A Study on Efficient Identification Schemes Secure against Concurrent Man-in-the-Middle Attacks

Hiroaki Anada

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Informatics
in the Graduate School of Information Security

INSTITUTE of INFORMATION SECURITY, JAPAN

2012

Abstract

We tackle the problem of constructing public key-based identification schemes (ID schemes) that are secure against concurrent man-in-the-middle attacks (cMiM attacks). Here, a concurrent man-in-the-middle attack means an active attack in which an adversary interacts with a victim verifier trying to impersonate a prover, while the adversary collects information of the secret key interacting concurrently with provers that have the same secret key.

Our approach is not based on the Σ-protocols but on key encapsulation mechanisms (KEMs). First, we propose a generic conversion from a KEM to an ID scheme. Although this is a straightforward conversion, there exists a possibility of realizing efficient cMiM secure ID schemes because a KEM only has to be secure against adaptive chosen ciphertext attacks on one-wayness (one-way-CCA2 secure) for the derived ID scheme to be cMiM secure.

Next, looking at the generic conversion as a design principle, we develop a series of concrete KEMs to get ID schemes. We start with El Gamal KEM and prove it to be secure against non-adaptive chosen ciphertext attacks on one-wayness (one-way-CCA1 secure) in the standard model based on some strong assumptions. Then, we apply a tag framework with the algebraic trick of Boneh and Boyen to El Gamal KEM to make it one-way-CCA2 secure. Here, its security is based on the Gap-Computational Diffie-Hellman (the Gap-CDH) assumption. Following this, we apply the CHK transformation and a target collision resistant hash function to exit the tag framework. 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 ID schemes obtained from our KEMs are cMiM secure and show the highest efficiency in both computational amount and message length as compared with previously known cMiM secure ID schemes.

Acknowledgements

It has been a great pleasure and an honor to study under the mentorship of Professor Seiko Arita for these three years. Prof. Arita was kind enough to take a chance when he agreed to supervise me three years ago. As an adviser, he patiently and skillfully guided me, always recommending what I should consult and what direction I should proceed in. Moreover, he has shared with me his large experience and deep perspective of the field. I appreciate all that he has done for me.

I was very lucky to meet Professor Eike Kiltz at the Fourth International Conference on Provable Security held at Malacca. The frank conversation I had with him helped progress of my study further. I would like Prof. Kiltz to accept my heartfelt acknowledgements.

I would also like to offer my sincere thanks to Professor Kaoru Kurosawa, who gave me much inspiration at Malacca. It gives me great pleasure to be able to show a gratitude to Prof. Kurosawa.

It must have been at least somewhat difficult to review this dissertation, and I offer my sincere thanks to Professor Noboru Kunihiro, Professor Kazuto Matsuo and Professor Naoshi Sato for the some.

I would like to thank Professor Hiroshi Doi, who guided me during my training. I also thank my colleague Akihiko Sakuma for the good times we had three years at the institute.

Finally, I would like to thank my parents and brothers for their warm words. I also must say that this dissertation would not have been possible without the heartfelt support and encouragement of my wife, Michiko. Last, I also find inspiration from my little daughter, Nozomi, who was born during my study.

Contents

Chapter 1

Introduction

In this chapter, we see what identification is and what kind of threats there are on identification. Then, we look at a corresponding model of a public key-based identification scheme (ID scheme) and corresponding models of attacks on an ID scheme. Next, we briefly review previous works on ID schemes and their merits and demerits. Then, we explain our strategy to overcome the demerits, and state what we have achieved as compared with related works. Finally, we explain the structure of the rest of this dissertation.

1.1 Identification and Impersonation

1.1.1 Identification

Identification is user authentication for a system or a service. In that process, a user is called a *prover* and an authenticator is called a *verifier*. It enables a prover to convince a verifier that the prover is certainly himself.

Some kinds of identification need some rounds of interaction between a prover and a verifier, where the prover tries to prove that he has some secret information which characterize himself. *Password-based* identification needs only one round and secret information is just a password. In contrast, in *public key-based* identification, a secret key which represents secret information and a public key which matches the secret key are used. A prover holds the secret key and a verifier refers to the public key. They interact for some rounds doing necessary computations until the verifier feels certain that the prover has the secret key.

By virtue of identification, only authorized persons can use a system or a service. As a result, identification makes a system or a service beneficial for users. If it were

not for identification, then someone malicious whom we call an *adversary* would login the system or the service and would steal and falsify information treated there. The aim of identification is to exclude such dangers.

1.1.2 Impersonation

Nevertheless, even if identification is composed to a system or a service, there are still threats in which adversary would intentionally try to pass identification by acting as if he would be a legitimate user having a secret key, which we call *impersonation*. Let us see some cases in the following.

Eavesdropping.

Suppose that a user enters his password into a login window in a password-based identification process. A user application (a prover `P`) sends the password to an authentication server (a verifier `V`) through a communication channel. If the password is sent without encryption, then an *eavesdropper* can easily steal the password (see Fig.1.1).

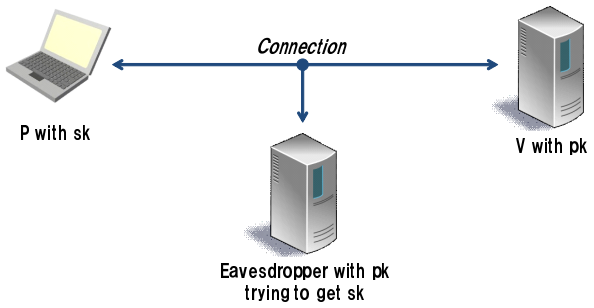

Figure 1.1: Eavesdropping.

Phishing.

Another case is what is called *phishing*. Suppose that a user often browses a website which imposes him to enter his password to login. Suppose, besides, that he usually visits the website by clicking the corresponding URL written on an e-mail. In a phishing technique, an adversary constructs a fake website that looks almost the same as the true website. Then, he sends an e-mail on which the URL of the fake website is written with attractive phrases. If the user unfortunately clicks the URL,

he is guided to the fake website (see Fig.1.2). He sadly enters his password into the login window and the adversary can easily steal the password.

Note that in this case, even if passwords are sent under encryption, the adversary can get the passwords because the encryption is under the control of the adversary.

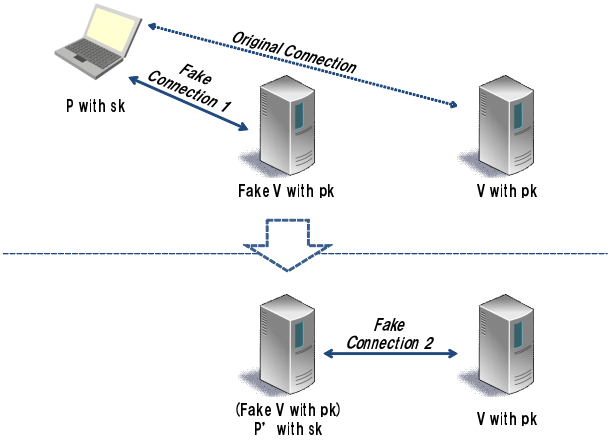

Figure 1.2: Phishing.

In each case above, once the adversary gets a password, he can act as if he is a legitimate prover, so he can easily impersonate the legitimate user who originally owns the password.

Fortunately, the two cases above can be avoided by public key-based identification because we do not have to send secret keys directly. But there are still some cases which cause considerable threats even for public key-based identification.

Man-in-the-Middle Connections and Impersonation.

