_{\$} \mathcal{K}$, instead of invoking $\mathsf{sKey}[\mathsf{sID}^*] \leftarrow \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ as did in $\mathsf{G}_{5,b}^{1.2}$.

Let $st[\mathsf{sID}^*] = (m_{11}, \tilde{sk}, \sigma, M_1 = (\tilde{pk}, c_1))$. If c_1 is received by some partner session $\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}^*)$, then oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, \overline{M}_1)$ computes the same m_{11} with sID^* and generates its session key with $\mathsf{sKey}[\mathsf{sID}] := \mathsf{PRF}(m_{11}, \overline{M}_1|\overline{M}_2) \oplus \mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$, where $\overline{M}_2$ is the output message of oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, \overline{M}_1)$. Therefore, all the information about m_{11} leaked to adversary $\mathcal{A}$ is limited by $\mathsf{PRF}(m_{11}, \overline{M}_1|\overline{M}_2)$. Recall that sID^* has no matching session, i.e., $\mathfrak{M}[\mathsf{sID}^*] = \emptyset$. So $\overline{M}_1|\overline{M}_2 \neq M_1|M_2$. Given that m_{11} is randomly distributed, we know that $\mathsf{PRF}(m_{11}, M_1|M_2)$ is pseudo-random, hence $\mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random as well. This yields

$$\left|\Pr\left[\mathsf{G}_{5,b}^{1.2} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{6,b}^{1.2} \Rightarrow 1\right]\right| \leq \mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRF}}),$$
$$\left|adv_5^{1.2} - adv_6^{1.2}\right| \leq 2\mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRF}}). \tag{22}$$

Now $\mathcal{A}$'s view is $\mathsf{G}_{6,b}^{1.2}$ is independent of b. So

$$adv_6^{1.2} = \left|\Pr\left[\mathsf{G}_{6,0}^{1.2} \Rightarrow 1\right] - \Pr\left[\mathsf{G}_{6,1}^{1.2} \Rightarrow 1\right]\right| = 0. \tag{23}$$

By (19), (20),(21), (22), and (23), we have

$$adv_1^{1.2} \leq 2 \cdot (\mathsf{Adv}_{\mathsf{TKEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{TKEM}}) + \mathsf{Adv}_{\mathsf{PRG}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRG}}) + \mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRF}}) + \delta). \tag{24}$$

By (6), (18), (24), we have

$$adv_1^1 = 2 \cdot \Big(\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{KEM}}) + (\ell+1) \cdot \mathsf{Adv}_{\mathsf{TKEM}}^{\mathsf{CCA}}(\mathcal{B}_{\mathsf{TKEM}}) + (\ell+2) \cdot \mathsf{Adv}_{\mathsf{PRG}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRG}})$$
$$+\mathsf{Adv}_{\mathsf{H}}^{\mathsf{tcr}}(\mathcal{B}_{\mathsf{H}}) + \mathsf{Adv}_{\mathsf{H}}^{\mathsf{owf}}(\mathcal{B}_{\mathsf{H}}) + (\ell+1) \cdot \mathsf{Adv}_{\mathsf{PRF}}^{\mathsf{ps}}(\mathcal{B}_{\mathsf{PRF}}) + (\ell+2)\delta\Big). \tag{25}$$

Case 2: $\mathsf{G}_{0,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{responder}$. In this case, $\mathfrak{M}(\mathsf{sID}^*) = \emptyset$ means that no matching session exists for sID^*, and $\mathsf{role}[\mathsf{sID}^*] = \mathsf{responder}$ implies that test session is held by a $\mathsf{responder}$.

Then, in session sID^*, responder i^* must suffer from an active attack from adversary $\mathcal{A}$. If j^* is further corrupted, then this is a trivial attack leading to $\mathsf{G}_{1,b} \Rightarrow 0$. Therefore, for $b \in \{0, 1\}$,

$$\Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{responder} \wedge crp[j^*]\right] = 0. \tag{26}$$

Consequently,

$$\Pr\left[\mathsf{G}_{1,b}^2 \Rightarrow 1\right] = \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{responder}\right]$$
$$= \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{responder} \wedge \neg crp[j^*]\right].$$

Since party j^* will never be corrupted in this case, m_{21} is uniformly distributed which further grantees the pseudo-randomness of session key $\mathsf{sKey}[\mathsf{sID}^*]$.

Now we consider $\mathsf{G}_{2,b}$ - $\mathsf{G}_{6,b}$ in Case 2, and denote it by $\mathsf{G}_{2,b}^2$ - $\mathsf{G}_{6,b}^2$.

Game $\mathsf{G}_{2,b}^2$. In $\mathsf{G}_{2,b}^2$, if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$ or $\mathsf{role}[\mathsf{sID}^*] \neq \mathsf{responder}$ or $crp[j^*] =$ **true**, $\mathcal{C}$ will return 0 directly. Hence, we have

$$\Pr\left[\mathsf{G}_{2,b}^2 \Rightarrow 1\right] = \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) = \emptyset \wedge \mathsf{role}[\mathsf{sID}^*] = \mathsf{responder} \wedge \neg crp[j^*]\right].$$

Game $\mathsf{G}_{3,b}^2$. It is the same as $\mathsf{G}_{2,b}^2$ except for the behavior of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$ and $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$.

Game	$\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*)$	$\mathsf{Der}_{\mathsf{init}}(\mathsf{sID})$	Remark
$\mathsf{G}_{1,b}$	Abort if $\neg\mathsf{TEST}(\mathsf{sID}^*) \vee (\mathsf{holder}[\mathsf{sID}^*], \mathsf{peer}[\mathsf{sID}^*]) \neq (i^*, j^*)$		with security loss $\mu^2\ell$
$\mathsf{G}^2_{2,b}$	Abort if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset \vee \mathsf{role}[\mathsf{sID}^*] \neq \mathsf{responder} \vee crp[j^*]$		$\mathsf{G}_{1,b}$ in Case 2
$\mathsf{G}^2_{3,b}$	$\mathsf{record}[c_2] := (seed_{i^*}, m_{21}, m_{22})$	if $\exists\mathsf{record}[c_2]$: use $\mathsf{record}[c_2]$ for $\mathsf{holder}[\mathsf{sID}] = j^*$	Correctness of KEM
$\mathsf{G}^2_{4,b}$	$seed_{i^*} \leftarrow_\$ \mathcal{K}$		CCA-security of KEM
$\mathsf{G}^2_{5,b}$	$m_{21}, m_{22} \leftarrow_\$ \mathcal{K}$		pseudo-randomness of PRG
$\mathsf{G}^2_{6,b}$	$\mathsf{sKey}[\mathsf{sID}^*] \leftarrow_\$ \mathcal{K}$		pseudo-randomness of PRF

Table 4: Brief description of $\mathsf{G}_{1,b}$ and hybrid games $\mathsf{G}^2_{2,b}$ - $\mathsf{G}^2_{6,b}$ for Case 2

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$. During the process, it will additionally record the intermediate values $(c_2, seed_{i^*}, m_{21}, m_{22})$ with $\mathsf{record}[c_2] := (seed_{i^*}, m_{21}, m_{22})$, where $(c_2, seed_{i^*}) \leftarrow \mathsf{KEM.Encap}$ and $(m_{21}, m_{22}) \leftarrow \mathsf{PRG}$.
- $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = j^*$, and the output c'_2 of $\mathsf{PKE.Dec}$ has ever been recorded with $\mathsf{record}[c'_2] := (seed_{i^*}, m_{21}, m_{22})$ by $\mathsf{Der}_{\mathsf{resp}}$, it will directly use the recorded values of $(seed_{i^*}, m_{21}, m_{22})$ for the generation of session key, rather than computing $seed_{i^*}$ with $\mathsf{KEM.Decap}$ and (m_{21}, m_{22}) with PRG as did in $\mathsf{G}^2_{2,b}$.

Due to the (1-δ)-correctness of KEM, we have

$$\left|\Pr\left[\mathsf{G}^2_{2,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^3_{3,b} \Rightarrow 1\right]\right| \leq \delta,$$
$$\left|adv^2_2 - adv^2_3\right| \leq 2\delta. \tag{27}$$

Game $\mathsf{G}^2_{4,b}$. It is the same as $\mathsf{G}^2_{3,b}$ except the behavior of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$. During the process, the value of $seed_{i^*}$ is randomly chosen in $\mathsf{G}^2_{4,b}$, instead of being generated by $\mathsf{KEM.Encap}$ in $\mathsf{G}^2_{3,b}$. The values of c_2, m_{21}, m_{22} are still the outputs of $\mathsf{KEM.Encap}$ and PRG, and $(c_2, seed_{j^*}, m_{21}, m_{22})$ is recorded in the same way as $\mathsf{G}^2_{3,b}$.

Due to the CCA security of KEM, we have

$$\left|\Pr\left[\mathsf{G}^2_{3,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^2_{4,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}),$$
$$\left|adv^2_3 - adv^2_4\right| \leq 2\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}). \tag{28}$$

Game $\mathsf{G}^2_{5,b}$. It is the same as $\mathsf{G}^2_{4,b}$ except the behavior of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$. During the process, the value of m_{21}, m_{22} are randomly chosen in $\mathsf{G}^2_{5,b}$, instead of being generated by PRG as in $\mathsf{G}^2_{4,b}$. The values of $c_2, seed_{i^*}$ are generated the same way as in $\mathsf{G}^2_{4,b}$. And $(c_2, seed_{i^*}, m_{21}, m_{22})$ is recorded in the same way as $\mathsf{G}^2_{4,b}$.

Given random $seed_{i^*}$, the output of $\mathsf{PRG}(seed_{i^*})$ is pseudo-random. We have

$$\left|\Pr\left[\mathsf{G}^2_{4,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^2_{5,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}),$$
$$\left|adv^2_4 - adv^2_5\right| \leq 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}). \tag{29}$$

Game $\mathsf{G}^2_{6,b}$. It is the same as $\mathsf{G}^2_{5,b}$ except for the generation of session key $\mathsf{sKey}[\mathsf{sID}^*]$ in Oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{TKEM.Decap}(sk_{i^*}, c_1, j^*) \neq \perp$, oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, M_1)$ uniformly samples $\mathsf{sKey}[\mathsf{sID}^*] \leftarrow_\$ \mathcal{K}$, instead of invoking $\mathsf{sKey}[\mathsf{sID}^*] \leftarrow \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ as did in $\mathsf{G}^2_{5,b}$. Besides, oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}^*, \mathsf{M}_1)$ still records the intermediate values $(c_2, seed_{i^*}, m_{21}, m_{22})$ with $\mathsf{record}[c_2] := (seed_{i^*}, m_{21}, m_{22})$ as did in $\mathsf{G}^2_{5,b}$.

