IEICE TRANS. FUNDAMENTALS, VOL.E95–A, NO.7 JULY 2012

PAPER

Identification Schemes from Key Encapsulation Mechanisms*

Hiroaki ANADA[†a)] ***and*** **Seiko ARITA**[†b)]**,** ***Members***

SUMMARY We propose a generic conversion from a key encapsulation mechanism (KEM) to an identification (ID) scheme. The conversion derives the security for ID schemes against concurrent man-in-the-middle (cMiM) attacks from the security for KEMs against adaptive chosen ciphertext attacks on one-wayness (one-way-CCA2). Then, regarding the derivation as a design principle of ID schemes, we develop a series of concrete one-way-CCA2 secure KEMs. We start with El Gamal KEM and prove it secure against non-adaptive chosen ciphertext attacks on one-wayness (one-way-CCA1) in the standard model. Then, we apply a tag framework with the algebraic trick of Boneh and Boyen to make it one-way-CCA2 secure based on the Gap-CDH assumption. Next, we apply the CHK transformation or a target collision resistant hash function to exit the tag framework. And finally, as it is better to rely on the CDH assumption rather than the Gap-CDH assumption, we apply the Twin DH technique of Cash, Kiltz and Shoup. The application is not "black box" and we do it by making the Twin DH technique compatible with the algebraic trick. The ID schemes obtained from our KEMs show the highest performance in both computational amount and message length compared with previously known ID schemes secure against concurrent man-in-the-middle attacks.

key words: *identification scheme, key encapsulation mechanism, one-way-CCA2 security, concurrent man-in-the-middle attack, the computational Diffie-Hellman assumption*

1. Introduction

Password-based identification protocols are broadly used to verify identities of entities. But they are exposed to a critical threat that, when a password happens to be sent without encryption through a communication channel, an adversary can eavesdrop the password and impersonate the prover easily. Another threat is that, if an adversary acts a verifier and the prover interacts with it without knowing that it is a fake verifier, then the adversary can catch the password even if it is sent under encryption.

Here the need of public key based ID schemes arises. In the public key framework, a prover holds a secret key and a verifier refers to a matching public key. They interact for some rounds doing necessary computations until the verifier feels certain that the prover has the secret key. The secret key is never revealed directly but embedded and hidden in messages by those computations.

However, even for such ID schemes, there is still a strong threat by the following active attack. Pretending to be a verifier, an adversary accesses a prover application (on a client PC, for instance), and invokes many clones of the application, which have independent states and random tapes, but the same secret key. Interacting with those clones, the adversary embeds some cheating trick in messages and collects information of the secret key from the responses of those clones. Afterwards, it tries to impersonate the prover against a true verifier (on a server, for instance) using that collected information. This situation is modeled as *concurrent (two-phase) attack* [7] in cryptography.

Another strong threat is *man-in-the-middle attack*, where an adversary stands between a prover and a verifier and interacts with both sides simultaneously. The advantageous point is that the adversary can interact with the prover adaptively according to the message from the verifier.

Now in the Internet environment where everyone is involved, attacks on ID schemes have become fairly strong and *concurrent man-in-the-middle attack (cMiM attack*, for short) [7] has become a real threat. In cMiM setting, an adversary stands between prover clones and a verifier. Interacting in some cheating way, the adversary collects information of the secret key from the prover clones, while the adversary interacts with the verifier simultaneously trying to impersonate the prover.

Historically, there have been two types of ID schemes. One is the Σ-protocol type [11] which is a kind of proofs of knowledge [6], [21] consisting of 3-round interaction, and the other is challenge-and-response type obtained in a natural way from encryption schemes or signature schemes. Most of known traditional ID schemes, such as the Schnorr scheme [37] and the Guillou-Quisquater (GQ) scheme [22], are the Σ-protocol type because they are faster than challenge-and-response type.

Unfortunately, the Schnorr scheme and the GQ scheme are not secure under man-in-the-middle attacks, hence there have been significant efforts to make ID schemes secure against such cMiM attacks based on the Σ-protocol. For example, Katz [25], [26] made an ID scheme of non-malleable proof of knowledge. But the security model is with timing constraint, not against full cMiM attacks. Gennaro [20] con-

Manuscript received November 4, 2011.
Manuscript revised February 25, 2012.

[†]The authors are with Institute of Information Security, Yokohama-shi, 221-0835 Japan.

*Preliminary versions of this paper appeared in Proceedings of the 4th International Conference on Provable Security — ProvSec 2010, Lecture Notes in Computer Science vol.6402, pp.18–34, Springer-Verlag [1], under the title of "Identification Schemes of Proofs of Ability Secure against Concurrent Man-in-the-Middle Attacks", and in Proceedings of the 4th International Conference on Cryptology — AfricaCrypt 2011, Lecture Notes in Computer Science vol.6737, pp.59–76, Springer-Verlag [2], under the same title as this paper.

a) E-mail: hiroaki.anada@gmail.com
b) E-mail: arita@iisec.ac.jp

DOI: 10.1587/transfun.E95.A.1136

structed an ID scheme of (fully) concurrently non-malleable proof of knowledge by employing a multi-trapdoor commitment. But it is no longer so fast as a challenge-and-response ID scheme obtained from, for instance, the Cramer-Shoup encryption scheme [14]. Moreover, the security is based on a strong type of assumption (the Strong Diffie-Hellman (SDH) assumption or the Strong RSA assumption).

One of the reasons why it is so difficult to construct an ID scheme secure against cMiM attacks seems that we are rooted in the category of Σ-protocols. Let us remember that challenge-and-response ID schemes obtained from IND-CCA2 secure encryption schemes (see [14] for example) and EUF-CMA secure signature schemes (see [4] for example) have already been secure against cMiM attacks.

1.1 Our Contribution

In the notion of encryption scheme, a key encapsulation mechanism (KEM) is the foundational concept for a hybrid construction with a data encryption mechanism (DEM). As a first contribution in this paper, we propose to use a KEM as an ID scheme analogous to the usage of an encryption scheme. That is, given a KEM, we derive a challenge-and-response ID scheme as follows. A verifier of a KEM-based ID scheme makes a pair of a random string and its encapsulation (that is, a ciphertext of the string) using a public key, and send the encapsulation as a challenge to the prover having the matching secret key. The prover decapsulates the encapsulation and returns the result as a response. The verifier checks whether or not the response is equal to the string. Although this is a straightforward conversion, it has never been mentioned in the literature, to the best of our knowledge.

As a generic property, KEM-based ID schemes have an advantage over (non-hybrid) encryption-based ID schemes. That is, a KEM only has to encapsulate *a random string* and *may generate it by itself*, while an encryption scheme has to encrypt *any string given as an input*. Consequently, KEM-based ID schemes have a possibility to be simpler and more efficient than encryption-based ID schemes.

In addition, as we will show in Sect. 3, a KEM only need to be one-way-CCA2 secure for the obtained ID scheme to be cMiM secure. In other words, the IND-CCA2 security, which is stronger than the one-way-CCA2 security, is rather excessive for deriving a cMiM secure ID scheme. Nonetheless by this time, most known encryption schemes and KEMs have been designed to possess IND-CCA2 security (because the purpose is not to derive ID schemes, of course).

Hence there arises a need to provide one-way-CCA2 secure KEMs. As a second contribution, we give concrete, discrete logarithm-based one-way-CCA2 secure KEMs. Starting with El Gamal KEM, we develop a series of five one-way-CCA2 secure KEMs applying techniques such as the selective tag with the algebraic trick [3], [27], the CHK transformation [12], the target collision-resistant hash function [33], [36], the Twin Diffie-Hellman technique[†] [13] and a modification to shorten message length.

It is true that there have already been a few one-way-CCA2 secure KEMs in discrete logarithm setting. In contrast to those KEMs, the feature of our KEMs is that they only need the smallest amount of computational cost and message length, while its security is based on the (Gap-) Computational Diffie-Hellman (CDH) assumption.

Finally, we point out another feature that provers of ID schemes obtained by our generic conversion are deterministic, which means our ID schemes are prover-resettable [5]. Moreover, they are also verifier-resettable because they consist of 2-round interaction. This is a remarkable property because, as is discussed by Yilek [41], resettable security is crucially helpful for virtual machine service in the Cloud Computing, for example.

1.2 Related Works

Recently, independently of us, Fujisaki [19] pointed out a fact similar to our generic construction above (that is, the conversion from a one-way-CCA2 secure KEM to a cMiM secure ID scheme). We discuss the conversion more precisely than it.

As for concrete constructions, the IND-CCA2 secure KEM of Cramer and Shoup (Cramer-Shoup KEM) [15], which is naturally a one-way-CCA2 secure KEM, performs comparably efficiently even now, while its security is based on the DDH assumption.

Hanaoka-Kurosawa [24] gave a one-way-CCA2 secure KEM based on the CDH assumption. It is directly comparable with our `KEM3` and our `KEM3` reduces the computation by at least 0.25 times a single exponentiation than Hanaoka-Kurosawa KEM.

Our `KEM2` and `KEM3` may be directly obtained from the KEM of Kiltz [28] and the KEM of Cash, Kiltz and Shoup [13], respectively. Their KEMs are described in the hashed DH setting and intended to be used in the KEM-DEM hybrid construction. Different from those, we show our KEM's security based on a weaker assumption, that is, the CDH assumption, rather than the hashed DH assumption, and we obtain a weaker security, that is, the one-way-CCA2 security, rather than the IND-CCA2 security. In addition, an application of the Twin DH technique to `KEM2` to get `KEM3` is not a "black box" and we do it by making the Twin DH technique compatible with the algebraic trick of Boneh and Boyen [3].

1.3 Organization of the Paper

In Sect. 2, we fix some notations and briefly review the notion of an ID scheme, a KEM and computational hardness assumptions. In Sect. 3, we propose a generic conversion from a KEM to an ID scheme. In Sect. 4, we discuss El

[†]Applying the Twin Diffie-Hellman technique was suggested to us by Prof. Kiltz [29].

Gamal KEM as a starting point. In Sect. 5, we employ the tag technique with the algebraic trick [3], [27] to get tKEM, which is one-way-CCA2 secure in the selective tag model based on the Gap-CDH assumption. In Sect. 6, we apply the CHK transformation [12] to leave the selective tag model and get KEM1. In Sect. 7, we further develop the series by using a target collision resistant hash function to get KEM2. In Sect. 8, we apply the Twin Diffie-Hellman technique [13] to KEM2 and get KEM3, which is secure based on the CDH assumption. In Sect. 9, we compare efficiency of the ID schemes from our KEMs with previously known ID schemes and KEMs. In Sect. 10, we conclude our work.

2. Preliminaries

The security parameter is denoted k. The bit length of a string s is denoted $|s|$. On an input 1^k, a probabilistic polynomial-time (PPT, for short) algorithm Grp runs and returns (q, g), where q is a prime of length k and g is a generator of a multiplicative cyclic group G_q of order q. Grp specifies elements and group operations of G_q. The ring of exponent domain of G_q, which consists of integers from 0 to $q-1$ with modulo q operation, is denoted $\mathbf{Z}_q$.

When an algorithm A on an input a returns z, we denote it as $z \leftarrow A(a)$. When A on an input a and B on an input b interact and B returns z, we denote it as $z \leftarrow \langle A(a), B(b)\rangle$. When A accesses an oracle O, we denote it as A^O. When A accesses n oracles $O_1, \ldots, O_n$ concurrently, we denote it as $A^{O_1|\cdots|O_n}$. Here "concurrent" means that A accesses oracles in an arbitrarily interleaved order of messages.

A probability of an event X is denoted Pr[X]. A probability of an event X on conditions $Y_1, \ldots, Y_m$ is denoted $\Pr[Y_1; \cdots; Y_m : X]$.

2.1 Identification Scheme

An *identification scheme* ID is a triple of PPT algorithms (K, P, V). K is a key generator which returns a pair of a public key and a matching secret key (pk, sk) on an input 1^k. P and V implement a prover and a verifier strategy, respectively. We require ID to satisfy the completeness condition that boolean decision by V(pk) after completing interaction with P(sk) is 1 with probability one. We say that V(pk) *accepts* if its boolean decision is 1.

2.1.1 Attacks on Identification Scheme

The aim of an adversary $\mathcal{A}$ that attacks an ID scheme ID is impersonation. We say that $\mathcal{A}$ *wins* when $\mathcal{A}$(pk) succeeds in making V(pk) accept.

An adversary $\mathcal{A}$ performs a *concurrent man-in-the-middle attack* as in the following experiment [5], [7].

$\mathbf{Exprmt}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(1^k)$
  $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k)$
  decision $\leftarrow \langle \mathcal{A}^{\mathtt{P}_1(\mathtt{sk})|\cdots|\mathtt{P}_n(\mathtt{sk})}(\mathtt{pk}), \mathtt{V}(\mathtt{pk})\rangle$
  If decision $= 1 \wedge \pi^* \notin \{\pi_i\}_{i=1}^n$ then return Win
  else return Lose.

In the above experiment, we denoted a transcript of interaction between $\mathtt{P}_i(\mathtt{sk})$ and $\mathcal{A}(\mathtt{pk})$ as π_i and a transcript between $\mathcal{A}(\mathtt{pk})$ and $\mathtt{V}(\mathtt{pk})$ as π^*. Here i runs from 1 to n and n is polynomial in k.

As a rule, man-in-the-middle adversary $\mathcal{A}$ is prohibited from relaying a transcript of a whole interaction with some prover clone to the verifier V(pk), as is described $\pi^* \notin \{\pi_i\}_{i=1}^n$ in the experiment. This is a natural and standard constraint to keep man-in-the-middle attack meaningful.

We define *the imp-cmim advantage of* $\mathcal{A}$ *over* ID as:

$$\mathbf{Adv}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(1^k) \text{ returns Win}].$$

We say that an ID is secure against concurrent man-in-the-middle attacks (cMiM secure, for short) if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(k)$ is negligible in k.

Suppose that an adversary $\mathcal{A}$ consists of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. The following experiment is called a *concurrent (two-phase) attack*.

$\mathbf{Exprmt}^{\text{imp-ca}}_{\mathcal{A},\mathtt{ID}}(1^k)$
  $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k), st \leftarrow \mathcal{A}_1^{\mathtt{P}_1(\mathtt{sk})|\cdots|\mathtt{P}_n(\mathtt{sk})}(\mathtt{pk})$
  decision $\leftarrow \langle \mathcal{A}_2(st), \mathtt{V}(\mathtt{pk})\rangle$
  If decision $= 1$ then return Win else return Lose.

$\mathbf{Adv}^{\text{imp-ca}}_{\mathcal{A},\mathtt{ID}}(k)$ is defined in the same way as $\mathbf{Adv}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(k)$.

The concurrent attack is a weaker model than the cMiM attack because of the constraint that the learning phase of $\mathcal{A}_1$ is limited to before the impersonation phase of $\mathcal{A}_2$.

The concurrent attack and the cMiM attack are classified to active attacks. In contrast, there is a *passive attack* described below.

$\mathbf{Exprmt}^{\text{imp-pa}}_{\mathcal{A},\mathtt{ID}}(1^k)$
  $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k)$
  If $\mathcal{A}_1(\mathtt{pk})$ makes a query, reply $\pi_i \leftarrow |\langle \mathtt{P}(\mathtt{sk}), \mathtt{V}(\mathtt{pk})\rangle|$
  $st \leftarrow \mathcal{A}_1(\{\pi_i\}_{i=1}^n)$
  decision $\leftarrow \langle \mathcal{A}_2(st), \mathtt{V}(\mathtt{pk})\rangle$
  If decision $= 1$ then return Win else return Lose.

In the above experiment, we denoted a transcript of a whole interaction between P(sk) and V(pk) as $\pi = |\langle \mathtt{P}(\mathtt{sk}), \mathtt{V}(\mathtt{pk})\rangle|$. $\mathbf{Adv}^{\text{imp-pa}}_{\mathcal{A},\mathtt{ID}}(k)$ is defined in the same way as $\mathbf{Adv}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(k)$.

The passive attack is a weaker model than the concurrent attack because of the constraint that $\mathcal{A}$ cannot choose messages in the learning phase.

2.1.2 Tag-Based Identification Scheme

A *tag-based ID scheme* tagID (see, for example, [1]) is a triple of PPT algorithms (K, P, V) and works in the same way as an ordinary ID scheme, except that a string t, called a *tag*,

is a priori given to P and V by the first round. An interaction between P and V depends on a given tag t.

As for attacks on tagID, we only consider here the *cMiM attack in the selective tag model* which is described by the following experiment.

$\mathbf{Exprmt}^{\text{stag-imp-cmim}}_{\mathcal{A},\mathtt{tagID}}(1^k)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k), \mathtt{t}^* \leftarrow \mathcal{A}(1^k)$
decision $\leftarrow \langle \mathcal{A}^{\mathtt{P}_1(\mathtt{t}_1,\mathtt{sk})|\cdots|\mathtt{P}_n(\mathtt{t}_n,\mathtt{sk})}(\mathtt{pk}), \mathtt{V}(\mathtt{t}^*,\mathtt{pk})\rangle$
If decision $= 1 \wedge \mathtt{t}^* \notin \{\mathtt{t}_i\}_{i=1}^n$
then return Win else return Lose.