A *man-in-the-middle attack* is well recognized; a typical instance of a man-in-the-middle attack in our real world is seen as follows. Suppose one uses a wireless connection service at an internet cafe. In a malicious secret way, an adversary's device is set so as to catch signals between a Note-PC and a legitimate access point (see Fig.1.3). The device tries to pretend both sides. That is, on the one hand, the device acts like the Note-PC for the legitimate access point (Fake Connection 0 in Fig.1.3). On the other hand, the device acts like the access point for the Note-PC

(Fake Connection 1 in Fig.1.3). A fake verifer (Fake V) and a fake prover (Fake P) on the device controls interactions with a user application on the Note-PC (a prover P) and a verifier on the access point (a verifier V), respectively. Fake V collects information of P's secret key, and once Fake V gets a certain kind of information, Fake P can impersonate P.

Note here that we do not consider *relay* of messages as a successful man-in-the-middle attack. Here relay means receiving and sending messages without any change. This constraint does not fit into some real cases, but even if we set the constraint, man-in-the-middle attacks remains our prime concern [25].

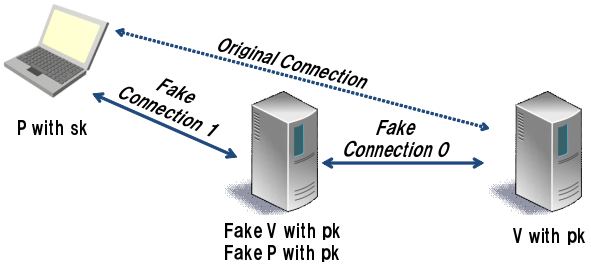

Figure 1.3: Man-in-the-Middle Connections and Impersonation.

Concurrent Connections and Impersonation.

Another case occurs especially in the Internet in the following way. Suppose one uses a synchronous filing service on a file server. It enables the one to maintain his files up-to-date even if he accesses them from different points simultaneously, such as from a Note-PC and from a smartphone (see Fig.1.4). A feature in this *concurrent connections* is that provers on the Note-PC and the smartphone ($\mathtt{P}_1$ and $\mathtt{P}_2$) have independent inner states and randomnesses, but the same secret key. Then, an adversary's fake verifier (Fake V), guiding and interacting with the provers $\mathtt{P}_1$ and $\mathtt{P}_2$ via fake connections, embeds some cheating trick in messages and collects information of the secret key from the responses of $\mathtt{P}_1$ and $\mathtt{P}_2$. Afterwards, the adversary's fake prover (Fake P) tries to impersonate the prover against a victim verifier on the file server (V) by using the collected information.

In the environment of the Cloud Computing, which realizes to access a system or a service from remote computers of small resource, the situation above really occurs and hence the threat by concurrent connections must be considered.

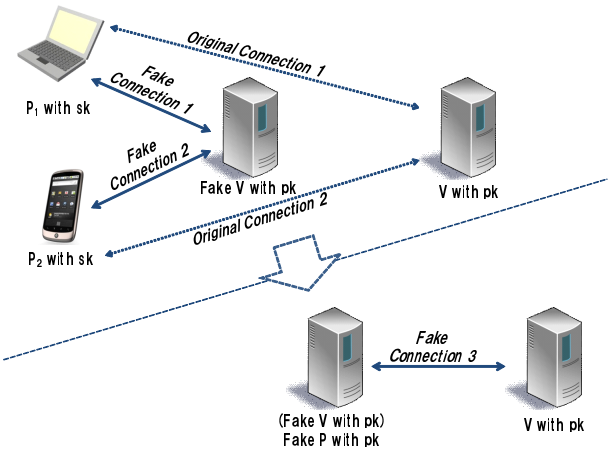

Figure 1.4: Concurrent Connections and Impersonation.

Concurrent Man-in-the-Middle Connections and Impersonation.

In the environment of the Internet where everyone is involved, the two cases above can be composed to lead a *concurrent man-in-the-middle connections* (cMiM connections, for short) [20]. In a cMiM connections (see Fig.1.5), an adversary's fake prover (Fake $\mathtt{P}$) and a fake verifier (Fake $\mathtt{V}$) stands between provers ($\mathtt{P}_1$ and $\mathtt{P}_2$) and a verifier ($\mathtt{V}$). Interacting in some cheating way, the adversary collects information of the secret key from $\mathtt{P}_1$ and $\mathtt{P}_2$, while the adversary, interacting with $\mathtt{V}$ simultaneously, tries to impersonate the prover,

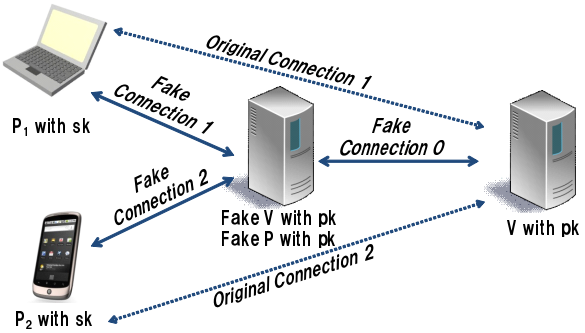

Figure 1.5: Concurrent Man-in-the-Middle Connections and Impersonation.

1.2 The Models of Attacks on ID Schemes

Real attacks on public key-based ID schemes which we have seen above are modeled as follows.

1.2.1 Passive Attack

In a *passive attack* [7], an adversary eavesdrops interactions between a prover and a verifier, and gets transcripts. Then, the adversary tries to impersonate the prover.

As we have seen in Section 1.1.2, password-based identification is basically vulnerable under passive attacks. In contrast, public key-based ID schemes can be secure against passive attacks. In other words, the security against passive attacks is a must requirement for public key-based ID schemes.

1.2.2 Active Attack

In an *active attack* [7], an adversary firstly acts as a verifier. It interacts with a prover on a user application and embeds some cheating trick in messages to collect information of the secret key from the responses of the prover. Secondly, after the completion of acting as a verifier, the adversary tries to impersonate the prover against a victim verifier on a server using that collected information.

1.2.3 Concurrent Attack

In a *concurrent attack* [7], an adversary firstly acts as a verifier. It interacts with provers on user applications which have independent inner states and random tapes, but the same secret key. It embeds some cheating trick in messages to collect information of the secret key from the responses from provers. Secondly, after the completion of acting as a verifier, the adversary tries to impersonate the prover against a victim verifier on a server using that collected information.

Active and concurrent attacks are real threats for public key-based ID schemes. A typical example is in an environment of the Cloud Computing, as we have seen in Section 1.1.2.

1.2.4 Man-in-the-Middle Attack

In a *man-in-the-middle attack* [25], an adversary stands between a prover and a verifier and interacts with both sides simultaneously. The advantageous point over

the active attack above is that the adversary can interact with the prover adaptively according to the received message from the verifier.

A man-in-the-middle attack actually happens in the Internet, as we have seen in Section 1.1.2.

1.2.5 Concurrent Man-in-the-Middle Attack

In a *concurrent man-in-the-middle attack* [20] (cMiM attack, for short), an adversary stands between provers and a verifier. Interacting in some cheating, concurrent way, the adversary collects information of the secret key from the provers. (Here "concurrent" means that the messages of those interactions between the adversary and the provers are allowed to be sent in arbitrary order.) At the same time, the adversary, trying to impersonate the prover, interacts with a victim verifier on a server simultaneously.

Among the attacks which we have seen above, the concurrent man-in-the-middle attack is strongest. In this dissertation, we will tackle the problem of preventing concurrent man-in-the-middle attacks on ID schemes.

1.3 Previous Works on Identification Schemes

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

As an example, the Schnorr ID scheme is described in Fig.1.6. As we can see, it needs 1 exponentiation in the prover computation and 2 exponentiations in the verifier computation.