For any partner session $\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}^*)$, If oracle $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, \overline{M}_2)$ computes the same c_2 as that in $\mathsf{record}[c_2] = (seed_{i^*}, m_{21}, m_{22})$, then sID and sID^* share the same m_{21}. Then $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, \overline{M}_2)$ may generate session key $\mathsf{sKey}[\mathsf{sID}] \leftarrow \mathsf{PRF}(\overline{m}_1, \overline{M}_1|\overline{M}_2) \oplus \mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$.
Therefore, all the information about m_{21} leaked to $\mathcal{A}$ is limited by $\mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$. Recall that sID^* has no matching session, i.e., $\mathfrak{M}[\mathsf{sID}^*] = \emptyset$. Therefore, $\overline{M}_1|\overline{M}_2 \neq M_1|M_2$. Given that m_{21} is randomly distributed, the security of PRF implies that $\mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random, hence $\mathsf{PRF}(\overline{m}_1, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random as well. This yields

$$\left|\Pr\left[\mathsf{G}^2_{5,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^2_{6,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}),$$
$$\left|adv^2_5 - adv^2_6\right| \leq 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}). \tag{30}$$

Now $\mathcal{A}$'s view is $\mathsf{G}^2_{6,b}$ is independent of b. So

$$adv^2_6 = \left|\Pr\left[\mathsf{G}^2_{6,0} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^2_{6,1} \Rightarrow 1\right]\right| = 0. \tag{31}$$

By (27),(28),(29), (30), and (31), we have

$$adv^2_1 \leq 2 \cdot (\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}) + \delta). \tag{32}$$

Case 3: $\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$. In this case, $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$ means that there is no active attack on the target test session sID^*.

We define
$\Pr\left[\mathsf{G}^3_{1,b} \Rightarrow 1\right] := \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) \neq \emptyset\right]$..
Game $\mathsf{G}^3_{2,b}$. In $\mathsf{G}^3_{2,b}$, if $\mathfrak{M}(\mathsf{sID}^*) = \emptyset$, $\mathcal{C}$ will abort and return 0 directly. Hence, we have $\Pr\left[\mathsf{G}^3_{2,b} \Rightarrow 1\right] = \Pr\left[\mathsf{G}^3_{1,b} \Rightarrow 1\right] = \Pr\left[\mathsf{G}_{1,b} \Rightarrow 1 \wedge \mathfrak{M}(\mathsf{sID}^*) \neq \emptyset\right]$,

$$adv^3_1 = adv^3_2. \tag{33}$$

Game $\mathsf{G}^3_{3,b}$. In $\mathsf{G}^3_{3,b}$, if $|\mathfrak{M}(\mathsf{sID}^*)| > 1$, then $\mathcal{C}$ will abort and return 0 directly. We now analyse $\Pr\left[|\mathfrak{M}(\mathsf{sID}^*)| > 1\right]$ depending on the role of sID^*.

(I) $\mathsf{role}[\mathsf{sID}^*] = \mathsf{initiator}$. In this case, $|\mathfrak{M}(\mathsf{sID}^*)| > 1$ means that there are at least two sessions sID_1 and sID_2 generating the same second round message $(\tilde{c}, m_{12} \oplus m_{22})$. According to $(1-\delta)$-correctness of PKE, $\mathsf{sID}_1, \mathsf{sID}_2$ must encrypt the same c_2 to get $\tilde{c}$ but using independent randomness. Further by the γ-spreadness of PKE, we get $\Pr\left[|\mathfrak{M}(\mathsf{sID}^*)| > 1\right] \leq \delta + 2^{-\gamma}$.

(II) $\mathsf{role}[\mathsf{sID}^*] = \mathsf{responder}$**.** In this case, $|\mathfrak{M}(\mathsf{sID}^*)| > 1$ means that there are at least two sessions sID_1 and sID_2 generating the same first round message $(c_1, \tilde{pk})$ using independent randomness. By the γ-diversity of PKE, we get $\Pr[|\mathfrak{M}(\mathsf{sID}^*)| > 1] \leq 2^{-\gamma}$.

Therefore, $\left|\Pr\left[\mathsf{G}^3_{3,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^3_{3,b} \Rightarrow 1\right]\right| \leq \Pr[|\mathfrak{M}(\mathsf{sID}^*)| > 1] \leq \delta + 2^{-\gamma}$,

$$\left|adv^3_2 - adv^3_3\right| \leq 2(\delta + 2^{-\gamma}). \tag{34}$$

Note that $\mathsf{G}^3_{3,b} \Rightarrow 1$ only if $|\mathfrak{M}(\mathsf{sID}^*)| = 1$, i.e., there exists only one session sID' matching the target test session sID^*.

Game $\mathsf{G}^3_{4,b}$. In $\mathsf{G}^3_{4,b}$, challenger $\mathcal{C}$ will first randomly choose a session sID' among the sessions of user j^*. At the end of $\mathsf{G}^3_{4,b}$, $\mathcal{C}$ will check whether $\mathfrak{M}(\mathsf{sID}^*) = \{\mathsf{sID}'\}$ (sID' is matching with sID^*). If not, $\mathcal{C}$ will abort and return 0 directly. User j^* has at most ℓ sessions, so $\Pr\left[\mathsf{G}^3_{3,b} \Rightarrow 1\right] = \ell \cdot \Pr\left[\mathsf{G}^3_{4,b} \Rightarrow 1\right]$,

$$adv^3_3 = \ell \cdot adv^3_4. \tag{35}$$

For the pair $(\mathsf{sID}^*, \mathsf{sID}')$, one role is $\mathsf{initiator}$ and the other $\mathsf{responder}$. For simplicity, we denote the initiator session by sID_I and the responder session by sID_R. Meanwhile, We define $(I, R) := (\mathsf{holder}[\mathsf{sID}_I], \mathsf{peer}[\mathsf{sID}_I])$.

If $\mathcal{A}$ both corrupts I and obtains its state by $\mathsf{StateReveal}(\mathsf{sID}_I)$, then this is a trivial attack leading to $\mathsf{G}^3_{4,b} \Rightarrow 0$. Therefore,

$$\begin{aligned}\Pr\left[\mathsf{G}^3_{4,b} \Rightarrow 1\right] \leq \Pr\left[\mathsf{G}^3_{4,b} \Rightarrow 1 \wedge \neg crp[I]\right] + \quad & \textbf{(Case 3.1)} \\ \Pr\left[\mathsf{G}^3_{4,b} \Rightarrow 1 \wedge \neg stRev[\mathsf{sID}_I]\right], \quad & \textbf{(Case 3.2)} \\ adv^3_4 \leq adv^{3.1}_4 + adv^{3.2}_4. &\end{aligned} \tag{36}$$

Case 3.1: $\mathsf{G}^3_{4,b} \Rightarrow 1 \wedge \neg crp[I]$. In this case, the initiator I is never corrupted. Hence m_{21} is uniformly distributed which further guarantees the pseudo-randomness of session key $\mathsf{sKey}[\mathsf{sID}^*]$.

Define $(\mathsf{G}^{3.1}_{i,b} \Rightarrow 1) := (\mathsf{G}^3_{i,b} \Rightarrow 1 \ \wedge \ \neg crp[I])$.

Game	$\mathsf{Der_{resp}}(\mathsf{sID}_R)$	$\mathsf{Der_{init}}(\mathsf{sID})$	Remark
$\mathsf{G}_{1,b}$	Abort if $\neg\mathsf{TEST}(\mathsf{sID}^*) \vee (\mathsf{holder}[\mathsf{sID}^*], \mathsf{peer}[\mathsf{sID}^*]) \neq (i^*, j^*)$		with security loss $\mu^2\ell$
$\mathsf{G}^3_{2,b}$	Abort if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$		$\mathsf{G}_{1,b}$ in Case 3
$\mathsf{G}^3_{3,b}$	Abort if $|\mathfrak{M}(\mathsf{sID}^*)| > 1$		γ-diversity and γ-spreadness of PKE
$\mathsf{G}^3_{4,b}$	Abort if $\mathfrak{M}(\mathsf{sID}) \neq \{\mathsf{sID}'\}$		with security loss ℓ
$\mathsf{G}^{3.1}_{5,b}$	Abort if initiator party I in $\mathsf{sID}^*, \mathsf{sID}'$ is corrupted		$\mathsf{G}^3_{4,b}$ in Case 3.1
$\mathsf{G}^{3.1}_{6,b}$	$\mathsf{record}[c_2] :=$ $(seed_R, m_{21}, m_{22})$	if $\exists\mathsf{record}[c_2]$: use $\mathsf{record}[c_2]$ for $\mathsf{holder}[\mathsf{sID}] = I$	Correctness of KEM
$\mathsf{G}^{3.1}_{7,b}$	$seed_R \leftarrow_\$ \mathcal{K}$		CCA-security of KEM
$\mathsf{G}^{3.1}_{8,b}$	$m_{21}, m_{22} \leftarrow_\$ \mathcal{K}$		pseudo-randomness of PRG
$\mathsf{G}^{3.1}_{9,b}$	$\mathsf{sKey}[\mathsf{sID}_R] \leftarrow_\$ \mathcal{K}$	$\mathsf{sKey}[\mathsf{sID}_I] := \mathsf{sKey}[\mathsf{sID}_R]$	pseudo-randomness of PRF

Table 5: Brief description of games $\mathsf{G}_{1,b}$, $\mathsf{G}^3_{2,b}$ - $\mathsf{G}^3_{4,b}$ and $\mathsf{G}^{3.1}_{5,b}$ - $\mathsf{G}^{3.1}_{9,b}$ for Case 3.1

Game $\mathsf{G}^{3.1}_{5,b}$. In $\mathsf{G}^{3.1}_{5,b}$, challenger $\mathcal{C}$ will abort and return 0 directly as long as $crp[I] = \textbf{true}$. We have

$$\Pr\left[\mathsf{G}^{3.1}_{5,b} \Rightarrow 1\right] = \Pr\left[\mathsf{G}^{3}_{4,b} \Rightarrow 1 \wedge \neg crp[I]\right],$$
$$adv^{3.1}_4 = adv^{3.1}_5. \tag{37}$$

Game $\mathsf{G}^{3.1}_{6,b}$. It is the same as $\mathsf{G}^{3.1}_{6,b}$ except for the behavior of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$ and $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}_I, M_2 = (\tilde{c}, C))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$. It will additionally record the intermediate values $(c_2, seed_R, m_{21}, m_{22})$ with $\mathsf{record}[c_2] := (seed_R, m_{21}, m_{22})$, where $(c_2, seed_R) \leftarrow \mathsf{KEM.Encap}$ and $(m_{21}, m_{22}) \leftarrow \mathsf{PRG}$.
- $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$.During the process, if $\mathsf{holder}[\mathsf{sID}] = I$, and the output c_2' of $\mathsf{PKE.Dec}$ has ever been recorded with $\mathsf{record}[c_2'] := (seed_R, m_{21}, m_{22})$ by $\mathsf{Der}_{\mathsf{resp}}$, it will directly use the recorded values of $(seed_R, m_{21}, m_{22})$ for the generation of session key, rather than computing $seed_R$ with $\mathsf{KEM.Decap}$ and (m_{21}, m_{22}) with PRG as did in $\mathsf{G}^{3.1}_{5,b}$.