In the above experiment, the adversary $\mathcal{A}$ firstly designates a tag $\mathtt{t}^*$ called a *target tag* and, after that, $\mathcal{A}$ is given a public key pk. In addition, before starting each interaction as a verifier, $\mathcal{A}$ provides a tag $\mathtt{t}_i(\neq \mathtt{t}^*)$ to each prover clone $\mathtt{P}_i(\mathtt{sk})$.
$\mathbf{Adv}^{\text{stag-imp-cmim}}_{\mathcal{A},\mathtt{tagID}}(k)$ is defined in the same way as $\mathbf{Adv}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(k)$.

2.2 Key Encapsulation Mechanism

A *key encapsulation mechanism (KEM) KEM* is a triple of PPT algorithms (K, Enc, Dec). K is a key generator which returns a pair of a public key and a matching secret key (pk, sk) on an input 1^k. Enc is an encapsulation algorithm which, on an input pk, returns a pair (K, ψ), where K is a random string and ψ is a ciphertext of K. Dec is a decapsulation algorithm which, on an input $(\mathtt{sk}, \psi)$, returns the decapsulation $\widehat{K}$ of ψ. We require KEM to satisfy the completeness condition that the decapsulation $\widehat{K}$ of a consistently generated ciphertext ψ by Enc is equal to the original string K with probability one. For this requirement, we simply force Dec deterministic.

2.2.1 Attacks on One-Wayness of KEM

An adversary $\mathcal{A}$ performs an *adaptive chosen ciphertext attack on one-wayness* of a KEM (called one-way-CCA2, for short) as in the following experiment [24], [35].

$\mathbf{Exprmt}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(1^k)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k), (K^*, \psi^*) \leftarrow \mathtt{Enc}(\mathtt{pk})$
$\widehat{K^*} \leftarrow \mathcal{A}^{\mathcal{DEC}(\mathtt{sk},\cdot)}(\mathtt{pk}, \psi^*)$
If $\widehat{K^*} = K^* \wedge \psi^* \notin \{\psi_i\}_{i=1}^{q_{dec}}$ then return Win
else return Lose.

In the above experiment, $\psi_i, i = 1, \ldots, q_{dec}$ mean ciphertexts for which $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answers. Here the number q_{dec} of queries is polynomial in k. Note that the challenge ciphertext ψ^* itself must not be queried to $\mathcal{DEC}(\mathtt{sk}, \cdot)$, as is described $\psi^* \notin \{\psi_i\}_{i=1}^{q_{dec}}$ in the experiment.

We define *the one-way-CCA2 advantage of $\mathcal{A}$ over KEM* as:

$$\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(1^k) \text{ returns Win}].$$

We say that a KEM is secure against adaptive chosen ciphertext attacks on one-wayness (one-way-CCA2 secure, for short) if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(k)$ is negligible in k. Note that if a KEM is IND-CCA2 secure [14], then it is one-way-CCA2 secure. So IND-CCA2 security is a stronger notion than one-way-CCA2 security.

Suppose that an adversary $\mathcal{A}$ consists of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. The following experiment is called a *non-adaptive chosen ciphertext attack on one-wayness* of a KEM (called one-way-CCA1, for short).

$\mathbf{Exprmt}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(1^k)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k), st \leftarrow \mathcal{A}_1^{\mathcal{DEC}(\mathtt{sk},\cdot)}(\mathtt{pk})$
$(K^*, \psi^*) \leftarrow \mathtt{Enc}(\mathtt{pk}), \widehat{K^*} \leftarrow \mathcal{A}_2(st, \psi^*)$
If $\widehat{K^*} = K^*$ then return Win else return Lose.

$\mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(k)$ is defined in the same way as $\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(k)$.

The non-adaptive chosen ciphertext attack is a weaker model than the adaptive one because of the constraint that the learning phase of $\mathcal{A}_1$ is limited to before the solving phase of $\mathcal{A}_2$.

The adaptive and non-adaptive chosen ciphertext attacks are classified to active attacks. In contrast, there is a *passive attack on one-wayness* of a KEM described below[†](which we call one-way-PA, for short).

$\mathbf{Exprmt}^{\text{ow-pa}}_{\mathcal{A},\mathtt{KEM}}(1^k)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k)$
If $\mathcal{A}_1(\mathtt{pk})$ makes a query, reply $(K_i, \psi_i) \leftarrow \mathtt{Enc}(\mathtt{pk})$
$st \leftarrow \mathcal{A}_1(\{(K_i, \psi_i)\}_{i=1}^{q_{pa}}), (K^*, \psi^*) \leftarrow \mathtt{Enc}(\mathtt{pk})$
$\widehat{K^*} \leftarrow \mathcal{A}_2(st, \psi^*)$
If $\widehat{K^*} = K^*$ then return Win else return Lose.

In the above experiment, the number q_{pa} of queries is polynomial in k.
$\mathbf{Adv}^{\text{ow-pa}}_{\mathcal{A},\mathtt{KEM}}(k)$ is defined in the same way as $\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(k)$.

The passive attack is a weaker model than the non-adaptive chosen ciphertext attack because of the constraint that $\mathcal{A}$ cannot choose ciphertexts in the learning phase.

2.2.2 Tag-Based Key Encapsulation Mechanism

A *tag-based key encapsulation mechanism (KEM) tagKEM* (see, for example, [27]) is a triple of PPT algorithms (K, Enc, Dec) and works in the same way as an ordinary KEM, except that a string t, called a *tag*, is a priori given to Enc and Dec as an input. A ciphertext ψ depends on a given tag t.

As for attacks on tagKEM, we only consider here the *adaptive chosen ciphertext attack on one-wayness* of a tag-based KEM *in the selective tag model* which is described by

[†]In the experiment, $\mathcal{A}_1$ may run Enc(pk) by itself to get (K_i, ψ_i).

the following experiment.

$\mathbf{Exprmt}^{\text{stag-ow-cca2}}_{\mathcal{A},\mathtt{tagKEM}}(1^k)$
$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k), \mathtt{t}^* \leftarrow \mathcal{A}(1^k)$
$(K^*,\psi^*) \leftarrow \mathtt{Enc}(\mathtt{pk},\mathtt{t}^*), \widehat{K^*} \leftarrow \mathcal{A}^{\mathcal{DEC}(\mathtt{sk},\cdot,\cdot)}(\mathtt{pk},\psi^*)$
If $\widehat{K^*} = K^* \wedge \mathtt{t}^* \notin \{\mathtt{t}_i\}_{i=1}^{q_{dec}}$
then return Win else return Lose.

In the above experiment, the adversary $\mathcal{A}$ firstly designates a tag $\mathtt{t}^*$ called a *target tag* and, after that, $\mathcal{A}$ is given a public key $\mathtt{pk}$. In addition, $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk},\cdot,\cdot)$ for the answer for a pair $(\mathtt{t}_i,\psi_i)$, where $\mathtt{t}_i(\neq \mathtt{t}^*)$ is a tag that $\mathcal{A}$ generates.
$\mathbf{Adv}^{\text{stag-ow-cca2}}_{\mathcal{A},\mathtt{tagKEM}}(k)$ is defined in the same way as $\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(k)$.

2.3 Computational Hardness Assumptions

We say a solver $\mathcal{S}$, a PPT algorithm, *wins* when $\mathcal{S}$ succeeds in solving a computational problem instance.

2.3.1 The CDH and the Gap-CDH Assumptions

A quadruple (g,X,Y,Z) of elements in G_q is called a Diffie-Hellman (DH) tuple if (g,X,Y,Z) is written as (g,g^x,g^y,g^{xy}) for some elements x and y in $\mathbf{Z}_q$. A CDH problem instance is a triple $(g,X=g^x,Y=g^y)$, where the exponents x and y are random and unknown to a solver. The CDH oracle $\mathcal{CDH}$ is an oracle which, queried about a CDH problem instance (g,X,Y), replies the correct answer $Z=g^{xy}$. A DDH problem instance is a quadruple (g,X,Y,Z). The DDH oracle $\mathcal{DDH}$ is an oracle which, queried about a DDH problem instance (g,X,Y,Z), replies the correct boolean decision whether (g,X,Y,Z) is a DH-tuple or not. A CDH problem solver is a PPT algorithm which, given a random CDH problem instance (g,X,Y) as an input, tries to return $Z=g^{xy}$. We define the following experiment.

$\mathbf{Exprmt}^{\text{cdh}}_{\mathcal{S},\mathtt{Grp}}(1^k)$
$(q,g) \leftarrow \mathtt{Grp}(1^k), x,y \leftarrow \mathbf{Z}_q, X := g^x, Y := g^y$
$Z \leftarrow \mathcal{S}(g,X,Y)$
If $Z=g^{xy}$ then return Win else return Lose.

We define the *CDH advantage of* $\mathcal{S}$ *over* $\mathtt{Grp}$ as:

$$\mathbf{Adv}^{\text{cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{cdh}}_{\mathcal{S},\mathtt{Grp}}(1^k) \text{ returns Win}].$$

We say that the CDH Assumption [34] holds for $\mathtt{Grp}$ if, for any PPT algorithm $\mathcal{S}$, $\mathbf{Adv}^{\text{cdh}}_{\mathcal{S},\mathtt{Grp}}(k)$ is negligible in k.

A CDH problem solver $\mathcal{S}$ that is allowed to access $\mathcal{DDH}$ polynomially many times is called a Gap-CDH problem solver. We define the following experiment.

$\mathbf{Exprmt}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(1^k)$
$(q,g) \leftarrow \mathtt{Grp}(1^k), x,y \leftarrow \mathbf{Z}_q, X := g^x, Y := g^y$
$Z \leftarrow \mathcal{S}^{\mathcal{DDH}}(g,X,Y)$
If $Z=g^{xy}$ then return Win else return Lose.

We define the *Gap-CDH advantage of* $\mathcal{S}$ *over* $\mathtt{Grp}$ as:

$$\mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(1^k) \text{ returns Win}].$$

We say that the Gap-CDH Assumption [34] holds for $\mathtt{Grp}$ if, for any PPT algorithm $\mathcal{S}$, $\mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k)$ is negligible in k.

2.3.2 The Twin Diffie-Hellman Technique

A 6-tuple (g,X_1,X_2,Y,Z_1,Z_2) of elements in G_q is called a *twin Diffie-Hellman tuple* if the tuple is written as $(g,g^{x_1},g^{x_2},g^y,g^{x_1y},g^{x_2y})$ for some elements x_1,x_2,y in $\mathbf{Z}_q$.

The following lemma of Cash, Kiltz and Shoup is used in Sect. 8 to decide whether a tuple is a twin DH tuple or not in the security proof.

Lemma 1 (Cash, Kiltz and Shoup [13] Theorem 2, "Trapdoor Test"). *Let X_1,r,s be mutually independent random variables, where X_1 takes values in G_q, and each of r,s is uniformly distributed over $\mathbf{Z}_q$. Define the random variable $X_2 := X_1^{-r}g^s$. Suppose that $\widehat{Y},\widehat{Z_1},\widehat{Z_2}$ are random variables taking values in G_q, each of which is defined independently of r. Then the probability that the truth value of $\widehat{Z_1}^r\widehat{Z_2} = \widehat{Y}^s$ does not agree with the truth value of $(g,X_1,X_2,\widehat{Y},\widehat{Z_1},\widehat{Z_2})$ being a twin DH tuple is at most $1/q$. Moreover, if $(g,X_1,X_2,\widehat{Y},\widehat{Z_1},\widehat{Z_2})$ is a twin DH tuple, then $\widehat{Z_1}^r\widehat{Z_2} = \widehat{Y}^s$ certainly holds.*

2.3.3 The Gap-DL Assumption

A discrete log (DL) problem instance consists of $(g,X=g^x)$, where the exponent x is random and unknown to a solver. A DL problem solver is a PPT algorithm which, given a random DL problem instance (g,X) as an input, tries to return x. A DL problem solver $\mathcal{S}$ that is allowed to access $\mathcal{CDH}$ polynomially many times is called a Gap-DL problem solver. We define the following experiment.

$\mathbf{Exprmt}^{\text{gap-dl}}_{\mathcal{S},\mathtt{Grp}}(1^k)$
$(q,g) \leftarrow \mathtt{Grp}(1^k), x \leftarrow \mathbf{Z}_q, X := g^x$
$x^* \leftarrow \mathcal{S}^{\mathcal{CDH}}(g,X)$
If $g^{x^*}=X$ then return Win else return Lose.

We define the *Gap-DL advantage of* $\mathcal{S}$ *over* $\mathtt{Grp}$ as:

$$\mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathtt{Grp}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{gap-dl}}_{\mathcal{S},\mathtt{Grp}}(1^k) \text{ returns Win}].$$

We say that the Gap-DL Assumption holds for $\mathtt{Grp}$ if, for any PPT algorithm $\mathcal{S}$, $\mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathtt{Grp}}(k)$ is negligible in k.

Although the Gap-DL Assumption is considered fairly strong, it is believed to hold for a certain class of cyclic groups [32].

2.3.4 The Knowledge-of-Exponent Assumption

Informally, the Knowledge-of-Exponent Assumption (KEA)

[8], [17] says that, given a randomly chosen $h \in G_q$ as an input, a PPT algorithm $\mathcal{H}$ can extend (g, h) to a DH-tuple (g, h, X, Z) *only when* $\mathcal{H}$ *knows the exponent* x *of* $X = g^x$. The formal definition is described as follows.

Let $\Lambda(1^k)$ be any distribution. Let $\mathcal{H}$ and $\mathcal{H}'$ be any PPT algorithms which take input of the form (g, h, λ). Here g is any fixed generator, h is a randomly chosen element in G_q, and λ is a string in $\{0,1\}^*$ output by $\Lambda(1^k)$ called auxiliary input [10], [16]. We define the following experiment.

$$\begin{aligned}&\mathbf{Exprmt}^{\mathrm{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(1^k)\\ &(q,g) \leftarrow \mathtt{Grp}(1^k), \lambda \leftarrow \Lambda(1^k), a \leftarrow \mathbf{Z}_q, h := g^a\\ &(g,h,X,Z) \leftarrow \mathcal{H}(g,h,\lambda), x' \leftarrow \mathcal{H}'(g,h,\lambda)\\ &\text{If } X^a = Z \wedge g^{x'} \neq X \text{ then return } \textsc{Win}\\ &\text{else return } \textsc{Lose}.\end{aligned}$$

Note that λ is independent of h in the experiment. This independence is crucial ([10], [16]).

We define the *KEA advantage of* $\mathcal{H}$ *over* $\mathcal{H}'$ *and* `Grp` as:

$$\mathbf{Adv}^{\mathrm{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{Exprmt}^{\mathrm{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(1^k) \text{ returns } \textsc{Win}].$$

An algorithm $\mathcal{H}'$ is called the *KEA extractor*. $\mathbf{Adv}^{\mathrm{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(k)$ can be considered the probability that the KEA extractor $\mathcal{H}'$ *fails* to extract the exponent x of $X = g^x$. We say that the KEA holds for `Grp` if, for any PPT algorithm $\mathcal{H}$, there exists a PPT algorithm $\mathcal{H}'$ such that for any distribution $\Lambda(1^k)$ $\mathbf{Adv}^{\mathrm{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(k)$ is negligible in k.

3. Identification Scheme from Key Encapsulation Mechanism

In this section, we propose a generic conversion from a KEM to an ID scheme. Then we observe a security derivation from a KEM to the obtained ID scheme. The derivation gives a design principle for getting an ID scheme with desired security. Especially, we confirm that a one-way-CCA2 secure KEM gives rise an ID scheme secure against concurrent man-in-the-middle attacks.

3.1 The Conversion

Let $\mathtt{KEM} = (\mathtt{K}, \mathtt{Enc}, \mathtt{Dec})$ be a KEM. Then an ID scheme `ID` is obtained in a natural way as shown in Fig. 1. The key generation algorithm is the same as that of `KEM`. The verifier `V`, given a public key `pk` as an input, invokes the encapsulation algorithm `Enc` on `pk` and gets its return (K, ψ). `V` sends ψ to `P`. The prover `P`, given a secret key `sk` as an input and receiving ψ as an input message, invokes the decapsulation algorithm `Dec` on $(\mathtt{sk}, \psi)$ and gets its return $\widehat{K}$. `P` sends $\widehat{K}$ to `V`. Finally the verifier `V`, receiving $\widehat{K}$ as an input message, verifies whether or not $\widehat{K}$ is equal to K. If so, then `V` returns 1 and otherwise, 0.

It is noteworthy that, a KEM only has to encapsulate a *random string* and *may generate it by itself*, while a (non-hybrid) encryption scheme has to encrypt *any string given as an input*. Consequently, KEM-based ID schemes has a possibility to be simpler and more efficient than encryption-based ID schemes.

