Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices*

Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, Kazuki Yoneyama

NTT Secure Platform Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan
yoneyama.kazuki@lab.ntt.co.jp

Abstract. An unresolved problem in research on authenticated key exchange (AKE) is to construct a secure protocol against advanced attacks such as key compromise impersonation and maximal exposure attacks without relying on random oracles. HMQV, a state of the art AKE protocol, achieves both efficiency and the strong security proposed by Krawczyk (we call it the CK^+ model), which includes resistance to advanced attacks. However, the security proof is given under the random oracle model. We propose a generic construction of AKE from a key encapsulation mechanism (KEM). The construction is based on a chosen-ciphertext secure KEM, and the resultant AKE protocol is CK^+ secure in the standard model. The construction gives the first CK^+ secure AKE protocols based on the hardness of integer factorization problem, code-based problems, or learning problems with errors. In addition, instantiations under the Diffie-Hellman assumption or its variant can be proved to have strong security without non-standard assumptions such as πPRF and KEA1. Furthermore, we extend the CK^+ model to identity-based (called the id-CK^+ model), and propose a generic construction of identity-based AKE (ID-AKE) based on identity-based KEM, which satisfies id-CK^+ security. The construction leads first strongly secure ID-AKE protocols under the hardness of integer factorization problem, or learning problems with errors.

Keywords: authenticated key exchange, CK^+ model, key encapsulation mechanism, identity-based authenticated key exchange

1 Introduction

1.1 Background

Establishing secure channels is one of the most important areas of cryptographic research. Secure channels provide secrecy and authenticity for both communication parties. When parties can share secret information via a public communication channel, secure channels would be constructed on (symmetric key) encryptions and message authentication codes with the shared secret information called session keys. Public-key cryptography can provide various solutions: one approach uses a *key encapsulation mechanism* (KEM) and another uses *authenticated key exchange* (AKE).

In KEM, a receiver has public information, called a *public key*, and the corresponding secret information, called a *secret key*. The public key is expected to be certified with the receiver's identity through an infrastructure such as a *public key infrastructure* (PKI). A sender who wants to share information, a *session key*, with the receiver sends a *ciphertext* of the information and, the receiver decrypts the ciphertext to extract the information. KEM can be easily constructed from *public-key encryption* (PKE) under the reasonable condition that the plaintext space is sufficiently large. The desirable security notion of KEM is formulated as the *indistinguishability against chosen ciphertext attacks* (IND-CCA).

In AKE, each party has public information, called a *static public key*, and the corresponding secret information, called a *static secret key*. The static public key is also expected to be

* An extended abstract of this paper appeared in PKC 2012 [FSXY12].

certified with a party's identity through an infrastructure such as PKI. A party who wants to share information with a party exchanges *ephemeral public keys*, generated from the corresponding *ephemeral secret keys*, and computes a *session state* from their static public keys, the corresponding static secret keys, the exchanged ephemeral public keys, and the corresponding ephemeral secret keys. Both parties then derive a *session key* from these values including the session state using a *key derivation procedure*. Many studies have investigated the security notion of AKE [BR93,CK01,Kra05,LLM07,SEVB10]. The first security notion of AKE based on indistinguishability was provided by Bellare and Rogaway [BR93] (BR model). The BR model captures basic security requirements for AKE such as known key security and impersonation resilience. However, the BR model cannot grasp more complicated situations where a static secret key or session state of a party has been exposed. Accordingly, Canetti and Krawczyk [CK01] defined the first security notion of AKE capturing the exposure of static secret keys and session state and called it the *Canetti-Krawczyk (CK) model*. Though the CK model represents exposure of information other than the target session of the adversary, some advanced attacks such as key compromise impersonation (KCI), the breaking of weak perfect forward secrecy (wPFS) and maximal exposure attacks (MEX) use secret information of the target session; thus, the CK model is not resilient to such attacks. KCI means that when given a static secret key, an adversary will try to impersonate some honest party in order to fool the owner of the exposed secret key. wPFS implies that an adversary cannot recover a session key if the adversary does not modify messages of the target session and the session is executed before the static secret keys are compromised. In MEX, an adversary tries to distinguish the session key from a random value under the disclosure of any pair of secret static keys and ephemeral secret keys of the initiator and the responder in the session except for both the static and ephemeral secret keys of the initiator or the responder. Resistance to MEX requires security against any exposure situation that was not presumed. For example, an implementer of AKE may pretend to generate secret keys in an insecure host machine in order to prevent the randomness generation mechanisms in a tamper-proof module such as a smart card. Additionally, if a pseudo-random number generator implemented in a system is poor, secret keys will be known to the adversary even when the generation of ephemeral secret keys is operated in a tamper-proof module. Most AKE protocols are proved in the CK model; however, it is unclear whether such protocols satisfy resistance to advanced attacks due to the limitations of the CK model. A state of the art AKE protocol HMQV [Kra05] satisfies all known security requirements for AKE, including resistance to KCI, wPFS[1], and MEX, as well as provable security in the CK model. In this paper, we call this security model the CK^+ model; it is known to be one of the 'strongest' models for AKE. LaMacchia et al. [LLM07] and Sarr et al. [SEVB10] also proposed very strong security models for AKE by re-formulating the concept of the CK^+ model; they called them the eCK model and the seCK model, respectively. These models allow an adversary to pose a query that directly reveals the ephemeral secret key of the target session. However, Cremers points out that the CK model and the eCK model are incomparable [Cre09,Cre11]; thus, the eCK model is not stronger than the CK model while the CK^+ model is. We will briefly show the difference between the

[1] HMQV does not provide full perfect forward secrecy (fPFS), which is the same as wPFS except that the adversary can modify messages of the target session. Some schemes [JKL04,GKR10,CF11,BGN11,Yon12,CF12] have achieved fPFS. However, the schemes [JKL04,GKR10] are clearly vulnerable to MEX; that is, the session key is computable if an adversary obtains an ephemeral secret key of parties in the target session. The schemes [CF11,BGN11,CF12] is resilient to MEX, but security is proved in the random oracle model. The other scheme [Yon12] limits instantiations to DH-based. Upgrading wPFS to fPFS is not that difficult; it can be done by simply adding MAC or a signature of ephemeral public keys. Thus, we do not discuss fPFS in this paper.

CK^+ model and these models. Since MEX includes any non-trivial exposure situation, HMQV (and CK^+ secure protocols) achieves surprisingly strong security.

1.2 Motivating Problem

HMQV is one of the most efficient protocols and satisfies one of the strongest security models (i.e., CK^+ security). However, the security proof is given in the random oracle model (ROM) under a specific number-theoretic assumption (Diffie-Hellman (DH) assumption). Moreover, to prove resistance to MEX, the knowledge-of-exponent assumption (KEA1) [Dam91] (a widely criticized assumption such as [Nao03]) is also necessary. Hence, one of the open problems in research on AKE is to construct a secure scheme in the CK^+ model without relying on random oracles under standard assumptions.

Boyd et al. [BCGNP08,BCGNP09,GBGNM09] gave a partial solution to this problem by noting that KEM and AKE are closely related and that it might be natural to construct AKE from KEM. They proposed a generic construction of AKE from KEM (BCGNP construction), and its security is proved in the CK model in the standard model (StdM). Also, the BCGNP construction is shown to satisfy resistance to KCI. However, it is unclear whether the BCGNP construction is secure when exposure of secret information occurs (i.e., resistance to MEX). In fact, the BCGNP construction fails to satisfy CK^+ security when we consider the following attack scenario: Two parties exchange ciphertexts of an IND-CCA secure KEM scheme and generate a session key from these. An adversary who obtains the ephemeral secret keys (randomness used in generating ciphertexts) of the parties can compute the session key and win the game. Though the BCGNP construction can be extended to satisfy wPFS, it is guaranteed under the DH assumption, not a general assumption. It is quite restrictive because it cannot be instantiated from the hardness of something other than the DH assumption such as an integer factoring problem, code-based problem, or lattice problem. Thus, we still have no AKE protocol that is secure in the 'strongest' model under just a general assumption without relying on random oracles (ROs).

1.3 Our Contribution

We fully solve the open problem by providing a generic construction of AKE from KEM. Our construction is a generalization of the BCGNP construction. The BCGNP construction uses IND-CCA KEM, a pseudo-random function (PRF), and a key derivation function (KDF) as building blocks. Our construction effectively follows the design principle of the BCGNP construction. However, we first point out that the security proof of the BCGNP construction is not complete. Specifically, a requirement for KEM has not been formulated. KEM keys must have enough min-entropy in order to make outputs of the KDF computationally indistinguishable from a uniformly random chosen element. The IND-CCA security does not imply min-entropy of KEM keys. Thus, the assumption that the KEM scheme satisfies such a property is additionally required. Fortunately, almost all IND-CCA KEM schemes satisfy that. Also, we need an IND-CPA secure KEM in addition to the BCGNP construction. Such an additional KEM can make our scheme wPFS and resilient to MEX. The resultant AKE protocol is CK^+ secure. Its security is proved under the existence of such KEMs, a KDF, and a PRF in the StdM. IND-CCA secure KEM schemes have been shown from the hardness of integer factoring [HK09a,MLLJ11], code-based problems [McE78,DMQN09], or lattice problems [PW08,Pei09,CHKP10,ABB10a,ABB10b,SSTX09,LPR10]. To the best of our knowledge, our generic construction provides the first CK^+ secure AKE protocols based on the hardness of the above problems. Regarding the DH assumption or its variant, our generic construction is

the first protocol that achieves CK$^+$ security in the StdM without non-standard assumptions (e.g., πPRF and KEA1).

We also rewrite the CK$^+$ model before proving the security of our generic construction in order to simplify the original model in [Kra05]. Specifically, the original model is defined as a mix of four definitions (i.e., the CK model, wPFS, and resistance to KCI and MEX); thus, the security proof must also be separated into four theorems, which may reduce the readability. Therefore, we reformulate the CK$^+$ model as follows: wPFS, resistance to KCI, and resistance to MEX are integrated into the experiment of the extended model by exhaustively classifying exposure patterns. This definition is handy to prove security and rigorously captures all required properties.

Moreover, we show an extension of the above result to the ID-based setting. It is natural to introduce ID-based cryptography in order to avoid the burden of key managements. In ID-based cryptography, it is assumed that a *key generate center* (KGC) exists. The KGC manages system parameters and a master secret key, and generates a static secret key of each party with the master secret key. However, this means that the master key is more powerful than static secret keys of parties. We need an additional security requirement, called master-key forward secrecy (mFS), that the session key is protected if the master secret key is revealed but ephemeral secret keys are not revealed. Thus, first, we formulate an ID-based version of the CK$^+$ model (called the id-CK$^+$ model) that captures mFS. Boyd et al. [BCGNP08,BCGNP09] gave a generic construction of ID-AKE based on ID-based chosen-ciphertext secure (IND-ID-CCA) ID-based KEM (IB-KEM). Next, we improve their ID-AKE construction as the same way as our generic AKE construction. IND-ID-CCA secure IB-KEM schemes have been shown from the hardness of bilinear pairing problems [BF01,BBS04], or lattice problems [PW08,Pei09,CHKP10,ABB10a,ABB10b,SSTX09,LPR10]. Our generic construction provides the first id-CK$^+$ secure ID-AKE protocols based on the hardness of the above problems in the StdM.

We summarize our contributions as follows:

1. We reformulate CK$^+$ and id-CK$^+$ models to gain readability of the security proofs.
2. We propose two-pass generic CK$^+$ secure AKE and two-pass generic id-CK$^+$ secure ID-AKE constructions in the StdM.
3. We achieve the first CK$^+$ secure AKE protocols based on the hardness of integer factorization problem, code-based problems, and lattice-based problems in the StdM.
4. We achieve the first CK$^+$ secure AKE protocol based on the DH assumption or its variant in the StdM without knowledge assumptions.
5. We achieve the first id-CK$^+$ secure ID-AKE protocols based on the hardness of bilinear pairing problems, and lattice-based problems in the StdM.

The proposed generic construction can allow a hybrid instantiation; that is, the initiator and the responder can use different KEMs under different assumptions. For example, the initiator uses a factoring-based KEM while the responder uses a lattice-based KEM.

2 Security Models

In this section, we recall the CK$^+$ model that was introduced by [Kra05]. We show a model specified to two pass protocols for simplicity. It can be trivially extended to any round protocol. Also, we show the id-CK$^+$ model as an extension of the CK$^+$ model.

Throughout this paper we use the following notations. If Set is a set, then by $m \in_R \mathsf{Set}$ we denote that m is sampled uniformly from Set. If $\mathcal{ALG}$ is an algorithm, then by $y \leftarrow \mathcal{ALG}(x; r)$ we denote that y is output by $\mathcal{ALG}$ on input x and randomness r (if $\mathcal{ALG}$ is deterministic, r is empty).

Table 1. Classification of attacks and proposed CK^+ model [Kra05] and eCK model [LLM07].

Cases in Def.2	ssk_A	esk_A	ssk_B	esk_B	attack type	CK^+ model	eCK model
2-(a)	r	ok	ok	n	KCI	✓	✓
2-(b)	ok	r	ok	n	MEX	✓	✓
2-(c)	r	ok	ok	r	KCI	✓	✓
2-(d)	ok	r	ok	r	MEX	✓	✓
2-(e)	r	ok	r	ok	wPFS	✓	✓
2-(f)	ok	r	r	ok	KCI	✓	✓

"2-(*)" means the corresponding case in Definition 2. "ssk_A" means the static secret key of owner A of test session sid^*, and "ssk_B" means the static secret key of peer B of test session sid^*. "esk_A" means the ephemeral secret key of test session sid^*, and "esk_B" means the ephemeral secret key of the matching session $\overline{\mathsf{sid}^*}$. "ok" means the secret key is not revealed, "r" means the secret key may be revealed, and "n" means no matching session exists. ✓ means that the model captures the attack.

Table 2. Comparison of CK^+ model [Kra05] and eCK model [LLM07].

	CK^+ model [Kra05]	eCK model [LLM07]
All non-trivial key exposure	✓	✓
Session state reveal	✓	χ
Adaptive key exposure	χ	✓

✓/χ means that the model does/does not capture the attack.

2.1 CK^+ vs. eCK

As indicated in Table 1, the CK^+ model captures all non-trivial patterns of exposure of static and ephemeral secret keys. The eCK model [LLM07], which is a variant of the CK model [CK01], also captures all non-trivial patterns of exposure, as in Table 1. Since the CK^+ model captures all non-trivial patterns of exposure of static and ephemeral secret keys, the CK^+ model can theoretically be seen as a completion of the AKE security model.