Due to the (1-δ)-correctness of KEM, we have

$$\left|\Pr\left[\mathsf{G}^{3.1}_{5,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.1}_{6,b} \Rightarrow 1\right]\right| \leq \ell\delta,$$
$$\left|adv^{3.1}_5 - adv^{3.1}_6\right| \leq 2\ell\delta. \tag{38}$$

Game $\mathsf{G}^{3.1}_{7,b}$. It is the same as $\mathsf{G}^{3.1}_{6,b}$ except for the behavior of Oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$. The value of $seed_R$ is randomly chosen in $\mathsf{G}^{3.1}_{7,b}$, instead of being generated by $\mathsf{KEM.Encap}$ in $\mathsf{G}^{3.1}_{6,b}$. The values of c_2, m_{21}, m_{22} are still the outputs of $\mathsf{KEM.Encap}$ and PRG, and $(c_2, seed_R, m_{21}, m_{22})$ is recorded in the same way as $\mathsf{G}^{3.1}_{6,b}$.

Due to the CCA security of KEM, we have

$$\left|\Pr\left[\mathsf{G}^{3.1}_{6,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.1}_{7,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}),$$
$$\left|adv^{3.1}_6 - adv^{3.1}_7\right| \leq 2\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}). \tag{39}$$

Game $\mathsf{G}^{3.1}_{8,b}$. It is the same as $\mathsf{G}^{3.1}_{8,b}$ except for the behavior of Oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$.

- $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$. The value of m_{21}, m_{22} are randomly chosen in $\mathsf{G}^{3.1}_{7,b}$, instead of being generated by PRG as in $\mathsf{G}^{3.1}_{7,b}$. The values of $c_2, seed_R$ are generated the same way as in $\mathsf{G}^{2}_{7,b}$. And $(c_2, seed_R, m_{21}, m_{22})$ is recorded in the same way as $\mathsf{G}^{2}_{7,b}$.

Given random $seed_R$, the output of $\mathsf{PRG}(seed_R)$ is pseudo-random. We have

$$\left|\Pr\left[\mathsf{G}^{3.1}_{7,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.1}_{8,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}),$$
$$\left|adv^{3.1}_7 - adv^{3.1}_8\right| \leq 2 \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}). \tag{40}$$

Game $\mathsf{G}^{3.1}_{9,b}$. It is the same as $\mathsf{G}^{3.1}_{8,b}$ except for the generation of session key $\mathsf{sKey}[\mathsf{sID}^*]$ in Oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$.

– $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$. During the process, if $\mathsf{TKEM.Decap}(sk_R, c_1, I) \neq \perp$, oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1)$ uniformly samples $\mathsf{sKey}[\mathsf{sID}_R] \leftarrow_{\$} \mathcal{K}$, instead of invoking $\mathsf{sKey}[\mathsf{sID}_R] \leftarrow \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ as did in $\mathsf{G}^2_{8,b}$. Besides, oracle $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID_R}, \mathsf{M_1})$ still records the intermediate values $(c_2, seed_R, m_{21}, m_{22})$ with $\mathsf{record}[c_2] := (seed_R, m_{21}, m_{22})$ as did in $\mathsf{G}^2_{8,b}$.
– $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}_I, M_2 = (\tilde{c}, C))$. During the process, if the query (sID_I, M_2) leads to $\mathsf{H}(m'_{12}) = \sigma$, oracle $\mathsf{Der}_{\mathsf{init}}$ computes $\mathsf{sKey}[\mathsf{sID}_I] := \mathsf{sKey}[\mathsf{sID}_R]$, instead of invoking $\mathsf{sKey}[\mathsf{sID}_I] \leftarrow \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ as did in $\mathsf{G}^{1.2}_{8,b}$.

For any partner session $\mathsf{sID} \in \mathfrak{P}(\mathsf{sID}_R) \setminus \{\mathsf{sID}_I\}$, if oracle $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, \overline{M}_2)$ computes the same c_2 as that in $\mathsf{record}[c_2] = (seed_R, m_{21}, m_{22})$, then sID and sID_R share the same m_{21}. Note that $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, \overline{M}_2)$ may generate session key $\mathsf{sKey}[\mathsf{sID}] \leftarrow \mathsf{PRF}(\overline{m}_1, \overline{M}_1|\overline{M}_2) \oplus \mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$.
Therefore, all information about m_{21} leaked to $\mathcal{A}$ is limited by $\mathsf{PRF}(m_{21}, \overline{M}_1|\overline{M}_2)$. Recall that sID_R only matches with sID_I. Therefore, $\overline{M}_1|\overline{M}_2 \neq M_1|M_2$. Given that m_{21} is randomly distributed, the security of PRF implies $\mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random, hence $\mathsf{PRF}(\overline{m}_1, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ is pseudo-random as well. This yields

$$\left|\Pr\left[\mathsf{G}^{3.1}_{8,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.1}_{9,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}),$$
$$\left|adv^{3.1}_8 - adv^{3.1}_9\right| \leq 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}). \quad (41)$$

Now $\mathcal{A}$'s view is $\mathsf{G}^{3.1}_{9,b}$ is independent of b. So

$$adv^{3.1}_9 = \left|\Pr\left[\mathsf{G}^{3.1}_{9,0} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.1}_{9,1} \Rightarrow 1\right]\right| = 0. \quad (42)$$

By (43), (38),(39),(40), (41), and (42), we have

$$adv^{3.1}_4 \leq 2 \cdot (\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}) + \delta\ell). \quad (43)$$

Case 3.2: $\mathsf{G}^3_{4,b} \Rightarrow 1 \wedge \neg stRev[\mathsf{sID}_I]$. In this case, the state of initiator session sID_I is never revealed. Hence m_{21} is uniformly distributed which further guarantees the pseudo-randomness of session key $\mathsf{sKey}[\mathsf{sID}^*]$.

Define $(\mathsf{G}^{3.2}_{i,b} \Rightarrow 1) := (\mathsf{G}^3_{i,b} \Rightarrow 1 \ \wedge \ \neg stRev[\mathsf{sID}_I])$. Next we will prove that $adv^{3.1}_1 = \mathsf{negl}(\lambda)$ with games $\mathsf{G}_{1,b}$, $\mathsf{G}^3_{2,b}$ - $\mathsf{G}^3_{4,b}$ and $\mathsf{G}^{3.2}_{5,b}$ - $\mathsf{G}^{3.2}_{11,b}$. The brief description of $\mathsf{G}_{1,b}$, $\mathsf{G}^3_{2,b}$ - $\mathsf{G}^3_{4,b}$ and $\mathsf{G}^{3.2}_{5,b}$ - $\mathsf{G}^{3.2}_{11,b}$ for Case 3.2 is shown in Table 6 and the full codes of games $\mathsf{G}^3_{2,b}$ - $\mathsf{G}^3_{4,b}$, $\mathsf{G}^{3.2}_{5,b}$ - $\mathsf{G}^{3.2}_{10,b}$ are shown in Fig. 9.
Game $\mathsf{G}^{3.2}_{5,b}$. In game $\mathsf{G}^{3.2}_{5,b}$, challenger $\mathcal{C}$ will abort the game and return 0 directly as long as $stRev[\mathsf{sID}_I] = \mathbf{true}$. Hence, we have

$$\Pr\left[\mathsf{G}^{3.2}_{5,b} \Rightarrow 1\right] = \Pr\left[\mathsf{G}^3_{4,b} \Rightarrow 1 \wedge \neg stRev[I]\right],$$
$$adv^{3.2}_4 = adv^{3.2}_5. \quad (44)$$

Game $\mathsf{G}^{3.2}_{6,b}$. It is the same as $\mathsf{G}^{3.2}_{5,b}$ except for the behavior of Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$ and $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}_I, M_2 = (\tilde{c}, C))$.

Game	$\mathsf{Der_{resp}}(\mathsf{sID}_R)$	$\mathsf{Der_{init}}(\mathsf{sID})$	Remark
$\mathsf{G}_{1,b}$	Abort if $\neg\mathsf{TEST}(\mathsf{sID}^*) \vee (\mathsf{holder}[\mathsf{sID}^*], \mathsf{peer}[\mathsf{sID}^*]) \neq (i^*, j^*)$		with security loss $\mu^2\ell$
$\mathsf{G}^3_{2,b}$	Abort if $\mathfrak{M}(\mathsf{sID}^*) \neq \emptyset$		$\mathsf{G}_{1,b}$ in Case 3
$\mathsf{G}^3_{3,b}$	Abort if $\vert\mathfrak{M}(\mathsf{sID}^*)\vert > 1$		γ-diversity and γ-spreadness of PKE
$\mathsf{G}^3_{4,b}$	Abort if $\mathfrak{M}(\mathsf{sID}) \neq \mathsf{sID}'$		with security loss ℓ
$\mathsf{G}^{3.2}_{5,b}$	Abort if the state of initiator session sID_I is revealed		$\mathsf{G}^3_{4,b}$ in Case 3.2
$\mathsf{G}^{3.2}_{6,b}$	$\mathsf{record}[\tilde{c}] := c_2, \mathsf{record}[c_2] := (seed_R, m_{21}, m_{22})$	if $\exists\mathsf{record}[\tilde{c}]$ or $\exists\mathsf{record}[c_2]$: use $\mathsf{record}[\tilde{c}]$ or $\mathsf{record}[c_2]$ for $\mathsf{holder}[\mathsf{sID}] = I$	Correctness of KEM and PKE
$\mathsf{G}^{3.2}_{7,b}$	$\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, 0)$		CPA-security of PKE
$\mathsf{G}^{3.2}_{8,b}$	$seed_R \leftarrow_{\$} \mathcal{K}$		output pseudo-randomness of KEM
$\mathsf{G}^{3.2}_{9,b}$	$m_{21}, m_{22} \leftarrow_{\$} \mathcal{K}$		pseudo-randomness of PRG
$\mathsf{G}^{3.2}_{10,b}$	$\mathsf{sKey}[\mathsf{sID}_R] \leftarrow_{\$} \mathcal{K}$	$\mathsf{sKey}[\mathsf{sID}_I] := \mathsf{sKey}[\mathsf{sID}_R]$	pseudo-randomness of PRF