Unfortunately, the Schnorr scheme and the GQ scheme are not secure under man-in-the-middle attacks[1] and hence there have been significant efforts to make the Σ-protocols secure against cMiM attacks. For example, Katz [25, 26] made an

[1] A simple man-in-the-middle attack (concerning to a related public key) is known for the Schnorr scheme and the GQ scheme [25].

Key Generation
– $\mathtt{K}$: 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 $(\mathtt{pk}, \mathtt{sk})$

Interaction
The 1st Round (Commitment)
– $\mathtt{P}$: given $\mathtt{sk}$ as an input;
- $a \leftarrow \mathbf{Z}_q, A := g^a$
- Send A to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{A} \mathtt{V}$$

The 2nd Round (Challenge)
– $\mathtt{V}$: given $\mathtt{pk}$ as an input and receiving A as an input message;
- $c \leftarrow \mathbf{Z}_q$
- Send c to $\mathtt{P}$

$$\mathtt{P} \xleftarrow{c} \mathtt{V}$$

The 3rd Round (Response)
– $\mathtt{P}$: receiving c as an input message;
- $r := a + xc$
- Send r to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{r} \mathtt{V}$$

The Final Decision (Verification)
– $\mathtt{V}$: receiving r as an input message;
- If $g^r = AX^c$ then return 1 else return 0

Figure 1.6: The Schnorr ID Scheme.

ID scheme of non-malleable proof of knowledge. But the security model is with timing constraint, not against full cMiM attacks. Gennaro [20] constructed 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).

As an example, the Gennaro Concurrently Non-malleable Proof of Knowledge [20] is described in Fig.1.7. As we will see in Chapter 7, it needs 5 exponentiations in the prover computation and 4.5 exponentiations in the verifier computation. In comparison with the Schnorr scheme, it needs 4 more exponentiations in the prover computation and 2.5 more exponentiations in the verifier computation.

That is, the Σ-protocols have become complicated so as to be secure against cMiM attacks.

One of the reasons why it is so difficult to construct an ID scheme secure against cMiM attacks based on Σ-protocols is as follows. A Σ-protocol is a 3-round proofs of knowledge which possess three properties: the completeness, the special soundness and the honest-verifier zero-knowledge. We have been thinking a Σ-protocol as a suitable base for an ID scheme because of its small amount of computational cost. But we note that we prove the soundness by constructing a *knowledge extractor*; when it comes to consider concurrent man-in-the-middle attacks, the knowledge extractor becomes too complex as it needs nested rewindings to extract a witness (that is, a secret key).

Let us remember that challenge-and-response ID schemes obtained from IND-CCA2 secure encryption schemes (see [14] for example) or EUF-CMA secure signature schemes (see [4] for example) have already been secure against concurrent man-in-the-middle attacks. But such challenge-and-response ID schemes have been considered to take more computational cost than Σ-protocols. Actually, to construct efficient IND-CCA2 secure encryption schemes and EUF-CMA secure signature schemes in the standard model, which are main concerns in the fields, are difficult tasks, and hence challenge-and-response ID schemes are not considered efficient.

Key Generation

– $\mathtt{K}$: given 1^k as an input;

- $(q, g) \leftarrow \mathtt{Grp}(1^k), x \leftarrow \mathbf{Z}_q, X := g^x$
- $P \leftarrow \mathtt{GenPrime}(1^k), N \leftarrow \mathtt{GenMod}(1^k), s \leftarrow \mathbf{Z}_N^*$
- Select $H, h : \{0,1\}^* \rightarrow \{0,1\}^k$
- $\mathtt{pk} := (q, g, X, P, N, s, H, h), \mathtt{sk} := (q, g, x, P, N, s, H, h)$, return $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Commitment)

– $\mathtt{P}$: given $\mathtt{sk}$ as an input;

- Do until e being a prime:
  $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$
  $e := 2 \cdot P \cdot H(\mathtt{vk}) + 1$
- $a \leftarrow \mathbf{Z}_q, A := g^a, t \leftarrow \mathbf{Z}_N^*$
- $C \leftarrow \mathtt{Com}_{N,s,e}(A, t) := s^{h(a)} t^e$ in $\mathbf{Z}_N^*$
- Send $(\mathtt{vk}, C)$ to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{\mathtt{vk}, C} \mathtt{V}$$

The 2nd Round (Challenge)

– $\mathtt{V}$: given $\mathtt{pk}$ as an input and receiving A as an input message;

- $c \leftarrow \mathbf{Z}_q$
- Send c to $\mathtt{P}$

$$\mathtt{P} \xleftarrow{c} \mathtt{V}$$

The 3rd Round (Response)

– $\mathtt{P}$: receiving c as an input message;

- $r := a + xc$ in $\mathbf{Z}_q$, $\sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}$(transcript)
- Send (A, t, r, σ) to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{A, t, r, \sigma} \mathtt{V}$$

The Final Decision (Verification)

– $\mathtt{V}$: receiving (A, t, r, σ) as an input message;

- If $C = \mathtt{Com}_{N,s,e}(A, t) = s^{h(a)} r^e$ and $g^r = AX^c$
  and $\mathtt{Vrfy}_{\mathtt{vk}}$(transcript) $= 1$ then return 1 else return 0

Figure 1.7: The Gennaro Concurrently Non-malleable Proof of Knowledge.

1.4 Our Contribution

As a first contribution in this dissertation, we propose a generic conversion to use a key encapsulation mechanism (KEM) as an ID scheme analogous to the usage of an encryption scheme. Among the notions of encryption schemes, a KEM is the foundational concept for a hybrid construction with a data encryption mechanism (DEM). 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 ciphertext as a challenge to the prover that has the matching secret key. The prover decapsulates the ciphertext and sends the result to the verifier as a response. The verifier checks whether or not the response is equal to the random string.

As a generic property, KEM-based ID schemes have two advantages over (usual) encryption-based ID schemes.

The first advantage is that 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 (usual) encryption-based ID schemes.

The second advantage is, as we will show in Section 3, that a KEM only need to be one-way-CCA2 secure for the derived 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 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 for ID schemes.

As a second contribution, we give concrete, discrete logarithm-based one-way-CCA2 secure KEMs for ID schemes. Starting with traditional El Gamal KEM, we develop a series of four one-way-CCA2 secure KEMs by 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] and the Twin Diffie-Hellman technique[2][13]. 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

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

our KEMs is that our KEMs need the smallest amount of computational cost and message length, while its security is based on the (Gap-) Computational Diffie-Hellman (CDH) assumption.

As for related works, Fujisaki [19] recently pointed out a fact similar to our generic conversion independently of us. We discuss the conversion more precisely than it. As for concrete constructions, the IND-CCA2 secure KEM of Cramer and Shoup [15] derives an efficient challenge-and-response ID scheme until now, though its security is based on the DDH assumption. Our `KEM1` and `KEM2` are secure based on the Gap-CDH assumption and they are faster than Cramer and Shoup KEM. Hanaoka-Kurosawa [24] construct a one-way-CCA2 secure KEM based on the CDH assumption along the way to develop an IND-CCA2 secure KEM. It is directly comparable with our `KEM3`, and our `KEM3` is faster than the KEM.

Finally, we point out another feature that the prover in our generic construction of an ID scheme is deterministic, and hence the derived ID scheme is prover-resettable [5]. Moreover, it is also verifier-resettable because they consist of 2-round interaction. This is a remarkable property because, as is discussed by and Bellare et al. [5] and Yilek [41], resettable security is crucially helpful for smartcards service and virtual machine service in the Cloud Computing.