Key Generation
– `K`: the same as that of `KEM`
Interaction
– `V`: given `pk` as an input;
  • Invoke `Enc` on `pk`: $(K, \psi) \leftarrow \mathtt{Enc}(\mathtt{pk})$
  • Send ψ to `P`
– `P`: given `sk` as an input and receiving ψ as an input message;
  • Invoke `Dec` on $(\mathtt{sk}, \psi)$: $\widehat{K} \leftarrow \mathtt{Dec}(\mathtt{sk}, \psi)$
  • Send $\widehat{K}$ to `V`
– `V`: receiving $\widehat{K}$ as an input message;
  • If $\widehat{K} = K$ then return 1 else return 0

Fig. 1 An ID Scheme `ID`=(`K`,`P`,`V`) Obtained from a KEM `KEM`= (`K`, `Enc`, `Dec`).

Given $(\mathtt{pk}, \psi^*)$ as an input;
Initial Setting
– Initialize its inner state
– Invoke $\mathcal{A}$ on `pk`
Answering $\mathcal{A}$'s Queries
– In the case that $\mathcal{A}$ queries `V`(`pk`) for the challenge message
  • Send ψ^* to $\mathcal{A}$
– In the case that $\mathcal{A}$ sends ψ to a prover clone `P`(`sk`)
  • If $\psi = \psi^*$, then $K := \perp$
  • else query $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for ψ: $K \leftarrow \mathcal{DEC}(\mathtt{sk}, \psi)$
  • Send K to $\mathcal{A}$
– In the case that $\mathcal{A}$ sends $\widehat{K^*}$ to `V`(`pk`) as the response message
  • Return $\widehat{K^*}$ as the answer for ψ^*

Fig. 2 A One-Way-CCA2 Adversary $\mathcal{B}$ Employing a cMiM Adversary $\mathcal{A}$ for the Proof of Theorem 1.

The following theorem is a security derivation from a KEM to the obtained ID scheme.

Theorem 1. *If a key encapsulation mechanism* `KEM` *is one-way-CCA2 secure, then the obtained identification scheme* `ID` *is cMiM secure. More precisely, for any PPT adversary* $\mathcal{A}$ *that attacks* `ID` *in the cMiM setting, there exists an PPT adversary* $\mathcal{B}$ *that attacks* `KEM` *in the one-way-CCA2 setting satisfying the following inequality.*

$$\mathbf{Adv}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(k) \leqslant \mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{B},\mathtt{KEM}}(k).$$

3.2 Proof of Theorem 1

Let `KEM` be a one-way-CCA2 secure KEM and `ID` be the obtained ID scheme by the conversion above. Let $\mathcal{A}$ be any given PPT adversary that attacks `ID`. Using $\mathcal{A}$ as a subroutine, we construct a PPT one-way-CCA2 adversary $\mathcal{B}$ that attacks `KEM` as shown in Fig. 2.

On an input `pk` and the challenge ciphertext ψ^*, $\mathcal{B}$ initializes its inner state and invokes $\mathcal{A}$ on an input `pk`.

In the case that $\mathcal{A}$ queries `V`(`pk`) for the challenge message, $\mathcal{B}$ sends ψ^* to $\mathcal{A}$.

In the case that $\mathcal{A}$ sends a message ψ to a prover clone `P`(`sk`), $\mathcal{B}$ checks whether or not ψ is equal to ψ^*. If so, then $\mathcal{B}$ puts $K = \perp$. Otherwise, $\mathcal{B}$ queries its decapsulation oracle

Table 1 Security Derivation from a KEM to the Obtained ID Scheme.

Security of a KEM against	$\Longrightarrow$	Security of the Obtained ID Scheme against
one-way-PA	$\Longrightarrow$	passive attack
one-way-CCA1	$\Longrightarrow$	concurrent (two-phase) attack
one-way-CCA2	$\Longrightarrow$	concurrent man-in-the-middle attack

$\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for the ciphertext ψ and gets its decapsulation K as a reply. $\mathcal{B}$ sends K to $\mathcal{A}$ as a response message.

In the case that $\mathcal{A}$ sends $\widehat{K^*}$ to $\mathtt{V}(\mathtt{pk})$ as the response message for the challenge message ψ^*, $\mathcal{B}$ returns $\widehat{K^*}$ as the answer for the challenge ciphertext ψ^*.

The view of $\mathcal{A}$ in $\mathcal{B}$ is the same as the real view of $\mathcal{A}$. This is obvious except for the case that ψ is equal to ψ^*. When $\mathcal{A}$ sent $\psi = \psi^*$, a transcript of the interaction between $\mathtt{P}(\mathtt{sk})$ and $\mathcal{A}(\mathtt{pk})$ would be wholly equal to a transcript of the interaction between $\mathcal{A}(\mathtt{pk})$ and $\mathtt{V}(\mathtt{pk})$, because the prover P is deterministic. This is ruled out, so $\mathcal{B}$'s response, $K = \perp$, is appropriate.

If $\mathcal{A}$ wins, then $\mathcal{B}$ wins. Hence the inequality in Theorem 1 follows. (*Q.E.D.*)

In an analogous ways, we can show the following facts. If a KEM is secure against non-adaptive chosen ciphertext attacks on one-wayness (one-way-CCA1), then the obtained ID scheme ID is secure against concurrent (two-phase) attacks. If a KEM is secure against passive attacks on one-wayness (one-way-PA), then the obtained ID scheme ID is secure against passive attacks.

Table 1 shows this security derivation. We can view the Table 1 as a design principle to construct an ID scheme with desired security.

We remark that the selective tag model for security proofs is compatible with the conversion. That is, if a KEM is secure in the selective tag model, then the obtained ID scheme is secure in the selective tag model.

3.3 Discussion

The prover P in Fig. 1 is deterministic. Therefore, the obtained ID scheme ID is prover-resettable [5]. Moreover, ID is also verifier-resettable because ID consists of 2-round interaction.

4. El Gamal KEM Revisited

In this section, we discuss El Gamal KEM EGKEM [18] as a starting point. The objective here is to see that EGKEM can be proven one-way-CCA1 secure. The obtained ID scheme from EGKEM is similar to the scheme of Stinson and Wu [38], [39]. Unlike their security proof being done in the random oracle model, we provide a security proof in the standard model (in Appendix A).

Key Generation
– K: given 1^k as an input;
• $(q, g) \leftarrow \mathtt{Grp}(1^k)$, $x \leftarrow \mathbf{Z}_q$, $X := g^x$
• $\mathtt{pk} := (q, g, X)$, $\mathtt{sk} := (q, g, x)$, return $(\mathtt{pk}, \mathtt{sk})$
Encapsulation
– Enc: given pk as an input;
• $a \leftarrow \mathbf{Z}_q$, $K := X^a$, $h := g^a$, $\psi := h$, return (K, ψ)
Decapsulation
– Dec: given sk and $\psi = h$ as an input;
• $\widehat{K} := h^x$, return $\widehat{K}$

Fig. 3 El Gamal KEM: EGKEM.

4.1 El Gamal KEM and Its Security

El Gamal KEM EGKEM consists of a triple (K, Enc, Dec). The construction is shown in Fig. 3.

On an input 1^k, the key generator K runs as follows. The group generator Grp returns (q, g) on an input 1^k. Then K chooses x from $\mathbf{Z}_q$, computes $X = g^x$ and sets $\mathtt{pk} = (q, g, X)$ and $\mathtt{sk} = (q, g, x)$. Then K returns (pk, sk).

On an input 1^k, the encapsulation algorithm Enc runs as follows. Enc chooses $a \in \mathbf{Z}_q$ at random, and computes $K = X^a$ and $h = g^a$, and puts $\psi = h$. The random string is K and its encapsulation is ψ. Enc returns the pair (K, ψ).

On an input sk and $\psi = h$, the decapsulation algorithm Dec runs as follows. Dec computes the decapsulation $\widehat{K} = h^x$. Then Dec returns $\widehat{K}$.

The security of EGKEM is stated as follows.

Theorem 2. *The key encapsulation mechanism EGKEM is one-way-CCA1 secure based on the Gap-DL assumption and the KEA for Grp. More precisely, for any PPT one-way-CCA1 adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ on EGKEM, there exists a PPT Gap-DL problem solver $\mathcal{S}$ on Grp and a PPT algorithm $\mathcal{H}$ for the KEA which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{EGKEM}}(k) \leqslant \mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathtt{Grp}}(k) + \mathbf{Adv}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(k).$$

Proof. A detailed proof is described in Appendix A as EGKEM and Theorem 2 are just our starting point. An outline is stated as follows. The CDH oracle enables the solver $\mathcal{S}$ to simulate the adversary $\mathcal{A}_1$'s decapsulation oracle perfectly. After that the KEA extractor works to extract the answer of a DL problem instance from $\mathcal{A}_2$'s return. (Note that the Gap-DL assumption and the KEA are compatible.)

4.2 Discussion

It is well-known that El Gamal KEM EGKEM is one-way-PA secure based on the CDH assumption. In contrast, Theorem 2 says a stronger fact though the assumptions are stronger ones.

5. A Tag-Based One-Way-CCA2 Secure KEM

In this section, owing the idea to the tag-based encryption

Key Generation
– K: given 1^k as an input;
• $(q, g) \leftarrow \mathtt{Grp}(1^k), x, y \leftarrow \mathbf{Z}_q, X := g^x, Y := g^y$
• $\mathtt{pk} := (q, g, X, Y)$, $\mathtt{sk} := (q, g, x, y)$, return $(\mathtt{pk}, \mathtt{sk})$
Tag-Receiving
– Enc and Dec are given a tag $\mathtt{t} \in \mathbf{Z}_q$
Encapsulation
– Enc: given pk as an input;
• $a \leftarrow \mathbf{Z}_q, K := X^a, h := g^a, d := (X^{\mathtt{t}}Y)^a, \psi := (h, d)$, return (K, ψ)
Decapsulation
– Dec: given sk and $\psi = (h, d)$ as an input;
• If $h^{\mathtt{t}x+y} \neq d$ then $\widehat{K} := h^x$ else $\widehat{K} := \perp$, return $\widehat{K}$

Fig. 4 A Tag-Based One-Way-CCA2 KEM: tKEM.

scheme of Kiltz [27], we apply a tag framework with the algebraic trick of Boneh and Boyen [3] to El Gamal KEM EGKEM to make it one-way-CCA2 secure.

5.1 A Tag-Based KEM and Its Security

A tag-based KEM tKEM consists of a triple (K, Enc, Dec). The construction is shown in Fig. 4.

On an input 1^k, the key generator K runs as follows. The group generator Grp returns (q, g) on an input 1^k. Then K chooses x and y from $\mathbf{Z}_q$, computes $X = g^x$ and $Y = g^y$, and sets $\mathtt{pk} = (q, g, X, Y)$ and $\mathtt{sk} = (q, g, x, y)$. Then K returns (pk, sk).

A string tag t is a priori given to Enc and Dec. In our construction, we set the tag t in $\mathbf{Z}_q$.

On an input 1^k, the encapsulation algorithm Enc runs as follows. Enc chooses a from $\mathbf{Z}_q$ at random, and computes $K = X^a$ and $h = g^a$. Additionally, Enc computes $d = (X^{\mathtt{t}}Y)^a$ and puts $\psi = (h, d)$. The random string is K and its encapsulation is ψ. Enc returns the pair (K, ψ).

On an input sk and $\psi = (h, d)$, the decapsulation algorithm Dec runs as follows. Dec verifies whether $(g, X^{\mathtt{t}}Y, h, d)$ is a DH-tuple. For this sake, Dec checks whether $h^{\mathtt{t}x+y} = d$ holds. If it does not hold, then Dec puts $\widehat{K} = \perp$. Otherwise, Dec computes the decapsulation $\widehat{K} = h^x$. Then Dec returns $\widehat{K}$.

The security of tKEM is stated as follows.

Theorem 3. *The key encapsulation mechanism tKEM is one-way-CCA2 secure in the selective tag model based on the Gap-CDH assumption. More precisely, for any PPT adversary $\mathcal{A}$ there exists a PPT Gap-CDH problem solver $\mathcal{S}$ which satisfies the following tight reduction.*

$$\mathbf{Adv}^{\text{stag-ow-cca2}}_{\mathcal{A},\mathtt{tKEM}}(k) \leqslant \mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k).$$

5.2 Proof of Theorem 3

An outline is stated as follows. The algebraic trick of the tag framework [3], [27] enables the solver $\mathcal{S}$ to simulate an adversary $\mathcal{A}$'s decapsulation oracle perfectly. The algebraic trick also enables $\mathcal{S}$ to embed a portion of a given CDH problem instance in the challenge ciphertext. It is easy to extract the answer for the CDH problem instance from $\mathcal{A}$'s return.

Given (q, g, V, W) as an input;
Initial Setting
– Initialize its inner state
– Invoke $\mathcal{A}$ on 1^k and get a target tag: $\mathtt{t}^* \leftarrow \mathcal{A}(1^k)$
– $u \leftarrow \mathbf{Z}_q, X := V, Y := X^{-\mathtt{t}^*}g^u$, $\mathtt{pk} := (q, g, X, Y)$
– $a^* \in \mathbf{Z}_q, h^* = Wg^{a^*}, d^* = (h^*)^u, \psi^* = (h^*, d^*)$
– Inputs $(\mathtt{pk}, \psi^*)$ into $\mathcal{A}$
Answering $\mathcal{A}$'s Queries and Extracting the Answer from Return
– Receive a tag: $\mathtt{t} \leftarrow \mathcal{A}$
– In the case that $\mathcal{A}$ queries $\mathcal{DEC}(\mathtt{sk}, \cdot, \cdot)$ for the answer for $(\mathtt{t}, \psi = (h, d))$;
• If $\mathcal{DDH}(g, h, X^{\mathtt{t}}Y, d) \neq 1$ then $\widehat{K} := \perp$
• else $\widehat{K} := (d/h^u)^{1/(\mathtt{t}-\mathtt{t}^*)}$ (: the case Simdec)
• Reply $\widehat{K}$ to $\mathcal{A}$
– In the case that $\mathcal{A}$ returns $\widehat{K^*}$;
• Return $Z := \widehat{K^*}/X^{a^*}$

Fig. 5 A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 3.

Let us proceed in detail. Let $\mathcal{A}$ be as in Theorem 3. Using $\mathcal{A}$ as a subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$. The construction is illustrated in Fig. 5.

$\mathcal{S}$ is given $q, g, V = g^v, W = g^w$ as a CDH problem instance, where v and w are random and unknown to $\mathcal{S}$. $\mathcal{S}$ initializes its inner state. $\mathcal{S}$ invokes $\mathcal{A}$ on an input 1^k and gets a target tag $\mathtt{t}^*$ from $\mathcal{A}$. $\mathcal{S}$ chooses u from $\mathbf{Z}_q$ at random and puts $X = V$ and $Y = X^{-\mathtt{t}^*}g^u$, sets $\mathtt{pk} = (q, g, X, Y)$. $\mathcal{S}$ chooses a^* from $\mathbf{Z}_q$ at random and puts $h^* = Wg^{a^*}, d^* = (h^*)^u$ and $\psi^* = (h^*, d^*)$. $\mathcal{S}$ inputs pk and ψ^* into $\mathcal{A}$.

Note that pk is correctly distributed. Note also that $\mathcal{S}$ knows neither x nor y, where x and y denote the discrete log of X and Y on a base g, respectively. Here the following holds;

$$y = -\mathtt{t}^*x + u.$$

$\mathcal{S}$ replies $\mathcal{A}$ in answer to $\mathcal{A}$'s queries as follows.

In the case that $\mathcal{A}$ queries $\mathcal{DEC}(\mathtt{sk}, \cdot, \cdot)$ for the answer for $(\mathtt{t}, \psi = (h, d))$, $\mathcal{S}$ verifies whether $(g, h, X^{\mathtt{t}}Y, d)$ is a DH-tuple. For this sake, $\mathcal{S}$ queries its DDH oracle $\mathcal{DDH}$ for the answer. If it is not satisfied then $\mathcal{S}$ puts $K = \perp$. Otherwise $\mathcal{S}$ puts $\widehat{K} = (d/h^u)^{1/(\mathtt{t}-\mathtt{t}^*)}$ (call this case Simdec). $\mathcal{S}$ replies $\widehat{K}$ to $\mathcal{A}$. Note that, in the selective tag model, $\mathcal{A}$ is prohibited from setting $\mathtt{t} = \mathtt{t}^*$ (that is, $\mathcal{A}$ must keep that $\mathtt{t} \neq \mathtt{t}^*$).

In the case that $\mathcal{A}$ returns $\widehat{K^*}$, $\mathcal{S}$ returns $Z = \widehat{K^*}/X^{a^*}$.

$\mathcal{S}$ is able to simulate the real view of $\mathcal{A}$ perfectly, as we see below.