Table 6: Brief description of $\mathsf{G}_{1,b}$, $\mathsf{G}^3_{2,b}$ - $\mathsf{G}^3_{4,b}$ and $\mathsf{G}^{3.2}_{5,b}$ - $\mathsf{G}^{3.2}_{11,b}$ for Case 3.2

- $\mathsf{Der_{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$. It will additionally record the intermediate values $(c_2, seed_R, m_{21}, m_{22})$ and $(\tilde{c}, c_2)$ with $\mathsf{record}[c_2] := (seed_R, m_{21}, m_{22})$ and $\mathsf{record}[\tilde{c}] := c_2$ respectively, where $(c_2, seed_R) \leftarrow \mathsf{KEM.Encap}(pk_I)$, $(m_{21}, m_{22}) \leftarrow \mathsf{PRG}(seed_R)$ and $\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, c_2)$.
- $\mathsf{Der_{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$. During the process, if $\mathsf{holder}[\mathsf{sID}] = I$, and the input $\tilde{c}$ has ever been recorded with $\mathsf{record}[\tilde{c}] := c_2'$ by $\mathsf{Der_{resp}}$, it will directly use the recorded value of c_2' for the generation of session key, rather than computing c_2' with $c_2' \leftarrow \mathsf{PKE.Dec}(\tilde{sk}, \tilde{c})$ as did in $\mathsf{G}^{3.2}_{5,b}$. Meanwhile, if c_2' has ever been recorded with $\mathsf{record}[c_2'] := (seed_R, m_{21}, m_{22})$ by $\mathsf{Der_{resp}}$, it will directly use the recorded value of $(seed_R, m_{21}, m_{22})$ for the generation of session key, rather than computing $seed_R$ with $seed_R \leftarrow \mathsf{KEM.Decap}(sk_I, c_2')$ and (m_{21}, m_{22}) with PRG as did in $\mathsf{G}^{3.2}_{5,b}$.

Due to the (1-δ)-correctness of PKE and KEM, we have

$$\left|\Pr\left[\mathsf{G}^{3.2}_{5,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.2}_{6,b} \Rightarrow 1\right]\right| \leq 2\ell\delta,$$
$$\left|adv^{3.2}_5 - adv^{3.2}_6\right| \leq 4\ell\delta. \tag{45}$$

Game $\mathsf{G}^{3.2}_{7,b}$. It is the same as $\mathsf{G}^{3.2}_{6,b}$ except for the generation of $\tilde{c}$ in Oracles $\mathsf{Der_{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$.

- $\mathsf{Der_{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$. It will compute $\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, 0)$, instead of $\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, c_2)$ as did in $\mathsf{G}^{3.2}_{6,b}$. It also records the intermediate values $(\tilde{c}, c_2)$ with $\mathsf{record}[\tilde{c}] := c_2$ in the same way as $\mathsf{G}^{3.2}_{6,b}$.

According to the CPA security of PKE, we have

$$\left|\Pr\left[\mathsf{G}^{3.2}_{6,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.2}_{7,b} \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}}),$$
$$\left|adv^{3.2}_6 - adv^{3.2}_7\right| \leq 2\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}}). \tag{46}$$

Note that $\tilde{c}$ is now independent of $seed_R$.

Game $\mathsf{G}^{3.2}_{8,b}$. It is the same as $\mathsf{G}^{3.2}_{7,b}$ except that in Oracles $\mathsf{Der_{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$, $seed_R$ is randomly chosen with $seed_R \leftarrow_{\$} \mathcal{K}$, instead of being generated by $(c_2, seed_R) \leftarrow \mathsf{KEM.Encap}(pk_I)$ as did in $\mathsf{G}^{3.2}_{7,b}$.

Due to the output pseudo-randomness of KEM, when the randomness r is randomly chosen, $seed_R$ generated by $(\cdot, seed_R) \leftarrow \mathsf{KEM.Encap}(pk_I)$ will also be uniformly distributed, even if sk_I is known to $\mathcal{A}$. So

$$\left|\Pr\left[\mathsf{G}^{3.2}_{7,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.2}_{8,b} \Rightarrow 1\right]\right| \le \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}),$$
$$\left|adv^{3.2}_7 - adv^{3.2}_8\right| \le 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}). \tag{47}$$

Game $\mathsf{G}^{3.2}_{9,b}$. It is the same as $\mathsf{G}^{3.2}_{8,b}$ except that in Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$, m_{21}, m_{22} are randomly chosen with $m_{21}, m_{22} \leftarrow_{\$} \mathcal{K}$, instead of being generated with $m_{21}|m_{22} \leftarrow \mathsf{PRG}(seed_R)$ as did in $\mathsf{G}^{3.2}_{8,b}$. Since $seed_R$ is uniformly distributed, the security of PRG implies that

$$\left|\Pr\left[\mathsf{G}^{3.2}_{8,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.2}_{9,b} \Rightarrow 1\right]\right| \le \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}),$$
$$\left|adv^{3.2}_8 - adv^{3.2}_9\right| \le 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}). \tag{48}$$

Game $\mathsf{G}^{3.2}_{10,b}$. It is the same as $\mathsf{G}^{3.2}_{9,b}$ except that Oracles $\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}_R, M_1 = (\tilde{pk}, c_1))$ uniformly samples session key $\mathsf{sKey}[\mathsf{sID}_R] \leftarrow_{\$} \mathcal{K}$ and $\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}_I, M_2 = (\tilde{c}, C))$ sets its session key as $\mathsf{sKey}[\mathsf{sID}_I] := \mathsf{sKey}[\mathsf{sID}_R]$.
Recall that $\mathsf{sKey}[\mathsf{sID}_I] := \mathsf{sKey}[\mathsf{sID}_R] = \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ in $\mathsf{G}^{3.2}_{9,b}$. Note that m_{21} is uniformly distributed in $\mathsf{G}^{3.2}_{10,b}$ and no information about m_{21} is leaked to $\mathcal{A}$. Then the security of PRF implies the pseudo-randomness of $\mathsf{PRF}(m_{21}, M_1|M_2)$, so $\mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ is also pseudo-random. Therefore,

$$\left|\Pr\left[\mathsf{G}^{3.2}_{9,b} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.2}_{10,b} \Rightarrow 1\right]\right| \le \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}),$$
$$\left|adv^{3.2}_9 - adv^{3.2}_{10}\right| \le 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}). \tag{49}$$

Finally, in $\mathsf{G}^{3.2}_{11,b}$, the test session key $\mathsf{sKey}[\mathsf{sID}^*]$ is independent of b. Hence,

$$adv^{3.2}_{11} = \left|\Pr\left[\mathsf{G}^{3.2}_{11,0} \Rightarrow 1\right] - \Pr\left[\mathsf{G}^{3.2}_{11,1} \Rightarrow 1\right]\right| = 0. \tag{50}$$

By (44), (45),(46),(48), (49), and (50), we have $adv^{3.2}_4 \le$

$$2 \cdot (\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}) + 2\ell\delta). \tag{51}$$

By (33), (34), (35), (36), (43), (51), we have

$$\begin{aligned} adv^3_1 \le 2\ell \cdot \Big(&\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}}) + \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + 2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) + \\ &2\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}) + \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + 4\ell\delta\Big) + 4(\delta + 2^{-\gamma}). \end{aligned} \tag{52}$$

By (1), (2), (3), (25), (32), (52), we have

$$\begin{aligned} \mathsf{Adv}^{\mathsf{IND\text{-}AA}}_{\mathsf{AKE},\mu,\ell,\mathcal{A}} = &2\mu^2\ell \cdot \Big((\ell+2) \cdot \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + (\ell+1) \cdot \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{B}_{\mathsf{TKEM}}) + \mathsf{Adv}^{\mathsf{tcr}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}) \\ &+\ell \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{B}_{\mathsf{KEM}}) + \mathsf{Adv}^{\mathsf{owf}}_{\mathsf{H}}(\mathcal{B}_{\mathsf{H}}) + (3\ell+2) \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{B}_{\mathsf{PRF}}) \\ &+\ell \cdot \mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}}) + (3\ell+3) \cdot \mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRG}}(\mathcal{B}_{\mathsf{PRG}}) + (4\ell^2+\ell+5) \cdot \delta + 2^{-\gamma+1}\Big). \end{aligned}$$

$\square$

Remark 2. If the building block PKE is replaced by a CCA-secure one, our AKE protocol can achieve unidirectional explicit authentication, where the initiator can authenticate the responder. Furthermore, m_{11} can be removed from AKE and the session key is $K := \mathsf{PRF}(m_{21}, M_1|M_2)$, instead of $K := \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m'_{21}, M_1|M_2)$. This change yields a smaller round-state size. The price is a slight increase in computation and communication complexity, due to the CCA-secure PKE. Let us first explain how unidirectional explicit authentication of responder is achieved by the message $M_2 = (\tilde{c}, C)$ with the help of CCA-secure PKE. If $\tilde{c}$ is an invalid ciphertext (from adversary), it either results in abort or leads to a different message m'_{22}, where $m'_{22} \leftarrow \mathsf{Dec}(sk_i, \mathsf{Dec}(\tilde{sk}, \tilde{c}))$. Consequently, $\mathsf{H}(m'_{22} \oplus C) \neq \sigma$ unless the one-wayness or TCR property of H is broken. Similarly, if C is modified by adversary, then $\mathsf{H}(m'_{22} \oplus C) \neq \sigma$ as well. In both cases, M_2 is rejected by the initiator. Next we explain why m_{11} can be removed when PKE is a CCA-secure one. In the security proof above, m_{11} only serves the proof of Case 1.2. In Case 1.2, the adversary does not corrupt j^*'s long-term secret key and does not get the ephemeral secret key $\tilde{sk}$ of PKE. Then Case 1.2 can take an analogous argument, just like how Case 1.1 makes use of the CCA security of KEM, to finish the proof without the help of m_{11}. By dropping m_{11}, the size of round state is shortened by at least λ bits.

5 Instantiations of Two-Message AKE

In this section, we will present instantiations of AKE in the standard model and the quantum random oracle model (QROM) respectively. To this end, we consider instantiations of the underlying building blocks of AKE.

In [1], Abe et al. presented a simple tranformation from any IND-CCA secure PKE with proper plaintext ciphertext to an IND-CCA secure TKEM. So we will seek IND-CCA secure PKE scheme instead of IND-CCA secure TKEM.

5.1 Instantiation of AKE in the Standard model

Here we show the instantiation of AKE from the LWE assumption.

- We take Peikeit's LWE-based PKE [16] as the underlying CCA secure PKE.
- We take Regev's LWE-based PKE [17] as the underlying CPA secure PKE.
- We take the LWE-based BPR-PRF [3] as the underlying PRF (PRG as well).
- We take the LMPR-Hash [15] as the underlying TCR hash function. Then the TCR security is based on the Short-Interger-Solution (SIS) assumption.