1.5 Organization of the Dissertation

In Chapter 2, we fix some notations and briefly review the notion of computational hardness assumptions. In Chapter 3, we state models of ID scheme, attacks and security proofs in algorithmic and probabilistic terms. In Chapter 4, we survey previous works on ID schemes. In Chapter 5, after looking at known definitions of a KEM, attacks on a KEM and security proofs, we propose a generic conversion from a KEM to an ID scheme. Using the conversion, we obtain a design principle of an ID scheme with a desired security. In Chapter 6, according to the design principle, we develop a series of concrete ID schemes. In Section 6.1, we discuss El Gamal KEM as a starting point. In Section 6.2, we employ a tag technique with the algebraic trick [3, 27] to obtain a tag-based KEM `tKEM`, which is one-way-CCA2 secure in the selective tag model based on the Gap-CDH assumption. In Section 6.3, we apply the CHK transformation [12] to leave the selective tag model and obtain `KEM1`. In

Section 6.4, we further develop the series by using a target collision resistant hash function to obtain KEM2. In Section 6.5, we apply the Twin Diffie-Hellman technique [13] to KEM2 and obtain KEM3, which is secure based on the CDH assumption. In Chapter 7, we compare the ID schemes from our KEMs with previously known ID schemes and KEMs. In Chapter 8, we conclude our work.

Chapter 2

Preliminaries

In this chapter, we prepare for notations and notions needed for what follows in this dissertation. First, we fix some algorithmic and probabilistic notations. Second, we introduce notions of computationally hard problems and assumptions, upon which the security of our ID schemes will rely on.

2.1 Algorithmic Notations

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 $\mathcal{O}$, we denote it as $A^{\mathcal{O}}$. When A accesses n oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$ concurrently, we denote it as $A^{\mathcal{O}_1|\cdots|\mathcal{O}_n}$. Here "concurrent" means that A accesses oracles in an arbitrarily interleaved order of messages.

2.2 Probabilistic Notations

A probability of an event X is denoted Pr[X]. A probability of an event X on conditions $\mathrm{Y}_1, \ldots, \mathrm{Y}_m$ is denoted $\Pr[\mathrm{Y}_1; \cdots ; \mathrm{Y}_m : \mathrm{X}]$.

2.3 Computationally Hard Problems and 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 Problems and 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:

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

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 $(g, g^{x_1}, g^{x_2}, g^y, g^{x_1 y}, g^{x_2 y})$ for some elements x_1, x_2, y in $\mathbf{Z}_q$.

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

Lemma 2.1 (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 Problems and 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 $\textsc{Win}$ else return $\textsc{Lose}$.

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

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

We say that the Gap-DL Assumption holds for `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) [17, 8] 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.

$\mathbf{Exprmt}^{\text{kea}}_{\mathcal{H},\mathcal{H}',\mathtt{Grp}}(1^k)$

$(q, g) \leftarrow \mathtt{Grp}(1^k), \lambda \leftarrow \Lambda(1^k), a \leftarrow \mathbf{Z}_q, h := g^a$

$(g, h, X, Z) \leftarrow \mathcal{H}(g, h, \lambda), x' \leftarrow \mathcal{H}'(g, h, \lambda)$

If $X^a = Z \wedge g^{x'} \neq X$ then return Win

else return Lose.

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* `Grp` *and* $\mathcal{H}'$ as:

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

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

2.4 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 of a verification key and a matching signing key $(\mathtt{vk}, \mathtt{sgk})$ 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 is 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.

Then, we define the *advantage of $\mathcal{F}$ over $\mathtt{OTS}$ in the game of existential unforgery against chosen message attack 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, for example, based on the existence of a one-way function ([31]).

2.5 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 $\mathit{Hfam}(1^k) = \{H_\kappa\}_{\kappa \in \mathit{Hkey}(1^k)}$. Here $\mathit{Hkey}(1^k)$ is a hash key space, $\kappa \in \mathit{Hkey}(1^k)$ 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 experi-

ment.

$\mathbf{Exprmt}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(1^k)$

$m \leftarrow \mathcal{CF}(1^k), \kappa \leftarrow Hkey(1^k), m' \leftarrow \mathcal{CF}(\kappa)$

If $H_\kappa(m) = H_\kappa(m')$ and $m \neq m'$ then return WIN

else return LOSE.

We define the *advantage of $\mathcal{CF}$ over Hfam in the game of target collision resistance* as follows.

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

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

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

Chapter 3

The Model of ID Schemes, Attacks and Security Proofs

In this chapter, we fix the models of ID schemes, attacks and security proofs for ID schemes against attacks. First, we introduce the model of ID scheme in algorithmic terms. Second, we set the models of attacks on an ID scheme and define what it is for an ID scheme to be secure against such attacks.

3.1 Identification Scheme

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

3.2 Attacks on ID Scheme

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

3.2.1 Passive Attack

One of the weakest and the most basic attacks is a *passive attack* described in the following experiment, which is the formalization of the one described in Section

1.2.1.

$\mathbf{Exprmt}^{\text{imp-pa}}_{\mathcal{A},\mathtt{ID}}(1^k)$

$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k)$

If $\mathcal{A}_1(\mathtt{pk})$ makes a query, reply $\pi_i \leftarrow |\langle \mathtt{P}(\mathtt{sk}), \mathtt{V}(\mathtt{pk})\rangle|$

$st \leftarrow \mathcal{A}_1(\{\pi_i\}_{i=1}^{q_{pa}})$

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 $\mathtt{P}(\mathtt{sk})$ and $\mathtt{V}(\mathtt{pk})$ as $\pi = |\langle \mathtt{P}(\mathtt{sk}), \mathtt{V}(\mathtt{pk})\rangle|$. The number of queries q_{pa} is polynomial in k.

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

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

We say that an ID is secure against passive attacks if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{imp-pa}}_{\mathcal{A},\mathtt{ID}}(k)$ is negligible in k.

3.2.2 Active Attack

Suppose that an adversary $\mathcal{A}$ consists of two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. The following experiment is called an *active (two-phase) attack*, which is the formalization of the one described in Section 1.2.2.

$\mathbf{Exprmt}^{\text{imp-aa}}_{\mathcal{A},\mathtt{ID}}(1^k)$

$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k), st \leftarrow \mathcal{A}_1^{\mathtt{P}(\mathtt{sk})}(\mathtt{pk})$

decision $\leftarrow \langle \mathcal{A}_2(st), \mathtt{V}(\mathtt{pk})\rangle$

If decision $= 1$ then return WIN else return LOSE.

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

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

We say that an ID is secure against active attacks if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{imp-aa}}_{\mathcal{A},\mathtt{ID}}(k)$ is negligible in k.

The active attack is a stronger model than the passive attack because $\mathcal{A}_1$ can choose messages of his choice in an interaction with $\mathtt{P}(\mathtt{sk})$.

3.2.3 Concurrent Attack

A stronger attack than the active attack is a *concurrent (two-phase) attack* described in the following experiment, which is the formalization of the one described in Section 1.2.3.

$\mathbf{Exprmt}^{\text{imp-ca}}_{\mathcal{A},\mathtt{ID}}(1^k)$

$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k)$, $st \leftarrow \mathcal{A}_1^{\mathtt{P}_1(\mathtt{sk})|\cdots|\mathtt{P}_n(\mathtt{sk})}(\mathtt{pk})$

decision $\leftarrow \langle \mathcal{A}_2(st), \mathtt{V}(\mathtt{pk}) \rangle$

If decision $= 1$ then return WIN else return LOSE.

In the above experiment, the number of prover clones n is polynomial in k.

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

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

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

The concurrent attack is a stronger model than the active attack because $\mathcal{A}_1$ can interact with (polynomially) many prover clones $(\{\mathtt{P}_i(\mathtt{sk})\}_{i=1}^n)$ concurrently.

3.2.4 Man-in-the-Middle Attack

Another stronger attack than the active attack is a *man-in-the-middle attack* described in the following experiment, which is the formalization of the one described in Section 1.2.4.