In Table 1, the six cases in Definition 2 are listed, and these six cases cover wPFS, resistance to KCI, and MEX as follows: Cases 2-(a), 2-(c), and 2-(f) capture KCI, since the adversary obtains the static secret key of one party and the ephemeral secret key of the other party of the test session. Case 2-(e) captures wPFS, since the adversary obtains the static secret keys of both parties of the test session. Cases 2-(b) and 2-(d) capture MEX, since the adversary obtains the ephemeral secret keys of both parties of the test session.

The main difference between the CK^+ model and the eCK model is that the CK^+ model captures the session state reveal attack, but the eCK model does not. Thus, we adopt the CK^+ model, which is stronger than the eCK model from the viewpoint of the session state reveal attack, in this paper.

Notice that the timing of the static and ephemeral key reveal differs in the eCK and CK^+ models. In the eCK model, an adversary can issue the static and ephemeral key reveal query adaptively. In contrast, in the CK^+ model, an adversary can issue a corrupt query to obtain the static key, and the ephemeral key is given to the adversary when it is determined. We summarize this in Table 2.

2.2 CK^+ Security Model

We denote a party by U_i, and party U_i and other parties are modeled as probabilistic polynomial-time (PPT) Turing machines w.r.t. security parameter κ. For party U_i, we denote static secret

(public) key by s_i (S_i) and ephemeral secret (public) key by x_i (X_i). Party U_i generates its own keys, s_i and S_i, and the static public key S_i is linked with U_i's identity in some systems like PKI.[2]

Session An invocation of a protocol is called a *session*. Session activation is done by an incoming message of the forms $(\Pi, \mathcal{I}, U_A, U_B)$ or $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, where we equate Π with a protocol identifier, $\mathcal{I}$ and $\mathcal{R}$ with role identifiers, and U_A and U_B with user identifiers. If U_A is activated with $(\Pi, \mathcal{I}, U_A, U_B)$, then U_A is called the session *initiator*. If U_B is activated with $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, then U_B is called the session *responder*. The initiator U_A outputs X_A, then may receive an incoming message of the forms $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ from the responder U_B, U_A then computes the session key SK if U_A received the message. On the contrary, the responder U_B outputs X_B, and computes the session key SK.

If U_A is the initiator of a session, the session is identified by $\mathsf{sid} = (\Pi, \mathcal{I}, U_A, U_B, X_A)$ or $\mathsf{sid} = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$. If U_B is the responder of a session, the session is identified by $\mathsf{sid} = (\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$. We say that U_A is the *owner* of session sid, if the third coordinate of session sid is U_A. We say that U_A is the *peer* of session sid, if the fourth coordinate of session sid is U_A. We say that a session is *completed* if its owner computes the session key. The *matching session* of $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ is session $(\Pi, \mathcal{R}, U_B, U_A, X_A, X_B)$ and vice versa.

Adversary The adversary $\mathcal{A}$, which is modeled as a probabilistic polynomial-time Turing machine, controls all communications between parties including session activation by performing the following adversary query.

- Send(message): The message has one of the following forms: $(\Pi, \mathcal{I}, U_A, U_B)$, $(\Pi, \mathcal{R}, U_B, U_A, X_A)$, or $(\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$. The adversary $\mathcal{A}$ obtains the response from the party.

To capture exposure of secret information, the adversary $\mathcal{A}$ is allowed to issue the following queries.

- $\mathsf{SessionKeyReveal}(\mathsf{sid})$: The adversary $\mathcal{A}$ obtains the session key SK for the session sid if the session is completed.
- $\mathsf{SessionStateReveal}(\mathsf{sid})$: The adversary $\mathcal{A}$ obtains the session state of the owner of session sid if the session is not completed (the session key is not established yet). The session state includes all ephemeral secret keys and intermediate computation results except for immediately erased information but does not include the static secret key.
- $\mathsf{Corrupt}(U_i)$: This query allows the adversary $\mathcal{A}$ to obtain all information of the party U_i. If a party is corrupted by a $\mathsf{Corrupt}(U_i, S_i)$ query issued by the adversary $\mathcal{A}$, then we call the party U_i *dishonest*. If not, we call the party *honest*.

Freshness For the security definition, we need the notion of freshness.

Definition 1 (Freshness). *Let* $\mathsf{sid}^* = (\Pi, \mathcal{I}, U_A, U_B, X_A, X_B)$ *or* $(\Pi, \mathcal{R}, U_A, U_B, X_B, X_A)$ *be a completed session between honest users* U_A *and* U_B*. If the matching session exists, then let* $\overline{\mathsf{sid}^*}$ *be the matching session of* sid^**. We say session* sid^* *is* fresh *if none of the following conditions hold:*

1. *The adversary* $\mathcal{A}$ *issues* $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$*, or* $\mathsf{SessionKeyReveal}(\overline{\mathsf{sid}^*})$ *if* $\overline{\mathsf{sid}^*}$ *exists,*

[2] Static public keys must be known to both parties in advance. They can be obtained by exchanging them before starting the protocol or by receiving them from a certificate authority. This situation is common for all PKI-based AKE schemes.

2. $\overline{\mathsf{sid}^*}$ *exists and the adversary* $\mathcal{A}$ *makes either of the following queries*
   – $\mathsf{SessionStateReveal}(\mathsf{sid}^*)$ *or* $\mathsf{SessionStateReveal}(\overline{\mathsf{sid}^*})$,
3. $\overline{\mathsf{sid}^*}$ *does not exist and the adversary* $\mathcal{A}$ *makes the following query*
   – $\mathsf{SessionStateReveal}(\mathsf{sid}^*)$.

Security Experiment For the security definition, we consider the following security experiment. Initially, the adversary $\mathcal{A}$ is given a set of honest users and makes any sequence of the queries described above. During the experiment, the adversary $\mathcal{A}$ makes the following query.

– $\mathsf{Test}(\mathsf{sid}^*)$: Here, sid^* must be a fresh session. Select random bit $b \in_U \{0,1\}$, and return the session key held by sid^* if $b = 0$, and return a random key if $b = 1$.

The experiment continues until the adversary $\mathcal{A}$ makes a guess b'. The adversary $\mathcal{A}$ *wins* the game if the test session sid^* is still fresh and if the guess of the adversary $\mathcal{A}$ is correct, i.e., $b' = b$. The advantage of the adversary $\mathcal{A}$ in the AKE experiment with the PKI-based AKE protocol Π is defined as

$$\mathrm{Adv}_{\Pi}^{\mathrm{AKE}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins] - \frac{1}{2}.$$

We define the security as follows.

Definition 2 (Security for PKI-based AKE). *We say that a PKI-based AKE protocol* Π *is* secure in the CK^+ model *if the following conditions hold:*

1. *If two honest parties complete matching sessions, then, except with negligible probability, they both compute the same session key.*
2. *For any PPT bounded adversary* $\mathcal{A}$, $\mathrm{Adv}_{\Pi}^{\mathrm{AKE}}(\mathcal{A})$ *is negligible in security parameter* κ *for the test session* sid^*,
   (a) *if* $\overline{\mathsf{sid}^*}$ *does not exist, and the static secret key of the owner of* sid^* *is given to* $\mathcal{A}$.
   (b) *if* $\overline{\mathsf{sid}^*}$ *does not exist, and the ephemeral secret key of* sid^* *is given to* $\mathcal{A}$.
   (c) *if* $\overline{\mathsf{sid}^*}$ *exists, and the static secret key of the owner of* sid^* *and the ephemeral secret key of* $\overline{\mathsf{sid}^*}$ *are given to* $\mathcal{A}$.
   (d) *if* $\overline{\mathsf{sid}^*}$ *exists, and the ephemeral secret key of* sid^* *and the ephemeral secret key of* $\overline{\mathsf{sid}^*}$ *are given to* $\mathcal{A}$.
   (e) *if* $\overline{\mathsf{sid}^*}$ *exists, and the static secret key of the owner of* sid^* *and the static secret key of the peer of* sid^* *are given to* $\mathcal{A}$.
   (f) *if* $\overline{\mathsf{sid}^*}$ *exists, and the ephemeral secret key of* sid^* *and the static secret key of the peer of* sid^* *are given to* $\mathcal{A}$.

Note that the items 2.a, 2.c, and 2.f correspond to resistance to KCI, item 2.e corresponds to wPFS, and items 2.b and 2.d correspond to resistance to MEX.

2.3 id-CK$^+$ Security Model

The id-CK^+ security model for ID-AKE is similarly defined as the CK^+ model. There are some differences between two models as follows:

– The KGC generates the master secret key and public parameter.
– Static secret keys of parties are generated by the KGC with the master secret key and IDs.
– An adversary may reveal the master secret key according to mFS.

Formulations of sessions, adversarial oracle queries, and freshness are not changed with the CK^+ model. To capture mFS, we modify the definition of security experiment.

Definition 3 (Security for ID-AKE). *We say that a ID-AKE protocol Π is secure in the* id-CK$^+$ *model if the following conditions hold:*

1. *If two honest parties complete matching sessions, then, except with negligible probability, they both compute the same session key.*
2. *For any PPT adversary $\mathcal{A}$,* $\mathrm{Adv}_{\Pi}^{\text{ID-AKE}}(\mathcal{A})$ *is negligible in security parameter κ for the* fresh *test session* sid^**,*
   (a) *if* $\overline{\mathsf{sid}^*}$ *does not exist, and the static secret key of the owner of* sid^* *is given to $\mathcal{A}$.*
   (b) *if* $\overline{\mathsf{sid}^*}$ *does not exist, and the ephemeral secret key of* sid^* *is given to $\mathcal{A}$.*
   (c) *if* $\overline{\mathsf{sid}^*}$ *exists, and the static secret key of the owner of* sid^* *and the ephemeral secret key of* $\overline{\mathsf{sid}^*}$ *are given to $\mathcal{A}$.*
   (d) *if* $\overline{\mathsf{sid}^*}$ *exists, and the ephemeral secret key of* sid^* *and the ephemeral secret key of* $\overline{\mathsf{sid}^*}$ *are given to $\mathcal{A}$.*
   (e) *if* $\overline{\mathsf{sid}^*}$ *exists, and the static secret key of the owner of* sid^* *and the static secret key of the peer of* sid^* *are given to $\mathcal{A}$.*
   (f) *if* $\overline{\mathsf{sid}^*}$ *exists, and the ephemeral secret key of* sid^* *and the static secret key of the peer of* sid^* *are given to $\mathcal{A}$.*
   (g) *if* $\overline{\mathsf{sid}^*}$ *exists, the master secret key msk is given to $\mathcal{A}$.*

Note that the items 2.a, 2.c, and 2.f correspond to resistance to KCI, item 2.e corresponds to wPFS, items 2.b and 2.d correspond to resistance to MEX, and item 2.g corresponds to mFS.

3 Generic AKE Construction from KEM without Random Oracles

In this section, we propose a generic construction of CK$^+$-secure AKE from KEM.

3.1 Preliminaries

Security Notions of KEM Schemes Here, we recall the definition of IND-CCA and IND-CPA security for KEM, and min-entropy of KEM keys as follows.

Definition 4 (Model for KEM Schemes). *A KEM scheme consists of the following 3-tuple* $(\mathsf{KeyGen}, \mathsf{EnCap}, \mathsf{DeCap})$*:*

- $(ek, dk) \leftarrow \mathsf{KeyGen}(1^\kappa, r_g)$ *: a key generation algorithm which on inputs 1^κ and $r_g \in \mathcal{RS}_G$, where κ is the security parameter and $\mathcal{RS}_G$ is a randomness space, outputs a pair of keys (ek, dk).*
- $(K, CT) \leftarrow \mathsf{EnCap}_{ek}(r_e)$ *: an encryption algorithm which takes as inputs encapsulation key ek and $r_e \in \mathcal{RS}_E$, outputs session key $K \in \mathcal{KS}$ and ciphertext $CT \in \mathcal{CS}$, where $\mathcal{RS}_E$ is a randomness space, $\mathcal{KS}$ is a session key space, and $\mathcal{CS}$ is a ciphertext space.*
- $K \leftarrow \mathsf{DeCap}_{dk}(CT)$ *: a decryption algorithm which takes as inputs decapsulation key dk and ciphertext $CT \in \mathcal{CS}$, and outputs session key $K \in \mathcal{KS}$.*

Definition 5 (IND-CCA and IND-CPA Security for KEM). *A KEM scheme is IND-CCA-secure for KEM if the following property holds for security parameter κ; For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,* $\mathbf{Adv}^{\text{ind-cca}} = |\Pr[r_g \leftarrow \mathcal{RS}_G; (ek, dk) \leftarrow \mathsf{KeyGen}(1^\kappa, r_g);\ (state) \leftarrow \mathcal{A}_1^{\mathcal{DO}(dk,\cdot)}(ek);\ b \leftarrow \{0,1\};\ r_e \leftarrow \mathcal{RS}_E; (K_0^*, CT_0^*) \leftarrow \mathsf{EnCap}_{ek}(r_e);\ K_1^* \leftarrow \mathcal{K};\ b' \leftarrow \mathcal{A}_2^{\mathcal{DO}(dk,\cdot)}(ek, (K_b^*, CT_0^*), state);\ b' = b] - 1/2| \leq negl$*, where $\mathcal{DO}$ is the decryption oracle, $\mathcal{K}$ is the space of session key and state is state information that $\mathcal{A}$ wants to preserve from $\mathcal{A}_1$ to $\mathcal{A}_2$. $\mathcal{A}$ cannot submit the ciphertext $CT = CT_0^*$ to $\mathcal{DO}$.*

We say a KEM scheme is IND-CPA-secure for KEM if $\mathcal{A}$ does not access $\mathcal{DO}$.

Definition 6 (Min-Entropy of KEM Key). *A KEM scheme is k-min-entropy KEM if for any ek, distribution $D_{\mathcal{KS}}$ of variable K defined by $(K, CT) \leftarrow \mathsf{EnCap}_{ek}(r_e)$, distribution D_{pub} of public information and random $r_e \in \mathcal{RS}_E$, $H_\infty(D_{\mathcal{KS}}|D_{pub}) \geq k$ holds, where H_∞ denotes min-entropy.*

Security Notion of Key Derivation Function. Let $KDF : Salt \times Dom \to Rng$ be a function with finite domain Dom, finite range Rng, and a space of non-secret random salt $Salt$.