Note that when PKE is used as KEM, the plaintext is uniformly chosen as the encapsulation key and independent of the secret key and the public key. Therefore, without the knowledge of ciphertext, the plaintext is uniform to the adversary even if the adversary obtains the public/secret key pair. Consequently, the output pseudo-randomness of KEM holds naturally in this case.

Since the LWE assumption implies the SIS assumption, we immediately obtain an LWE-based two-message AKE in the standard model.

$\mathsf{G}^3_{2,b}$, $\mathsf{G}^3_{3,b}$, $\mathsf{G}^{3.2}_{4,b}$, $\mathsf{G}^{3.2}_{5,b}$, $\mathsf{G}^{3.2}_{6,b}$, $\mathsf{G}^{3.2}_{7,b}$, $\mathsf{G}^{3.2}_{8,b}$, $\mathsf{G}^{3.2}_{9,b}$, $\mathsf{G}^{3.2}_{10,b}$

$\mathsf{sID}^* \leftarrow_{\$} [\ell]$
$\mathsf{sID}' \leftarrow_{\$} [\ell]$
$(i^*, j^*) \leftarrow_{\$} [\mu] \times [\mu]$
for $i \in [\mu]$:
  $(pk_i, sk_i) \leftarrow \mathsf{AKE.Gen}$
$b' \leftarrow \mathcal{A}^{O(\cdot)}(pk_1, \ldots, pk_\mu)$
If $\mathsf{Trivial}(\mathsf{sID}^*)$: Return 0
If $\mathfrak{M}(\mathsf{sID}^*) = \emptyset$: Return 0 //Not Case 3
If $|\mathfrak{M}(\mathsf{sID}^*)| > 1$: Return 0
If $\mathfrak{M}(\mathsf{sID}^*) \neq \{\mathsf{sID}'\}$:Return 0
Return b'

<u>$\mathsf{EST}(i, j)$</u> :
$cnt := cnt + 1$, $\mathsf{sID} := cnt$
If $\mathsf{sID} = \mathsf{sID}^* \wedge (i, j) \neq (i^*, j^*)$: abort //Not Case 3
If $\mathsf{sID} = \mathsf{sID}' \wedge (i, j) \neq (j^*, i^*)$: abort
$(\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}]) := (i, j)$
$(stRev[\mathsf{sID}], rev[\mathsf{sID}]) := (\mathbf{false}, \mathbf{false})$
Return sID

<u>$\mathsf{StateReveal}(\mathsf{sID})$</u> :
If $\mathsf{sID} = \mathsf{sID}_I$: abort
$stRev[\mathsf{sID}] := \mathbf{true}$
Return $st[\mathsf{sID}]$

<u>$\mathsf{TEST}(\mathsf{sID})$</u> : //Only one query
If $\mathsf{sID} \neq \mathsf{sID}^*$: abort //Not Case 3
If $\mathsf{sKey}[sid^*] = \perp$: Return $\perp$
$K_0^* \leftarrow_{\$} \mathcal{K}, K_1^* := \mathsf{sKey}[\mathsf{sID}^*]$
Return K_b^*

<u>$\mathsf{Init}(\mathsf{sID})$</u> :
If $\mathsf{holder}[\mathsf{sID}] = \perp \vee \mathsf{sent}[\mathsf{sID}] \neq \perp$: Return $\perp$
$\mathsf{role}[\mathsf{sID}] = \mathsf{initiator}$
If $\mathsf{sID} = \mathsf{sID}^*$: $\mathsf{sID}_I := \mathsf{sID}^*, \mathsf{sID}_R := \mathsf{sID}'$
If $\mathsf{sID} = \mathsf{sID}'$: $\mathsf{sID}_I := \mathsf{sID}', \mathsf{sID}_R := \mathsf{sID}^*$
$(i, j) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
$(c_1, seed_i) \leftarrow \mathsf{TKEM.Encap}(pk_j, i)$
$m_{11} | m_{12} \leftarrow \mathsf{PRG}(seed_i)$
$\sigma := \mathsf{H}(m_{12})$
$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{PKE.Gen}$
$M_1 := (\tilde{pk}, c_1)$
$st[\mathsf{sID}] := (m_{11}, \tilde{sk}, \sigma, M_1)$; $\mathsf{sent}[\mathsf{sID}] := M_1$
Return M_1

<u>$\mathsf{Der}_{\mathsf{resp}}(\mathsf{sID}, M_1 = (\tilde{pk}, c_1))$</u> :
If $\mathsf{holder}[\mathsf{sID}] = \perp \vee \mathsf{role}[\mathsf{sID}] = \mathsf{initiator} \vee \mathsf{sKey}[\mathsf{sID}] \neq \perp$:
  Return $\perp$
$\mathsf{role}[\mathsf{sID}] = \mathsf{responder}$
$(j, i) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
If $\mathsf{TKEM.Decap}(sk_j, c_1, i) = \perp$: Return $\perp$
$seed_i' \leftarrow \mathsf{TKEM.Decap}(sk_j, c_1, i)$
$m_{11}' | m_{12}' \leftarrow \mathsf{PRG}(seed_i')$
$(c_2, seed_j) \leftarrow \mathsf{KEM.Encap}(pk_i)$
$\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, c_2)$
If $\mathsf{sID} = \mathsf{sID}_R$: $\tilde{c} \leftarrow \mathsf{PKE.Enc}(\tilde{pk}, 0)$
If $\mathsf{sID} = \mathsf{sID}_R$: $\mathsf{record}[\tilde{c}] := c_2$
If $\mathsf{sID} = \mathsf{sID}_R$: $seed_j \leftarrow_{\$} \mathcal{K}$
$m_{21} | m_{22} \leftarrow \mathsf{PRG}(seed_j)$
If $\mathsf{sID} = \mathsf{sID}_R$: $m_{21}, m_{22} \leftarrow_{\$} \mathcal{K} \times \mathcal{K}$
If $\mathsf{sID} = \mathsf{sID}_R$:
  $\mathsf{record}[c_2] := (seed_j, m_{21}, m_{22})$
$C := m_{12}' \oplus m_{22}$
$M_2 := (\tilde{c}, C)$
$K := \mathsf{PRF}(m_{11}', M_1 | M_2) \oplus \mathsf{PRF}(m_{21}, M_1 | M_2)$
$\mathsf{sKey}[\mathsf{sID}] := K$
If $\mathsf{sID} = \mathsf{sID}_R$: $\mathsf{sKey}[\mathsf{sID}] \leftarrow_{\$} \mathcal{K}$
$(\mathsf{recv}[\mathsf{sID}], \mathsf{sent}[\mathsf{sID}]) := (M_1, M_2)$
Return M_2

<u>$\mathsf{Der}_{\mathsf{init}}(\mathsf{sID}, M_2 = (\tilde{c}, C))$</u> :
If $st[\mathsf{sID}] = \perp \vee \mathsf{sKey}[\mathsf{sID}] \neq \perp$: Return $\perp$
$(i, j) := (\mathsf{holder}[\mathsf{sID}], \mathsf{peer}[\mathsf{sID}])$
Parse $st[\mathsf{sID}] = (m_{11}, \tilde{sk}, \sigma, M_1 = (\tilde{pk}, c_1))$
If $\mathsf{PKE.Dec}(\tilde{sk}, \tilde{c}) = \perp$:
  Return $\perp$
$c_2' \leftarrow \mathsf{PKE.Dec}(\tilde{sk}, \tilde{c})$
If $\mathsf{holder}[\mathsf{sID}] = I \wedge \mathsf{record}[\tilde{c}] \neq \emptyset$:
  $c_2' := \mathsf{record}[\tilde{c}]$
If $\mathsf{KEM.Decap}(sk_i, c_2') = \perp$:
  Return $\perp$
$seed_j' \leftarrow \mathsf{KEM.Decap}(sk_i, c_2')$
$m_{21}' | m_{22}' \leftarrow \mathsf{PRG}(seed_j')$
If $\mathsf{holder}[\mathsf{sID}] = I \wedge \mathsf{record}[c_2'] \neq \emptyset$:
  Parse $\mathsf{record}[c_2'] = (seed_j, m_{21}, m_{22})$
  $(seed_{j'}, m_{21}', m_{22}') := (seed_j, m_{21}, m_{22})$
$m_{12}' := C \oplus m_{22}'$
If $\mathsf{H}(m_{12}') \neq \sigma$:
  $\mathsf{sKey}[\mathsf{sID}] := \mathbf{reject}$
  Return $\perp$
$K := \mathsf{PRF}(m_{11}, M_1 | M_2) \oplus \mathsf{PRF}(m_{21}', M_1 | M_2)$
$\mathsf{sKey}[\mathsf{sID}] := K$
If $\mathsf{sID} = \mathsf{sID}_I$: $\mathsf{sKey}[\mathsf{sID}] := \mathsf{sKey}[\mathsf{sID}_R]$
$\mathsf{recv}[\mathsf{sID}] := M_2$

Fig. 9: Game $\mathsf{G}^3_{2,b}$ - $\mathsf{G}^3_{4,b}$, $\mathsf{G}^{3.2}_{5,b}$ - $\mathsf{G}^{3.2}_{10,b}$. Queries to $\mathcal{O}_{\mathsf{AKE}} := \{\mathsf{REVEAL}, \mathsf{CORRUPT}\}$ are defined as in the original game in Fig. 6.

In fact, there are many other choices for the building blocks, so our generic construction actually leads to many two-message AKE schemes from standard assumptions in the standard model.

5.2 AKE from CPA-secure PKE in the QROM

5.2.1 PRF and TCR. We simply take hash function as PRF (and PRG) and TCR.

- We take a hash function $H_1 : \mathcal{K} \times \mathcal{X} \to \mathcal{K}$ as a PRF.
- We take a hash function $H_2 : \mathcal{K} \to \Sigma$, where $\mathcal{K} = \Sigma \times \Sigma$ as a TCR.

The securities of PRF and TCR have already proved in QROM, as shown in Lemma 1 and Lemma 2.

Lemma 1 (PRF from QROM, Corollary 1 from [5]). *Let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a quantum-accessible random oracle. This function $\mathsf{PRF}(k, x) := H(k, x)$ may be used as a quantum-accessible PRF with a key $k \leftarrow_{\$} \mathcal{K}$. For any PRF-adversary $\mathcal{A}$ making at most q queries to H and any number of queries to F_k, its advantage satisfies $\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{PRF}}(\mathcal{A}) \le 2q/\sqrt{|\mathcal{K}|}$.*

Lemma 2 (TCR Hash from QROM, Theorem 3.1 from [21]). *There is a universal constant α such that the following holds. Let $H : \mathcal{K} \to \Sigma$ be a quantum-accessible random oracle. Then any algorithm making q quantum queries to H outputs a collision for H with probability at most $\alpha(q+1)^3/|\Sigma|$.*

5.2.2 KEM and TKEM from FO transformation in QROM. Lately, Don et al. [7] proved FO-transform with explicit rejection can be applied in QROM. Hence, an IND-CCA secure KEM can be constructed from IND-CPA secure PKE, via FO-transform. The constructed scheme $\mathsf{KEM}_{\mathsf{FO}}$ is shown in Fig. 10 and its security is given in Lemma 3.