$\mathbf{Exprmt}^{\text{imp-mim}}_{\mathcal{A},\mathtt{ID}}(1^k)$

$(\mathtt{pk},\mathtt{sk}) \leftarrow \mathtt{K}(1^k)$

decision $\leftarrow \langle \mathcal{A}^{\mathtt{P}(\mathtt{sk})}(\mathtt{pk}), \mathtt{V}(\mathtt{pk}) \rangle$

If decision $= 1 \wedge \pi^* \neq \pi$ then return WIN

else return LOSE.

In the above experiment, we denoted a transcript of interaction between $\mathtt{P}(\mathtt{sk})$ and $\mathcal{A}(\mathtt{pk})$ as π and a transcript between $\mathcal{A}(\mathtt{pk})$ and $\mathtt{V}(\mathtt{pk})$ as π^*.

As a rule, a man-in-the-middle adversary $\mathcal{A}$ is prohibited from relaying a transcript of a whole interaction with the prover $\mathtt{P}(\mathtt{sk})$ to the verifier $\mathtt{V}(\mathtt{pk})$, as is described

$\pi^* \neq \pi$ in the above experiment. This is a natural and standard constraint when we discuss man-in-the-middle attacks [25].

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

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

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

3.2.5 Concurrent Man-in-the-Middle Attack

An adversary $\mathcal{A}$ performs a *concurrent man-in-the-middle attack* as in the following experiment [5, 7], which is the formalization of the one described in Section 1.2.5.

$\mathbf{Exprmt}^{\text{imp-cmim}}_{\mathcal{A},\mathtt{ID}}(1^k)$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k)$

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^*.

As a rule, a concurrent man-in-the-middle adversary $\mathcal{A}$ is prohibited from relaying a transcript of a whole interaction with some prover clone $\mathtt{P}_i(\mathtt{sk})$ to the verifier $\mathtt{V}(\mathtt{pk})$, as is described $\pi^* \notin \{\pi_i\}_{i=1}^n$ in the experiment. This is a natural constraint when we discuss man-in-the-middle attacks [25].

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

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

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.

3.3 Tag-Based Identification Scheme

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

$\mathtt{t}$, called a *tag*, is a priori given to $\mathtt{P}$ and $\mathtt{V}$ by the first round. An interaction between $\mathtt{P}$ and $\mathtt{V}$ depends on a given tag $\mathtt{t}$.

As for attacks on $\mathtt{tagID}$, 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 $\mathtt{pk}$. In addition, before starting each interaction as a verifier, $\mathcal{A}$ provides a tag $\mathtt{t}_i(\neq \mathtt{t}^*)$ to each prover clone $\mathtt{P}_i(\mathtt{sk})$.

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

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

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

Chapter 4

A Survey of Previous Works on ID Schemes

In this chapter, we survey previous works on ID schemes from the view point of efficiency so as to elucidate merits and demerits of techniques employed in those works.

As we mentioned in Introduction, there have been two types of ID schemes. One is a kind of proofs of knowledge [21, 6] called the Σ-protocols [11], and the other is challenge-and-response type obtained from encryption schemes or signature schemes in a natural way. First, we survey five Σ-protocols. Second, we see a challenge-and-response ID scheme from the Cramer-Shoup encryption scheme. Then, we discuss merits and demerits of techniques employed in them.

4.1 Σ-protocol ID Schemes

In this section, we survey five Σ-protocol ID schemes: the Schnorr, the Guillou-Quisquater, the Katz (in the discrete logarithm setting and the RSA setting) and the Gennaro ID schemes.

4.1.1 The Schnorr ID Scheme

The Schnorr ID scheme [37] is described in Fig.1.6 in Introduction.

Let G_q denote a cyclic group of a prime order q and g denote a generator.

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

The interaction proceeds as follows.

The first round (Commitment): given `sk` as an input, the prover `P` chooses a from $\mathbf{Z}_q$ at random and computes $A = g^a$. `P` sends A to the verifier `V`.

The second round (Challenge): given `pk` as an input and receiving A as an input message, `V` chooses c from $\mathbf{Z}_q$.

The third round (Response): receiving c as an input message, `P` computes $r := a + xc$ in $\mathbf{Z}_q$. `P` sends r to `V`.

The final decision (Verification): receiving r as an input message, `V` computes and verifies whether $g^r = AX^c$ holds in G_q or not. `V` returns 1 or 0.

The security of the Schnorr ID scheme is stated as follows.

Theorem 4.1 (Bellare and Palacio [7]) *The Schnorr ID scheme is secure against concurrent two-phase attacks based on the One More Discrete Logarithm assumption for* `Grp`.

For the One More Discrete Logarithm (omdl, for short) assumption, see [7]. We only denote the idea in [7] for the proof of Theorem 4.1. Given any PPT adversary $\mathcal{A}$, we construct a omdl problem solver $\mathcal{S}$ using $\mathcal{A}$ as subroutine. $\mathcal{S}$ is able to make response messages in answer to $\mathcal{A}$'s challenge messages (that is, $\mathcal{S}$ can simulate prover clones) with the aid of *the Discrete Logarithm oracle*. After completing the simulation, $\mathcal{S}$ extracts the answer of omdl problem instances by the technique of *rewinding*.

4.1.2 The Guillou-Quisquater ID Scheme

The Guillou-Quisquater (GQ) ID scheme [22] is described in Fig.4.1.

Let N denote an RSA modulus of length k, that is, a product of two primes p and q of length $k/2$. Let $e < \phi(N)$ denote an odd prime which satisfies $\gcd(e, \phi(N)) = 1$.

On an input 1^k, the key generator `K` runs as follows. On an input 1^k, The RSA parameter generator `GenRSA` returns (N, e). Then, `K` chooses x from $\mathbf{Z}_N$, computes $X = x^e$ and sets $\mathtt{pk} = (N, e, X)$ and $\mathtt{sk} = (N, e, x)$. Then, `K` returns (`pk`, `sk`).

The interaction proceeds as follows.

The first round (Commitment): given `sk` as an input, the prover `P` chooses a from $\mathbf{Z}_N$ at random and computes $A = a^e$. `P` sends A to the verifier `V`.

The second round (Challenge): given pk as an input and receiving A as an input message, V chooses c from $\mathbf{Z}_q$.

The third round (Response): receiving c as an input message, P computes $r := ax^c$ in $\mathbf{Z}_N^*$. P sends r to V.

The final decision (Verification): receiving r as an input message, V computes and verifies whether $r^e = AX^c$ holds in $\mathbf{Z}_N^*$ or not. V returns 1 or 0.

The security of the GQ ID scheme is stated as follows.

Theorem 4.2 (Bellare and Palacio [7]) *The Guillou-Quisquater ID scheme is secure against concurrent two-phase attacks based on the RSA One More Inversion assumption for GenRSA.*

For the RSA One More Inversion (rsa-omi, for short) assumption, see [7]. We only denote the idea in [7] for the proof of Theorem 4.2. Given any PPT adversary $\mathcal{A}$, we construct an rsa-omi problem solver $\mathcal{S}$ using $\mathcal{A}$ as subroutine. $\mathcal{S}$ is able to make response messages in answer to $\mathcal{A}$'s challenge messages (that is, $\mathcal{S}$ can simulate prover clones) with the aid of *the RSA-Inversion oracle*. After completing the simulation, $\mathcal{S}$ extracts the answer of rsa-omi problem instances by the technique of *rewinding*.

4.1.3 The Katz Non-malleable Proof of Knowledge ID Scheme

The Katz Non-malleable Proof of Knowledge in the Discrete Logarithm Setting.

The Katz Non-malleable Proofs of Knowledge [25, 26] in the Discrete Logarithm setting is described in Fig.4.2.