Firstly, the challenge ciphertext $\psi^* = (h^*, d^*)$ is consistent and correctly distributed. This is because the distribution of $\psi^* = (h^*, d^*)$ is equal to that of the real consistent ciphertext $\psi = (h, d)$. To see it, note that $w + a^*$ is substituted for a;

$$h^* = Wg^{a^*} = g^{w+a^*},$$
$$d^* = (g^{w+a^*})^u = (g^u)^{w+a^*} = (X^{\mathtt{t}^*}Y)^{w+a^*}.$$

Secondly, in the case Simdec, $\mathcal{S}$ can simulate the decap-

sulation oracle $\mathcal{DEC}(\mathtt{sk},\cdot,\cdot)$ perfectly. This is because $\widehat{K}$ is equal to h^x by the following equalities;

$$d/h^u = h^{\mathtt{t}x+y-u} = h^{(\mathtt{t}-\mathtt{t}^*)x+(\mathtt{t}^*x+y-u)} = h^{(\mathtt{t}-\mathtt{t}^*)x}.$$

As a whole, $\mathcal{S}$ simulates the real view of $\mathcal{A}$ perfectly.

Now we evaluate the Gap-CDH advantage of $\mathcal{S}$. If $\mathcal{A}$ wins, then $(g, h^*, X, \widehat{K^*})$ is a DH-tuple and the following holds (note that we have set $X = V$, so $x = v$);

$$\widehat{K^*} = (g^x)^{w+a^*} = g^{vw}X^{a^*}.$$

Hence the return Z is equal to $\widehat{K^*}/X^{a^*} = g^{vw}$, which is the correct answer for the input (g, V, W). That is, $\mathcal{S}$ wins. Therefore, the probability that $\mathcal{S}$ wins is lower bounded by the probability that $\mathcal{A}$ wins;

$$\Pr[\mathcal{S}\text{ wins}] \geqslant \Pr[\mathcal{A}\text{ wins}].$$

That is; $\quad \mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \geqslant \mathbf{Adv}^{\text{stag-ow-cca2}}_{\mathcal{A},\mathtt{tKEM}}(k)$. (*Q.E.D.*)

5.3 Discussion

We note that the selective tag model for security proofs is compatible with our generic conversion. That is, the obtained ID scheme from tKEM is cMiM secure in the selective tag model.

6. A One-Way-CCA2 Secure KEM by the CHK Transformation

In this section, we apply a generic method, that is, the CHK transformation [12], to the tag-based KEM tKEM to exit the tag framework. Along the technique, we replace the tag t by a one-time verification key vk of a strong one-time signature OTS. (The definition of strong one-time signatures is summarized in Appendix B.)

6.1 A KEM by the CHK Transformation and Its Security

A KEM KEM1 by the CHK transformation of the tag-based KEM tKEMconsists of a triple (K, Enc, Dec). The construction is shown in Fig. 6.

On an input 1^k, the key generator K runs as follows. The group generator Grp returns (q, g) on an input 1^k. Then K chooses x and y from $\mathbf{Z}_q$, puts $X = g^x$ and $Y = g^y$, and sets $\mathtt{pk} = (q, g, X, Y)$ and $\mathtt{sk} = (q, g, x, y)$. Then K returns (pk, sk).

On an input 1^k, the encapsulation algorithm Enc runs as follows. Enc runs signing key generator SGK on an input 1^k to get (vk, sgk), where vk is a verification key and sgk is the matching signing key. Enc chooses a from $\mathbf{Z}_q$ at random, and computes $K = X^a$ and $h = g^a$. Enc computes $d := (X^{\mathtt{vk}}Y)^a$ (here, if it is needed for OTS, we take a hash value $H(\mathtt{vk})$ of vk before the exponentiation so that $H(\mathtt{vk}) \in \mathbf{Z}_q$). Enc runs $\mathtt{Sign}_{\mathtt{sgk}}$ on message (h, d) to get a signature σ. Enc puts $\psi = (\mathtt{vk}, (h, d), \sigma)$. The random string is K and its encapsulation is ψ. Enc returns the pair (K, ψ).

Key Generation
– K: given 1^k as an input;
- $(q, g) \leftarrow \mathtt{Grp}(1^k)$, $x, y \leftarrow \mathbf{Z}_q$, $X := g^x$, $Y := g^y$
- $\mathtt{pk} := (q, g, X, Y)$, $\mathtt{sk} := (q, g, x, y)$, return (pk, sk)

Encapsulation
– Enc: given pk as an input;
- $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$
- $a \leftarrow \mathbf{Z}_q$, $K := X^a$, $h := g^a$, $d := (X^{\mathtt{vk}}Y)^a$, $\sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}((h, d))$
- $\psi := (\mathtt{vk}, (h, d), \sigma)$
- Return (K, ψ)

– Dec: given sk and $\psi = (\mathtt{vk}, (h, d), \sigma)$ as an input;
- If $\mathtt{Vrfy}_{\mathtt{vk}}((h, d), \sigma) \neq 1$ or $h^{(\mathtt{vk})x+y} \neq d$ then $\widehat{K} := \perp$ else $\widehat{K} := h^x$
- Return $\widehat{K}$

Fig. 6 A One-Way-CCA2 KEM by the CHK Transformation: KEM1.

On an input sk and $\psi = (\mathtt{vk}, (h, d), \sigma)$, the decapsulation algorithm Dec runs as follows. Dec verifies whether the signature σ for the message (h, d) is valid under vk and whether $(g, X^{\mathtt{vk}}Y, h, d)$ is a DH-tuple. For the latter, Dec checks whether $h^{(\mathtt{vk})x+y} = d$ holds. If at least one of them does not hold then Dec puts $\widehat{K} = \perp$. Otherwise, Dec computes the decapsulation $\widehat{K} = h^x$. Then Dec returns $\widehat{K}$.

The security of KEM1 is stated as follows.

Theorem 4. *The key encapsulation mechanism KEM1 is one-way-CCA2 secure based on the Gap-CDH assumption and one-time security of a strong one-time signature employed. More precisely, for any PPT adversary $\mathcal{A}$ there exist a PPT Gap-CDH problem solver $\mathcal{S}$ and a PPT forger $\mathcal{F}$ which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM1}}(k) \leqslant \mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) + \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathtt{OTS}}(k).$$

Proof. A detailed proof is described in Appendix C as it is an adaptation of the routine established by [12] to our situation. An outline is stated as follows. The proof goes almost the same way as the proof of Theorem 3 except that we have to treat a case that the verification key $\mathtt{vk}^*$ in the challenge ciphertext is equal to a verification key vk in a decapsulation query. The EUF-CMA property in the strong sense of a one-time signature employed assures that the case happens only with negligible probability.

6.2 Discussion

When we use a one-time signature such that $|\mathtt{vk}| \geqslant |q| = k$, we have to take a hash value $H(\mathtt{vk})$ by a hash function $H : \{0,1\}^* \to \mathbf{Z}_q$ just before doing a computation in $\mathbf{Z}_q$. In this setting, the security statement in Theorem 4 needs a target collision resistance for $\{H\}$ to make a collision case (that is, the case that $\mathtt{vk} \neq \mathtt{vk}^*$ and $H(\mathtt{vk}) = H(\mathtt{vk}^*)$) negligible.

7. A One-Way-CCA2 Secure KEM with a Target Collision Resistant Hash Function

In this section, depending on the specific structure, we use a specific tool, that is, a target collision resistant hash function (a TCR hash function, for short) to exit the tag framework of tKEM. Along the technique, we replace the tag t by a TCR

Key Generation
– K: given 1^k as an input;
• $(q, g) \leftarrow \mathtt{Grp}(1^k), x, y \leftarrow \mathbf{Z}_q$
• $X := g^x, Y := g^y, \kappa \leftarrow Hkey(1^k)$
• $\mathtt{pk} := (q, g, X, Y, \kappa), \mathtt{sk} := (q, g, x, y, \kappa)$, return $(\mathtt{pk}, \mathtt{sk})$
Encapsulation
– Enc: given pk as an input;
• $a \leftarrow \mathbf{Z}_q, K := X^a, h := g^a, \tau \leftarrow H_\kappa(h), d := (X^\tau Y)^a$
• $\psi := (h, d)$
• Return (K, ψ)
Decapsulation
– Dec: given sk and $\psi = (h, d)$ as an input;
• $\tau \leftarrow H_\kappa(h)$
• If $h^{\tau x+y} \neq d$ then $\widehat{K} :=\perp$ else $\widehat{K} := h^x$
• Return $\widehat{K}$

Fig. 7 A One-Way-CCA2 KEM with a TCR Hash Function: KEM2.

hash function value. (The definition of a TCR hash function family $Hfam(1^k) = \{H_\kappa\}_{\kappa \in Hkey(1^k)}$ is summarized in Appendix D.)

7.1 A KEM with a TCR Hash Function and Its Security

A KEM KEM2 with TCR hash function consists of a triple (K, Enc, Dec). The construction is shown in Fig. 7.

On an input 1^k, the key generator K runs as follows. The group generator Grp returns (q, g) on an input 1^k. Then K chooses x and y from $\mathbf{Z}_q$ and computes $X = g^x$ and $Y = g^y$. In addition, K chooses a hash key κ from a hash key space $Hkey(1^k)$. The hash key κ indicates a specific hash function H_κ with values in $\mathbf{Z}_q$ in a hash function family $Hfam(1^k) = \{H_\kappa\}_{\kappa \in Hkey(1^k)}$. K sets $\mathtt{pk} = (q, g, X, Y, \kappa)$ and $\mathtt{sk} = (q, g, x, y, \kappa)$. Then K returns $(\mathtt{pk}, \mathtt{sk})$.

On an input 1^k, the encapsulation algorithm Enc runs as follows. Enc chooses a from $\mathbf{Z}_q$ at random and computes $h = g^a$. Enc computes the hash value $\tau \leftarrow H_\kappa(h)$ and computes $d = (X^\tau Y)^a$. Enc puts $\psi = (h, d)$. The random string is K and its encapsulation is ψ. Then Enc returns the pair (K, ψ).

On an input sk and $\psi = (h, d)$, the decapsulation algorithm Dec runs as follows. Dec computes the hash value $\tau \leftarrow H_\kappa(h)$. Dec verifies whether the quadruple $(g, X^\tau Y, h, d)$ is a DH-tuple. For this sake, P checks whether $h^{\tau x+y} = d$ holds. If it does not hold, then Dec puts $\widehat{K} =\perp$. Otherwise, Dec computes the decapsulation $\widehat{K} = h^x$. Then Dec returns $\widehat{K}$.

The security of KEM2 is stated as follows.

Theorem 5. *The key encapsulation mechanism KEM2 is one-way-CCA2 secure based on the Gap-CDH assumption and the target collision resistance of a hash function family $Hfam(1^k) = \{H_\kappa\}_{\kappa \in Hkey(1^k)}$. More precisely, for any PPT adversary $\mathcal{A}$ there exist a PPT Gap-CDH problem solver $\mathcal{S}$ and a PPT collision-finder $\mathcal{CF}$ on Hfam which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM2}}(k) \leqslant \mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) + \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k).$$

Given (q, g, V, W) as an input;
Initial Setting
– Initialize its inner state
– $a^* \leftarrow \mathbf{Z}_q, h^* := Wg^{a^*}$
– $\kappa \leftarrow Hkey(1^k), \tau^* \leftarrow H_\kappa(h^*)$
– $u \leftarrow \mathbf{Z}_q, X := V, Y := X^{-\tau^*} g^u, d^* = (h^*)^u$
– $\mathtt{pk} := (q, g, X, Y, \kappa), \psi^* := (h^*, d^*)$
– Invoke $\mathcal{A}$ on pk and ψ^*
Answering $\mathcal{A}$'s Queries and Extracting the Answer from Return
– In the case that $\mathcal{A}$ queries $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (h, d)$;
• $\tau \leftarrow H_\kappa(h)$
• If $\mathcal{DDH}(g, X^\tau Y, h, d) \neq 1$ then $\widehat{K} :=\perp$
• else
If $\tau \neq \tau^*$ then $\widehat{K} := (d/h^u)^{1/(\tau-\tau^*)}$ (: the case Simdec)
else abort (: the case Abort)
• Reply $\widehat{K}$ to $\mathcal{A}$
– In the case that $\mathcal{A}$ returns $\widehat{K}^*$;
• Return $Z := \widehat{K}^*/X^{a^*}$

Fig. 8 A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 5.

7.2 Proof of Theorem 5

An outline is stated as follows. The proof goes almost the same way as the proof of Theorem 3 except that we have to treat a case that the hash value τ^* of h^* in the challenge ciphertext is equal to a hash value τ of h in a decapsulation query. The TCR property of a TCR hash function family employed assures that the case happens only with negligible probability.

Let us proceed in detail. Let $\mathcal{A}$ be as in Theorem 5. Using $\mathcal{A}$ as a subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$. The construction is illustrated in Fig. 8.

$\mathcal{S}$ is given $q, g, V = g^v, W = g^w$ as an input, where v and w are random and unknown to $\mathcal{S}$. $\mathcal{S}$ initializes its inner state. $\mathcal{S}$ chooses a^* from $\mathbf{Z}_q$ at random and computes $h^* = Wg^{a^*}$. Then $\mathcal{S}$ chooses κ from $Hkey(1^k)$ and computes $\tau^* \leftarrow H_\kappa(h^*)$. $\mathcal{S}$ chooses u from $\mathbf{Z}_q$ at random, puts $X = V$ and computes $Y = X^{-\tau^*} g^u$ and $d^* = (h^*)^u$. $\mathcal{S}$ sets $\mathtt{pk} = (q, g, X, Y)$ and puts $\psi^* = (h^*, d^*)$.

Note that pk is correctly distributed. Note also that $\mathcal{S}$ knows neither x nor y, where x and y denote the discrete log of X and Y on a base g, respectively. Here the following holds;

$$y = -\tau^* x + u.$$

$\mathcal{S}$ invokes $\mathcal{A}$ on an input $(\mathtt{pk}, \psi^*)$. $\mathcal{S}$ replies to $\mathcal{A}$'s queries as follows.

In the case that $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\phi = (h, d)$, $\mathcal{S}$ computes $\tau \leftarrow H_\kappa(h)$. $\mathcal{S}$ verifies whether $(g, X^\tau Y, h, d)$ is a DH-tuple. For this sake, $\mathcal{S}$ queries its decision oracle $\mathcal{DDH}$. If the answer is "False", then $\mathcal{S}$ puts $\widehat{K} =\perp$. Otherwise, if $\tau \neq \tau^*$, then $\mathcal{S}$ computes $\widehat{K} = (d/h^u)^{1/(\tau-\tau^*)}$ (call this case Simdec). If $\tau = \tau^*$, then $\mathcal{S}$ aborts (call this case Abort). Then $\mathcal{S}$ replies $\widehat{K}$ to $\mathcal{A}$ except for the case Abort.

In the case that $\mathcal{A}$ returns $\widehat{K}^*$ as the answer for ψ^*, $\mathcal{S}$ returns $Z = \widehat{K}^*/X^{a^*}$.

$\mathcal{S}$ is able to simulate the real view of $\mathcal{A}$ perfectly until the case Abort happens, as we see below.

Firstly, the challenge ciphertext $\psi^* = (h^*, d^*)$ is consistent and correctly distributed. This is because the distribution of $\psi^* = (h^*, d^*)$ is equal to that of the real consistent ciphertext $\psi = (h, d)$. To see it, note that $w + a^*$ is substituted for a;

$$h^* = Wg^{a^*} = g^{w+a^*},$$
$$d^* = (g^{w+a^*})^u = (g^u)^{w+a^*} = (X^{\tau^*}Y)^{w+a^*}.$$

Secondly, in the case Simdec, $\mathcal{S}$ can simulate the decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ perfectly. This is because $\widehat{K}$ is equal to h^x by the following equalities;

$$d/h^u = h^{\tau x+y-u} = h^{(\tau-\tau^*)x+(\tau^* x+y-u)} = h^{(\tau-\tau^*)x}.$$

As a whole, $\mathcal{S}$ simulates the real view of $\mathcal{A}$ perfectly until the case Abort happens.

Now we evaluate the CDH advantage of $\mathcal{S}$. When $\mathcal{A}$ wins, $(g, X, h^*, \widehat{K^*})$ is a DH-tuple, so the following hold (note that we have set $X = V$, so $x = v$);

$$\widehat{K^*} = (g^x)^{w+a^*} = g^{vw}X^{a^*}.$$

Hence the return Z is equal to $\widehat{K^*}/X^{a^*} = g^{vw}$, which is the correct answer for the input (g, V, W). That is, $\mathcal{S}$ wins. Therefore the probability that $\mathcal{S}$ wins is lower bounded by the probability that $\mathcal{A}$ wins and Abort does not happen.

$$\begin{aligned}\Pr[\mathcal{S}\text{ wins}] &\geqslant \Pr[\mathcal{A}\text{ wins } \wedge \neg\text{Abort}]\\ &\geqslant \Pr[\mathcal{A}\text{ wins}] - \Pr[\text{Abort}].\end{aligned}$$

Hence we get the following inequality.

$$\mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \geqslant \mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM2}}(k) - \Pr[\text{Abort}].$$

So our task being left is to show that Pr[Abort] is negligible in k.