$\mathsf{Encap}(pk\ ,\tau\ )$	$\mathsf{Decap}(sk, c\ ,\tau\ ):$
$m \leftarrow_{\$} \mathcal{M}$	$m' := \mathsf{Dec}(sk, c)$
$c := \mathsf{Enc}(pk, m; G(m\ \vert\tau\ ))$	If $m' = \perp$ or $\mathsf{Enc}(pk, m'; G(m'\ \vert\tau\ )) \neq c$:
$K := H(m\ \vert\tau\ )$	Return $\perp$
Return (c, K)	Else return $K := H(m'\ \vert\tau\ )$

Fig. 10: $\mathsf{KEM}_{\mathsf{FO}}$ from FO transformation (without gray box) and $\mathsf{TKEM}_{\mathsf{FO}}$ from FO transformation (with gray box).

Lemma 3 (IND-CCA Security of $\mathsf{KEM}_{\mathsf{FO}}$, Theorem 6.1 from [7]). *If* PKE *is a $(1-\delta)$-correct IND-CPA secure public key encryption scheme satisfying γ-spreadness and G,H are quantum-accessible random oracles, then the $\mathsf{KEM}_{\mathsf{FO}}$ in Fig. 10 is IND-CCA secure.*

Lemma 1 implies output pseudo-randomness of $\mathsf{KEM}_{\mathsf{FO}}$ as shown below.

Lemma 4 (Output Pseudo-Randomness of $\mathsf{KEM}_{\mathsf{FO}}$). *For any adversary $\mathcal{A}$ against output pseudo-randomness of $\mathsf{KEM}_{\mathsf{FO}}$, issuing at most q (quantum) queries to H, its advantage satisfies $\mathsf{Adv}^{\mathsf{ps}}_{\mathsf{KEM}}(\mathcal{A}) \leq 2q/\sqrt{|\mathcal{M}|}$.*

Proof. The output pseudo-randomness of $\mathsf{KEM}_{\mathsf{FO}}$ requires the two distributions $\{\ H(m) \mid m \leftarrow_{\$} \mathcal{M}\ \}$ and $\{\ K \mid K \leftarrow_{\$} \mathcal{K}\ \}$ are computational indistinguishable even if $\mathcal{A}$ makes at most q (quantum) queries to H. Lemma 1 already shows that H can be used as a PRF. Consequently, $H(m)$ is pseudo-random to $\mathcal{A}$ since m is randomly chosen. $\square$

Now we extend FO-transform to Tagged KEM in QROM. The construction of Tagged KEM is almost the same as $\mathsf{KEM}_{\mathsf{FO}}$. We just attach the tag τ to message m (m') as the input of G and H. Assume PKE is IND-CPA secure with γ-spreadness. The construction of $\mathsf{TKEM}_{\mathsf{FO}}$ from PKE is shown in Fig. 10.

In Lemma 5, we show that the IND-CCA security of $\mathsf{TKEM}_{\mathsf{FO}}$ can be reduced to IND-CPA security of PKE in QROM.

Lemma 5 (IND-CCA security of $\mathsf{TKEM}_{\mathsf{FO}}$). *If PKE is a $(1-\delta)$-correct IND-CPA secure public key encryption scheme satisfying γ-spreadness and G,H are quantum-accessible random oracles, then $\mathsf{TKEM}_{\mathsf{FO}}$ in Fig. 10 is IND-CCA secure.*

The intuition for the proof of Lemma 5 is as follows. Suppose that $G : \mathcal{M} \times \mathcal{T} \rightarrow \mathcal{K}$ is a quantum-accessible random oracle, then for each $\tau \in \mathcal{T}$, $G_\tau : \mathcal{M} \rightarrow \mathcal{K}$ defined by $G_\tau(m) := G(m, \tau)$ is also a quantum-accessible random oracle. Hence, the proof of Lemma 5 almost verbatim follows that of Lemma 3. We omit it here and put it in Appendix B.

5.2.3 The Final AKE in QROM. Given the above instantiations of PRG, PRF, TCR Hash, and KEM and TKEM constructed from CPA-secure PKE in QROM, we immediately obtain a generic construction of AKE from CPA-secure PKE in QROM. For further optimization, we replace the computation of session key $K := \mathsf{PRF}(m_{11}, M_1|M_2) \oplus \mathsf{PRF}(m_{21}, M_1|M_2)$ with hash function $K := H(m_{11}|m_{21}|M_1|M_2)$. With the following quantum-accessible random oracles, we obtain the final construction of our AKE protocol in Fig. 11.

- $G : \mathcal{K} \times \mathcal{T} \rightarrow \mathcal{R}$, which is used to generate randomness in PKE.
- $\mathsf{H} : \mathcal{K} \rightarrow \Sigma$, which is used as a target collision resistant hash function. Here $\mathcal{K} = \Sigma \times \Sigma$,
- $H_1 : \mathcal{K} \times \mathcal{T} \rightarrow \mathcal{K}$, which is used to generate encapsulation key.
- $H_2 : \mathcal{K} \times \{0,1\} \rightarrow \mathcal{K}$, which is used as a pseudo-random generator.
- $H : \{0,1\}^* \rightarrow \mathcal{K}$, which is used to generate session key.

Acknowledgements. Shengli Liu and You Lyu were partially sponsored by Guangdong Major Project of Basic and Applied Basic Research under Grant No. 2019B030302008, National Natural Science Foundation of China under Grant No. 61925207 and the National Key R&D Program of China under Grant No. 2022YFB2701500.

$\underline{\mathsf{Init}(sk_i, pk_j):}$
$m_1 \leftarrow_\$ \mathcal{K}$
$c_1 \leftarrow \mathsf{Enc}(pk_j, m_1; G(m_1|i))$
$seed_i \leftarrow H_1(m_1|i)$
$m_{11} \leftarrow H_2(seed_i|0)$
$m_{12} \leftarrow H_2(seed_i|1)$
$\sigma := \mathsf{H}(m_{12})$
$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{PKE.Gen}$
$M_1 := (\tilde{pk}, c_1)$
$st := (m_{11}, \tilde{sk}, \sigma, M_1)$
Return (M_1, st)

$\underline{\mathsf{Der}_{\mathsf{resp}}(sk_j, pk_i, M_1):}$
Parse $M_1 = (\tilde{pk}, c_1)$
$m_1' \leftarrow \mathsf{Dec}(sk_j, c_1)$
If $m_1' = \perp \vee \mathsf{Enc}(pk_j, m_1'; G(m_1'|i)) \neq c_1$:
  Return $\perp$
else:
  $seed_i' := H_1(m_1'|i)$
$m_{11}' \leftarrow H_2(seed_i'|0); m_{12}' \leftarrow H_2(seed_i'|1)$
$m_2 \leftarrow_\$ \mathcal{K}$
$c_2 \leftarrow \mathsf{Enc}(pk_i, m_2; G(m_2))$
$seed_j \leftarrow H_1(m_2)$
$m_{21} \leftarrow H_2(seed_j|0); m_{22} \leftarrow H_2(seed_j|1)$
$\tilde{c} \leftarrow \mathsf{Enc}(\tilde{pk}, c_2)$
$C := m_{12}' \oplus m_{22}$
$M_2 := (\tilde{c}, C)$
$K := H(m_{11}'|m_{21}|M_1|M_2)$
Return (M_2, K)

$\underline{\mathsf{Der}_{\mathsf{init}}(sk_i, pk_j, M_2, st):}$
Parse $M_2 = (\tilde{c}, C)$
Parse $st = (m_{11}, \tilde{sk}, \sigma, M_1 = (\tilde{pk}, c_1))$
If $\mathsf{Dec}(\tilde{sk}, \tilde{c}) = \perp$:
  Return $\perp$
$c_2' \leftarrow \mathsf{Dec}(\tilde{sk}, \tilde{c})$
$m_2' \leftarrow \mathsf{Dec}(sk_i, c_2')$
If $\mathsf{Enc}(pk_i, m_2'; G(m_2')) \neq c_2'$:
  Return $\perp$
else:
  $seed_j' := H_1(m_2')$
$m_{21}' \leftarrow H_2(seed_j'|0), m_{22}' \leftarrow H_2(seed_j'|1)$
If $\mathsf{H}(C \oplus m_{22}') \neq \sigma$:
  Return $\perp$
$K := H(m_{11}|m_{21}'|M_1|M_2)$
Return K

Fig. 11: Generic construction of AKE from CPA-secure PKE in QROM.

References

1. Abe, M., Gennaro, R., Kurosawa, K.: Tag-kem/dem: A new framework for hybrid encryption. J. Cryptol. **21**(1), 97–130 (2008), https://doi.org/10.1007/s00145-007-9010-x
2. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: CRYPTO 2019. https://doi.org/10.1007/978-3-030-26951-7_10
3. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: EUROCRYPT 2012. https://doi.org/10.1007/978-3-642-29011-4_42
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO 1993. https://doi.org/10.1007/3-540-48329-2_21
5. Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: TCC 2019. https://doi.org/10.1007/978-3-030-36033-7_3
6. Boyd, C., Cliff, Y., Nieto, J.M.G., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: ACISP2008. https://doi.org/10.1007/978-3-540-70500-0_6
7. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: EUROCRYPT 2022. https://doi.org/10.1007/978-3-031-07082-2_24
8. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: PKC 2012. https://doi.org/10.1007/978-3-642-30057-8_28
9. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: CRYPTO 2021. https://doi.org/10.1007/978-3-030-84259-8_23
10. Hashimoto, K., Katsumata, S., Kwiatkowski, K., Prest, T.: An efficient and generic construction for signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. In: PKC 2021. https://doi.org/10.1007/978-3-030-75248-4_15
11. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: PKC 2020. https://doi.org/10.1007/978-3-030-45388-6_14

12. Huguenin-Dumittan, L., Vaudenay, S.: On ind-qcca security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In: EUROCRYPT 2022. https://doi.org/10.1007/978-3-031-07082-2_22
13. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: EUROCRYPT 2021. https://doi.org/10.1007/978-3-030-77870-5_5
14. Krawczyk, H.: HMQV: A high-performance secure diffie-hellman protocol. In: CRYPTO 2005. https://doi.org/10.1007/11535218_33
15. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: FSE 2008. https://doi.org/10.1007/978-3-540-71039-4_4
16. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: STOC 2009. https://doi.org/10.1145/1536414.1536461
17. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005. https://doi.org/10.1145/1060590.1060603
18. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: CCS 2020. https://doi.org/10.1145/3372297.3423350
19. Unruh, D.: Revocable quantum timed-release encryption. In: EUROCRYPT 2014. https://doi.org/10.1007/978-3-642-55220-5_8
20. Xue, H., Lu, X., Li, B., Liang, B., He, J.: Understanding and constructing AKE via double-key key encapsulation mechanism. In: ASIACRYPT 2018. https://doi.org/10.1007/978-3-030-03329-3_6
21. Zhandry, M.: A note on the quantum collision and set equality problems. Quantum Inf. Comput. **15**(7&8), 557–567 (2015), https://doi.org/10.26421/QIC15.7-8-2

Appendix

A Useful Lemmas for Quantum Random Oracles

A.1 Extractable Quantum Random Oracle Simulation

We first recall some definitions and main theorem in [7].