Let G_q denote a cyclic group of a prime order q and g denote a generator.

On an input 1^k, the key generator K runs as follows. On an input 1^k, The group generator Grp returns (q, g_0) on an input 1^k. Then, K chooses x from $\mathbf{Z}_q$ and computes $X = g_0^x$. Next chooses g_1, h from $\mathbf{Z}_q$ at random and selects a hash function $H : \{0,1\}^* \to \mathbf{Z}_q$. K sets pk $= (q, g_0, X, g_1, h, H)$ and sk $= (q, g_0, x, g_1, h, H)$, and returns (pk, sk).

The interaction proceeds as follows.

The first round (Commitment): given sk as an input, the prover P runs a signing key generator SGK(1^k) and gets a pair of a verification key and a signing key (vk, sgk).

Key Generation

– K: given 1^k as an input;

- $(N, e) \leftarrow \mathtt{GenRSA}(1^k), x \leftarrow \mathbf{Z}_N^*, X := x^e$ in $\mathbf{Z}_N^*$
- $\mathtt{pk} := (N, e, X), \mathtt{sk} := (N, e, x)$, return $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Commitment)

– P: given sk as an input;

- $a \leftarrow \mathbf{Z}_N^*, A := a^e$
- Send A to V

$$\mathtt{P} \xrightarrow{A} \mathtt{V}$$

The 2nd Round (Challenge)

– V: given pk as an input and receiving A as an input message;

- $c \leftarrow \mathbf{Z}_q$
- Send c to P

$$\mathtt{P} \xleftarrow{c} \mathtt{V}$$

The 3rd Round (Response)

– P: receiving c as an input message;

- $r := ax^c$ in $\mathbf{Z}_N^*$
- Send r to V

$$\mathtt{P} \xrightarrow{r} \mathtt{V}$$

The Final Decision (Verification)

– V: receiving r as an input message;

- If $r^e = AX^c$ in $\mathbf{Z}_N^*$ then return 1 else return 0

Figure 4.1: The Guillou-Quisquater ID Scheme.

Then, P chooses a_1, r_2, c_2 from $\mathbf{Z}_q$ at random. P computes $H(\mathtt{vk})$ to get α. P computes $A_1 = g_0^{a_1}$ and $A_2 = g_0^{r_2}/(g_1^{\alpha}h)^{c_2}$. P sends $(\mathtt{vk}, A_1, A_2)$ to the verifier V.

The second round (Challenge): given pk as an input and receiving $(\mathtt{vk}, A_1, A_2)$ as an input message, V chooses c from $\mathbf{Z}_q$. V sends c to P.

The third round (Response): receiving c as an input message, P computes $c_1 = c - c_2$ in $\mathbf{Z}_q$ and $r_1 = a_1 + xc_1$ in $\mathbf{Z}_q$. Then, P runs $\mathtt{Sign}_{\mathtt{sgk}}$ on input the whole transcript $(= (A_1, A_2, c, c_1, r_1, r_2))$ and gets a signature σ. P and sends (c_1, r_1, r_2, σ) to V.

The final decision (Verification): receiving (c_1, r_1, r_2, σ) as an input message, V computes $H(\mathtt{vk})$ to get α and computes $c_2 = c - c_1$ in $\mathbf{Z}_e$. V computes and verifies whether $g_0^{r_1} = A_1 X^{c_1}$ and $g_0^{r_2} = A_2(g_1^{\alpha}h)^{c_2}$ holds in G_q and $\mathtt{Vrfy}_{\mathtt{vk}}(\text{transcript}, \sigma) = 1$. V returns 1 or 0.

The security of the Katz Non-malleable Proof of Knowledge in the Discrete Logarithm setting is stated as follows.

Theorem 4.3 (Katz [26]) *The Katz Non-malleable Proof of Knowledge in the Discrete Logarithm setting is secure against man-in-the-middle attacks based on the assumption that the Discrete Logarithm problem is hard for* ***Grp*** *against expected polynomial time algorithms, and one-time security of an employed one-time signature.*

We only denote the idea in [26] for the proof of Theorem 4.3. Given any expected polynomial time adversary $\mathcal{A}$, we construct the Discrete Logarithm problem solver $\mathcal{S}$ using $\mathcal{A}$ as subroutine. $\mathcal{S}$ is able to make response messages in answer to $\mathcal{A}$'s challenge messages (that is, $\mathcal{S}$ can simulate prover clones) by the technique of *OR-Proof*. Moreover, $\mathcal{S}$ extracts the answer of a Discrete Logarithm problem instance by the technique of *rewinding*.

The Katz Non-malleable Proof of Knowledge in the RSA Setting.

The Katz Non-malleable Proofs of Knowledge [25, 26] in the RSA setting is described in Fig.4.3.

Let N denote an RSA modulus of length k, that is, a product of two primes p and q of length $k/2$. Let $e < \phi(N)$ denote an odd prime which satisfies $\gcd(e, \phi(N)) = 1$.

Key Generation

– $\mathtt{K}$: given 1^k as an input;

- $q \leftarrow \mathtt{Grp}(1^k), g_0, g_1, h \leftarrow G_q$
- $x \leftarrow \mathbf{Z}_q, X := g_0^x$, select $H : \{0,1\}^* \rightarrow \mathbf{Z}_q$
- $\mathtt{pk} := (q, g, X, g_0, g_1, h, H), \mathtt{sk} := (q, g, x, g_0, g_1, h, H)$, return $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Commitment)

– $\mathtt{P}$: given $\mathtt{sk}$ as an input;

- $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$
- $a_1, r_2, c_2 \leftarrow \mathbf{Z}_q$
- $\alpha \leftarrow H(\mathtt{vk})$
- $A_1 := g_0^{a_1}, A_2 := g_0^{r_2}/(g_1^{\alpha}h)^{c_2}$
- Send $(\mathtt{vk}, A_1, A_2)$ to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{\mathtt{vk}, A_1, A_2} \mathtt{V}$$

The 2nd Round (Challenge)

– $\mathtt{V}$: given $\mathtt{pk}$ as an input and receiving $(\mathtt{vk}, A_1, A_2)$ as an input message;

- $c \leftarrow \{0,1\}^l$
- Send c to $\mathtt{P}$

$$\mathtt{P} \xleftarrow{c} \mathtt{V}$$

The 3rd Round (Response)

– $\mathtt{P}$: receiving c as an input message;

- $c_1 := c - c_2$ in $\mathbf{Z}_q$
- $r_1 := a_1 + xc_1$
- $\sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}(\text{transcript})$
- Send (c_1, r_1, r_2, σ) to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{c_1, r_1, r_2, \sigma} \mathtt{V}$$

The Final Decision (Verification)

– $\mathtt{V}$: receiving (c_1, r_1, r_2, σ) as an input message;

- $\alpha \leftarrow H(\mathtt{vk}), c_2 := c - c_1$ in $\mathbf{Z}_q$
- If $g_0^{r_1} = A_1 X^{c_1}$ and $g_0^{r_2} = A_2(g_1^{\alpha}h)^{c_2}$ and $\mathtt{Vrfy}_{\mathtt{vk}}(\text{transcript}, \sigma) = 1$ then return 1 else return 0

Figure 4.2: The Katz Non-malleable Proof of Knowledge in the Discrete Logarithm Setting.

On an input 1^k, the key generator K runs as follows. On an input 1^k, The RSA parameter generator GenRSA returns (N, e). Then, K chooses x from $\mathbf{Z}_N^*$ and computes $X = x^e$. Next K chooses g, h from $\mathbf{Z}_N^*$ at random and selects a hash function $H : \{0,1\}^* \to \mathbf{Z}_e$. K sets $\mathtt{pk} = (N, e, X, g, h, H)$ and $\mathtt{sk} = (N, e, x, g, h, H)$, and returns (pk, sk).