Claim 7.1. *The probability that* Abort *occurs is negligible in k.*

Proof of Claim 7.1. Using $\mathcal{A}$ as a subroutine, we construct a target collision finder $\mathcal{CF}$ on *Hfam* as follows. Given 1^k as an input, $\mathcal{CF}$ initializes its inner state. $\mathcal{CF}$ gets (q, g) from $\mathtt{Grp}(1^k)$. $\mathcal{CF}$ chooses a^* from $\mathbf{Z}_q$ at random, computes $h^* = g^{a^*}$ and returns h^*. $\mathcal{CF}$ receives a random hash key κ and computes $\tau^* \leftarrow H_\kappa(h^*)$. Then $\mathcal{CF}$ chooses $x, y \in \mathbf{Z}_q$ at random and computes $X = g^x, Y = g^y$. $\mathcal{CF}$ computes $d^* = (X^{\tau^*}Y)^{a^*}$ and puts $\psi^* = (h^*, d^*)$. Finally $\mathcal{CF}$ sets $\mathtt{pk} = (q, g, X, Y, \kappa)$, $\mathtt{sk} = (q, g, x, y, \kappa)$ and invokes $\mathcal{A}$ on $\mathtt{pk}$ and ψ^*.

In the case that $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (h, d)$, $\mathcal{CF}$ computes $\tau \leftarrow H_\kappa(h)$ and verifies whether $(g, X^\tau Y, h, d)$ is a DH-tuple. $\mathcal{CF}$ does the same as an honest decapsulation algorithm does because $\mathcal{CF}$ has the secret key $\mathtt{sk}$. If it is not a DH-tuple, $\mathcal{CF}$ sets $\widehat{K} = \perp$. Otherwise, if $\tau \neq \tau^*$, then $\mathcal{CF}$ replies $\widehat{K} = h^x$ to $\mathcal{A}$. If $\tau = \tau^*$, then $\mathcal{CF}$ returns h and stops (call this case Collision).

Note that the view of $\mathcal{A}$ in $\mathcal{CF}$ is the same as the real view until the case Collision happens. Especially, the view of $\mathcal{A}$ in $\mathcal{CF}$ is the same as the view of $\mathcal{A}$ in $\mathcal{S}$ until the case Abort or the case Collision happens. So we have:

$$\Pr[\text{Abort}] = \Pr[\text{Collision}].$$

Notice that the case Collision implies that the following conditions hold;

$$\begin{cases} & (g, X^\tau Y, h, d) \text{ is a DH-tuple}\\ \text{and} & (g, X^{\tau^*}Y, h^*, d^*) \text{ is a DH-tuple}\\ \text{and} & \tau = \tau^*. \end{cases}$$

If in addition to the above conditions h were equal to h^*, then d would be equal to d^*, which would mean that $\mathcal{A}$ queried the challenge ciphertext ψ^* to its decapsulation oracle. This is ruled out. Hence it must hold that

$$h \neq h^*.$$

Namely, in the case Collision, $\mathcal{CF}$ succeeds in obtaining a target collision. So we have:

$$\Pr[\text{Collision}] = \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},\mathit{Hfam}}(k).$$

Combining the two equalities, we get

$$\Pr[\text{Abort}] = \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},\mathit{Hfam}}(k).$$

The right hand side is negligible in k by the assumption in Theorem 5. (*Q.E.D.*)

7.3 Discussion

As we will see in Sect. 9, the obtained ID scheme from KEM2 shows the highest performance in both computational amount and message length compared with previous ID schemes secure against cMiM attacks.

8. A One-Way-CCA2 Secure KEM by the Twin Diffie-Hellman Technique

In this section, as it is better to rely on the CDH assumption rather than the Gap-CDH assumption, we apply the Twin DH technique of Cash, Kiltz and Shoup [13], [29] to KEM2. An application of the Twin DH technique to KEM2 to get KEM3 is not a "black box" and we do it by making the Twin DH technique compatible with the algebraic trick of Boneh and Boyen [3] introducing necessary random variables.

8.1 A KEM by the Twin Diffie-Hellman Technique and Its Security

A KEM KEM3 by the Twin Diffie-Hellman technique consists of a triple $(\mathtt{K}, \mathtt{Enc}, \mathtt{Dec})$. The construction is shown in

Key Generation
– K: given 1^k as an input;
- $(q, g) \leftarrow \mathtt{Grp}(1^k), \kappa \leftarrow Hkey(1^k)$
- $x_1, x_2, y_1, y_2 \leftarrow \mathbf{Z}_q, X_1 := g^{x_1}, X_2 := g^{x_2}, Y_1 := g^{y_1}, Y_2 := g^{y_2}$
- $\mathtt{pk} := (q, g, X_1, X_2, Y_1, Y_2, \kappa), \mathtt{sk} := (q, g, x_1, x_2, y_1, y_2, \kappa)$
- Return $(\mathtt{pk}, \mathtt{sk})$

Encapsulation
– Enc: given pk as an input;
- $a \leftarrow \mathbf{Z}_q, h := g^a, \tau \leftarrow H_\kappa(h)$
- $d_1 := (X_1^\tau Y_1)^a, d_2 := (X_2^\tau Y_2)^a, K := X_1^a, \psi = (h, d_1, d_2)$
- Return (K, ψ)

Decapsulation
– Dec: given $\mathtt{sk}, \psi = (h, d_1, d_2)$ as an input;
- $\tau \leftarrow H_\kappa(h)$
- If $h^{\tau x_1 + y_1} \neq d_1$ or $h^{\tau x_2 + y_2} \neq d_2$ then $\widehat{K} :=\perp$ else $\widehat{K} := h^{x_1}$
- Return $\widehat{K}$

Fig. 9 A One-Way-CCA2 KEM by the Twin DH Technique: KEM3.

Fig. 9.

On an input 1^k, the key generator K runs as follows. The group generator Grp returns (q, g) on an input 1^k. Then K chooses x_1, x_2, y_1, y_2 from $\mathbf{Z}_q$ and computes $X_1 = g^{x_1}, X_2 = g^{x_2}, Y_1 = g^{y_1}, Y_2 = g^{y_2}$. In addition, K chooses a hash key κ from a hash key space $Hkey(1^k)$. The hash key κ indicates a specific hash function H_κ with values in $\mathbf{Z}_q$ in a hash function family $Hfam(1^k)$. K sets $\mathtt{pk} = (q, g, X_1, X_2, Y_1, Y_2, \kappa)$ and $\mathtt{sk} = (q, g, x_1, x_2, y_1, y_2, \kappa)$. Then K returns (pk, sk).

On an input pk, the encapsulation algorithm Enc runs as follows. Enc chooses a from $\mathbf{Z}_q$ at random and computes $h = g^a$ and the hash value $\tau \leftarrow H_\kappa(h)$. Then Enc computes $d_1 = (X_1^\tau Y_1)^a, d_2 = (X_2^\tau Y_2)^a$ and $K = X_1^a$. The random string is K and its encapsulation is $\psi = (h, d_1, d_2)$. Note here that $(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ is a twin DH tuple. Enc returns the pair (K, ψ).

On an input sk and $\psi = (h, d_1, d_2)$, the decapsulation algorithm Dec runs as follows. Dec computes the hash value $\tau \leftarrow H_\kappa(h)$. Then Dec verifies whether $\psi = (h, d_1, d_2)$ is a consistent ciphertext, that is, whether $(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ is a twin DH tuple or not. For this sake, Dec checks whether $h^{\tau x_1 + y_1} = d_1$ and $h^{\tau x_2 + y_2} = d_2$ hold. If at least one of them does not hold, then Dec puts $\widehat{K} =\perp$. Otherwise, Dec computes the decapsulation $\widehat{K} = h^{x_1}$. Then Dec returns $\widehat{K}$.

The security of KEM3 is stated as follows.

Theorem 6. *The key encapsulation mechanism KEM3 is one-way-CCA2 secure based on the CDH assumption and the target collision resistance of a hash function family employed. More precisely, for any PPT one-way-CCA2 adversary $\mathcal{A}$ on KEM3 that queries decapsulation oracle at most q_{dec} times, there exist a PPT CDH problem solver $\mathcal{S}$ on Grp and a PPT collision-finder $\mathcal{CF}$ on Hfam which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM1}}(k) \leqslant \frac{q_{dec}}{q} + \mathbf{Adv}^{\text{cdh}}_{\mathcal{S},\mathtt{Grp}}(k) + \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k).$$

8.2 Proof of Theorem 6

An outline is stated as follows. The proof goes almost the same way as the proof of Theorem 5 except that we have to treat an argument for the Trapdoor Test (Lemma 1) which decides whether a given tuple is a twin DH tuple or not.

Given (q, g, V, W) as an input;
Initial Setting
– Initialize its inner state
– $a^* \leftarrow \mathbf{Z}_q, h^* := Wg^{a^*}$
– $\kappa \leftarrow Hkey(1^k), \tau^* \leftarrow H_\kappa(h^*)$
– $r, s \leftarrow \mathbf{Z}_q, X_1 := V, X_2 := X_1^{-r} g^s$
– $u_1, u_2 \leftarrow \mathbf{Z}_q, Y_1 := X_1^{-\tau^*} g^{u_1}, Y_2 := X_2^{-\tau^*} g^{u_2}$
– $d_1^* := (h^*)^{u_1}, d_2^* := (h^*)^{u_2}$
– $\mathtt{pk} := (q, g, X_1, X_2, Y_1, Y_2, \kappa), \psi^* := (h^*, d_1^*, d_2^*)$
– Invoke $\mathcal{A}$ on pk and ψ^*
Answering $\mathcal{A}$'s Queries and Extracting the Answer from Return
– In the case that $\mathcal{A}$ queries $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (h, d_1, d_2)$
- If $\psi = \psi^*$, then put $\widehat{K} :=\perp$
- else (: the case Consistency-Check)
  $\tau \leftarrow H_\kappa(h), \widehat{Y} := h^{\tau - \tau^*}, \widehat{Z_1} := d_1/h^{u_1}, \widehat{Z_2} := d_2/h^{u_2}$
  If $\widehat{Z_1}^r \widehat{Z_2} \neq \widehat{Y}^s$, then $\widehat{K} :=\perp$
  else
    If $\tau \neq \tau^*$, then $\widehat{K} := \widehat{Z_1}^{1/(\tau - \tau^*)}$ (: the case Simdec)
    else abort (: the case Abort)
- Reply $\widehat{K}$ to $\mathcal{A}$

– In the case that $\mathcal{A}$ returns $\widehat{K^*}$ as the answer for ψ^*
- Return $Z := \widehat{K^*}/X_1^{a^*}$

Fig. 10 A CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 6.

Let us proceed in detail. Let $\mathcal{A}$ be any given adversary that attacks KEM3 in one-way-CCA2 setting. Using $\mathcal{A}$ as a subroutine, we construct a PPT CDH problem solver $\mathcal{S}$ as shown in Fig. 10, where the algebraic trick [3] and the Twin Diffie-Hellman technique [13] are essentially used.

$\mathcal{S}$ is given $q, g, V = g^v, W = g^w$ as an input, where v and w are random and unknown to $\mathcal{S}$. $\mathcal{S}$ initializes its inner state. $\mathcal{S}$ chooses a^* from $\mathbf{Z}_q$ at random and computes $h^* = Wg^{a^*}$. Then $\mathcal{S}$ chooses κ from $Hkey(1^k)$ and computes $\tau^* \leftarrow H_\kappa(h^*)$. $\mathcal{S}$ chooses r and s from $\mathbf{Z}_q$ at random, and puts $X_1 = V, X_2 = X_1^{-r} g^s$. $\mathcal{S}$ chooses u_1 and u_2 from $\mathbf{Z}_q$ at random, and computes $Y_1 = X_1^{-\tau^*} g^{u_1}, Y_2 = X_2^{-\tau^*} g^{u_2}$. $\mathcal{S}$ computes $d_1^* = (h^*)^{u_1}, d_2^* = (h^*)^{u_2}$. $\mathcal{S}$ sets $\mathtt{pk} = (q, g, X_1, X_2, Y_1, Y_2, \kappa), \psi^* = (h^*, d_1^*, d_2^*)$ and invokes $\mathcal{A}$ on an input $(\mathtt{pk}, \psi^*)$.

Note that pk is correctly distributed. Note also that $\mathcal{S}$ does not know x_1, x_2, y_1, y_2 at all, where x_1, x_2, y_1, y_2 denote the discrete log of X_1, X_2, Y_1, Y_2 on a base g, respectively. Here the following holds.

$$y_i = -\tau^* x_i + u_i, \quad i = 1, 2. \tag{1}$$

$\mathcal{S}$ replies to $\mathcal{A}$'s queries as follows.

In the case that $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (h, d_1, d_2)$, $\mathcal{S}$ checks whether ψ is equal to ψ^* or not. If $\psi = \psi^*$, then $\mathcal{S}$ puts $\widehat{K} =\perp$. Otherwise, $\mathcal{S}$ computes $\tau \leftarrow H_\kappa(h)$ and verifies whether $\psi = (h, d_1, d_2)$ is consistent or not (call this case Consistency-Check). That is, $\mathcal{S}$ verifies whether $(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ is a twin DH tuple as follows. Put $\widehat{Y} = h^{\tau - \tau^*}, \widehat{Z_1} = d_1/h^{u_1}$ and $\widehat{Z_2} = d_2/h^{u_2}$. If $\widehat{Z_1}^r \widehat{Z_2} \neq \widehat{Y}^s$, then it is not a twin DH tuple and $\mathcal{S}$ puts $\widehat{K} =\perp$. Otherwise, $\mathcal{S}$ *decides* that it is a twin DH tuple. Then, if $\tau \neq \tau^*$, $\mathcal{S}$ computes $\widehat{K} = \widehat{Z_1}^{1/(\tau - \tau^*)}$ (call

this case SIMDEC). Otherwise ($\tau = \tau^*$), $\mathcal{S}$ aborts (call this case ABORT). $\mathcal{S}$ replies $\widehat{K}$ to $\mathcal{A}$ except for the case ABORT.

In the case that $\mathcal{A}$ returns $\widehat{K^*}$, $\mathcal{S}$ returns $Z = \widehat{K^*}/X^{a^*}$.

$\mathcal{S}$ is able to simulate the real view of $\mathcal{A}$ perfectly until the case ABORT happens, except for a negligible case, as we see below.

Firstly, the challenge ciphertext $\psi^* = (h^*, d_1^*, d_2^*)$ is consistent and correctly distributed. This is because the distribution of $\psi^* = (h^*, d_1^*, d_2^*)$ is equal to that of the real consistent ciphertext $\psi = (h, d_1, d_2)$. To see it, note that $w + a^*$ is substituted for a;

$$h^* = Wg^{a^*} = g^{w+a^*},$$
$$d_i^* = (g^{w+a^*})^{u_i} = (g^{u_i})^{w+a^*} = (X_i^{\tau^*} Y_i)^{w+a^*}, \quad i = 1, 2.$$

Secondly, $\mathcal{S}$ can simulate the decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ perfectly except for a negligible case. To see it, note that the consistency check really works though it may involve a negligible error case, which is explained by the following two claims.

Claim 8.1. *$(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ is a twin DH tuple if and only if $(g, X_1, X_2, \widehat{Y}, \widehat{Z_1}, \widehat{Z_2})$ is a twin DH tuple for $\widehat{Y} = h^{\tau-\tau^*}, \widehat{Z_1} = d_1/h^{u_1}$ and $\widehat{Z_2} = d_2/h^{u_2}$.*

Proof of Claim 8.1. This claim is proven by direct calculations. See Appendix E.

Claim 8.2. *If $\widehat{Z_1}^r \widehat{Z_2} = \widehat{Y}^s$ holds for $\widehat{Y} = h^{\tau-\tau^*}, \widehat{Z_1} = d_1/h^{u_1}$ and $\widehat{Z_2} = d_2/h^{u_2}$, then $(g, X_1, X_2, \widehat{Y}, \widehat{Z_1}, \widehat{Z_2})$ is a twin DH tuple except for an error case that occurs at most $1/q$ probability. Conversely, if $(g, X_1, X_2, \widehat{Y}, \widehat{Z_1}, \widehat{Z_2})$ is a twin DH tuple, then $\widehat{Z_1}^r \widehat{Z_2} = \widehat{Y}^s$ certainly holds.*

Proof of Claim 8.2. We observe that each of $\widehat{Y} = h^{\tau-\tau^*}, \widehat{Z_1} = d_1/h^{u_1}$ and $\widehat{Z_2} = d_2/h^{u_2}$ is given independently of r. So we can apply Lemma 1 in Sect. 2. (*Q.E.D.*)

Let us define the event OVERLOOK as:

$$\text{OVERLOOK} \stackrel{\text{def}}{=} \begin{cases} & \widehat{Z_1}^r \widehat{Z_2} = \widehat{Y}^s \text{ holds} \\ \text{and} & (g, X_1, X_2, \widehat{Y}, \widehat{Z_1}, \widehat{Z_2}) \text{ is not} \\ & \quad \text{a twin DH tuple.} \end{cases}$$

Then, by Claim 8.2, the probability that OVERLOOK occurs is at most $1/q$ for each consistency check. So for at most q_{dec} consistency checks, CONSISTENCY-CHECK$_i$, $i = 1, \ldots, q_{dec}$, the probability that at least one corresponding OVERLOOK$_i$ occurs is at most q_{dec}/q. That is;

$$\Pr\left[\bigvee_{i=1}^{q_{dec}} \text{OVERLOOK}_i\right] \leqslant \frac{q_{dec}}{q}. \tag{2}$$

q_{dec} is polynomial and q is exponential in k, so the right hand side is negligible in k.