Definition 7 (Key Derivation Function [GS04]). *We say function KDF is a key derivation function (KDF) if the following condition holds for a security parameter κ: For any PPT adversary $\mathcal{A}$ and any distribution D_{Rng} over Rng with $H_\infty(D_{Rng}) \geq \kappa$, $|\Pr[y \in_R Rng; 1 \leftarrow \mathcal{A}(y)] - \Pr[x \in_R Dom; s \in_R Salt; y \leftarrow KDF(s, x); 1 \leftarrow \mathcal{A}(y)]| \leq negl$.*

For example, concrete constructions of such a computationally secure KDF are given in [Kra10,DSGKM12] from a computational extractor and a PRF.

Security Notion of Pseudo-Random Function. Let κ be a security parameter and $\mathsf{F} = \{F_\kappa : Dom_\kappa \times \mathcal{FS}_\kappa \to Rng_\kappa\}_\kappa$ be a function family with a family of domains $\{Dom_\kappa\}_\kappa$, a family of key spaces $\{\mathcal{FS}_\kappa\}_\kappa$ and a family of ranges $\{Rng_\kappa\}_\kappa$.

Definition 8 (Pseudo-Random Function). *We say that function family $\mathsf{F} = \{F_\kappa\}_\kappa$ is a PRF family if for any PPT distinguisher $\mathcal{D}$, $\mathbf{Adv}^{\mathrm{prf}} = |\Pr[1 \leftarrow \mathcal{D}^{F_\kappa(\cdot)}] - \Pr[1 \leftarrow \mathcal{D}^{RF_\kappa(\cdot)}]| \leq negl$, where $RF_\kappa : Dom_\kappa \to Rng_\kappa$ is a truly random function.*

3.2 Construction

Our construction (GC) is based on an IND-CCA secure KEM, an IND-CPA secure KEM, PRFs, and a KDF. While the requirements for the underlying building blocks are not stronger than those for the previous generic construction [BCGNP08,BCGNP09], GC achieves stronger security (i.e., CK^+ security) without random oracles.

Necessity of Min-Entropy of KEM Key In the BCGNP construction, a KEM scheme is only assumed to be IND-CCA. However, it is not enough to prove the security. Both parties derive the session key by applying decapsulated KEM keys to a strong randomness extractor before applying them to PRFs. This extractor guarantees to output a statistically indistinguishable value from a uniform randomly chosen element from the same space. It requires as input a (public) seed and a KEM session key with min-entropy κ, where κ is a security parameter. IND-CCA states that no PPT adversary can distinguish the KEM key from a random element, but this does not directly guarantee min-entropy of the KEM session key. Thus, we must also assume that min-entropy of the KEM session key is equal or larger than κ. This property is not very strong; almost all IND-CCA secure schemes satisfy it. We will discuss later about this property of concrete KEM schemes.

Also, we can improve the efficiency of the session key derivation procedure of the BCGNP construction by using a KDF instead of a strong randomness extractor. On input a value having sufficient min-entropy, a strong randomness extractor outputs a value which is *statistically indistinguishable* from a uniformly chosen random value. Indeed, such statistical indistinguishability is not necessary to prove the security of our construction. *Computational indistinguishability* is sufficient, and the KDF [GS04] is suitable. Such a technique is also used in [Yon13b,Yon13a].

Design Principle The main ideas to achieve CK^+ security are to use the *twisted PRF* trick and *session-specific* key generation.

First, we have to consider resistance to MEX. The most awkward pattern of MEX is the disclosure of ephemeral secret keys of the initiator and the responder. If we use KEM naturally, all randomness used to generate ciphertexts is exposed as ephemeral secret keys; thus, the adversary can obtain encrypted messages without knowing secret keys. Hence, we have to avoid using ephemeral secret keys as randomness of KEM directly. A possible solution is to generate randomness from the static secret key as well as the ephemeral secret key by using a technique such as the ordinary NAXOS trick [LLM07]. Though this trick leads to security against exposure of ephemeral secret keys, the trick must apply an RO to the concatenation of the static and ephemeral secret keys, and it uses the output as a quasi-ephemeral secret key. It is unsuitable for our purpose to construct secure protocols in the StdM. Thus, we use a trick to achieve the same properties as the NAXOS trick but without ROs. We call it the twisted PRF trick.[3] This trick uses two PRFs (F, F') with reversing keys; we choose two ephemeral keys (r, r') and compute $F_\sigma(r) \oplus F'_{r'}(\sigma')$, where σ and σ' are static secret keys. The twisted PRF trick is especially effective in the following two scenarios: exposure of both ephemeral secret keys of the initiator and the responder, and exposure of the static secret key of the initiator and the ephemeral secret key of the responder (i.e., corresponding to KCI). If (r, r') is exposed, $F_\sigma(r)$ cannot be computed without knowing σ. Similarly, if σ and σ' are exposed, $F'_{r'}(\sigma')$ cannot be computed without knowing r'. In our KEM-based generic construction, the output of the twisted PRF is used as randomness for the encapsulation algorithm.

Next, we have to consider the scenario in which static secret keys are exposed as the attack scenario in wPFS. We cannot achieve a CK^+ secure scheme by any combination of KEMs using static secret keys as decapsulation keys against exposure of both static secret keys of the initiator and the responder because an adversary can obtain all information that the parties can obtain by using static secret keys. Our solution is to generate session-specific decapsulation and encapsulation keys. The initiator sends the temporary encapsulation key to the responder, the responder encapsulates a KEM key with the temporary encapsulation key, and the initiator decapsulates the ciphertext. Since this procedure does not depend on the static secret keys, the KEM key is hidden even if both static secret keys of the initiator and the responder are exposed. Note that security of KEM for temporary use only requires IND-CPA. The session-specific key generation is effective for achieving wPFS.

As the BCGNP construction [BCGNP08,BCGNP09], we use IND-CCA secure KEM schemes to exchange ciphertexts. The CCA security is necessary to simulate SessionStateReveal queries in the security proof. When we prove security in the case where ephemeral secret keys are exposed, the simulator needs to embed the challenge ciphertext in the ephemeral public key in the test session. Then, the static secret key to decrypt the challenge ciphertext is not known; that is, the simulator must respond to the SessionStateReveal query for a session owned by the same parties as the test session without knowing the static secret key. Hence, the simulator needs the power of the decryption oracle to obtain intermediate computation results corresponding to the SessionStateReveal query.

Generic Construction GC The protocol of GC from KEMs (KeyGen, EnCap, DeCap) and (wKeyGen, wEnCap, wDeCap) is as follows.

Public Parameters. Let κ be the security parameter, $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and $G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$ be pseudo-random functions, where $\mathcal{FS}$ is the key space of PRFs ($|\mathcal{FS}| =$

[3] A similar trick is used in the Okamoto AKE scheme [Oka07].

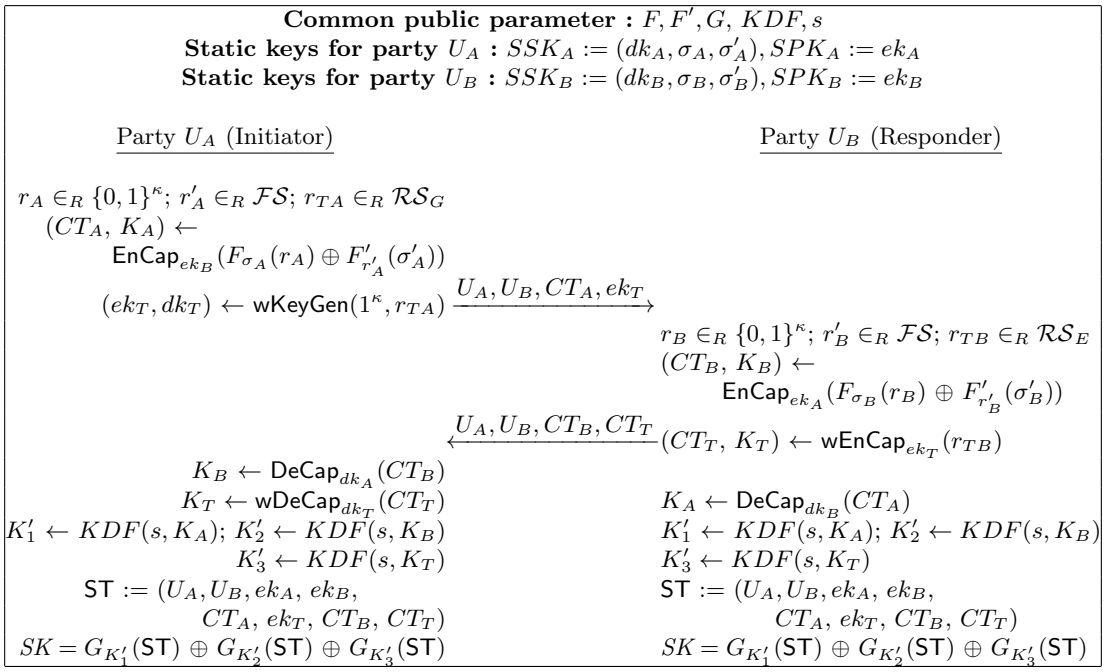

Fig. 1. Generic construction GC

κ), $\mathcal{RS}_E$ is the randomness space of encapsulation algorithms, and $\mathcal{RS}_G$ is the randomness space of key generation algorithms, and let $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ be a KDF with a non-secret random salt $s \in Salt$, where $Salt$ is the salt space and $\mathcal{KS}$ is a space of KEM session keys. These are provided as some of the public parameters.

Secret and Public Keys. Party U_P randomly selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r \in_R \mathcal{RS}_G$, and runs $(ek_P, dk_P) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$. Party U_P's SSK and SPK are $((dk_P, \sigma_P, \sigma'_P), ek_P)$.

Key Exchange. Party U_A with secret and public keys $((dk_{A,1}, \sigma_A, \sigma'_A), ek_A)$ as the initiator, and party U_B with secret and public keys $((dk_{B,1}, \sigma_B, \sigma'_B), ek_B)$ as the responder, perform the following two-pass key exchange protocol.

1. Party U_A randomly chooses ephemeral secret keys $r_A \in_R \{0,1\}^\kappa$, $r'_A \in_R \mathcal{FS}$ and $r_{TA} \in \mathcal{RS}_G$. Party U_A computes $(CT_A, K_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus F'_{r'_A}(\sigma'_A))$ and $(ek_T, dk_T) \leftarrow \mathsf{wKeyGen}(1^\kappa, r_{TA})$ and sends (U_A, U_B, CT_A, ek_T) to party U_B.
2. Upon receiving (U_A, U_B, CT_A, ek_T), party U_B chooses the ephemeral secret keys $r_B \in_R \{0,1\}^\kappa$, $r'_B \in_R \mathcal{FS}$ and $r_{TB} \in_R \mathcal{RS}_E$, computes $(CT_B, K_B) \leftarrow \mathsf{EnCap}_{ek_A}(F_{\sigma_B}(r_B) \oplus F'_{r'_B}(\sigma'_B))$ and $(CT_T, K_T) \leftarrow \mathsf{wEnCap}_{ek_T}(r_{TB})$, and sends (U_A, U_B, CT_B, CT_T) to party U_A.
   Party U_B computes $K_A \leftarrow \mathsf{DeCap}_{dk_B}(CT_A)$, $K'_1 \leftarrow KDF(s, K_A)$, $K'_2 \leftarrow KDF(s, K_B)$ and $K'_3 \leftarrow KDF(s, K_T)$, sets the session transcript $\mathsf{ST} = (U_A, U_B, ek_A, ek_B, CT_A, ek_T, CT_B, CT_T)$ and the session key $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$, completes the session, and erases all session states.
3. Upon receiving (U_A, U_B, CT_B, CT_T), party U_A computes $K_B \leftarrow \mathsf{DeCap}_{dk_A}(CT_B)$, $K_T \leftarrow \mathsf{wDeCap}_{dk_T}(CT_T)$, $K'_1 \leftarrow KDF(s, K_A)$, $K'_2 \leftarrow KDF(s, K_B)$ and $K'_3 \leftarrow KDF(s, K_T)$, sets the session transcript $\mathsf{ST} = (U_A, U_B, ek_A, ek_B, CT_A, ek_T, CT_B, CT_T)$ and the session key $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$, completes the session, and erases all session states.

The session state of a session owned by U_A contains ephemeral secret keys (r_A, r_{TA}), encapsulated KEM key K_A and ad-hoc decryption key dk_T. Other information that is computed after receiving the message from the peer is immediately erased when the session key is established. Similarly, the session state of a session owned by U_B contains ephemeral secret keys (r_B, r_{TB}) and encapsulated KEM keys K_B and K_T.

Other intermediate values (e.g., decapsulated KEM keys, and outputs of KDF) are not contained in session state. After receiving the message from the peer all intermediate computations are executed without stopping, and such values are immediately erased after finishing the session. Boyd et al. [BCGNP08] showed that protocols based on KEM would be trivially broken if an adversary learned these values by SessionStateReveal.

Remark 1. Obviously, we can use arbitrary combinations of KEM schemes in the generic construction. This means that each party can rely on a different assumption from the peer. Since our construction does not contain any direct operation between derivatives of KEM schemes, it is no problem that randomness spaces, public keys, or ciphertext are distinct from each other.

3.3 Security

We show the following theorem.

Theorem 1. *If* (KeyGen, EnCap, DeCap) *is IND-CCA secure KEM and is* κ*-min-entropy KEM,* (wKeyGen, wEnCap, wDeCap) *is IND-CPA secure KEM and is* κ*-min-entropy KEM,* F, F', G *are PRFs, and* KDF *is a KDF, then AKE construction* GC *is* CK^+*-secure.*

The proof of Theorem 1 is shown in Appendix A. Here, we give an overview of the security proof.

We have to consider the following four exposure patterns in the CK^+ security model (matching cases):

2-(c) the static secret key of the initiator and the ephemeral secret key of the responder
2-(d) both ephemeral secret keys
2-(e) both static secret keys
2-(f) the ephemeral secret key of the initiator and the static secret key of the responder

In case 2-(c), K_A is protected by the security of CT_A because r'_A is not exposed; therefore, $F'_{r'_A}(\sigma'_A)$ is hidden and dk_B is not exposed. In case 2-(d), K_A and K_B are protected by the security of CT_A and CT_B because σ_A and σ_B are not exposed; therefore, $F_{\sigma_A}(r_A)$ and $F_{\sigma_B}(r_B)$ are hidden and dk_A and dk_B are not exposed. In case 2-(e), K_T is protected by the security of CT_T because dk_T and r_{TB} are not exposed. In case 2-(f), K_B is protected by the security of CT_B because r'_B is not exposed; therefore, $F'_{r'_B}(\sigma'_B)$ is hidden and dk_A is not exposed. Then, we transform the CK^+ security game since the session key in the test session is randomly distributed. First, we change part of the twisted PRF in the test session into a random function because the key of part of the twisted PRF is hidden from the adversary; therefore, the randomness of the protected KEM can be randomly distributed. Second, we change the protected KEM key into a random key for each pattern; therefore, the input of KDF is randomly distributed and has sufficient min-entropy. Third, we change the output of KDF into randomly chosen values. Finally, we change one of the PRFs (corresponding to the protected KEM) into a random function. Therefore, the session key in the test session is randomly distributed; thus, there is no advantage to the adversary. We can show a similar proof in non-matching cases.