The interaction proceeds as follows.

The first round (Commitment): given sk as an input, the prover P runs a signing key generator $\mathtt{SGK}(1^k)$ and gets a pair of a verification key and a signing key (vk, sgk). Then, P chooses a_1, r_2 from $\mathbf{Z}_N^*$ and c_2 from $\mathbf{Z}_N$ at random. P computes $H(\mathtt{vk})$ to get α. P computes $A_1 = a_1^e$ and $A_2 = r_2^e/(g^\alpha h)^{c_2}$. P sends $(\mathtt{vk}, A_1, A_2)$ to the verifier V.

The second round (Challenge): given pk as an input and receiving $(\mathtt{vk}, A_1, A_2)$ as an input message, V chooses c from $\mathbf{Z}_e$. V sends c to P.

The third round (Response): given c as an input message, P computes $c_1 = c - c_2$ in $\mathbf{Z}_e$ and $r_1 = a_1 x^{c_1}$ in $\mathbf{Z}_e$. Then, P runs $\mathtt{Sign}_{\mathtt{sgk}}$ on input the whole transcript $(= (A_1, A_2, c, c_1, r_1, r_2))$ and gets a signature σ. P and sends (c_1, r_1, r_2, σ) to V.

The final decision (Verification): given (c_1, r_1, r_2, σ) as an input message, V computes $H(\mathtt{vk})$ to get α and computes $c_2 = c - c_1$ in $\mathbf{Z}_e$. V computes and verifies whether $r_1^e = A_1 X^{c_1}$ and $r_2^e = A_2(g^\alpha h)^{c_2}$ holds in $\mathbf{Z}_N^*$ and $\mathtt{Vrfy}_{\mathtt{vk}}(\text{transcript}, \sigma) = 1$. V returns 1 or 0.

The security of the Katz Non-malleable Proof of Knowledge in the RSA setting is stated as follows.

Theorem 4.4 (Katz [26]) *The Katz Non-malleable Proof of Knowledge in the RSA setting is secure against man-in-the-middle attacks based on the assumption that the RSA problem is hard for* ***Grp*** *against expected polynomial time algorithms, and one-time security of an employed one-time signature.*

We only denote the idea in [26] for the proof of Theorem 4.4. Given any expected polynomial time adversary $\mathcal{A}$, we construct the RSA problem solver $\mathcal{S}$ using $\mathcal{A}$ as subroutine. $\mathcal{S}$ is able to make response messages in answer to $\mathcal{A}$'s challenge messages (that is, $\mathcal{S}$ can simulate prover clones) by the technique of *OR-Proof*. Moreover, $\mathcal{S}$ extracts the answer of a RSA problem instance by the technique of *rewinding*.

Key Generation

– $\mathtt{K}$: given 1^k as an input;

- $(N, e) \leftarrow \mathtt{GenRSA}(1^k), x \leftarrow \mathbf{Z}_N^*, X := x^e$ in $\mathbf{Z}_N^*$
- $g, h \leftarrow \mathbf{Z}_N^*$, select $H : \{0,1\}^* \rightarrow \mathbf{Z}_e$
- $\mathtt{pk} := (N, e, X, g, h, H), \mathtt{sk} := (N, e, x, g, h, H)$, return $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Commitment)

– $\mathtt{P}$: given $\mathtt{sk}$ as an input;

- $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$
- $r_1, R_2 \leftarrow \mathbf{Z}_N^*, c_2 \leftarrow \mathbf{Z}_e$
- $\alpha \leftarrow H(\mathtt{vk})$
- $A_1 := r_1^e, A_2 := R_2^e/(g^\alpha h)^{c_2}$
- Send $(\mathtt{vk}, A_1, A_2)$ to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{\mathtt{vk}, A_1, A_2} \mathtt{V}$$

The 2nd Round (Challenge)

– $\mathtt{V}$: given $\mathtt{pk}$ as an input and receiving $(\mathtt{vk}, A_1, A_2)$ as an input message;

- $c \leftarrow \mathbf{Z}_e$
- Send c to $\mathtt{P}$

$$\mathtt{P} \xleftarrow{c} \mathtt{V}$$

The 3rd Round (Response)

– $\mathtt{P}$: receiving c as an input message;

- $c_1 := c - c_2$ in $\mathbf{Z}_e$
- $R_1 := x^{c_1} r_1$
- $\sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}(\text{transcript})$
- Send (c_1, R_1, R_2, σ) to $\mathtt{V}$

$$\mathtt{P} \xrightarrow{c_1, R_1, R_2, \sigma} \mathtt{V}$$

The Final Decision (Verification)

– $\mathtt{V}$: receiving (c_1, R_1, R_2, σ) as an input message;

- $\alpha \leftarrow H(\mathtt{vk}), c_2 := c - c_1$ in $\mathbf{Z}_N^*$
- If $R_1^e = A_1 X^{c_1}$ and $R_2^e = A_2(g^\alpha h)^{c_2}$ and $\mathtt{Vrfy}_{\mathtt{vk}}(\text{transcript}, \sigma) = 1$ then return 1 else return 0

Figure 4.3: The Katz Non-malleable Proof of Knowledge in the RSA Setting.

4.1.4 The Gennaro Concurrently Non-malleable Proof of Knowledge ID Scheme

The Gennaro Concurrently Non-malleable Proof of Knowledge [20] is described in Fig.1.7 in Introduction.

Let N denote an RSA modulus of length $l(k)$, that is, a product of two primes p and q of length $l(k)/2$. Here $l(k)$ is a polynomial in k. Let $e < \phi(N)$ denote an odd prime which satisfies $\gcd(e, \phi(N)) = 1$.

On an input 1^k, the key generator $\mathtt{K}$ runs as follows. On an input 1^k, The group generator $\mathtt{Grp}$ returns (q, g). q is of length k. $\mathtt{K}$ chooses x from $\mathbf{Z}_q$ and computes $X = g^x$. Next $\mathtt{K}$ runs a prime generator $\mathtt{GenPrime}$ and gets P, and runs the RSA modulus generator $\mathtt{GenMod}$ and gets N which is of length $l(k)$. $\mathtt{K}$ chooses s from $\mathbf{Z}_N^*$. $\mathtt{K}$ selects a hash function $H : \{0,1\}^* \to \mathbf{Z}_e$. $\mathtt{K}$ sets $\mathtt{pk} = (q, g, X, N, s, H)$ and $\mathtt{sk} = (q, g, x, N, s, H)$, and returns $(\mathtt{pk}, \mathtt{sk})$.

The interaction proceeds as follows.

The first round (Commitment): given $\mathtt{sk}$ as an input, the prover $\mathtt{P}$ runs a signing key generator $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$ until $e = 2 \cdot PH(\mathtt{vk}) + 1$ being a prime. Then, $\mathtt{P}$ chooses a from $\mathbf{Z}_q$ and computes $A = g^a$ and $\mathtt{P}$ chooses t from $\mathbf{Z}_N^*$. $\mathtt{P}$ computes a commitment $C \leftarrow \mathtt{Com}_{N,s,e}(A, t) := s^{h(a)} t^e$ in $\mathbf{Z}_N^*$. $\mathtt{P}$ sends $(\mathtt{vk}, C)$ to the verifier $\mathtt{V}$.

The second round (Challenge): given $\mathtt{pk}$ as an input and receiving $(\mathtt{vk}, C)$ as an input message, $\mathtt{V}$ chooses c from $\mathbf{Z}_q$. $\mathtt{V}$ sends c to $\mathtt{P}$.