Suppose $\mathcal{S}$ has confirmed that a decapsulation query $\psi = (h, d_1, d_2)$ passed the consistency check. In that case, $(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ is a twin DH tuple (except for a negligible case OVERLOOK), so $d_1 = h^{\tau x_1 + y_1}$ holds. If, in addition, $\mathcal{S}$ is in the case SIMDEC (that is, $\tau \neq \tau^*$), then the answer $\widehat{K} = \widehat{Z_1}^{1/(\tau-\tau^*)}$ of $\mathcal{S}$ to $\mathcal{A}$ is correct. This is because $\widehat{K} = (d_1/h^{u_1})^{1/(\tau-\tau^*)}$ is equal to h^{x_1} by the following equalities.

$$d_1/h^{u_1} = h^{\tau x_1 + y_1 - u_1} = h^{(\tau-\tau^*)x_1 + (\tau^* x_1 + y_1 - u_1)} = h^{(\tau-\tau^*)x_1},$$

where we use the equality (1).

As a whole, $\mathcal{S}$ simulates the real view of $\mathcal{A}$ perfectly until the case ABORT happens except for the negligible case OVERLOOK.

Now we evaluate the CDH advantage of $\mathcal{S}$. When $\mathcal{A}$ wins, $(g, X_1, h^*, \widehat{K^*})$ is a DH tuple, so the following holds (note that we have set $X_1 = V$, so $x_1 = v$);

$$\widehat{K^*} = g^{x_1(w+a^*)} = g^{vw} X_1^{a^*}.$$

Hence the return Z is equal to $\widehat{K^*}/X_1^{a^*} = g^{vw}$, which is the correct answer for the input (g, V, W). That is, $\mathcal{S}$ wins. Therefore, the probability that $\mathcal{S}$ wins is lower bounded by the probability that $\mathcal{A}$ wins, OVERLOOK$_i$ never occurs for $i = 1, \ldots, q_{dec}$ and ABORT does not happen:

$$\begin{aligned}
&\Pr[\mathcal{S} \text{ wins}] \\
&\geqslant \Pr\left[\mathcal{A} \text{ wins } \wedge \left(\bigwedge_{i=1}^{q_{dec}} (\neg \text{OVERLOOK}_i)\right) \wedge (\neg \text{ABORT})\right] \\
&\geqslant \Pr[\mathcal{A} \text{ wins}] - \Pr\left[\left(\bigvee_{i=1}^{q_{dec}} \text{OVERLOOK}_i\right) \vee \text{ABORT}\right] \\
&= \Pr[\mathcal{A} \text{ wins}] \\
&\quad - \left(\Pr\left[\bigvee_{i=1}^{q_{dec}} \text{OVERLOOK}_i\right] + \Pr\left[\left(\bigwedge_{i=1}^{q_{dec}} (\neg \text{OVERLOOK}_i)\right) \wedge \text{ABORT}\right]\right).
\end{aligned}$$

Using the inequality (2), we get:

$$\begin{aligned}
\mathbf{Adv}^{\text{cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \geqslant &\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM3}}(k) - \frac{q_{dec}}{q} \\
&- \Pr\left[\left(\bigwedge_{i=1}^{q_{dec}} (\neg \text{OVERLOOK}_i)\right) \wedge \text{ABORT}\right].
\end{aligned}$$

So our task being left is to show that the last term probability is negligible in k.

Claim 8.3. *The probability that $(\bigwedge_{i=1}^{q_{dec}} (\neg \text{OVERLOOK}_i)) \wedge$ ABORT occurs is negligible in k.*

Proof of Claim 8.3. Using $\mathcal{A}$ as a subroutine, we construct a PPT target collision finder $\mathcal{CF}$ on *Hfam* as follows. Given 1^k as an input, $\mathcal{CF}$ initializes its inner state. $\mathcal{CF}$ gets (q, g) from $\mathtt{Grp}(1^k)$. $\mathcal{CF}$ chooses $a^* \in \mathbf{Z}_q$ at random, computes $h^* = g^{a^*}$ and outputs h^*. $\mathcal{CF}$ receives a random hash key κ and computes $\tau^* \leftarrow H_\kappa(h^*)$. Then $\mathcal{CF}$ makes a secret key and public key honestly by itself: $\mathtt{sk} = (q, g, x_1, x_2, y_1, y_2, \kappa), \mathtt{pk} = (q, g, X_1, X_2, Y_1, Y_2, \kappa)$. Finally, $\mathcal{CF}$ computes $d_1^* = (X_1^{\tau^*} Y_1)^{a^*}, d_2^* = (X_2^{\tau^*} Y_2)^{a^*}$ and puts

$\psi^* = (h^*, d_1^*, d_2^*)$. $\mathcal{CF}$ invokes $\mathcal{A}$ on $\mathtt{pk}$ and ψ^*.

In the case that $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (h, d_1, d_2)$, $\mathcal{CF}$ checks whether ψ is equal to ψ^* or not. If $\psi = \psi^*$, then $\mathcal{CF}$ replies $K = \perp$ to $\mathcal{A}$. Otherwise ($\psi \neq \psi^*$), $\mathcal{CF}$ computes $\tau \leftarrow H_\kappa(h)$ and verifies whether $\psi = (h, d_1, d_2)$ is consistent. $\mathcal{CF}$ can do it in the same way as $\mathtt{Dec}$ does because $\mathcal{CF}$ has the secret key $\mathtt{sk}$. If it is not consistent, $\mathcal{CF}$ replies $K = \perp$ to $\mathcal{A}$. Otherwise, if $\tau \neq \tau^*$, then $\mathcal{CF}$ replies $K = h^{x_1}$ to $\mathcal{A}$. Else if $\tau = \tau^*$, then $\mathcal{CF}$ returns h and stops (call this case Collision).

The view of $\mathcal{A}$ in $\mathcal{CF}$ is the same as the real view until the case Collision happens.

Observe here the following. If Overlook never occurs in $\mathcal{S}$, then only consistent queries (ψs) have the chance to cause a collision ($\tau = \tau^*$) as is the case in $\mathcal{CF}$. Hence we have:

$$\Pr\left[\left(\bigwedge_{i=1}^{q_{dec}} (\neg \text{Overlook}_i)\right) \wedge \text{Abort}\right] \leqslant \Pr[\text{Collision}]. \quad (3)$$

On the other hand, notice that Collision implies the following.

$$\begin{cases} & (g, X_1^{\tau^*} Y_1, X_2^{\tau^*} Y_2, h^*, d_1^*, d_2^*)\text{: a twin DH tuple} \\ \text{and} & \exists (g, X_1^{\tau} Y_1, X_2^{\tau} Y_2, h, d_1, d_2)\text{: a twin DH tuple} \\ \text{and} & \tau = \tau^*. \end{cases}$$

If in addition to the above conditions h were equal to h^*, then (d_1, d_2) would be equal to (d_1^*, d_2^*), which would mean that $\mathcal{A}$ queried the challenge ciphertext ψ^* to its decapsulation oracle. This is ruled out. Hence it must hold that:

$$h \neq h^*.$$

Namely, in the case Collision, $\mathcal{CF}$ succeeds in obtaining a target collision. So we have:

$$\Pr[\text{Collision}] = \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k). \quad (4)$$

Combining (3) and (4), we get

$$\Pr\left[\left(\bigwedge_{i=1}^{q_{dec}} (\neg \text{Overlook}_i)\right) \wedge \text{Abort}\right] \leqslant \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k).$$

The right hand side is negligible in k by the assumption in Theorem 6. (*Q.E.D.*)

8.3 Discussion

To reduce the length of ciphertext $\psi = (h, d_1, d_2)$, we can replace the term d_2 with its hash value $v_2 := H_\kappa(d_2)$. Let us call this KEM $\mathtt{KEM\tilde{3}}$. The security statement and the obtained ID scheme is described in Appendix F.

9. Efficiency Comparison

In this section, we evaluate the efficiency of the ID schemes from our KEMs comparing with other ID schemes, especially, which are secure against concurrent man-in-the-middle attacks in the standard model. Among ID schemes secure against concurrent man-in-the-middle attacks, the ID scheme from $\mathtt{KEM2}$ shows the highest performance in both computational amount and message length. Among ID schemes whose security is based on the CDH assumption, the ID scheme from $\mathtt{KEM\tilde{3}}$ shows the highest performance in both.

Comparable ID schemes are divided into four categories. The first category is Σ-protocols, the second category is challenge-and-response ID schemes obtained from EUF-CMA secure signature schemes, the third category is the ones obtained from IND-CCA2 secure (non-hybrid) encryption schemes and the fourth category is the ones obtained from KEMs.

In the first category, to the best of our knowledge, the Gennaro scheme [20] is the most efficient. Here we consider one of the schemes in [20], that is, the Schnorr ID scheme plus a multi-trapdoor commitment scheme in the RSA setting. But it is no more efficient than the ID scheme obtained from the Cramer-Shoup encryption [14] (Cramer-Shoup ID, for short).

In the second category, all the known signature schemes in the standard model, including the Short Signature [4] and the Waters Signature [40], are costly as ID schemes than Cramer-Shoup ID.

In the third category, Cramer-Shoup ID is the most efficient.

In the fourth category, Cramer-Shoup KEM (CS3b in [15]) (which we denote as $\mathtt{CSKEM}$), which is IND-CCA2 secure based on the DDH assumption, is the most efficient. We remark that the KEM part of Kurosawa-Desmedt hybrid encryption scheme [30] is not comparable because the KEM is not one-way-CCA2 secure [23]. Among ID schemes whose security is based on the CDH assumption, an ID scheme from the one-way-CCA2 secure KEM of Hanaoka-Kurosawa [24] (which we denote as $\mathtt{HKKEM}$) is the most comparable.

Table 2 shows a comparison of these ID schemes, plus the Schnorr ID scheme and an ID scheme from El Gamal KEM, with the ID schemes from our KEMs.

In Table 2, we compare computational amount by counting the number of exponentiations in a prover $\mathtt{P}$ and a verifier $\mathtt{V}$. We also compare the maximum message length.

In Computational Amount column, notation (s, t) or (s, t, u) means that the prover $\mathtt{P}$ or the verifier $\mathtt{V}$ includes s single exponentiations, t double exponentiations and u triple exponentiations of the form g^a, $g^a h^b$ and $g^a h^b i^c$, respectively.

A simple estimation, which is denoted by index 1, is done by evaluating the amount of a double and a triple exponentiation as two times and three times as large as the amount of a single exponentiation, respectively. That is, the total amount $T_1(\mathtt{P})$ for a prover $\mathtt{P}$ and $T_1(\mathtt{V})$ for a verifier $\mathtt{V}$

Table 2 Efficiency Comparison of the ID Schemes from Our KEMs with Previous ID Schemes and KEMs.
OMDL, GDL, SDH, GCDH and SRSA mean assumptions of the One-More DL, the Gap-DL, the Strong DH, the Gap-CDH and the Strong RSA, respectively.
OW-CCA* means the one-way-CCA* security and ca means concurrent (two-phase) attack.
(s, t) (or (s, t, u)) means s single exponentiations, t double exponentiations (and u triple exponentiations).
kg and vf mean a key generation and a verification of a one-time signature, respectively.
$T_1(\mathtt{P}) := s + 2t$ and $T_1(\mathtt{V}) := s + 2t + 3u$ (plus 0.1 or 0.05 for kg or vf, respectively) and
$T_2(\mathtt{P}) := s + 1.75t$ and $T_2(\mathtt{V}) := s + 1.75t + 1.875u$ (plus 0.1 or 0.05 for kg or vf, respectively).
$T_1 := T_1(\mathtt{P}) + T_1(\mathtt{V})$ and $T_2 := T_2(\mathtt{P}) + T_2(\mathtt{V})$.
G and h mean an element in G_q and a hash value in $\mathbf{Z}_q$, respectively.
vk and σ mean a verification key and a signature of a one-time signature, respectively.

ID Sch. or KEM	Security Assump.	Security as KEM	Security as ID Sch.	Computational Amount							Max.Msg. Length
				P	$T_1(\mathtt{P})$	$T_2(\mathtt{P})$	V	$T_1(\mathtt{V})$	$T_2(\mathtt{V})$	$T1$	$T2$	
SchID	OMDL	-	ca	(1,0)	1.00	1.00	(0,1,0)	2.00	1.75	3.00	2.75	G
GenID	DL&SRSA	-	cMiM	(3,1)+kg	5.10	4.85	$(\frac{1}{2},2,0)$+vf	4.55	4.05	9.65	8.90	$G + \mathrm{vk}$
EGKEM	GDL&KEA	OW-CCA1	ca	(1,0)	1.00	1.00	(2,0,0)	2.00	2.00	3.00	3.00	G
CSKEM	DDH	IND-CCA2	cMiM	(3,0)	3.00	3.00	(3,1,0)	5.00	4.75	8.00	7.75	$3G$
HKKEM	CDH	OW-CCA2	cMiM	(3,0)	3.00	3.00	(2,0,2)	8.00	5.75	11.00	8.75	$3G$
Our tKEM	GCDH(stag)	OW-CCA2	cMiM	(2,0)	2.00	2.00	(2,1,0)	4.00	3.75	6.00	5.75	$2G$
Our KEM1	GCDH	OW-CCA2	cMiM	(2,0)+vf	2.05	2.05	(2,1,0)+kg	4.10	3.85	6.15	5.90	$2G + \mathrm{vk} + \sigma$
Our KEM2	GCDH	OW-CCA2	cMiM	(2,0)	2.00	2.00	(2,1,0)	4.00	3.75	6.00	5.75	$2G$
Our KEM3	CDH	OW-CCA2	cMiM	(3,0)	3.00	3.00	(2,2,0)	6.00	5.50	9.00	8.50	$3G$
Our KEM$\tilde{3}$	CDH	OW-CCA2	cMiM	(3,0)	3.00	3.00	(2,2,0)	6.00	5.50	9.00	8.50	$2G + h$

are evaluated as $s + 2t$ and $s + 2t + 3u$ times as large as a signal exponentiation, respectively.

When some techniques for exponentiations are applicable, better estimations can be achieved. Here we only consider a basic technique for modular exponentiations by El Gamal and Shamir [18], [42]. By that technique, a double exponentiation is 1.75 times as large as a single exponentiation and a triple exponentiation is 1.875 times as large as a single exponentiation. As a result, $T_1(\mathtt{P})$ and $T_1(\mathtt{V})$ are evaluated as $s + 1.75t$ and $s + 1.75t + 1.875u$ times as large as a single exponentiation, respectively.

We note that a one-time signature does not cost so much. In Table 2, the Lamport signature [31] in mind, we estimate that a key generation and a verification amount to at most 0.10 times and 0.05 times as large as a single exponentiation, respectively (see [9], for example).

T_1 and T_2 in Computational Amount column of Table 2 show total computational amount for each ID scheme. That is, $T_1 = T_1(\mathtt{P}) + T_1(\mathtt{V})$ and $T_2 = T_2(\mathtt{P}) + T_2(\mathtt{V})$.

As shown in the column, the ID scheme from our KEM2 achieves $T_1 = 6.00, T_2 = 5.75$ with the maximum message length $2G$, which is the highest performance among ID schemes secure against cMiM attacks.

Also, the ID scheme from our KEM$\tilde{3}$ shows the highest performance in both computational amount ($T_1 = 9.00, T_2 = 8.50$) and message length ($2G + h$) among ID schemes secure against cMiM attacks based on the CDH assumption. Our KEM$\tilde{3}$ reduces T_1 by 2.00 exponentiations and T_2 by 0.25 exponentiations from those of HKKEM.

10. Conclusions

We proposed a generic conversion from a KEM to an ID scheme. Then, starting with El Gamal KEM, we develop a series of five one-way-CCA2 secure KEMs applying techniques such as the selective tag with the algebraic trick [3], [27], the CHK transformation [12], the target collision-resistant hash functions, the Twin Diffie-Hellman technique [13] and a modification for shortening message length. The ID schemes from our KEMs show the highest performance in both computational amount and message length compared with previous ID schemes secure against cMiM attacks.

Acknowledgements

We appreciate sincere suggestions offered by Prof. Kiltz [29] and we would like to thank Prof. Kurosawa for inspiring words, both at ProvSec 2010. We also thank anonymous reviewers of AfricaCrypt 2011 for careful reading and valuable comments.