3.4 Instantiations

3.5 Diffie-Hellman-based

We can achieve various AKE schemes as concrete instantiations based on the hardness of the DH problem and its variants. These are derived from the generic construction GC in Section 3. For example, we can apply efficient IND-CCA KEM schemes to GC from the decisional DH [CS98,KD04] (DDH), computational DH [HK08,HJKS10], hashed DH [Kil07] and bilinear DH assumptions [BMW05].

We can easily show that these schemes have κ-min-entropy KEM keys. The KEM part of the Cramer-Shoup PKE consists of $g_1^{zr} \in G$, where G is a finite cyclic group of order prime p, g_1^z is part of ek, and r is uniformly chosen randomness, and $|r|$ is 2κ. Thus, g_1^{zr} has min-entropy larger than κ. Similarly, other schemes are also κ-min-entropy KEM.

The significant advantage of our instantiations in the StdM is reasonable assumption. First, HMQV satisfies the same security model as our construction. However, it requires the KEA1 assumption and relies on ROs. Since it has been criticized, in particular because the KEA1 assumption does not appear to be "efficiently falsifiable" as Naor put it [Nao03], this assumption is quite undesirable. Also, it was shown that there exist some protocols that are secure in the ROM but are insecure if ROs are replaced by any specific function [CGH98]. A disadvantage of our construction to HMQV is that HMQV is a one-round protocol but our scheme is not. One-round protocols mean that the initiator and the responder can send their messages independently and simultaneously. Conversely, in our scheme, the responder must wait to receive the message from the initiator. Next, the AKE scheme by Okamoto [Oka07] is secure in the StdM. However, it is not proved in the CK^+ model and needs to assume existence of πPRF. πPRF is a stronger primitive than ordinary PRF, and it is not known how to construct πPRF concretely. On the contrary, our instantiations only require the standard notions of KEM and pseudo-random function security. Moreover, the BCGNP construction [BCGNP08,BCGNP09] is secure in the StdM with standard assumption. However, the security is not proved in the CK^+ model.[4] Thus, DH-based AKE schemes from GC are first CK^+ secure schemes in the StdM with standard assumptions.

For example, our scheme can be instantiated with the Cramer-Shoup KEM [CS04] as an IND-CCA KEM, and with the ElGamal KEM as an IND-CPA KEM under the DDH assumption. Communication complexity (for two parties) of this instantiation is $8|p|$, where $|p|$ is the length of a group element. Computational complexity (for two parties) of this instantiation is 4 multi-exponentiations and 12 regular exponentiations (all symmetric operations such as hash function/KDF/PRF and multiplications are ignored). We show a comparison between this instantiation and previous schemes in Table 3.

3.6 Factoring-based

We can achieve several new AKE protocols as concrete instantiations based on the hardness of integer factorization and its variants such as the RSA problem.

Some instantiations in the StdM are based on the hardness of the integer factorization problem. The Hofheinz-Kiltz PKE [HK09a] and the Mei-Li-Lu-Jia PKE [MLLJ11] are IND-CCA secure in the StdM under the factoring assumption. Furthermore, by applying the fact [HK09b]

[4] The BCGNP construction with an additional exchange of a DH value (called Protocol 2 in [BCGNP08,BCGNP09]) can be proved in the CK model, and it satisfies wPFS and resistance to KCI. We can extend the security of Protocol 2 to the CK^+ security with the twisted PRF trick. If IND-CPA KEM in GC is instantiated with the ElGamal KEM, our scheme is the same as Protocol 2 with the twisted PRF trick. Thus, our scheme can also be seen as a generalization of the BCGNP construction.

Table 3. Comparison of previous DH-based schemes and an instantiation of our scheme

	Model	Resource	Assumption	Computation (#[multi,regular]-exp.)	Communication complexity	
[Kra05]	CK^+	ROM	gap DH & KEA1	$[2, 2]$	$2\|p\|$	512
[Oka07]	eCK	StdM	DDH & πPRF	$[6, 6]$	$9\|p\|$	2304
[BCGNP09]	CK & KCI	StdM	DDH	$[4, 8]$	$6\|p\|$	1536
Ours	CK^+	StdM	DDH	$[4, 12]$	$8\|p\|$	2048

For concreteness the expected ciphertext overhead for a 128-bit implementation is also given. Note that computational costs are estimated without any pre-computation technique.

that if a scheme is secure under the CDH assumption in $\mathbb{Z}_N^*$, it is also secure under the factoring assumption, we can obtain more efficient factoring-based KEM schemes from IND-CCA secure KEM under the CDH assumption such as [HK08,HJKS10]. Thus, we can obtain first CK^+ secure AKE protocols in the StdM under the integer factorization assumption. Also, we have other instantiations based on the hardness of RSA inversion. By applying the Chevallier-Mames-Joye PKE [CMJ09] and the Kiltz-Mohassel-O'Neill PKE [KMO10], which are IND-CCA secure in the StdM under the instance-independent RSA assumption to GC, we can obtain first CK^+ secure AKE protocols in the StdM under the RSA-type assumption.

We can regard a message in PKE as a KEM key when the message space is larger than κ and messages are uniformly chosen randomness. In this case, it is obvious that such a KEM scheme is κ-min-entropy KEM.

3.7 Code-based

We can achieve new AKE protocols as concrete instantiations based on code-based problems.

For the AKE protocol in the StdM, we can apply Dowsley et al.'s PKE [DMQN09] that is IND-CCA secure in the StdM under the McEliece and LPN assumptions to GC. (See Ref. [DMQN09] for definitions of these assumptions.) This is the first CK^+ secure AKE protocol without ROs based on a code-based problem.

As for factoring-based PKE, code-based PKE schemes are also κ-min-entropy KEM when the message space is larger than κ and messages are uniformly chosen randomness.

Remark 2. Bernstein et al. [BLP11] estimated the size of a public key of the original McEliece at about 2 Mbits for 128-bit security. If we employ "wild" McEliece by Bernstein et al. [BLP10] rather than the original McEliece PKE, the size of the public key is reduced to 750K bits. Our generic construction contains the public key of the KEM from the temporary key generation in the first round message. If the randomized McEliece PKE by Nojima et al. [NIKM08] is employed as the IND-CPA secure KEM, which is IND-CPA secure and requires the same size for the public key as the original, the communication complexity of the resultant AKE scheme is high. However, the way to construct an efficient and CK^+ secure AKE scheme from codes is an open problem.

3.8 Lattice-based

We also achieve new concrete AKE protocols based on the worst-case hardness of the (ring-)LWE problems derived from our generic constructions.

PKE schemes [PW08,Pei09,CHKP10,ABB10a,ABB10b,SSTX09,LPR10,MP12] which are IND-CCA secure in the StdM are easily converted into IND-CCA secure KEM schemes. Also, PRFs are obtained from one-way functions [Ajt96,MR07,LM06,PR06] and directly constructed from

the (ring-)LWE assumptions with sub-exponential parameters [BPR12]. Thus, by applying these building blocks to GC, we can obtain first CK^+ secure AKE protocols in the StdM under the (ring-)LWE assumption. Unfortunately, the obtained AKE protocols are still theoretical since these PKE schemes require huge keys, say, of the quadratic or cubic order of the security parameter, and thus, an efficient and direct construction of PRFs from the (ring-)LWE assumption with polynomial parameters has not yet been achieved.

As for factoring-based PKE, lattice-based PKE schemes are also κ-min-entropy KEM when the message space is larger than κ and messages are uniformly chosen randomness.

4 Generic ID-AKE Construction from IB-KEM without Random Oracles

In this section, we propose a generic construction of id-CK^+-secure ID-AKE from IB-KEM.

4.1 Preliminaries

Here, we recall the definition of IND-sID-CCA/CPA security (selective-ID IND-CCA/CPA security) for IB-KEM, and min-entropy of KEM keys as follows.

Definition 9 (Model for ID-based KEM Schemes). *A IB-KEM scheme consists of the following 4-tuple* $(\mathsf{MKeyGen}, \mathsf{KeyDer}, \mathsf{EnCap}, \mathsf{DeCap})$*:*

- $(mpk, msk) \leftarrow \mathsf{MKeyGen}(1^\kappa, r_g)$ *: a key generation algorithm which on inputs* 1^κ *and* $r_g \in \mathcal{RS}_G$*, where* κ *is the security parameter and* $\mathcal{RS}_G$ *is a randomness space, outputs master public key and secret key* (mpk, msk)*.*
- $dk \leftarrow \mathsf{KeyDer}(mpk, msk, ID, r_g)$ *: a key derivation algorithm which on inputs master public and secret keys* (mpk, msk)*, identity string* ID *and* $r_g \in \mathcal{RS}_G$*, where* $\mathcal{RS}_G$ *is a randomness space, outputs decapsulation key* dk *corresponding to* ID*.*
- $(K, CT) \leftarrow \mathsf{EnCap}_{mpk,ID}(r_e)$ *: an encryption algorithm which takes as inputs master public key* mpk*, identity string* ID*, and* $r_e \in \mathcal{RS}_E$*, outputs session key* $K \in \mathcal{KS}$ *and ciphertext* $CT \in \mathcal{CS}$*, where* $\mathcal{RS}_E$ *is a randomness space,* $\mathcal{KS}$ *is a session key space, and* $\mathcal{CS}$ *is a ciphertext space.*
- $K \leftarrow \mathsf{DeCap}_{dk}(CT)$ *: a decryption algorithm which takes as inputs decapsulation key* dk *and ciphertext* $CT \in \mathcal{CS}$*, outputs session key* $K \in \mathcal{KS}$*.*

Here, we recall the definition of IND-sID-CCA/CPA security (selective-ID IND-CCA/CPA security) for IB-KEM as follows.

Definition 10 (IND-CCA/CPA security for ID-based KEM). *A IB-KEM scheme is* (t, ϵ)*-IND-ID-CCA-secure for IB-KEM if the following property holds for security parameter* κ*; For any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *with a time-complexity at most* t*,* $\mathbf{Adv}^{\mathrm{id-ind-cca}} = |\Pr[(mpk, msk) \leftarrow \mathsf{MKeyGen}(1^\kappa, r_g);$ $(ID^*, state) \leftarrow \mathcal{A}_1^{\mathcal{DO}(\cdot,\cdot),\mathcal{KO}(msk,\cdot)}(mpk);\ b \leftarrow \{0,1\};\ (K_0^*, CT_0^*) \leftarrow \mathsf{EnCap}_{mpk,ID^*}(r);\ K_1^* \leftarrow \mathcal{K};$ $b' \leftarrow \mathcal{A}_2^{\mathcal{DO}(\cdot,\cdot),\mathcal{KO}(msk,\cdot)}(mpk,\ (K_b^*, CT_0^*), state);\ b' = b] - 1/2| \leq \epsilon$*, where* $\mathcal{DO}(ID, CT)$ *is the decryption oracle,* $\mathcal{KO}(msk, ID)$ *is the key derivation oracle,* $\mathcal{K}$ *is the space of session key,* $state$ *is state information which* $\mathcal{A}$ *wants to preserve from* $\mathcal{A}_1$ *to* $\mathcal{A}_2$ *and* $\mathcal{A}$ *runs in at most* t *steps.* $\mathcal{A}$ *cannot make query* $\mathcal{DO}(ID^*, CT_0^*)$*, and cannot make query* $\mathcal{KO}(msk, ID^*)$*.*

We say IB-KEM scheme is IND-CPA secure if adversary $\mathcal{A}$ *cannot access to the decryption oracle* $\mathcal{DO}$*.*

We say IB-KEM scheme is IND-sID-CCA/CPA secure if adversary $\mathcal{A}$ *outputs target identity string* ID^* *at the beginning of the game.*

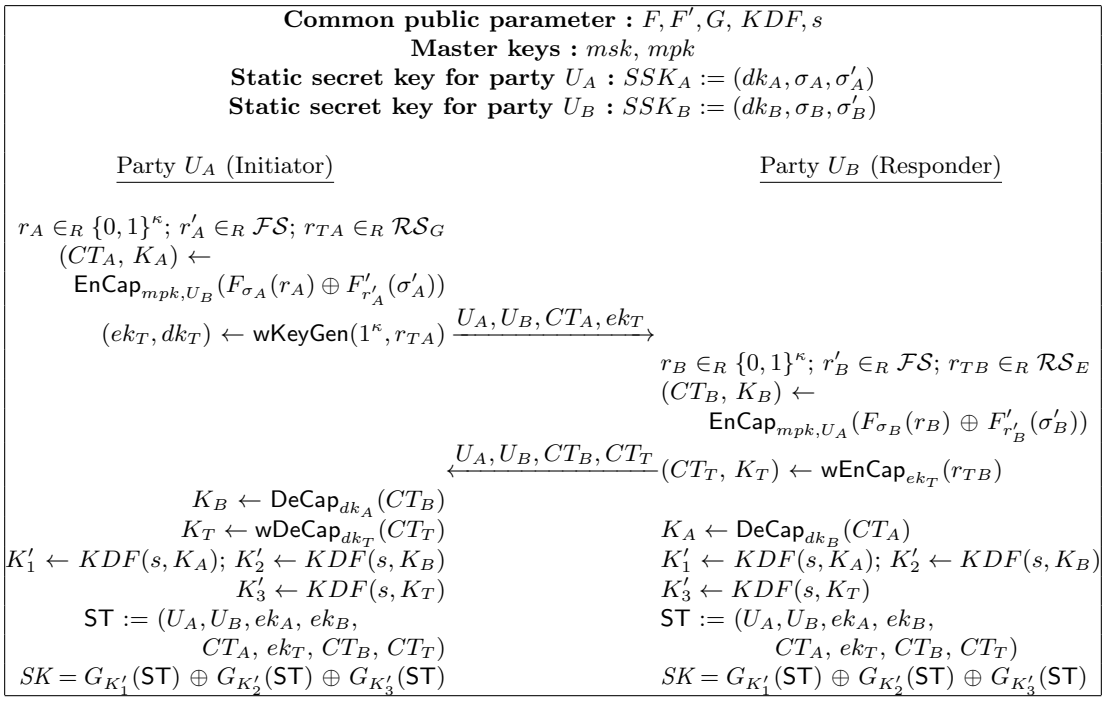

Fig. 2. Generic construction ID-GC

We define the notion of k-min-entropy for KEM keys as follows.