Definition 15. *Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{C}$ be an arbitrary fixed function with $\mathcal{Y} = \{0,1\}^n$, we define*

$$\Gamma(f) := \max_{x,c} |\{y | f(x,y) = c\}|.$$

Definition 16. *Let $R \subseteq \mathcal{X} \times \{0,1\}^n$ be a relation. We define*

$$\Gamma(R) := \max_{x\in\mathcal{X}} |\{y \in \{0,1\}^n | (x,y) \in R\}|.$$

Next we recall the Theorem 4.3 and Proposition 4.4 in Lemma 6 (we only list the entries which will be used in the following proof.)

Lemma 6 (Theorem 4.3 in [7]). *For a fixed function $f : \mathcal{X} \times \{0,1\}^n \to \mathcal{C}$, there is an efficient simualor $\mathcal{S}$ that has two interfaces $\mathcal{S}.RO : \mathcal{X} \to \{0,1\}^n$ and $\mathcal{S}.E : \mathcal{C} \to \mathcal{X} \cup \{\perp\}$ and has the following properties:*

(1) If $\mathcal{S}.E$ is unused, $\mathcal{S}$ is perfectly indistinguishable from the random oracle RO.
(2.a) Any two subsequent independent queries to $\mathcal{S}.RO$ commute. We refer two subsequent queries as being independent if the input to one query does not depend on the output of the other.
(2.b) Any two subsequent independent queries to $\mathcal{S}.E$ commute.
(2.c) Any two subsequent independent queries to $\mathcal{S}.E$ and $\mathcal{S}.RO$ $8\sqrt{2\Gamma(f)/2^n}$-almost-commute.
(4.b) If $h = \mathcal{S}.RO(x)$ and $\hat{x} = \mathcal{S}.E(f(x,h))$ are two subsequent classical queries such that no prior query to $S.E$ has been made, then

$$\Pr[\hat{x} = \perp] \leq 2 \cdot 2^{-n}.$$

(4.c) (Proposition 4.4 in [7].) Let $R' \subseteq \mathcal{X} \times \mathcal{C}$ be a relation. Consider a query algorithm $\mathcal{A}$ that makes q queries to the $\mathcal{S}.RO$ interface of $\mathcal{S}$ but no query to $\mathcal{S}.E$, outputting some $(c_1,\ldots,c_\ell) \in \mathcal{C}^\ell$. For each i, let $\hat{x}_i$ then be obtained by making an additional query to $\mathcal{S}.E$ on input t_i. Then

$$\Pr_{\substack{(c_1,\ldots,c_\ell)\leftarrow\mathcal{A}^{\mathcal{S}.RO}\\ \hat{x}_i \leftarrow \mathcal{S}.E(c_i)}} \left[\exists i : (\hat{x}_i, c_i) \in R'\right] \leq 128 \cdot q^2 \Gamma(R)/2^n,$$

where $R \subseteq \mathcal{X} \times \mathcal{Y}$ is the relation $(x,y) \in R \Leftrightarrow (x, f(x,y)) \in R'$.

A.2 O2H Lemma

Now we recall the well-know O2H lemma proposed in [19]. We will use its general version which is Theorem 3 in [2].

Lemma 7. *Let $S \subseteq X$ be random. Let $G, H : X \to Y$ be random functions satisfying $\forall x \notin S, G(x) = H(x)$. Let z be a random bitstring. (S, G, H, z may have arbitrary joint distribution.)*

Let A be quantum oracle algorithm with query number q.

Let B^H be an oracle algorithm that on input z does the following: pick $i \leftarrow_\$ \{1, ..., q\}$, run $A^H(z)$ until (just before) the i-th query, measure all query input registers in the computational basis, output the set T of measurement outcome.

Then,

$$\left|\Pr\left[b = 1 : b \leftarrow A^H(z)\right] - \Pr\left[b = 1 : b \leftarrow A^G(z)\right]\right| \leq 2q\sqrt{\Pr\left[S \cap T \neq \emptyset : T \leftarrow B^H(z)\right]}.$$

B FO transformation of TKEM

This part follows the proof of CCA-security of KEM from FO-transformation in [7]. The sequence of hybrid games and proof are almost the same as proof in [7]. We will show them for completeness.

The construction of TKEM is shown in Fig 10. We will prove the following theorem.

Theorem 2. *Let* PKE *be a $(1-\delta)$-correct public key encryption scheme satisfying γ-spreadness. Let $\mathcal{A}$ be any IND-CCA adversary against TKEM, making $q_D \geq 1$ queries to the decapsulation oracle* Decap *and q_G and q_H (quantum) queries to $G : \mathcal{M} \times \mathcal{T} \to \mathcal{R}$ and $H : \mathcal{M} \times \mathcal{T} \to \mathcal{K}$, respectively, where G and H are modeled as random oracles. Let $q := q_G + q_H + 2q_D$. Then, there exists a IND-CPA adversary $\mathcal{B}_{\mathsf{PKE}}$ against* PKE *with*

$$\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}) \leq 2q\sqrt{\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{\mathsf{PKE}})} + 24q^2(\sqrt{\delta} + 2^{-\gamma/4}).$$

Proof. We first analyze the sequence of hybrids for a fixed key pair (pk, sk). Let δ_{sk} be the maximum probability of a decryption error and g_{sk} be the maximum probability of any ciphertext, so that $E[\delta_{sk}] \leq \delta$ and $E[g_{sk}] \leq 2^{-\gamma}$ with the expectation over $(pk, sk) \leftarrow \mathsf{Gen}$.

Game 0. In Game 0, we first sample a random oracle F and define $G(m|\tau) := F(0|m|\tau)$ and $H(m|\tau) := F(1|m|\tau)$. Now we consider both challenger and adversary access this single random oracle F. When convenient, we sometimes refer to $F(0|\cdot)$ as G and $F(1|\cdot)$ as H. These changes do not affect the adversary's view or the game's outcome. Game 0 is still the IND-CCA game for TKEM with fixed key pair (pk, sk). Therefore,

$$\Pr\left[b = b' \text{ in } \textbf{Game 0}\right] = \frac{1}{2} + \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}).$$

Game 1. In Game 1, we introduce a new oracle F'. Let m^* denote the message encrypted in challenge ciphertext c^* and τ^* denote the tag submitted by adversary $\mathcal{A}_{sk}$. We define $F'(0|m^*|\tau^*) := r'$ and $F'(1|m^*|\tau^*) := k'$ for uniformly random $r' \in \mathcal{R}$ and $k' \in \mathcal{K}$, while letting $F'(b|m|\tau) = F(b|m|\tau)$ for $(m, \tau) \neq (m^*, \tau^*)$ and $b \in \{0, 1\}$. Note that F' is still a purely random function. We also define $G' := F'(0|\cdot)$ and $H' := F'(1|\cdot)$.

Game 1 is the same as Game 0, except for the following changes. After (m^*, τ^*) have been produced and before $\mathcal{A}$ can access to Decap oracle, challenger $\mathcal{C}$ first queries $r' = G'(m^*|\tau^*)$ then computes $c' := \mathsf{Enc}(pk, m^*; r')$. Furthermore, Decap oracle changes as follows.

- $\mathsf{Decap}(sk, c, \tau)$: If $(c, \tau) = (c', \tau^*)$, it will compute the decrypt result using random oracle G and H. Otherwise, it will use random oracle G' and H'.

We claim that

$$\Pr[b = b' \text{ in } \mathbf{Game\ 1}] = \Pr[b = b' \text{ in } \mathbf{Game\ 0}] = \frac{1}{2} + \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}_{sk}).$$

For any decryption query (c_i, τ_i), let $m_i \leftarrow \mathsf{Dec}(sk, c_i)$. Then there are three cases:

1. $m_i \neq m^*$ 2. $m_i = m^*$ but $\tau_i \neq \tau^*$ 3. $m_i = m^*$ and $\tau_i = \tau^*$

In case 1 and case 2, we have $F'(b|m_i|\tau_i) = F(b|m_i|\tau_i)$. In case 3, we either have $c_i = c'$, where nothing changes by definition of the game, or else $\mathsf{Enc}(pk, m^*; G(m^*|\tau^*)) = c^* \neq c_i$ and $\mathsf{Enc}(pk, m^*; G'(m^*|\tau^*)) = c' \neq c_i$, and hence the re-encryption check fails and $K_i = \perp$ in both Game 0 and Game 1. Therefore, the view of $\mathcal{A}$ in Game 0 and Game 1 is the same.

Game 2. Game 2 is identical to Game 1, except that $\mathcal{C}$ uses F' to reply all Decap calls (also for $(c_i, \tau_i) = (c', \tau^*)$) and all random oracle calls made by $\mathcal{A}$. The challenge ciphertext $c^* = \mathsf{Enc}(pk, m^*; G(m^*|\tau^*))$ and key $K_0 = H(m^*|\tau^*)$ are still computed using F. Note that in Game 2, $K_0 = H(m^*|\tau^*)$ is independent of m^*, τ^* and F', exactly as K_1 is, which means that $\mathcal{A}_{sk}$ can only win with probability $\frac{1}{2}$.