References

[1] H. Anada and S. Arita, "Identification schemes of proofs of ability secure against concurrent man-in-the-middle attacks," Proc. ProvSec 2010, Malacca, Malaysia, Oct. 2010, Lect. Notes Comput. Sci., vol.6402, pp.18–34, Springer-Verlag, Heidelberg, Germany.

[2] H. Anada and S. Arita, "Identification schemes from key encapsulation mechanisms," Proc. AFRICACRYPT 2011, Dakar, Senegal, July 2011, Lect. Notes Comput. Sci., vol.6737, pp.59–76, Springer-Verlag, Heidelberg, Germany.

[3] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," Proc. EUROCRYPT 2004, Interlaken, Switzerland, May 2004, Lect. Notes Comput. Sci., vol.3027, pp.223–238, Springer-Verlag, Heidelberg, Germany.

[4] D. Boneh and X. Boyen, "Short signatures without random oracles," Proc. EUROCRYPT 2004, Interlaken, Switzerland, May 2004, Lect. Notes Comput. Sci., vol.3027, pp.56–73, Springer-Verlag, Heidelberg, Germany.

[5] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali, "Identification protocols secure against reset attacks," Proc. EUROCRYPT 2001, Innsbruck, Austria, May 2001, Lect. Notes Comput. Sci.,

vol.2045, pp.495–511, Springer-Verlag, Heidelberg, Germany.
[6] M. Bellare and O. Goldreich, "On defining proofs of knowledge," Proc. CRYPTO'92, Santa Barbara, CA, USA, Aug. 1992, Lect. Notes Comput. Sci., vol.740, pp.390–420, Springer-Verlag, Heidelberg, Germany.
[7] M. Bellare and A. Palacio, "GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks," Proc. CRYPTO 2002, Santa Barbara, CA, USA, Aug. 2002, Lect. Notes Comput. Sci., vol.2442, pp.162–177, Springer-Verlag, Heidelberg, Germany.
[8] M. Bellare and A. Palacio, "The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols," Proc. CRYPTO 2004, Santa Barbara, CA, USA, Aug. 2004, Lect. Notes Comput. Sci., vol.3152, pp.273–289, Springer-Verlag, Heidelberg, Germany.
[9] M. Bellare and S. Shoup, "Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles," Proc. PKC 2007, Beijing, China, April 2007, Lect. Notes Comput. Sci., vol.4450, pp.201–216, Springer-Verlag, Heidelberg, Germany.
[10] R. Canetti and R.R. Dakdouk, "Extractable perfectly one-way functions," Proc. ICALP 2008, Reykjavik, Iceland, July 2008, Lect. Notes Comput. Sci., vol.5126, pp.449–460, Springer-Verlag, Heidelberg, Germany.
[11] R. Cramer, I. Damgård, and J.B. Nielsen, "Multiparty computation from threshold homomorphic encryption," Proc. EUROCRYPT 2001, Innsbruck, Austria, May 2001, Lect. Notes Comput. Sci., vol.2045, pp.280–300, Springer-Verlag, Heidelberg, Germany.
[12] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," Proc. EUROCRYPT 2004, Interlaken, Switzerland, May 2004, Lect. Notes Comput. Sci., vol.3027, pp.207–222, Springer-Verlag, Heidelberg, Germany.
[13] D. Cash, E. Kiltz, and V. Shoup, "The twin Diffie-Hellman problem and applications," Proc. EUROCRYPT 2008, Istanbul, Turkey, April 2008, Lect. Notes Comput. Sci., vol.4965, pp.127–145, Springer-Verlag, Heidelberg, Germany. Full version available at Cryptology ePrint Archive, 2008/067, http://eprint.iacr.org/
[14] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," Proc. CRYPTO'98, Santa Barbara, CA, USA, Aug. 1998, Lect. Notes Comput. Sci., vol.1462, pp.13–25, Springer-Verlag, Heidelberg, Germany.
[15] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," SIAM J. Comput., vol.33, no.1, pp.167–226, Aug. 2003.
[16] R.R. Dakdouk, Theory and Application of Extractable Functions, Doctor of Philosophy Dissertation, Yale University, New Haven, CT, USA, 2009.
[17] I. Damgård, "Towards practical public key systems secure against chosen ciphertext attacks," Proc. CRYPTO'91, Santa Barbara, CA, USA, Aug. 1991, Lect. Notes Comput. Sci., vol.576, pp.445–456, Springer-Verlag, Heidelberg, Germany.
[18] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," Proc. CRYPTO'84, Santa Barbara, California, USA, Aug. 1984, Lect. Notes Comput. Sci., vol.196, pp.10–18, Springer-Verlag, Heidelberg, Germany.
[19] E. Fujisaki, "New constructions of efficient simulation-sound commitments using encryption," Proc. 2011 Symposium on Cryptography and Information Security, SCIS 2011, Kokura, Japan, Jan. 2011, 1A2-3, The Institute of Electronics, Information and Communication Engineers, Tokyo, Japan.
[20] R. Gennaro, "Multi-trapdoor commitments and their applications to non-malleable protocols," Proc. CRYPTO 2004, Santa Barbara, CA, USA, Aug. 2004, Lect. Notes Comput. Sci., vol.3152, pp.220–236, Springer-Verlag, Heidelberg, Germany.
[21] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," SIAM J. Comput., vol.18, no.1, pp.186–208, Feb. 1989.
[22] L. Guillou and J.J. Quisquater, "A paradoxical identity-based signature scheme resulting from zero-knowledge," Proc. CRYPTO'88, Santa Barbara, CA, USA, Aug. 1988, Lect. Notes Comput. Sci., vol.403, pp.216–231, Springer-Verlag, Heidelberg, Germany.
[23] J. Herranz, D. Hofheinz, and E. Kiltz, "The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure," Cryptology ePrint Archive, 2006/207, http://eprint.iacr.org/
[24] G. Hanaoka and K. Kurosawa, "Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption," Proc. ASIACRYPT 2008, Melbourne, Australia, Dec. 2008, Lect. Notes Comput. Sci., vol.5350, pp.308–325, Springer-Verlag, Heidelberg, Germany. Full version available at Cryptology ePrint Archive, 2008/211, http://eprint.iacr.org/
[25] J. Katz, Efficient Cryptographic Protocols Preventing "Man-in-the-Middle" Attacks, Doctor of Philosophy Dissertation, Columbia University, New York, NY, USA, 2002.
[26] J. Katz, "Efficient and non-malleable proofs of plaintext knowledge and applications," Proc. EUROCRYPT 2003, Warsaw, Poland, May 2003, Lect. Notes Comput. Sci., vol.2656, pp.211–228, Springer-Verlag, Heidelberg, Germany.
[27] E. Kiltz, "Chosen-ciphertext security from tag-based encryption," Proc. TCC 2006, New York, NY, USA, March 2006, Lect. Notes Comput. Sci., vol.3876, pp.581–600, Springer-Verlag, Heidelberg, Germany.
[28] E. Kiltz, "Chosen-ciphertext security from hashed gap Diffie-Hellman," Proc. PKC 2007, New York, NY, USA, March 2006, Lect. Notes Comput. Sci., vol.3876, pp.581–600, Springer-Verlag, Heidelberg, Germany.
[29] E. Kiltz, Personal communication at ProvSec 2010, Malacca, 2010.
[30] K. Kurosawa and Y. Desmedt, "A new paradigm of hybrid encryption scheme," Proc. CRYPTO 2004, Santa Barbara, CA, USA, Aug. 2004, Lect. Notes Comput. Sci., vol.3152, pp.426–442, Springer-Verlag, Heidelberg, Germany.
[31] L. Lamport, "Constructing digital signatures from a one-way function," Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.
[32] U. Maurer and S. Wolf, "Lower bounds on generic algorithms in groups," Proc. EUROCRYPT'98, Espoo, Finland, May-June 1998, Lect. Notes Comput. Sci., vol.1403, pp.72–84, Springer-Verlag, Heidelberg, Germany.
[33] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," Proc. 21st Symposium on Theory of Computing, Seattle, Washington, USA, May 1989, pp.33–43, Association for Computing Machinery, New York, USA.
[34] T. Okamoto and D. Pointcheval, "The gap-problems: A new class of problems for the security of cryptographic schemes," Proc. PKC 2001, Cheju Island, Korea, Feb. 2001, Lect. Notes Comput. Sci., vol.1992, pp.104–118, Springer-Verlag, Heidelberg, Germany.
[35] D. Pointcheval, "Chosen-ciphertext security for any one-way cryptosystem," Proc. PKC 2000, Melbourne, Victoria, Australia, Jan. 2000, Lect. Notes Comput. Sci., vol.1751, pp.129–146, Springer-Verlag, Heidelberg, Germany.
[36] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," Proc. 22nd Annual Symposium on Theory of Computing, Baltimore, MD, USA, May 1990, pp.387–384, Association for Computing Machinery, New York, USA.
[37] C.P. Schnorr, "Efficient identification and signatures for smart cards," Proc. CRYPTO'89, Santa Barbara, CA, USA, Aug. 1989, Lect. Notes Comput. Sci., vol.435, pp.239–252, Springer-Verlag, Heidelberg, Germany.
[38] D.R. Stinson and J. Wu, "An efficient and secure two-flow zero-knowledge identification protocol," J. Mathematical Cryptology, vol.1, no.3, pp.201–220, Aug. 2007.
[39] J. Wu and D.R. Stinson, "An efficient identification protocol and the knowledge-of-exponent assumption," Cryptology ePrint Archive, 2007/479, http://eprint.iacr.org/
[40] B. Waters, "Efficient identity-based encryption without random oracles," Proc. EUROCRYPT 2005, Aarhus, Denmark, May 2005,

Lect. Notes Comput. Sci., vol.3494, pp.114–127, Springer-Verlag, Heidelberg, Germany.

[41] S. Yilek, "Resettable public-key encryption: How to encrypt on a virtual machine," Proc. the Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 2010, Lect. Notes Comput. Sci., vol.5985, pp.41–56, Springer-Verlag, Heidelberg, Germany.

[42] S.M. Yen, C.S. Laih, and A.K. Lenstra, "Multi-exponentiation," IEE Proc. Computers and Digital Techniques, vol.141, no.6, pp.325–326, Nov. 1994.

Appendix A: Proof of Theorem 2

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be as in Theorem 2. Using $\mathcal{A}$ as a subroutine, we construct a Gap-DL problem solver $\mathcal{S}$. The construction is illustrated in Fig. A· 1.

$\mathcal{S}$ is given $q, g, X = g^x$ as a DL problem instance, where x is random and unknown to $\mathcal{S}$. $\mathcal{S}$ initializes its inner state, sets $\mathtt{pk} = (q, g, X)$ and invokes $\mathcal{A}_1$ on an input $\mathtt{pk}$.

In the first phase, in the case that $\mathcal{A}_1$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = h$, $\mathcal{S}$ queries its CDH oracle $\mathcal{CDH}$ for the answer for a CDH problem instance (g, h, X) and gets Z as a reply. Then $\mathcal{S}$ replies $\widehat{K} = Z$ to $\mathcal{A}$. In the case that $\mathcal{A}_1$ returns the inner state st, $\mathcal{S}$ ends the first phase and proceeds to the second phase.

In the second phase, $\mathcal{S}$ chooses a^* from $\mathbf{Z}_q$ at random and computes $\psi^* = h^* = g^{a^*}$. Then $\mathcal{S}$ invokes $\mathcal{A}_2$ on an input (st, ψ^*). In the case that $\mathcal{A}_2$ returns $\widehat{K}^*$, $\mathcal{S}$ invokes the KEA extractor $\mathcal{H}'$ on (g, h^*, st). Here $\mathcal{H}'$ is the KEA extractor associated with the algorithm $\mathcal{H}$ below.

$\mathcal{H}(g, h^*, st)$:

$\widehat{K}^* \leftarrow \mathcal{A}_2(st, h^*), Z := \widehat{K}^*$, return$(g, h^*, X, Z)$.

Note that the auxiliary input st is independent of h^*.

If $\mathcal{H}'$ returns x^*, then $\mathcal{S}$ returns x^*.

It is obvious that $\mathcal{S}$ simulates A's decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ perfectly with the aid of CDH oracle $\mathcal{CDH}$.

Now we evaluate the Gap-DL advantage of $\mathcal{S}$. Let Ext denote the event that $g^{x^*} = X$ holds (that is, $\mathcal{H}'$ succeeds in extracting the exponent of $g^{x^*} = X$). If Ext occurs, then the solver $\mathcal{S}$ wins. So we have:

$$\Pr[\mathcal{S} \text{ wins}] \geqslant \Pr[\text{Ext}].$$

Then we do the following deformation;

$$\begin{aligned}&\Pr[\mathcal{S} \text{ wins}]\\ &\geqslant \Pr[\mathcal{A} \text{ wins} \wedge \text{Ext}] + \Pr[\neg(\mathcal{A} \text{ wins}) \wedge \text{Ext}]\\ &\geqslant \Pr[\mathcal{A} \text{ wins} \wedge \text{Ext}]\\ &= \Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A} \text{ wins} \wedge \neg\text{Ext}].\end{aligned}$$

Here $\mathcal{A}$ wins if and only if $\widehat{K}^* = Z = X^{a^*}$ holds. Therefore;

$$\Pr[\mathcal{S} \text{ wins}] \geqslant \Pr[\mathcal{A} \text{ wins}] - \Pr[X^{a^*} = Z \wedge g^{x^*} \neq X].$$

That means what we want.

$$\mathbf{Adv}^{\text{gap-dl}}_{\mathcal{S},\mathtt{Grp}}(k) \geqslant \mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{EGKEM}}(k) - \mathbf{Adv}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(k).$$

(*Q.E.D.*)

Given (q, g, X) as an input;
Initial Setting
– Initialize its inner state
– $\mathtt{pk} := (q, g, X)$, invoke $\mathcal{A}_1$ on $\mathtt{pk}$
The First Phase : Answering $\mathcal{A}_1$'s Queries
– In the case that $\mathcal{A}_1$ queries $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = h$;
  • $Z \leftarrow \mathcal{CDH}(g, h, X)$, reply $\widehat{K} := Z$ to $\mathcal{A}_1$
– In the case that $\mathcal{A}_1$ returns the inner state st;
  • Proceed to the Second Phase
The Second Phase : Extracting the Answer from $\mathcal{A}_2$'s Return
  • $a^* \leftarrow \mathbf{Z}_q, \psi^* := h^* := g^{a^*}$
  • Invoke $\mathcal{A}_2$ on (st, ψ^*)
– In the case that $\mathcal{A}_2$ returns $\widehat{K}^*$;
  • Invoke $\mathcal{H}'$ on (g, h^*, st) : $x^* \leftarrow \mathcal{H}'(g, h^*, st)$
  • Return x^*

Fig. A· 1 A Gap-DL Problem Solver $\mathcal{S}$ for the Proof of Theorem 2.

Appendix B: One-Time Signatures

A *one-time signature* $\mathtt{OTS}$ is a triple of PPT algorithms $(\mathtt{SGK}, \mathtt{Sign}, \mathtt{Vrfy})$. $\mathtt{SGK}$ is a signing key generator which returns a pair $(\mathtt{vk}, \mathtt{sgk})$ of a verification key and a matching signing key on an input 1^k. $\mathtt{Sign}$ and $\mathtt{Vrfy}$ are a signing algorithm and a verification algorithm, respectively. We require $\mathtt{OTS}$ to be existentially unforgeable against chosen message attack by any PPT forger $\mathcal{F}$ (the EUF-CMA property). The following experiment represents the strong version.

$\mathbf{Exprmt}^{\text{euf-cma}}_{\mathcal{F},\mathtt{OTS}}(1^k)$
$(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k), m \leftarrow \mathcal{F}(\mathtt{vk}), \sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}(m)$,
$(m', \sigma') \leftarrow \mathcal{F}(\mathtt{vk}, (m, \sigma))$
If $\mathtt{Vrfy}_{\mathtt{vk}}(m', \sigma') = 1 \wedge (m', \sigma') \neq (m, \sigma)$
then return Win else return Lose.

We define the *EUF-CMA advantage of $\mathcal{F}$ over $\mathtt{OTS}$ in the strong sense* as follows.

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathtt{OTS}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{euf-cma}}_{\mathcal{F},\mathtt{OTS}}(1^k) \text{ returns Win}].$$

We say that $\mathtt{OTS}$ has the *one-time security in the strong sense* if, for any PPT algorithm $\mathcal{F}$, $\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathtt{OTS}}(k)$ is negligible in k. We also say that $\mathtt{OTS}$ is a *strong one-time signature*, or, $\mathtt{OTS}$ has the *EUF-CMA property in the strong sense*.

One-time signatures can be constructed based on the existence of a one-way function (for example, [31]).

Appendix C: Proof of Theorem 4

Let $\mathcal{A}$ be as in Theorem 4. Using $\mathcal{A}$ as a subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$. The construction is illustrated in Fig. A· 2.