Definition 11 (Min-Entropy of KEM Keys). *A IB-KEM scheme is k-min-entropy IB-KEM if for any ID, mpk, distribution $D_{\mathcal{KS}}$ of variable K defined by $(K, CT) \leftarrow \mathsf{EnCap}_{mpk,ID}(r_e)$, distribution D_{pub} of public information and random $r_e \in \mathcal{RS}_E$, $H_\infty(D_{\mathcal{KS}}|D_{pub}) \geq k$ holds, where H_∞ denotes min-entropy.*

4.2 Construction

We propose a generic construction ID-GC of id-CK$^+$ secure ID-AKE without ROs from IND-sID-CCA secure IB-KEM, IND-CPA secure KEM, PRFs, and a KDF.

Design Principle To modify the generic construction GC of PKI-based AKE, we must remove static public keys from the protocol. Thus, we use IND-sID-CCA secure IB-KEM instead of IND-CCA secure KEM. In initialization, each party receives a static secret key based on the ID from the KGC. To send CT_A or CT_B each party encapsulates a KEM session key with the ID of the peer by using the encapsulation algorithm of IB-KEM. Hence, static public keys are not necessary. IND-CPA secure KEM for session-specific key generation can be still used in the ID-based setting because it is not necessary to put any information to static public keys in order to generate ek_T and CT_T.

Generic Construction ID-GC The protocol of ID-GC from IB-KEM (MKeyGen, KeyDer, EnCap, DeCap) and KEM (wKeyGen, wEnCap, wDeCap) is provided as follows.

Public Parameters. Let κ be the security parameter, $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and $G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$ be pseudo-random functions, where $\mathcal{FS}$ is the key space of PRFs ($|\mathcal{FS}| = \kappa$), $\mathcal{RS}_E$ is the randomness space of encapsulation algorithms, and $\mathcal{RS}_G$ is the randomness space

of key generation algorithms, and let $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ be a KDF a non-secret random salt $s \in Salt$, where $Salt$ is the salt space and $\mathcal{KS}$ is a space of KEM session keys. These are provided as some of the public parameters.

Master Secret and Public Keys. The KGC randomly selects $r \in \mathcal{RS}_G$, and generates master public and secret keys $(mpk, msk) \leftarrow \mathsf{MKeyGen}(1^\kappa, r)$, where $\mathcal{RS}_G$ is the randomness space of $\mathsf{MKeyGen}$.

Secret Key. For party U_P, the KGC randomly selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r' \in \mathcal{RS}_G$, and runs the key derivation algorithm $dk_P \leftarrow \mathsf{KeyDer}(mpk, msk, U_P, r')$, where $\mathcal{RS}_G$ is the randomness space of KeyDer. Party U_P's static secret key is $(dk_P, \sigma_P, \sigma'_P)$.

Key Exchange. Party U_A with secret and public keys $((dk_{A,1}, \sigma_A, \sigma'_A), ek_A)$ as the initiator, and party U_B with secret and public keys $((dk_{B,1}, \sigma_B, \sigma'_B), ek_B)$ as the responder, perform the following two-pass key exchange protocol.

1. Party U_A randomly chooses ephemeral secret keys $r_A \in_R \{0,1\}^\kappa$, $r'_A \in_R \mathcal{FS}$ and $r_{TA} \in \mathcal{RS}_G$. Party U_A computes $(CT_A, K_A) \leftarrow \mathsf{EnCap}_{mpk,U_B}(F_{\sigma_A}(r_A) \oplus F'_{r'_A}(\sigma'_A))$ and $(ek_T, dk_T) \leftarrow \mathsf{wKeyGen}(1^\kappa, r_{TA})$ and sends (U_A, U_B, CT_A, ek_T) to party U_B.
2. Upon receiving (U_A, U_B, CT_A, ek_T), party U_B chooses the ephemeral secret keys $r_B \in_R \{0,1\}^\kappa$, $r'_B \in_R \mathcal{FS}$ and $r_{TB} \in_R \mathcal{RS}_E$, computes $(CT_B, K_B) \leftarrow \mathsf{EnCap}_{mpk,U_A}(F_{\sigma_B}(r_B) \oplus F'_{r'_B}(\sigma'_B))$ and $(CT_T, K_T) \leftarrow \mathsf{wEnCap}_{ek_T}(r_{TB})$, and sends (U_A, U_B, CT_B, CT_T) to party U_A. Party U_B computes $K_A \leftarrow \mathsf{DeCap}_{dk_B}(CT_A)$, $K'_1 \leftarrow KDF(s, K_A)$, $K'_2 \leftarrow KDF(s, K_B)$ and $K'_3 \leftarrow KDF(s, K_T)$, sets the session transcript $\mathsf{ST} = (U_A, U_B, ek_A, ek_B, CT_A, ek_T, CT_B, CT_T)$ and the session key $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$, completes the session, and erases all session states.
3. Upon receiving (U_A, U_B, CT_B, CT_T), party U_A computes $K_B \leftarrow \mathsf{DeCap}_{dk_A}(CT_B)$, $K_T \leftarrow \mathsf{wDeCap}_{dk_T}(CT_T)$, $K'_1 \leftarrow KDF(s, K_A)$, $K'_2 \leftarrow KDF(s, K_B)$ and $K'_3 \leftarrow KDF(s, K_T)$, sets the session transcript $\mathsf{ST} = (U_A, U_B, ek_A, ek_B, CT_A, ek_T, CT_B, CT_T)$ and the session key $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$, completes the session, and erases all session states.

The session state of a session owned by U_A contains ephemeral secret keys (r_A, r_{TA}), encapsulated KEM key K_A and ad-hoc decryption key dk_T. Other information that is computed after receiving the message from the peer is immediately erased when the session key is established. Similarly, the session state of a session owned by U_B contains ephemeral secret keys (r_B, r_{TB}) and encapsulated KEM keys K_B and K_T.

Security The generic construction $\mathsf{ID\text{-}GC}$ is id-CK$^+$ secure ID-AKE without random oracles as follows.

Theorem 2. *If* $(\mathsf{MKeyGen}, \mathsf{KeyDer}, \mathsf{EnCap}, \mathsf{DeCap})$ *is IND-sID-CCA secure and* κ*-min-entropy IB-KEM,* $(\mathsf{wKeyGen}, \mathsf{wEnCap}, \mathsf{wDeCap})$ *is IND-CPA secure and* κ*-min-entropy KEM,* F, F', G *are PRFs, and* KDF *is a KDF, then ID-AKE construction* $\mathsf{ID\text{-}GC}$ *is* id-CK$^+$ *secure.*

The proof of Theorem 2 is shown in Appendix B. Here, we give an overview of the security proof.

In addition to the case of Theorem 1, we have to consider exposure of the master secret key according to Definition 3 in the id-CK$^+$ security model: Other cases are almost same as Theorem 1.

If msk is revealed, dk_A and dk_B are also revealed, and an adversary can know K_A and K_B. However, K_T is protected by the security of CT_T because dk_T and r_{TB} are not exposed because these values are generated only from ephemeral secret keys. We can prove the case by a similar way as Theorem 1.

4.3 Instantiations

In this section, from our generic construction ID-GC, we provide some ID-AKE protocols as concrete instantiations based on the lattices and pairings.

4.4 Lattice-based Instantiations

From our generic construction ID-GC in Section 4, we achieve new concrete ID-AKE protocols from the (ring-)LWE assumption.[5]

The existing IND-sID-CPA secure HIBE schemes [CHKP10,ABB10a,LPR10,LS12] in the StdM are easily converted into IND-sID-CCA secure IBE schemes by the CHK conversion [BCHK07] and they yield IND-sID-CCA secure IB-KEM schemes. Also, PRFs are obtained from one-way functions [Ajt96,MR07,LM06,PR06] under the (ring-)LWE assumption with standard parameters and a direct construction [BPR12] from the (ring-)LWE assumption with sub-exponential parameters. Applying our generic construction ID-GC with these building blocks, we can obtain first id-CK$^+$ secure ID-AKE protocols in the StdM under the (ring-)LWE assumption.

We finally note that the obtained IB-KEM scheme from [LS12] enjoys quasi-linear-time key-generation, encapsulation, and decapsulation.

4.5 Pairing-based Instantiations

From our generic construction ID-GC in Section 4, we can achieve various ID-AKE schemes as concrete instantiations based on the hardness of the bilinear DH (BDH) problem and its variants. For example, we can apply efficient IND-sID-CCA IB-KEM schemes to ID-GC from the decisional BDH (DBDH), decisional linear (DLIN) or the DBDH Inversion (DBDHI) [BB04] with the BCHK transformation [BCHK07].

We can easily show that these schemes are κ-min-entropy KEM. The KEM part of the Boneh-Boyen IBE consists of $e(g,\hat{g})^{\alpha\beta s} \in G_T$, where G_T is a finite cyclic bilinear group of order prime p, $e(g,\hat{g})^{\alpha\beta}$ is part of public parameters, and s is uniformly chosen randomness, and $|s|$ is larger than κ. Thus, $e(g,\hat{g})^{\alpha\beta s}$ has min-entropy larger than κ.

The significant advantage of our instantiations is security in the StdM. Most of previous ID-AKE schemes [CCS07,HC09,FG10,FSU10] are proved in the ROM. Moreover, though the generic construction of ID-AKE in [BCGNP08,BCGNP09] is secure in the StdM, its security is not proved in the id-CK$^+$ model. Thus, pairing-based ID-AKE schemes from ID-GC are first id-CK$^+$ secure schemes in the StdM.

For example, our scheme can be instantiated with the Boyen-Mei-Waters ID-based KEM [BMW05] as an IND-sID-CCA KEM under the DBDH assumption, and with the ElGamal KEM as an IND-CPA KEM under the DDH assumption. Communication complexity (for two parties) of this instantiation is $8|p|$, where $|p|$ is the length of a group element. Computational complexity (for two parties) of this instantiation is 8 pairings and 14 regular exponentiations (all symmetric operations such as hash function/KDF/PRF and multiplications are ignored). We show a comparison between this instantiation and previous schemes in Table 4.

[5] The hardness of the (ring-)LWE problems are reduced to the worst-case hardness of the (ideal) lattice problems.

Table 4. Comparison of previous pairing-based schemes and an instantiation of our scheme

	Model	Resource	Assumption	Computation (#[pairing,exp.])	Communication complexity	
[CCS07]	id-BR & KCI & mFS	ROM	BDH	[4, 6]	$2\|p\|$	512
[HC09]	id-eCK	ROM	BDH	[4, 6]	$2\|p\|$	512
[FG10]†	id-CK & KCI & mFS	ROM	strong DH	[0, 8]	$4\|p\|$	1024
[FSU10]	id-eCK	ROM	gap BDH	[4, 6]	$2\|p\|$	512
[BCGNP09]	id-CK & KCI	StdM	DBDH	[8, 10]	$6\|p\|$	1536
Ours	id-CK$^+$	StdM	DBDH & DDH	[8, 14]	$8\|p\|$	2048

For concreteness the expected ciphertext overhead for a 128-bit implementation is also given. Note that computational costs are estimated without any pre-computation technique.
† Non-pairing protocol

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient Lattice (H)IBE in the Standard Model. In *EUROCRYPT 2010*, pages 553–572, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. In *CRYPTO 2010*, pages 98–115, 2010.

[Ajt96] Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC 1996*, pages 99–108, 1996. See also ECCC TR96-007.

[BB04] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *EUROCRYPT 2004*, pages 223–238, 2004. See also Cryptology ePrint Archive-2004/172.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO 2004*, pages 41–55, 2004.

[BCGNP08] Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. Efficient One-Round Key Exchange in the Standard Model. In *ACISP 2008*, pages 69–83, 2008.

[BCGNP09] Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. One-round key exchange in the standard model. In *IJACT 1(3)*, pages 181–199, 2009.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *SIAM J. Comput. 36(5)*, pages 1301–1328, 2007.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*, pages 213–229, 2001.

[BGN11] Colin Boyd and Juan Manuel González Nieto. On Forward Secrecy in One-Round Key Exchange. In *IMA Int. Conf. 2011*, pages 451–468, 2011.

[BLP10] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Wild McEliece. In *SAC 2010*, pages 143–158, 2010.

[BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller Decoding Exponents: Ball-Collision Decoding. In *CRYPTO 2011*, pages 743–760, 2011.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM Conference on Computer and Communications Security 2005*, pages 320–329, 2005.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom Functions and Lattices. In *EUROCRYPT 2012*, pages 719–737, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *CRYPTO 1993*, pages 232–249, 1993.

[CCS07] Liqun Chen, Zhaohui Cheng, and Nigel P. Smart. Identity-based Key Agreement Protocols From Pairings. In *Int. J. Inf. Sec. 6(4)*, pages 213–241, 2007.

[CF11] Cas J. F. Cremers and Michele Feltz. One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability. In *Cryptology ePrint Archive: 2011/300*, 2011.

[CF12] Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal. In *ESORICS 2012*, pages 734–751, 2012.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In *STOC 1998*, pages 131–140, 1998.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. In *EUROCRYPT 2010*, pages 523–552, 2010.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT 2001*, pages 453–474, 2001.

[CMJ09] Benoît Chevallier-Mames and Marc Joye. Chosen-Ciphertext Secure RSA-Type Cryptosystems. In *ProvSec 2009*, pages 32–46, 2009.

[Cre09] Cas J. F. Cremers. Session-state Reveal Is Stronger Than Ephemeral Key Reveal: Attacking the NAXOS Authenticated Key Exchange Protocol. In *ACNS 2009*, pages 20–33, 2009.

[Cre11] Cas J. F. Cremers. Examining Indistinguishability-Based Security Models for Key Exchange Protocols: The case of CK, CK-HMQV, and eCK. In *ASIACCS 2011*, pages 80–91, 2011.

[CS98] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO 1998*, pages 13–25, 1998.

[CS04] Ronald Cramer and Victor Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. In *SIAM Journal on Computing 33*, pages 167–226, 2004.