Hence, we have

$$|\Pr[b = b' \text{ in } \mathbf{Game\ 1}] - \Pr[b = b' \text{ in } \mathbf{Game\ 2}]| = \mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}_{sk}).$$

By the Lemma 7 (O2H Lemma), we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}_{sk}) &= |\Pr[b = b' \text{ in } \mathbf{Game\ 1}] - \Pr[b = b' \text{ in } \mathbf{Game\ 2}]| \\ &\leq 2(q_G + q_H + 2)\sqrt{\Pr[(m', \tau') = (m^*, \tau^*) \text{ in } \mathbf{Game\ 3}]},\end{aligned} \tag{53}$$

where **Game 3** is introduced by the O2H lemma to extract an input where F and F' differs. There are total $q_G + q_H$ quantum queries to G' and H', and $2q_D$ classical queries incurred by Decap oracle. Note that Game 1 and Game 2 have the same behavior of $\mathsf{Decap}(c, \tau)$ except $\mathcal{A}$ queries $\mathsf{Decap}(c', \tau^*)$. Hence, $\mathcal{B}$ can only consider the $(q_G + q_H)$ quantum queries plus two classical queries incurred by $\mathsf{Decap}(c', \tau^*)$. $\mathcal{B}$ first randomly chooses these q_G+q_H+2 queries and measures the input of j-th query to F and obtains (m, τ). Then $\mathcal{B}$ sets $(m', \tau') := (m, \tau)$ and returns whether $(m', \tau') = (m^*, \tau^*)$. Besides, rather than measuring Decap's classical query to G' and H' upon decryption query $(c_i, \tau_i) = (c', \tau^*)$, we can equivalently set $m' := m_i = \mathsf{Dec}(sk, c')$ and $\tau' := \tau^*$.

Since we are concerned with the measurement outcome (m', τ') only, it is irrelevant whether the game stops right after the measurement, or it continues until $\mathcal{A}$ outputs b'.

Game 4. Game 4 is the same as Game 3 except for the behavior of random oracle G'. In Game 4, we replace G' with the extractable RO-simulator $\mathcal{S}$ from Lemma 6 for the fixed function $f : (\mathcal{M}\times\mathcal{T})\times\mathcal{R} \to \mathcal{C}, ((m, \tau), r) \mapsto \mathsf{Enc}(pk, m; r)$. Furthermore, at the very end of the game, we invoke the extractor interface $\mathcal{S}.E$ to compute $(\hat{m}_i, \hat{\tau}_i) \leftarrow \mathcal{S}.E(c_i)$ for each (c_i, τ_i) that $\mathcal{A}$ queried to Decap. By (1) in Lemma 6, given that the $\mathcal{S}.E$ queries take place only after the run of $\mathcal{A}$, we have

$$\Pr\left[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 4}\right] = \Pr\left[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 3}\right]. \quad (54)$$

Furthermore, consider (4.c) in Lemma 6 for relation $R' := \{((m, \tau), c) : \mathsf{Dec}(sk, c) \neq m\}$, then the event

$$P^\dagger := [\forall i : \hat{m}_i = m_i \vee \hat{m}_i = \perp]$$

holds except with probability $\epsilon_1 := 128(q_G + q_D)^2 \Gamma(R)/|\mathcal{R}|$ for Γ_R, which here means that $\Gamma(R)/|\mathcal{R}| = \delta_{sk}$. Thus,

$$\Pr\left[(m', \tau') = (m^*, \tau^*) \wedge P^\dagger \text{ in } \textbf{Game 4}\right] \geq \Pr\left[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 4}\right] - \epsilon_1. \quad (55)$$

Game 5. Game 5 is the same as Game 4, except for the behavior of oracle $\mathsf{Decap}(c, \tau)$.

- $\mathsf{Decap}(c, \tau)$: It first computes $m \leftarrow \mathsf{Dec}(sk, c)$ and queries $g := \mathcal{S}.RO(m|\tau)$. Then, it queries $\hat{m}|\hat{\tau} \leftarrow \mathcal{S}.E(c)$ immediately instead of querying $S.E(c)$ after the game simulation ends.

Since the result of $\hat{m}$ is never used in Game 5, any queries to $\mathcal{S}.E(c)$ are independent of queries to $\mathcal{S}.E(c)$ and $\mathcal{S}.RO(m|\tau)$. By (2.b) and (2.c) of Theorem 6, each swap of a $\mathcal{S}.RO$ with a $\mathcal{S}.E$ query affects the final probability by at most $8\sqrt{2\Gamma(f)/|\mathcal{R}|} = 8\sqrt{2g_{sk}}$. Since there are q_D queries of $\mathcal{S}.E$ and $(q_D + q_G)$ queries of $\mathcal{S}.RO$, we have Game 4 and Game 5 are the same except with probability $8q_D(q_D + q_G)\sqrt{2g_{sk}}$.

Suppose we just swap the latest $\mathcal{S}.E(c_{q_D})$ query to oracle Decap, i.e., no query to $\mathcal{S}.E$ before $\mathcal{S}.E(c_{q_D})$. Let $(\hat{m}_{q_D}, \hat{\tau}_{q_D}) \leftarrow \mathcal{S}.E(c_{q_D})$. By (4.b) of Theorem 6, $\hat{m}_{q_D} = \perp$ implies $\mathsf{Enc}(pk, m_{q_D}; \mathcal{S}.RO(m_{q_D}|\tau_{q_D})) \neq c_{q_D})$ expect with probability $2 \cdot 2^{-n}$. Repeating the above step and applying the union bound, we find that event $P^\dagger$ implies

$$P := [\forall i : \hat{m}_i = m_i \vee (\hat{m}_i = \perp \wedge \mathsf{Enc}(pk, m_i; \mathcal{S}.RO(m_i|\tau_i)) \neq c_i)]$$

except with probability $q_D \cdot 2 \cdot 2^{-n}$. Thus,

$$\Pr\left[(m', \tau') = (m^*, \tau^*) \wedge P \text{ in } \textbf{Game 5}\right] \geq \Pr\left[(m', \tau') = (m^*, \tau^*) \wedge P^\dagger \text{ in } \textbf{Game 4}\right] - \epsilon_2, \quad (56)$$

with $\epsilon_2 := 2q_D \cdot \big((q_G + q_D) \cdot 4\sqrt{2g_{sk}} + 2^{-n}\big)$.

Game 6. Game 6 is the same as Game 5, except for the behavior of oracle $\mathsf{Decap}(c, \tau)$.

- $\mathsf{Decap}(c, \tau)$: It first computes $m \leftarrow \mathsf{Dec}(sk, c)$ and queries $g := \mathcal{S}.RO(m|\tau)$, $\hat{m}|\hat{\tau} \leftarrow \mathcal{S}.E(c)$. Then, it returns $K := \perp$ if $\hat{m} = \perp$ and returns $K := H'(\hat{m}|\tau)$ if $\hat{m} \neq \perp$, rather than returning $\perp$ if $\mathsf{Enc}(pk, m; g) \neq c$ and returning $K := H'(m|\tau)$ if $\mathsf{Enc}(pk, m; g) = c$ as did in Game 5.

Here, we note that if the event
$$P_i = [\hat{m}_i = m_i \vee (\hat{m}_i = \perp \wedge \mathsf{Enc}(pk, m_i; \mathcal{S}.RO(m_i|\tau_i)) \neq c_i)]$$
holds for a given i, then the above change will not affect Decap's response K_i. Therefore, we have

$$\Pr\big[(m', \tau') = (m^*, \tau^*) \wedge P \text{ in } \textbf{Game 6}\big] = \Pr\big[(m', \tau') = (m^*, \tau^*) \wedge P \text{ in } \textbf{Game 5}\big]. \tag{57}$$

Game 7. Game 7 is the same as Game 6 except for the method of obtaining m'. In Game 7, m' is obtained by the following two ways:

1. With probability $(q_G + q_H)/(q_G + q_H + 2q_D)$, m' is obtained by measuring a random query of $\mathcal{A}$ to either $\mathcal{S}.RO$ or H'.
2. With probability $2q_D/(q_G + q_H + 2q_D)$, m' is obtained by set $m' := \hat{m}_{i'}$, where $i' \leftarrow_\$ [q_D]$.

Recall that in Game 6, m' is obtained by measuring a random query of $\mathcal{A}$ to either $\mathcal{S}.RO$ or H' with probability $(q_G + q_H)/(q_G + q_H + 2)$, or by outputting $\hat{m}_i$ with $(c_i, \tau_i) = (c', \tau^*)$ with probability $2/(q_G + q_H + 2)$. Since conditioned on the first case being chosen or the latter with $i = i'$, Game 7 coincides with Game 6, we have

$$\Pr\big[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 7}\big] \geq \frac{q_G + q_H + 2}{q_G + q_H + 2q_D} \cdot \Pr\big[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 6}\big]. \tag{58}$$

Note that after Game 7, $\mathcal{B}$ does not need to compute $c' := \mathsf{Enc}(pk, m^*; G'(m^*|\tau^*))$. In other words, $\mathcal{B}$ does not need to use m^* in the game.

Game 8. In Game 8, we drop out all the $\mathcal{S}.RO(m_i|\tau_i)$ queries from Decap oracle, or equivalently, move them to the end of the execution of the game. Invoking once again (2.c) of Theorem 6, we then get

$$\Pr\big[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 8}\big] \geq \Pr\big[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 7}\big] - \epsilon_3, \tag{59}$$

for $\epsilon_3 = (q_D + 1) \cdot q_G \cdot 8\sqrt{2g_{sk}}$.

Note that in Game 8, $\mathcal{B}$ can simulate Decap oracle without knowledge of the secret key sk, and thus we can construct a OW-CPA (IND-CPA) attacker $\mathcal{B}_{sk}$ against PKE, which takes as input a public key pk and an encryption c^* of a random message $m^* \in \mathcal{M}$, and outputs m^* with the given probability, i.e.,

$$\Pr\left[(m', \tau') = (m^*, \tau^*) \text{ in } \textbf{Game 8}\right] \leq \mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}). \tag{60}$$

By (53), (54), (55), (56), (57), (58), (59), (60) and setting $q := q_H + q_G + 2q_D$, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{TKEM}}(\mathcal{A}_{sk}) &\le 2(q_G + q_H + 2)\sqrt{\frac{q}{q_H + q_D + 2}(\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{sk}) + \epsilon_3) + \epsilon_1 + \epsilon_2} \\
&\le 2q\sqrt{\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{sk}) + \epsilon_2 + \epsilon_3} + 2q\sqrt{\epsilon_1} \\
&\le 2q(\sqrt{\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{B}_{sk})} + \sqrt{\epsilon_2 + \epsilon_3}) + 2q\sqrt{\epsilon_1}, \\
\sqrt{\epsilon_2 + \epsilon_3} &= \sqrt{2q_D \cdot 4(q_G + q_D + q_G + 1/q_D)\sqrt{2g_{sk}} + 2^{-n}} \\
&\le 6\sqrt{q_G q_D}(g_{sk}^{1/4} + 2^{-n/2}) \le 12q \cdot g_{sk}^{1/4}.
\end{aligned}$$

Finally, taking the expectation over $(pk, sk) \leftarrow \mathsf{Gen}$ and applying Jensen's inequality, we prove that the construction of Fig 10 is a secure IND-CCA TKEM. $\square$

Table of Contents