$\mathcal{S}$ is given $q, g, V = g^v, W = g^w$ as a CDH problem

Given (q, g, V, W) as an input;
Initial Setting
– Initialize inner state
– $(\mathtt{vk}^*, \mathtt{sgk}^*) \leftarrow \mathtt{SGK}(1^k)$
– $u \leftarrow \mathbf{Z}_q, X := V, Y := X^{-\mathtt{vk}^*} g^u, \mathtt{pk} := (q, g, X, Y)$
– $a^* \leftarrow \mathbf{Z}_q, h^* := Wg^{a^*}, d^* := (h^*)^u, \sigma^* \leftarrow \mathtt{Sign}_{\mathtt{sgk}^*}((h^*, d^*))$
– $\psi^* := (\mathtt{vk}^*, (h^*, d^*), \sigma^*)$, invoke $\mathcal{A}$ on $\mathtt{pk}$ and ψ^*
Answering $\mathcal{A}$'s Queries and Extracting the Answer from Return
– In the case that $\mathcal{A}$ queries $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (\mathtt{vk}, (h, d), \sigma)$;
• If $\mathtt{Vrfy}_{\mathtt{vk}}((h, d), \sigma) \neq 1$ or $\mathcal{DDH}(g, X^{\mathtt{vk}} Y, h, d) \neq 1$ then $\widehat{K} := \perp$
• else
If $\mathtt{vk} \neq \mathtt{vk}^*$ then $\widehat{K} := (d/h^u)^{1/(\mathtt{vk}-\mathtt{vk}^*)}$ (: the case Simdec)
else abort (: the case Abort)
• Reply $\widehat{K}$ to $\mathcal{A}$
– In the case that $\mathcal{A}$ returns $\widehat{K}^*$;
• Return $Z := \widehat{K}^* / X^{a^*}$

Fig. A· 2 A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 4.

instance, where v and w are random and unknown to $\mathcal{S}$.

$\mathcal{S}$ initializes its inner state. Invoking $\mathtt{SGK}$ on an input 1^k, $\mathcal{S}$ gets $(\mathtt{vk}^*, \mathtt{sgk}^*)$. $\mathcal{S}$ chooses u from $\mathbf{Z}_q$ at random and puts $X = V$ and $Y = X^{-\mathtt{vk}^*} g^u$ and sets $\mathtt{pk} = (q, g, X, Y)$.

Note that $\mathtt{pk}$ is correctly distributed. Note also that $\mathcal{S}$ knows neither x nor y, where x and y denote the discrete log of X and Y on a base g, respectively. Here the following holds;

$$y = -\mathtt{vk}^* x + u.$$

$\mathcal{S}$ chooses a^* from $\mathbf{Z}_q$ at random and $\mathcal{S}$ puts $h^* = Wg^{a^*}$ and $d^* = (h^*)^u$. $\mathcal{S}$ gets a signature σ^* from $\mathtt{Sign}_{\mathtt{sgk}^*}((h^*, d^*))$. Then $\mathcal{S}$ puts $\psi^* = (\mathtt{vk}^*, (h^*, d^*), \sigma^*)$. $\mathcal{S}$ invokes $\mathcal{A}$ on an input $\mathtt{pk}$.

$\mathcal{S}$ replies $\mathcal{A}$ in answer to $\mathcal{A}$'s decapsulation queries as follows. In the case that $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (\mathtt{vk}, (h, d), \sigma)$, $\mathcal{S}$ verifies whether $((h, d), \sigma)$ is valid under $\mathtt{vk}$ and whether $(g, X^{\mathtt{vk}} Y, h, d)$ is a DH-tuple. $\mathcal{S}$ checks the latter by querying its DDH oracle $\mathcal{DDH}$ for the answer. If at least one of them is not satisfied, then $\mathcal{S}$ puts $\widehat{K} = \perp$.

Otherwise, if $\mathtt{vk} \neq \mathtt{vk}^*$ then $\mathcal{S}$ puts $\widehat{K} = (d/h^u)^{1/(\mathtt{vk}-\mathtt{vk}^*)}$ (call this case Simdec). If $\mathtt{vk} = \mathtt{vk}^*$, $\mathcal{S}$ aborts (call this case Abort). $\mathcal{S}$ replies $\widehat{K}$ to $\mathcal{A}$ except for the case Abort.

In the case that $\mathcal{A}$ returns $\widehat{K}^*$ as the answer for ψ^*, $\mathcal{S}$ returns $Z = \widehat{K}^* / X^{a^*}$.

$\mathcal{S}$ is able to simulate the real view of $\mathcal{A}$ perfectly until the case Abort happens, as we see below.

Firstly, the challenge ciphertext $\psi^* = (\mathtt{vk}^*, (h^*, d^*), \sigma^*)$ is consistent and correctly distributed. This is because the distribution of (h^*, d^*) is equal to that of the real consistent one (h, d). To see it, note that $w + a^*$ is substituted for a;

$$h^* = Wg^{a^*} = g^{w+a^*},$$
$$d^* = (g^{w+a^*})^u = (g^u)^{w+a^*} = (X^{\mathtt{vk}^*} Y)^{w+a^*}.$$

Secondly, in the case Simdec, $\mathcal{S}$ can simulate the decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ perfectly. This is because $\widehat{K}$ is equal to h^x by the following equalities;

$$d/h^u = h^{(\mathtt{vk})x+y-u} = h^{(\mathtt{vk}-\mathtt{vk}^*)x+(\mathtt{vk}^* x+y-u)} = h^{(\mathtt{vk}-\mathtt{vk}^*)x}.$$

As a whole, $\mathcal{S}$ simulates the real view of $\mathcal{A}$ perfectly until the case Abort happens.

Now we evaluate the Gap-CDH advantage of $\mathcal{S}$. When $\mathcal{A}$ wins, $(g, X, h^*, \widehat{K}^*)$ is a DH-tuple, so the following holds (note that we have set $X = V$, so $x = v$);

$$\widehat{K}^* = (g^x)^{w+a^*} = g^{vw} X^{a^*}.$$

So $\mathcal{S}$ wins because its return Z is g^{vw}. Therefore the probability that $\mathcal{S}$ wins is lower bounded by the probability that $\mathcal{A}$ wins and the case Abort does not happen;

$$\begin{aligned}\Pr[\mathcal{S} \text{ wins}] &\geqslant \Pr[\mathcal{A} \text{ wins } \wedge \neg \text{Abort}] \\ &\geqslant \Pr[\mathcal{A} \text{ wins}] - \Pr[\text{Abort}].\end{aligned}$$

That is;

$$\mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \geqslant \mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM1}}(k) - \Pr[\text{Abort}].$$

So our task being left is to show that $\Pr[\text{Abort}]$ is negligible in k.

Claim C.1. *The probability that the case* Abort *occurs is negligible in k.*

Proof of Claim C.1. Using $\mathcal{A}$ as a subroutine, we construct a signature forger $\mathcal{F}$ on $\mathtt{OTS}$ as follows. Given $\mathtt{vk}^*$ as an input, $\mathcal{F}$ initializes its inner state. $\mathcal{F}$ chooses x and y from $\mathbf{Z}_q$ at random and computes $X = g^x$ and $Y = g^y$. $\mathcal{F}$ chooses a^* from $\mathbf{Z}_q$ at random and computes $h^* = g^{a^*}$ and $d^* = (X^{\mathtt{vk}^*} Y)^{a^*}$. Then $\mathcal{F}$ queries its signing oracle $\mathcal{SIGN}_{\mathtt{sgk}^*}$ for a signature on a message (h^*, d^*) and gets a signature σ^*. Putting $\psi^* = (\mathtt{vk}^*, (h^*, d^*), \sigma^*)$, $\mathcal{F}$ invokes $\mathcal{A}$ on an input $(\mathtt{pk}, \psi^*)$.

In the case that $\mathcal{A}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ for the answer for $\psi = (\mathtt{vk}, (h, d), \sigma)$, $\mathcal{F}$ verifies whether the signature is valid and whether $(g, X^{\mathtt{vk}} Y, h, d)$ is a DH-tuple. For the latter, $\mathcal{F}$ does the same as an honest decapsulation algorithm does because $\mathcal{F}$ has the secret key $\mathtt{sk}$. If the signature is not valid or the tuple is not a DH-tuple, $\mathcal{F}$ sets $\widehat{K} = \perp$. Otherwise, if $\mathtt{vk} \neq \mathtt{vk}^*$, then $\mathcal{F}$ replies $\widehat{K} = h^x$ to $\mathcal{A}$. If $\mathtt{vk} = \mathtt{vk}^*$, then $\mathcal{F}$ returns $((h, d), \sigma)$ and stops (call this case Forge).

Note that the view of $\mathcal{A}$ in $\mathcal{F}$ is the same as the real view until the case Forge happens. Especially, the view of $\mathcal{A}$ in $\mathcal{F}$ is the same as the view of $\mathcal{A}$ in $\mathcal{S}$ until the case Abort or the case Forge happens. So we have:

$$\Pr[\text{Abort}] = \Pr[\text{Forge}].$$

Notice that the case Forge implies that the following equalities hold;

$$\mathtt{vk} = \mathtt{vk}^*, ((h, d), \sigma) \neq ((h^*, d^*), \sigma^*).$$

This is because that, if $((h, d), \sigma)$ were equal to $((h^*, d^*), \sigma^*)$, then $\mathcal{A}$ would have queried the challenge ciphertext ψ^* to its decapsulation oracle. This is ruled out.

Hence in the case Forge, $\mathcal{F}$ succeeds in making an existential forgery in the strong sense. That is;

$$\Pr[\textsc{Forge}] = \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathsf{OTS}}(k).$$

Combining the two equalities, we get

$$\Pr[\textsc{Abort}] = \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{F},\mathsf{OTS}}(k).$$

The right hand side is negligible in k by the assumption in Theorem 4. (*Q.E.D.*)

Appendix D: Target Collision Resistant Hash Functions

Target collision resistant (TCR) hash functions [33], [36] are treated as a family. Let us denote a function family as $Hfam(1^k) = \{H_\kappa\}_{\kappa \in Hkey(1^k)}$. Here $Hkey(1^k)$ is a hash key space, $\kappa \in Hkey(1^k)$ is a hash key and H_κ is a function from $\{0,1\}^*$ to $\{0,1\}^k$. We may assume that H_κ is from $\{0,1\}^*$ to $\mathbf{Z}_q$, where q is a prime of length k.

Given a PPT algorithm $\mathcal{CF}$, a collision finder, we consider the following experiment.

$\mathbf{Exprmt}^{\text{tcr}}_{\mathcal{CF},Hfam}(1^k)$
$m \leftarrow \mathcal{CF}(1^k), \kappa \leftarrow Hkey(1^k), m' \leftarrow \mathcal{CF}(\kappa)$
If $H_\kappa(m) = H_\kappa(m')$ and $m \neq m'$ then return Win
else return Lose.

We define the *TCR advantage of $\mathcal{CF}$ over Hfam* as follows.

$$\mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{tcr}}_{\mathcal{CF},Hfam}(1^k) \text{ returns Win}].$$

We say that *Hfam* is *a TCR function family*, or, *Hfam* has the *TCR property* if, for any PPT algorithm $\mathcal{CF}$, $\mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k)$ is negligible in k.

In theory, TCR hash function families can be constructed based on the existence of a one-way function [33], [36].

Appendix E: Proof of Claim 8.1

Assume that $(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ is a twin DH tuple and put

$$X_i^\tau Y_i =: g^{\alpha_i}, h =: g^\beta, d_i =: g^{\alpha_i\beta},\ i = 1,2.$$

Then $h^{\tau-\tau^*} = g^{\beta(\tau-\tau^*)}$. Note that we have set

$$Y_i := X_i^{-\tau^*} g^{u_i},\ i = 1,2.$$

So $X_i^\tau Y_i = X_i^\tau X_i^{-\tau^*} g^{u_i} = X_i^{\tau-\tau^*} g^{u_i}$ and we have

$$g^{\alpha_i - u_i} = X_i^{\tau-\tau^*},\ i = 1,2.$$

Hence

$$d_i/h^{u_i} = g^{\alpha_i\beta}/g^{\beta u_i} = g^{(\alpha_i-u_i)\beta} = X_i^{\beta(\tau-\tau^*)},\ i = 1,2.$$

This means $(g, X_1, X_2, \widehat{Y}, \widehat{Z_1}, \widehat{Z_2})$ is a twin DH tuple for $\widehat{Y} = h^{\tau-\tau^*}$, $\widehat{Z_1} = d_1/h^{u_1}$ and $\widehat{Z_2} = d_2/h^{u_2}$.

The converse is also verified by setting the goal to be $d_i = g^{\alpha_i\beta}$, $i = 1,2$ and starting with the assumption that $\widehat{Z_i} = d_i/h^{u_i} = X_i^{\beta(\tau-\tau^*)}$, $i = 1,2$. (*Q.E.D.*)

Appendix F: KEM$\tilde{3}$ and the Obtained ID Scheme

In KEM$\tilde{3}$, the ciphertext turns into $\psi = (h, d_1, v_2), v_2 := H_\kappa(d_2)$. So the consistency check for the index 2 becomes:

whether $H_\kappa(h^{\tau x_2 + y_2}) = v_2$ or not.

The Trapdoor Test in the security proof is deformed as follows.

$$\begin{aligned}\widehat{Z_1}^r \widehat{Z_2} = \widehat{Y}^s &\iff (d_1/h^{u_1})^r (d_2/h^{u_2}) = (h^{\tau-\tau^*})^s \\ &\iff d_1^{-r} h^{ru_1+u_2+s(\tau-\tau^*)} = d_2 \\ &\Longrightarrow H_\kappa(d_1^{-r} h^{ru_1+u_2+s(\tau-\tau^*)}) = v_2.\end{aligned}$$

The last equality may cause a collision, so the security statement for KEM$\tilde{3}$ needs the collision resistance assumption of a hash function family *Hfam* employed (the name of game "cr" in $\mathbf{Adv}^{\text{cr}}_{\mathcal{CF}',Hfam}(k)$ below means collision resistance).

Corollary of Theorem 6. *The key encapsulation mechanism KEM$\tilde{3}$ is one-way-CCA2 secure based on the CDH assumption, the target collision resistance and the collision resistance of a hash function family employed. More precisely, for any PPT one-way-CCA2 adversary $\mathcal{A}$ on KEM$\tilde{3}$ that queries its decapsulation oracle at most q_{dec} times, there exist a PPT CDH problem solver $\mathcal{S}$ on Grp, a PPT collision-finder $\mathcal{CF}$ and $\mathcal{CF}'$ on Hfam which satisfy the following tight reduction.*

$$\begin{aligned}\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathsf{KEM}\tilde{3}}(k) \leqslant &\frac{q_{dec}}{q} + \mathbf{Adv}^{\text{cdh}}_{\mathcal{S},\mathsf{Grp}}(k) + \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF},Hfam}(k) \\ &+ \mathbf{Adv}^{\text{cr}}_{\mathcal{CF}',Hfam}(k).\end{aligned}$$

The ID scheme obtained from KEM$\tilde{3}$ is shown in Fig. A·3. By the above tuning, the maximum message length reduces to two elements $2G$ of Grp plus one hash value h by H_κ.

Key Generation
– K: given 1^k as an input;
- $(q, g) \leftarrow \mathsf{Grp}(1^k), \kappa \leftarrow Hkey(1^k)$
- $x_1, x_2, y_1, y_2 \leftarrow \mathbf{Z}_q, X_1 := g^{x_1}, X_2 := g^{x_2}, Y_1 := g^{y_1}, Y_2 := g^{y_2}$
- $\mathsf{pk} := (q, g, X_1, X_2, Y_1, Y_2, \kappa), \mathsf{sk} := (q, g, x_1, x_2, y_1, y_2, \kappa)$
- Return $(\mathsf{pk}, \mathsf{sk})$

Interaction
– S: given pk as an input;
- $a \leftarrow \mathbf{Z}_q, h := g^a, \tau \leftarrow H_\kappa(h)$
- $d_1 := (X_1^\tau Y_1)^a, v_2 := H_\kappa((X_2^\tau Y_2)^a), K := X_1^a, \psi = (h, d_1, v_2)$
- Send ψ to R

– R: given sk as an input and receiving $\psi = (h, d_1, v_2)$ as an input message;
- $\tau \leftarrow H_\kappa(h)$
- If $h^{\tau x_1 + y_1} \neq d_1$ or $H_\kappa(h^{\tau x_2 + y_2}) \neq v_2$ then $\widehat{K} := \perp$ else $\widehat{K} := h^{x_1}$
- Send $\widehat{K}$ to S

– S: receiving $\widehat{K}$ as an input message;
- If $\widehat{K} = K$ then return 1 else return 0

Fig. A·3 An ID Scheme Obtained from KEM$\tilde{3}$.

Hiroaki Anada received his B.E. and M.E. in mathematics from Waseda University. He is interested in interactive proofs and has recently received Ph.D. from Institute of Information Security, Yokohama, Japan.

Seiko Arita received his B.E. and M.E. from Kyoto University, and Ph.D. from Chuo University. He has been interested in prime numbers, algebraic curves and cryptographic protocols. He is with Institute of Information Security, Yokohama, Japan.