[Dam91] Ivan Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *CRYPTO 1991*, pages 445–456, 1991.

[DMQN09] Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. A CCA2 Secure Public Key Encryption Scheme Based on the McEliece Assumptions in the Standard Model. In *CT-RSA 2009*, pages 240–251, 2009.

[DSGKM12] Dana Dachman-Soled, Rosario Gennaro, Hugo Krawczyk, and Tal Malkin. Computational Extractors and Pseudorandomness. In *TCC 2012*, pages 383–403, 2012.

[FG10] Dario Fiore and Rosario Gennaro. Making the Diffie-Hellman Protocol Identity-Based. In *CT-RSA 2010*, pages 165–178, 2010.

[FSU10] Atsushi Fujioka, Koutarou Suzuki, and Berkant Ustaoglu. Ephemeral Key Leakage Resilient and Efficient ID-AKEs That Can Share Identities, Private and Master Keys. In *Pairing 2010*, pages 187–205, 2010.

[FSXY12] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices. In *Public Key Cryptography 2012*, pages 467–484, 2012.

[GBGNM09] M. Choudary Gorantla, Colin Boyd, Juan Manuel González Nieto, and Mark Manulis. Generic One Round Group Key Exchange in the Standard Model. In *ICISC 2009*, pages 1–15, 2009.

[GKR10] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Okamoto-Tanaka Revisited: Fully Authenticated Diffie-Hellman with Minimal Overhead. In *ACNS 2010*, pages 309–328, 2010.

[GS04] Rosario Gennaro and Victor Shoup. A Note on An Encryption Scheme of Kurosawa and Desmedt. In *Cryptology ePrint Archive: 2004/194*, 2004.

[HC09] Hai Huang and Zhenfu Cao. An ID-based Authenticated Key Exchange Protocol Based on Bilinear Diffie-Hellman Problem. In *ASIACCS 2009*, pages 333–342, 2009.

[HJKS10] Kristiyan Haralambiev, Tibor Jager, Eike Kiltz, and Victor Shoup. Simple and Efficient Public-Key Encryption from Computational Diffie-Hellman in the Standard Model. In *Public Key Cryptography 2010*, pages 1–18, 2010.

[HK08] Goichiro Hanaoka and Kaoru Kurosawa. Efficient Chosen Ciphertext Secure Public Key Encryption under the Computational Diffie-Hellman Assumption. In *ASIACRYPT 2008*, pages 308–325, 2008.

[HK09a] Dennis Hofheinz and Eike Kiltz. Practical Chosen Ciphertext Secure Encryption from Factoring. In *EUROCRYPT 2009*, pages 313–332, 2009.

[HK09b] Dennis Hofheinz and Eike Kiltz. The Group of Signed Quadratic Residues and Applications. In *CRYPTO 2009*, pages 637–653, 2009.

[JKL04] Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. One-Round Protocols for Two-Party Authenticated Key Exchange. In *ACNS 2004*, pages 220–232, 2004.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *CRYPTO 2004*, pages 426–442, 2004.

[Kil07] Eike Kiltz. Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. In *Public Key Cryptography 2007*, pages 282–297, 2007.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive Trapdoor Functions and Chosen-Ciphertext Security. In *EUROCRYPT 2010*, pages 673–692, 2010.

[Kra05] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO 2005*, pages 546–566, 2005.

[Kra10] Hugo Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In *CRYPTO 2010*, pages 631–648, 2010.

[LLM07] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger Security of Authenticated Key Exchange. In *ProvSec 2007*, pages 1–16, 2007.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized Compact Knapsacks are Collision Resistant. In *ICALP (2) 2006*, pages 144–155, 2006.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT 2010*, pages 1–23, 2010.
[LS12] Adeline Langlois and Damien Stehle. Hardness of decision (R)LWE for any modulus. In *Cryptology ePrint Archive: 2012/091*, 2012.
[McE78] Robert J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. In *Deep Space Network progress Report*, 1978.
[MLLJ11] Qixiang Mei, Bao Li, Xianhui Lu, and Dingding Jia. Chosen Ciphertext Secure Encryption under Factoring Assumption Revisited. In *Public Key Cryptography 2011*, pages 210–227, 2011.
[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *EUROCRYPT 2012*, pages 700–718, 2012.
[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. Preliminary version in *FOCS 2004*, 2004.
[Nao03] Moni Naor. On Cryptographic Assumptions and Challenges. In *CRYPTO 2003*, pages 96–109, 2003.
[NIKM08] Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the McEliece cryptosystem without random oracles. *Designs, Codes and Cryptography*, 49(1-3):289–305, 2008.
[Oka07] Tatsuaki Okamoto. Authenticated Key Exchange and Key Encapsulation in the Standard Model. In *ASIACRYPT 2007*, pages 474–484, 2007.
[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC 2009*, pages 333–342, 2009.
[PR06] Chris Peikert and Alon Rosen. Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. In *TCC 2006*, pages 145–166, 2006.
[PW08] Chris Peikert and Brent Waters. Lossy Trapdoor Functions and Their Applications. In *STOC 2008*, pages 187–196, 2008.
[SEVB10] Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. A New Security Model for Authenticated Key Agreement. In *SCN 2010*, pages 219–234, 2010.
[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient Public Key Encryption Based on Ideal Lattices. In *ASIACRYPT 2009*, pages 617–635, 2009.
[Yon12] Kazuki Yoneyama. One-Round Authenticated Key Exchange with Strong Forward Secrecy in the Standard Model against Constrained Adversary. In *IWSEC 2012*, pages 69–86, 2012.
[Yon13a] Kazuki Yoneyama. Generic Construction of Two-Party Round-Optimal Attribute-Based Authenticated Key Exchange without Random Oracles. In *IEICE Transactions 96-A(6)*, pages 1112–1123, 2013.
[Yon13b] Kazuki Yoneyama. One-Round Authenticated Key Exchange with Strong Forward Secrecy in the Standard Model against Constrained Adversary. In *IEICE Transactions 96-A(6)*, pages 1124–1138, 2013.

A Proof of Theorem 1

In the experiment of CK^+ security, we suppose that sid^* is the session identity for the test session, and that there are N users and at most ℓ sessions are activated. Let κ be the security parameter, and let $\mathcal{A}$ be a PPT (in κ) bounded adversary. Suc denotes the event that $\mathcal{A}$ wins. We consider the following events that cover all cases of the behavior of $\mathcal{A}$.

- Let E_1 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the initiator and the static secret key of the initiator is given to $\mathcal{A}$.
- Let E_2 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the initiator and the ephemeral secret key of sid^* is given to $\mathcal{A}$.
- Let E_3 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the responder and the static secret key of the responder is given to $\mathcal{A}$.
- Let E_4 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the responder and the ephemeral secret key of sid^* is given to $\mathcal{A}$.
- Let E_5 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and both static secret keys of the initiator and the responder are given to $\mathcal{A}$.
- Let E_6 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and both ephemeral secret keys of sid^* and $\overline{\mathsf{sid}^*}$ are given to $\mathcal{A}$.

– Let E_7 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and the static secret key of the owner of sid^* and the ephemeral secret key of $\overline{\mathsf{sid}}^*$ are given to $\mathcal{A}$.
– Let E_8 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and the ephemeral secret key of sid^* and the static secret key of the owner of $\overline{\mathsf{sid}}^*$ are given to $\mathcal{A}$.

To finish the proof, we investigate events $E_i \wedge Suc$ $(i = 1, \ldots, 8)$ that cover all cases of event Suc.

A.1 Event $E_1 \wedge Suc$

We change the interface of oracle queries and the computation of the session key. These instances are gradually changed over seven hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, the session key in the test session does not contain information of the bit b. Thus, the adversary clearly only output a random guess. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_6$ and the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$ by $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_i)$.

Hybrid experiment $\mathbf{H}_0$: This experiment denotes the real experiment for CK^+ security and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as the advantage of the real experiment.

Hybrid experiment $\mathbf{H}_1$: In this experiment, if session identities in two sessions are identical, the experiment halts.

When two ciphertexts from different randomness are identical and two public keys from different randomness are identical, session identities in two sessions are also identical. In the IND-CCA secure KEM, such an event occurs with negligible probability. Thus, $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_1) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_0)| \leq negl$.

Hybrid experiment $\mathbf{H}_2$: In this experiment, the experiment selects a party U_A and integer $i \in [1, \ell]$ randomly in advance. If $\mathcal{A}$ poses Test query to a session except i-th session of U_A, the experiment halts.

Since guess of the test session matches with $\mathcal{A}$'s choice with probability $1/N^2\ell$, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_2) \geq 1/N^2\ell \cdot \mathbf{Adv}(\mathcal{A}, \mathbf{H}_1)$.

Hybrid experiment $\mathbf{H}_3$: In this experiment, the computation of (CT_A^*, K_A^*) in the test session is changed. Instead of computing $(CT_A^*, K_A^*) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus F'_{r'_A}(\sigma'_A))$, it is changed as $(CT_A^*, K_A^*) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus RF(\sigma'_A))$, where we suppose that U_B is the intended partner of U_A in the test session.

We construct a distinguisher $\mathcal{D}$ between PRF $F^* : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$ and a random function RF from $\mathcal{A}$ in $\mathbf{H}_2$ or $\mathbf{H}_3$. $\mathcal{D}$ performs the following steps.

Setup. $\mathcal{D}$ chooses pseudo-random functions $F : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$ and $G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. Also, $\mathcal{D}$ embeds F^* into F'. These are provided as a part of the public parameters. Also, $\mathcal{D}$ sets all N users' static secret and public keys. $\mathcal{D}$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r \in_R \mathcal{RS}_G$, and runs $(ek_P, dk_P) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$. Party U_P's SSK and SPK are $((dk_P, \sigma_P, \sigma'_P), ek_P)$. U_A's static key $(dk_A, \sigma_A, \sigma'_A)$ is given to $\mathcal{A}$.

Next, $\mathcal{D}$ sets the ephemeral public key of i-th session of U_A (i.e., the test session) as follows: $\mathcal{D}$ selects ephemeral secret keys $r_A^* \in \{0,1\}^\kappa$, $r_A'^* \in \mathcal{FS}$ and $r_{TA}^* \in \mathcal{RS}_G$ randomly. Then, $\mathcal{D}$

poses σ'_A to his oracle (i.e., F^* or a random function RF) and obtains $x \in \mathcal{RS}_E$. $\mathcal{D}$ computes $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r^*_A) \oplus x)$ and $(dk^*_T, ek^*_T) \leftarrow \mathsf{KeyGen}(r^*_{TA})$, and sets the ephemeral public key (CT^*_A, ek^*_T) of i-th session of U_A.

Simulation. $\mathcal{D}$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{D}$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: If $P = A$ and the session is i-th session of U_A, $\mathcal{D}$ returns the ephemeral public key (CT^*_A, ek^*_T) computed in the setup. Otherwise, $\mathcal{D}$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: $\mathcal{D}$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{D}$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Otherwise, $\mathcal{D}$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:
   (a) If the session sid is not completed, $\mathcal{D}$ returns an error message.
   (b) Otherwise, $\mathcal{D}$ returns the recorded value SK.
5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{D}$ responds the ephemeral secret key and intermediate computation results of sid as the definition. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.
6. $\mathsf{Corrupt}(U_P)$: $\mathcal{D}$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{D}$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess $b' = 0$, $\mathcal{D}$ outputs that the oracle is the PRF F^*. Otherwise, $\mathcal{D}$ outputs that the oracle is a random function RF.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_2$ if the oracle is the PRF F^*. Otherwise, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_3$. Thus, if the advantage of $\mathcal{D}$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_3) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_2)| \leq negl$.

Hybrid experiment $\mathbf{H}_4$: In this experiment, the computation of K^*_A in the test session is changed again. Instead of computing $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus RF(\sigma'_A))$, it is changed as choosing $K^*_A \leftarrow \mathcal{KS}$ randomly, where we suppose that U_B is the intended partner of U_A in the test session.

We construct an IND-CCA adversary $\mathcal{S}$ from $\mathcal{A}$ in $\mathbf{H}_3$ or $\mathbf{H}_4$. $\mathcal{S}$ performs the following steps.

Init. $\mathcal{S}$ receives the public key ek^* as a challenge.

Setup. $\mathcal{S}$ chooses pseudo-random functions $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and $G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. These are provided as a part of the public parameters. Also, $\mathcal{S}$ sets all N users' static secret and public keys except U_B. $\mathcal{S}$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r \in_R \mathcal{RS}_G$, and runs $(ek_P, dk_P) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$. Party U_P's SSK and SPK are $((dk_P, \sigma_P, \sigma'_P), ek_P)$. U_A's static key $(dk_A, \sigma_A, \sigma'_A)$ is given to $\mathcal{A}$.

Next, $\mathcal{S}$ sets ek^* as the static public key of U_B. Also, $\mathcal{S}$ receives the challenge (K^*, CT^*) from the challenger.

Simulation. $\mathcal{S}$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{S}$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: If $P = A$ and the session is i-th session of U_A, $\mathcal{S}$ computes ek_T obeying the protocol and returns the ephemeral public key (CT^*, ek_T). Otherwise, $\mathcal{S}$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: If $\bar{P} = B$ and $CT_P \neq CT^*$, $\mathcal{S}$ poses CT_P to the decryption oracle, obtains K_P, computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Else if $\bar{P} = B$ and $CT_P = CT^*$, $\mathcal{S}$ sets $K_P = K^*$, computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{S}$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records (Π, U_P, $U_{\bar{P}}$, (CT_P, ek_T), ($CT_{\bar{P}}$, CT_T)) as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{S}$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Else if $P = A$ and the session is i-th session of U_A, $\mathcal{S}$ computes the session key SK obeying the protocol except that $K_A^* = K^*$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{S}$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:
   (a) If the session sid is not completed, $\mathcal{S}$ returns an error message.
   (b) Otherwise, $\mathcal{S}$ returns the recorded value SK.
5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{S}$ responds the ephemeral secret key and intermediate computation results of sid as the definition. If the owner of sid is U_B, $\mathcal{S}$ poses ciphertexts received by U_B to the decryption oracle and can simulate all intermediate computation results. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.
6. $\mathsf{Corrupt}(U_P)$: $\mathcal{S}$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{S}$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess b', $\mathcal{S}$ outputs b'.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{S}$ is same as the experiment $\mathbf{H}_3$ if the challenge is (K_1^*, CT_0^*). Otherwise, the simulation by $\mathcal{S}$ is same as the experiment $\mathbf{H}_4$. Also, both K_A^* in two experiments have κ-min-entropy because $(\mathsf{KeyGen}, \mathsf{EnCap}, \mathsf{DeCap})$ is κ-min-entropy KEM. Thus, if the advantage of $\mathcal{S}$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_4) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_3)| \leq negl$.

Hybrid experiment $\mathbf{H}_5$: In this experiment, the computation of $K_1'^*$ in the test session is changed. Instead of computing $K_1'^* \leftarrow KDF(s, K_A^*)$, it is changed as choosing $K_1'^* \in \mathcal{FS}$ randomly.

Since K_A^* is randomly chosen in $\mathbf{H}_4$, it has sufficient min-entropy. Thus, by the definition of the KDF, $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_5) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_4)| \leq negl$.

Hybrid experiment $\mathbf{H}_6$: In this experiment, the computation of SK in the test session is changed. Instead of computing $SK = G_{K_1'}(\mathsf{ST}) \oplus G_{K_2'}(\mathsf{ST}) \oplus G_{K_3'}(\mathsf{ST})$, it is changed as

$SK = x \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$ where $x \in \{0,1\}^\kappa$ is chosen randomly and we suppose that U_B is the intended partner of U_A in the test session.

We construct a distinguisher $\mathcal{D}'$ between PRF $F^* : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$ and a random function RF from $\mathcal{A}$ in $\mathbf{H}_5$ or $\mathbf{H}_6$. $\mathcal{D}'$ performs the following steps.

Setup. $\mathcal{D}'$ chooses pseudo-random functions $F : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, $F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, sets $G = F^*$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. These are provided as a part of the public parameters. Also, $\mathcal{D}'$ sets all N users' static secret and public keys. $\mathcal{D}'$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r \in_R \mathcal{RS}_G$, and runs $(ek_P, dk_P) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$. Party U_P's SSK and SPK are $((dk_P, \sigma_P, \sigma'_P), ek_P)$. U_A's static key $(dk_A, \sigma_A, \sigma'_A)$ is given to $\mathcal{A}$.

Simulation. $\mathcal{D}'$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{D}'$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: $\mathcal{D}'$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: $\mathcal{D}'$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{D}'$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Else if $P = A$ and the session is i-th session of U_A, $\mathcal{D}'$ poses ST to his oracle (i.e., F^* or a random function RF), obtains $x \in \{0,1\}^\kappa$, computes the session key $SK = x \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{D}'$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:
   (a) If the session sid is not completed, $\mathcal{D}'$ returns an error message.
   (b) Otherwise, $\mathcal{D}'$ returns the recorded value SK.
5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{D}'$ responds the ephemeral secret key and intermediate computation results of sid as the definition. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.
6. $\mathsf{Corrupt}(U_P)$: $\mathcal{D}'$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{D}'$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess $b' = 0$, $\mathcal{D}'$ outputs that the oracle is the PRF F^*. Otherwise, $\mathcal{D}'$ outputs that the oracle is a random function RF.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{D}'$ is same as the experiment $\mathbf{H}_5$ if the oracle is the PRF F^*. Otherwise, the simulation by $\mathcal{D}'$ is same as the experiment $\mathbf{H}_6$. Thus, if the advantage of $\mathcal{D}'$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_6) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_5)| \leq negl$.

In $\mathbf{H}_6$, the session key in the test session is perfectly randomized. Thus, $\mathcal{A}$ cannot obtain any advantage from Test query.

Therefore, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_6) = 0$ and $\Pr[E_1 \wedge Suc]$ is negligible.

A.2 Event $E_2 \wedge Suc$

The proof in this case is essentially same as the event $E_1 \wedge Suc$. There is a difference in the experiment $\mathbf{H}_3$. In the event $E_1 \wedge Suc$, instead of computing $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus$

$F'_{r'_A}(\sigma'_A)$), it is changed as $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus RF(\sigma'_A))$, where we suppose that U_B is the intended partner of U_A in the test session. In the event $E_2 \wedge Suc$, it is changed as $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(RF(r_A) \oplus F'_{r'_A}(\sigma'_A))$. Since $\mathcal{A}$ cannot obtain σ_A by the freshness definition in this event, we can construct a distinguisher $\mathcal{D}$ from $\mathcal{A}$ in the similar manner in the proof of the event $E_1 \wedge Suc$.

A.3 Event $E_3 \wedge Suc$

The proof in this case is essentially same as the event $E_1 \wedge Suc$. There is differences in experiments $\mathbf{H}_3$ and $\mathbf{H}_4$. In $\mathbf{H}_3$ of the event $E_1 \wedge Suc$, instead of computing $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus F'_{r'_A}(\sigma'_A))$, it is changed as $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus RF(\sigma'_A))$, where we suppose that U_B is the intended partner of U_A in the test session. In $\mathbf{H}_3$ of the event $E_3 \wedge Suc$, instead of computing $(CT^*_B, K^*_B) \leftarrow \mathsf{EnCap}_{ek_A}(F_{\sigma_B}(r_B) \oplus F'_{r'_B}(\sigma'_B))$, it is changed as $(CT^*_B, K^*_B) \leftarrow \mathsf{EnCap}_{ek_A}(F_{\sigma_B}(r_B) \oplus RF(\sigma'_B))$. In $\mathbf{H}_4$ of the event $E_1 \wedge Suc$, instead of computing $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus RF(\sigma'_A))$, it is changed as choosing $K^*_A \leftarrow \mathcal{KS}$ randomly. In $\mathbf{H}_4$ of the event $E_3 \wedge Suc$, instead of computing $(CT^*_B, K^*_B) \leftarrow \mathsf{EnCap}_{ek_A}(F_{\sigma_B}(r_B) \oplus RF(\sigma'_B))$, it is changed as choosing $K^*_B \leftarrow \mathcal{KS}$ randomly. Since $\mathcal{A}$ cannot obtain σ_B by the freshness definition in this event, we can construct a distinguisher $\mathcal{D}$ from $\mathcal{A}$ in the similar manner in the proof of the event $E_1 \wedge Suc$.

A.4 Event $E_4 \wedge Suc$

The proof in this case is essentially same as the event $E_2 \wedge Suc$. There is differences in experiments $\mathbf{H}_3$ and $\mathbf{H}_4$. In $\mathbf{H}_3$ of the event $E_2 \wedge Suc$, instead of computing $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(F_{\sigma_A}(r_A) \oplus F'_{r'_A}(\sigma'_A))$, it is changed as $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(RF(r_A) \oplus F'_{r'_A}(\sigma'_A))$, where we suppose that U_B is the intended partner of U_A in the test session. In $\mathbf{H}_3$ of the event $E_3 \wedge Suc$, instead of computing $(CT^*_B, K^*_B) \leftarrow \mathsf{EnCap}_{ek_A}(F_{\sigma_B}(r_B) \oplus F'_{r'_B}(\sigma'_B))$, it is changed as $(CT^*_B, K^*_B) \leftarrow \mathsf{EnCap}_{ek_A}(RF(r_B) \oplus F'_{r'_B}(\sigma'_B))$. In $\mathbf{H}_4$ of the event $E_2 \wedge Suc$, instead of computing $(CT^*_A, K^*_A) \leftarrow \mathsf{EnCap}_{ek_B}(RF(r_A) \oplus F'_{r'_A}(\sigma'_A))$, it is changed as choosing $K^*_A \leftarrow \mathcal{KS}$ randomly. In $\mathbf{H}_4$ of the event $E_3 \wedge Suc$, instead of computing $(CT^*_B, K^*_B) \leftarrow \mathsf{EnCap}_{ek_A}(RF(r_B) \oplus F'_{r'_B}(\sigma'_B))$, it is changed as choosing $K^*_B \leftarrow \mathcal{KS}$ randomly. Since $\mathcal{A}$ cannot obtain σ_B by the freshness definition in this event, we can construct a distinguisher $\mathcal{D}$ from $\mathcal{A}$ in the similar manner in the proof of the event $E_1 \wedge Suc$.

A.5 Event $E_5 \wedge Suc$

We change the interface of oracle queries and the computation of the session key. These instances are gradually changed over six hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, the session key in the test session does not contain information of the bit b. Thus, the adversary clearly only output a random guess. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_5$ and the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$ by $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_i)$.

Hybrid experiment $\mathbf{H}_0$: This experiment denotes the real experiment for CK^+ security and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as the advantage of the real experiment.

Hybrid experiment $\mathbf{H}_1$: In this experiment, if session identities in two sessions are identical, the experiment halts.

By the same as the event $E_1 \wedge Suc$, $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_1) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_0)| \leq negl$.

Hybrid experiment $\mathbf{H}_2$: In this experiment, the experiment selects a party U_A and integer $i \in [1, \ell]$ randomly in advance. If $\mathcal{A}$ poses Test query to a session except i-th session of U_A, the experiment halts.

By the same as the event $E_1 \wedge Suc$, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_2) \geq 1/N^2\ell \cdot \mathbf{Adv}(\mathcal{A}, \mathbf{H}_1)$.

Hybrid experiment $\mathbf{H}_3$: In this experiment, the computation of K_T^* in the test session is changed. Instead of computing $(CT_T^*, K_T^*) \leftarrow \mathsf{wEnCap}_{ek_T}(r_{TB})$, it is changed as choosing $K_T^* \leftarrow \mathcal{KS}$ randomly, where we suppose that U_B is the intended partner of U_A in the test session.

We construct an IND-CPA adversary $\mathcal{S}$ from $\mathcal{A}$ in $\mathbf{H}_2$ or $\mathbf{H}_3$. $\mathcal{S}$ performs the following steps.

Init. $\mathcal{S}$ receives the public key ek^* as a challenge.

Setup. $\mathcal{S}$ chooses pseudo-random functions $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and $G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. These are provided as a part of the public parameters. Also, $\mathcal{S}$ sets all N users' static secret and public keys. $\mathcal{S}$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r \in_R \mathcal{RS}_G$, and runs $(ek_P, dk_P) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$. Party U_P's SSK and SPK are $((dk_P, \sigma_P, \sigma'_P), ek_P)$. U_A's static key $(dk_A, \sigma_A, \sigma'_A)$ and U_B's static key $(dk_B, \sigma_B, \sigma'_B)$ are given to $\mathcal{A}$.

Next, $\mathcal{S}$ receives the challenge (K^*, CT^*) from the challenger.

Simulation. $\mathcal{S}$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{S}$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: If $P = A$ and the session is i-th session of U_A, $\mathcal{S}$ computes CT_A obeying the protocol and returns the ephemeral public key (CT_A, ek^*). Otherwise, $\mathcal{S}$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: If $\bar{P} = B$, $\mathcal{S}$ computes $CT_{\bar{P}}$ and the session key SK obeying the protocol except that $K_T = K^*$, returns the ephemeral public key $(CT_{\bar{P}}, CT^*)$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{S}$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{S}$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Else if $P = A$ and the session is i-th session of U_A, $\mathcal{S}$ computes the session key SK obeying the protocol except that $K_T^* = K^*$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{S}$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:
   (a) If the session sid is not completed, $\mathcal{S}$ returns an error message.
   (b) Otherwise, $\mathcal{S}$ returns the recorded value SK.
5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{S}$ responds the ephemeral secret key and intermediate computation results of sid as the definition. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.

6. $\mathsf{Corrupt}(U_P)$: $\mathcal{S}$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{S}$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess b', $\mathcal{S}$ outputs b'.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{S}$ is same as the experiment $\mathbf{H}_2$ if the challenge is (K_1^*, CT_0^*). Otherwise, the simulation by $\mathcal{S}$ is same as the experiment $\mathbf{H}_3$. Also, both K_T^* in two experiments have κ-min-entropy because $(\mathsf{wKeyGen}, \mathsf{wEnCap}, \mathsf{wDeCap})$ is κ-min-entropy KEM. Thus, if the advantage of $\mathcal{S}$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_3) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_2)| \leq negl$.

Hybrid experiment $\mathbf{H}_4$: In this experiment, the computation of $K_3'^*$ in the test session is changed. Instead of computing $K_3'^* \leftarrow KDF(s, K_T^*)$, it is changed as choosing $K_3'^* \in \mathcal{FS}$ randomly.

Since K_T^* is randomly chosen in $\mathbf{H}_3$, it has sufficient min-entropy. Thus, by the definition of the KDF, $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_4) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_3)| \leq negl$.

Hybrid experiment $\mathbf{H}_5$: In this experiment, the computation of SK in the test session is changed. Instead of computing $SK = G_{K_1'}(\mathsf{ST}) \oplus G_{K_2'}(\mathsf{ST}) \oplus G_{K_3'}(\mathsf{ST})$, it is changed as $SK = G_{K_1'}(\mathsf{ST}) \oplus G_{K_2'}(\mathsf{ST}) \oplus x$ where $x \in \{0,1\}^\kappa$ is chosen randomly and we suppose that U_B is the intended partner of U_A in the test session.

We construct a distinguisher $\mathcal{D}'$ between PRF $F^* : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$ and a random function RF from $\mathcal{A}$ in $\mathbf{H}_4$ or $\mathbf{H}_5$. $\mathcal{D}'$ performs the following steps.

Setup. $\mathcal{D}'$ chooses pseudo-random functions $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and sets $G = F^*$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. These are provided as a part of the public parameters. Also, $\mathcal{D}'$ sets all N users' static secret and public keys. $\mathcal{D}'$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r \in_R \mathcal{RS}_G$, and runs $(ek_P, dk_P) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$. Party U_P's SSK and SPK are $((dk_P, \sigma_P, \sigma'_P), ek_P)$. U_A's static key $(dk_A, \sigma_A, \sigma'_A)$ and U_B's static key $(dk_B, \sigma_B, \sigma'_B)$ are given to $\mathcal{A}$.

Simulation. $\mathcal{D}'$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{D}'$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: $\mathcal{D}'$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: If $P = A$ and the session is partnered with i-th session of U_A, $\mathcal{D}'$ poses ST to his oracle (i.e., F^* or a random function RF), obtains $x \in \{0,1\}^\kappa$, computes the session key $SK = G_{K_1'}(\mathsf{ST}) \oplus G_{K_2'}(\mathsf{ST}) \oplus x$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{D}'$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{D}'$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Else if $P = A$ and the session is i-th session of U_A, $\mathcal{D}'$ poses ST to his oracle (i.e., F^* or a random function RF), obtains $x \in \{0,1\}^\kappa$, computes the session key $SK = G_{K_1'}(\mathsf{ST}) \oplus G_{K_2'}(\mathsf{ST}) \oplus x$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{D}'$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:

(a) If the session sid is not completed, $\mathcal{D}'$ returns an error message.
(b) Otherwise, $\mathcal{D}'$ returns the recorded value SK.

5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{D}'$ responds the ephemeral secret key and intermediate computation results of sid as the definition. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.
6. $\mathsf{Corrupt}(U_P)$: $\mathcal{D}'$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{D}'$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess $b' = 0$, $\mathcal{D}'$ outputs that the oracle is the PRF F^*. Otherwise, $\mathcal{D}'$ outputs that the oracle is a random function RF.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{D}'$ is same as the experiment $\mathbf{H}_4$ if the oracle is the PRF F^*. Otherwise, the simulation by $\mathcal{D}'$ is same as the experiment $\mathbf{H}_5$. Thus, if the advantage of $\mathcal{D}'$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_5) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_4)| \leq negl$.

In $\mathbf{H}_5$, the session key in the test session is perfectly randomized. Thus, $\mathcal{A}$ cannot obtain any advantage from Test query.

Therefore, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_5) = 0$ and $\Pr[E_5 \wedge Suc]$ is negligible.

A.6 Event $E_6 \wedge Suc$

The proof in this case is essentially same as the event $E_2 \wedge Suc$. The situation that the ephemeral secret key of $\overline{\mathsf{sid}}^*$ is given to $\mathcal{A}$ is the same as sid has no matching session because $\mathcal{A}$ can decide arbitrary ephemeral key. Thus, the proof in this event follows that in the event $E_2 \wedge Suc$.

A.7 Event $E_7 \wedge Suc$

The proof in this case is essentially same as the event $E_1 \wedge Suc$. The situation that the ephemeral secret key of $\overline{\mathsf{sid}}^*$ is given to $\mathcal{A}$ is the same as sid has no matching session because $\mathcal{A}$ can decide arbitrary ephemeral key. Thus, the proof in this event follows that in the event $E_1 \wedge Suc$.

A.8 Event $E_8 \wedge Suc$

The proof in this case is essentially same as the event $E_4 \wedge Suc$. The situation that the ephemeral secret key of $\overline{\mathsf{sid}}^*$ is given to $\mathcal{A}$ is the same as $\overline{\mathsf{sid}}^*$ has no matching session because $\mathcal{A}$ can decide arbitrary ephemeral key. Thus, the proof in this event follows that in the event $E_4 \wedge Suc$.

B Proof of Theorem 2

In the experiment of id-CK$^+$ security, we suppose that sid^* is the session identity for the test session, and that there are N users and at most ℓ sessions are activated. Let κ be the security parameter, and let $\mathcal{A}$ be a PPT (in κ) bounded adversary. Suc denotes the event that $\mathcal{A}$ wins. We consider the following events that cover all cases of the behavior of $\mathcal{A}$.

- Let E_1 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the initiator and the static secret key of the initiator is given to $\mathcal{A}$.
- Let E_2 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the initiator and the ephemeral secret key of sid^* is given to $\mathcal{A}$.
- Let E_3 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the responder and the static secret key of the responder is given to $\mathcal{A}$.

– Let E_4 be the event that the test session sid^* has no matching session $\overline{\mathsf{sid}}^*$, the owner of sid^* is the responder and the ephemeral secret key of sid^* is given to $\mathcal{A}$.
– Let E_5 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and both static secret keys of the initiator and the responder are given to $\mathcal{A}$.
– Let E_6 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and both ephemeral secret keys of sid^* and $\overline{\mathsf{sid}}^*$ are given to $\mathcal{A}$.
– Let E_7 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and the static secret key of the owner of sid^* and the ephemeral secret key of $\overline{\mathsf{sid}}^*$ are given to $\mathcal{A}$.
– Let E_8 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and the ephemeral secret key of sid^* and the static secret key of the owner of $\overline{\mathsf{sid}}^*$ are given to $\mathcal{A}$.
– Let E_9 be the event that the test session sid^* has matching session $\overline{\mathsf{sid}}^*$, and master secret key is given to $\mathcal{A}$.

To finish the proof, we investigate events $E_i \wedge Suc$ $(i = 1, \ldots, 9)$ that cover all cases of event Suc. Though proofs of events are essentially same as the case of Theorem 1, $E_9 \wedge Suc$ is the characteristic event for Theorem 2. Thus, we only show the proof of event $E_9 \wedge Suc$.

B.1 Event $E_9 \wedge Suc$

We change the interface of oracle queries and the computation of the session key. These instances are gradually changed over six hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, the session key in the test session does not contain information of the bit b. Thus, the adversary clearly only output a random guess. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_5$ and the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$ by $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_i)$.

Hybrid experiment $\mathbf{H}_0$: This experiment denotes the real experiment for id-CK$^+$ security and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as the advantage of the real experiment.

Hybrid experiment $\mathbf{H}_1$: In this experiment, if session identities in two sessions are identical, the experiment halts.

When two ciphertexts from different randomness are identical, session identities in two sessions are also identical. In the IND-sID-CCA secure IB-KEM, such an event occurs with negligible probability. Thus, $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_1) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_0)| \leq negl$.

Hybrid experiment $\mathbf{H}_2$: In this experiment, the experiment selects a party U_A and integer $i \in [1, \ell]$ randomly in advance. If $\mathcal{A}$ poses Test query to a session except i-th session of U_A, the experiment halts.

Since guess of the test session matches with $\mathcal{A}$'s choice with probability $1/N^2\ell$, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_2) \geq 1/N^2\ell \cdot \mathbf{Adv}(\mathcal{A}, \mathbf{H}_1)$.

Hybrid experiment $\mathbf{H}_3$: In this experiment, the computation of K_T^* in the test session is changed. Instead of computing $(CT_T^*, K_T^*) \leftarrow \mathsf{wEnCap}_{ek_T}(r_{TB})$, it is changed as choosing $K_T^* \leftarrow \mathcal{KS}$ randomly, where we suppose that U_B is the intended partner of U_A in the test session.

We construct an IND-CPA adversary $\mathcal{S}$ from $\mathcal{A}$ in $\mathbf{H}_2$ or $\mathbf{H}_3$. $\mathcal{S}$ performs the following steps.

Init. $\mathcal{S}$ receives the public key ek^* as a challenge.

Setup. $\mathcal{S}$ chooses pseudo-random functions $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and $G : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. These are provided as a part of the public parameters. Also, $\mathcal{S}$ sets the master public and secret key, and all N users' static secret keys. $\mathcal{S}$ selects $r \in \mathcal{RS}_G$, and generates master public and secret keys $(mpk, msk) \leftarrow \mathsf{MKeyGen}(1^\kappa, r)$, where $\mathcal{RS}_G$ is the randomness space of $\mathsf{MKeyGen}$. Then, $\mathcal{S}$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r' \in \mathcal{RS}_G$, and runs the key derivation algorithm $dk_P \leftarrow \mathsf{KeyDer}(mpk, msk, U_P, r')$, where $\mathcal{RS}_G$ is the randomness space of KeyDer. Party U_P's static secret key is $(dk_P, \sigma_P, \sigma'_P)$. The master key msk is given to $\mathcal{A}$.

Next, $\mathcal{S}$ receives the challenge (K^*, CT^*) from the challenger.

Simulation. $\mathcal{S}$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{S}$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: If $P = A$ and the session is i-th session of U_A, $\mathcal{S}$ computes CT_A obeying the protocol and returns the ephemeral public key (CT_A, ek^*). Otherwise, $\mathcal{S}$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: If $\bar{P} = B$, $\mathcal{S}$ computes $CT_{\bar{P}}$ and the session key SK obeying the protocol except that $K_T = K^*$, returns the ephemeral public key $(CT_{\bar{P}}, CT^*)$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{S}$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{S}$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Else if $P = A$ and the session is i-th session of U_A, $\mathcal{S}$ computes the session key SK obeying the protocol except that $K^*_T = K^*$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{S}$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:
   (a) If the session sid is not completed, $\mathcal{S}$ returns an error message.
   (b) Otherwise, $\mathcal{S}$ returns the recorded value SK.
5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{S}$ responds the ephemeral secret key and intermediate computation results of sid as the definition. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.
6. $\mathsf{Corrupt}(U_P)$: $\mathcal{S}$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{S}$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess b', $\mathcal{S}$ outputs b'.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{S}$ is same as the experiment $\mathbf{H}_2$ if the challenge is (K^*_1, CT^*_0). Otherwise, the simulation by $\mathcal{S}$ is same as the experiment $\mathbf{H}_3$. Also, both K^*_T in two experiments have κ-min-entropy because $(\mathsf{wKeyGen}, \mathsf{wEnCap}, \mathsf{wDeCap})$ is κ-min-entropy KEM. Thus, if the advantage of $\mathcal{S}$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_3) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_2)| \leq negl$.

Hybrid experiment $\mathbf{H}_4$: In this experiment, the computation of K'^*_3 in the test session is changed. Instead of computing $K'^*_3 \leftarrow KDF(s, K^*_T)$, it is changed as choosing $K'^*_3 \in \mathcal{FS}$ randomly.

Since K_T^* is randomly chosen in $\mathbf{H}_3$, it has sufficient min-entropy. Thus, by the definition of the KDF, $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_4) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_3)| \leq negl$.

Hybrid experiment $\mathbf{H}_5$: In this experiment, the computation of SK in the test session is changed. Instead of computing $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus G_{K'_3}(\mathsf{ST})$, it is changed as $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus x$ where $x \in \{0,1\}^\kappa$ is chosen randomly and we suppose that U_B is the intended partner of U_A in the test session.

We construct a distinguisher $\mathcal{D}'$ between PRF $F^* : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$ and a random function RF from $\mathcal{A}$ in $\mathbf{H}_4$ or $\mathbf{H}_5$. $\mathcal{D}'$ performs the following steps.

Setup. $\mathcal{D}'$ chooses pseudo-random functions $F, F' : \{0,1\}^* \times \mathcal{FS} \to \mathcal{RS}_E$, and sets $G = F^*$, where $\mathcal{FS}$ is the key space of PRFs, and a KDF $KDF : Salt \times \mathcal{KS} \to \mathcal{FS}$ with a non-secret random salt $s \in Salt$. These are provided as a part of the public parameters. Also, $\mathcal{S}$ sets the master public and secret key, and all N users' static secret keys. $\mathcal{S}$ selects $r \in \mathcal{RS}_G$, and generates master public and secret keys $(mpk, msk) \leftarrow \mathsf{MKeyGen}(1^\kappa, r)$, where $\mathcal{RS}_G$ is the randomness space of $\mathsf{MKeyGen}$. Then, $\mathcal{S}$ selects $\sigma_P \in_R \mathcal{FS}$, $\sigma'_P \in_R \{0,1\}^\kappa$ and $r' \in \mathcal{RS}_G$, and runs the key derivation algorithm $dk_P \leftarrow \mathsf{KeyDer}(mpk, msk, U_P, r')$, where $\mathcal{RS}_G$ is the randomness space of KeyDer. Party U_P's static secret key is $(dk_P, \sigma_P, \sigma'_P)$. The master key msk is given to $\mathcal{A}$.

Simulation. $\mathcal{D}'$ maintains the list $\mathcal{L}_{SK}$ that contains queries and answers of $\mathsf{SessionKeyReveal}$. $\mathcal{D}'$ simulates oracle queries by $\mathcal{A}$ as follows.

1. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}})$: $\mathcal{D}'$ computes the ephemeral public key (CT_P, ek_T) obeying the protocol, returns it and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T))$.
2. $\mathsf{Send}(\Pi, \mathcal{R}, U_{\bar{P}}, U_P, (CT_P, ek_T))$: If $P = A$ and the session is partnered with i-th session of U_A, $\mathcal{D}'$ poses ST to his oracle (i.e., F^* or a random function RF), obtains $x \in \{0,1\}^\kappa$, computes the session key $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus x$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{D}'$ computes the ephemeral public key $(CT_{\bar{P}}, CT_T)$ and the session key SK obeying the protocol, returns the ephemeral public key, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
3. $\mathsf{Send}(\Pi, \mathcal{I}, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$: If $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not recorded, $\mathcal{D}'$ records the session $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ is not completed. Else if $P = A$ and the session is i-th session of U_A, $\mathcal{D}'$ poses ST to his oracle (i.e., F^* or a random function RF), obtains $x \in \{0,1\}^\kappa$, computes the session key $SK = G_{K'_1}(\mathsf{ST}) \oplus G_{K'_2}(\mathsf{ST}) \oplus x$, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$. Otherwise, $\mathcal{D}'$ computes the session key SK obeying the protocol, and records $(\Pi, U_P, U_{\bar{P}}, (CT_P, ek_T), (CT_{\bar{P}}, CT_T))$ as the completed session and SK in the list $\mathcal{L}_{SK}$.
4. $\mathsf{SessionKeyReveal}(\mathsf{sid})$:
   (a) If the session sid is not completed, $\mathcal{D}'$ returns an error message.
   (b) Otherwise, $\mathcal{D}'$ returns the recorded value SK.
5. $\mathsf{SessionStateReveal}(\mathsf{sid})$: $\mathcal{D}'$ responds the ephemeral secret key and intermediate computation results of sid as the definition. Note that the $\mathsf{SessionStateReveal}$ query is not posed to the test session from the freshness definition.
6. $\mathsf{Corrupt}(U_P)$: $\mathcal{D}'$ responds the static secret key and all unerased session states of U_P as the definition.
7. $\mathsf{Test}(\mathsf{sid})$: $\mathcal{D}'$ responds to the query as the definition.
8. If $\mathcal{A}$ outputs a guess $b' = 0$, $\mathcal{D}'$ outputs that the oracle is the PRF F^*. Otherwise, $\mathcal{D}'$ outputs that the oracle is a random function RF.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{D}'$ is same as the experiment $\mathbf{H}_4$ if the oracle is the PRF F^*. Otherwise, the simulation by $\mathcal{D}'$ is same as the experiment $\mathbf{H}_5$. Thus, if the advantage of $\mathcal{D}'$ is negligible, then $|\mathbf{Adv}(\mathcal{A}, \mathbf{H}_5) - \mathbf{Adv}(\mathcal{A}, \mathbf{H}_4)| \leq negl$.

In $\mathbf{H}_5$, the session key in the test session is perfectly randomized. Thus, $\mathcal{A}$ cannot obtain any advantage from Test query.

Therefore, $\mathbf{Adv}(\mathcal{A}, \mathbf{H}_5) = 0$ and $\Pr[E_5 \wedge Suc]$ is negligible.