The third round (Response): receiving c as an input message, $\mathtt{P}$ computes $r = a + xc$ in $\mathbf{Z}_q$. Then, $\mathtt{P}$ runs $\mathtt{Sign}_{\mathtt{sgk}}$ on input the whole transcript $(= (X, C, c, A, t, r))$ and gets a signature σ. $\mathtt{P}$ and sends (A, t, r, σ) to $\mathtt{V}$.

The final decision (Verification): receiving (A, t, r, σ) as an input message, $\mathtt{V}$ checks if $C = \mathtt{Com}_{N,s,e}(A, t) = s^{h(a)} r^e$ and $g^r = AX^c$ and $\mathtt{Vrfy}_{\mathtt{vk}}(\text{transcript}) = 1$. If it is true, then return 1, and otherwise, return 0

The security of the Gennaro Concurrently Non-malleable Proof of Knowledge is stated as follows.

Theorem 4.5 (Gennaro [20]) *The Gennaro Concurrently Non-malleable Proof of Knowledge is secure against concurrent man-in-the-middle attacks based on the Discrete Logarithm assumption, the Strong RSA assumption and one-time security of an employed one-time signature.*

We only denote the idea in [20] for the proof of Theorem 4.5. Given any PPT opener $\mathcal{A}$ on the multi-trapdoor commitment, we construct the Strong RSA problem solver $\mathcal{S}$ using $\mathcal{A}$ as subroutine.

4.2 Challenge-and-Response ID Schemes

In this section, we see a challenge-and-response ID scheme from an encryption scheme: the Cramer-Shoup encryption scheme.

4.2.1 An ID Scheme from the Cramer-Shoup Encryption

An ID scheme from the Cramer-Shoup encryption scheme [14] is described in Fig.4.4.

On an input 1^k, the key generator K runs as follows. On an input 1^k, The group generator Grp returns q. Then, K chooses g_1, g_2 from G_q. K chooses x_1, x_2, y_1, y_2, z from $\mathbf{Z}_q$ and computes $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$. Next K selects a hash function $H : \{0,1\}^* \to \{0,1\}^k$. K sets $\mathtt{pk} = (q, g_1, g_2, c, d, h, H)$ and $\mathtt{sk} = (q, g_1, g_2, x_1, x_2, y_1, y_2, z, H)$, and returns (pk, sk).

The interaction proceeds as follows.

The first round (Challenge): given pk as an input, the verifier V chooses $m \leftarrow G_q$ and $r \leftarrow \mathbf{Z}_q$ at random. Then, V computes $u_1 := g_1^r, u_2 := g_2^r, e := h^r m, \alpha \leftarrow H(u_1, u_2, e)$ and $v := c^r d^{r\alpha}$. V sends $\psi := (u_1, u_2, e, v)$ to P.

The second round (Response): given sk as an input and receiving $\psi := (u_1, u_2, e, v)$ as an input message, the prover P runs $\alpha \leftarrow H(u_1, u_2, e)$. P checks whether $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$ or not. If it is true, then P computes $\widehat{m} := e/u_1^z$, and otherwise, puts $m :=\perp$. P sends $\widehat{m}$ to V.

The final decision (Verification): receiving $\widehat{m}$ as an input message, V checks whether $\widehat{m} = m$ or not. V returns 1 or 0.

The security of the ID scheme from the Cramer-Shoup encryption scheme is stated as follows.

Theorem 4.6 (Corollary of the Theorem in Cramer and Shoup [15])
The ID scheme from the Cramer-Shoup encryption scheme is secure against concurrent man-in-the-middle attacks based on the Decisional Diffie-Hellman assumption for Grp and target collision resistance of a hash function family $\{H\}$ employed.

We only denote the idea in for the proof of Theorem 4.6. IND-CCA2 security for the Cramer and Shoup encryption scheme is done based on the Decisional Diffie-Hellman assumption and target collision resistance of an employed hash function [15]. One-way CCA2 security is proven by contradiction to the the IND-CCA2 security.

Key Generation

– K: given 1^k as an input;

- $q \leftarrow \mathtt{Grp}(1^k), g_1, g_2 \leftarrow G_q, x_1, x_2, y_1, y_2, z \leftarrow \mathbf{Z}_q$
- $c := g_1^{x_1} g_2^{x_2}, d := g_1^{y_1} g_2^{y_2}, h := g_1^z$
- Select $H : \{0,1\}^* \rightarrow \{0,1\}^k$
- $\mathtt{pk} := (q, g_1, g_2, c, d, h, H), \mathtt{sk} := (q, g_1, g_2, x_1, x_2, y_1, y_2, z, H)$
- Return $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Challenge)

– V: given pk as an input;

- $m \leftarrow G_q, r \leftarrow \mathbf{Z}_q$
- $u_1 := g_1^r, u_2 := g_2^r, e := h^r m, \alpha \leftarrow H(u_1, u_2, e), v := c^r d^{r\alpha}$
- Send $\psi := (u_1, u_2, e, v)$ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)

– P: given sk as an input and receiving $\psi := (u_1, u_2, e, v)$ as an input message;

- $\alpha \leftarrow H(u_1, u_2, e)$
- If $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v$
  then $\widehat{m} := e/u_1^z$ else $m :=\perp$
- Send $\widehat{m}$ to V

$$\mathtt{P} \xrightarrow{\widehat{m}} \mathtt{V}$$

The Final Decision (Verification)

– V: receiving $\widehat{m}$ as an input message;

- If $\widehat{m} = m$ then return 1 else return 0

Figure 4.4: An ID Scheme from the Cramer-Shoup Encryption Scheme.

4.3 Discussion

As we can see above, Σ-protocols are originally simple and fast for implementation [7]. But once we want to make them secure against concurrent man-in-the-middle attacks, the security proofs need strong assumptions, or, the protocols become complicated [26, 20]. This is because they basically need *nested rewindings* to extract witnesses in the proofs of the special soundness.

In contrast, concerning the concurrent man-in-the-middle security, the challenge-and-response ID schemes from the Cramer-Shoup [14] is relatively simple. It is even faster than the ID scheme of Gennaro [20].

So our strategy afterwards is the latter. That is to say, we are going to carry out a research on challenge-and-response ID schemes.

Chapter 5

A Generic Conversion from KEM to ID Scheme

In this chapter, 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 obtaining 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.

The achievements in this chapter are announced in AfricaCrypt 2011 [2].

5.1 Key Encapsulation Mechanism

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

5.2 Attacks on One-Wayness of KEM

5.2.1 Passive Attack

One of the weakest and the most basic attacks is a *passive attack on one-wayness* of a KEM (which we call one-way-PA, for short) described in the following experiment [1].

$\mathbf{Exprmt}^{\text{ow-pa}}_{\mathcal{A},\mathtt{KEM}}(1^k)$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k)$

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 of queries q_{pa} is polynomial in k.

We define *the ow-pa advantage of $\mathcal{A}$ over KEM* as:

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

We say that an KEM is secure against passive attacks if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{ow-pa}}_{\mathcal{A},\mathtt{KEM}}(k)$ is negligible in k.

5.2.2 Non-adaptive Chosen Ciphertext Attack

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.

We define *the one-way-CCA1 advantage of $\mathcal{A}$ over KEM* as:

$$\mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(1^k) \text{ returns WIN}].$$

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

We say that an KEM is secure against non-adaptive chosen ciphertext attacks (one-way-CCA1-secure, for short) if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{ow-cca1}}_{\mathcal{A},\mathtt{KEM}}(k)$ is negligible in k.

The non-adaptive chosen ciphertext attack is a stronger model than the passive attack because $\mathcal{A}_1$ can choose ciphertexts of his choice in its queries to $\mathcal{DEC}$.

5.2.3 Adaptive Chosen Ciphertext Attack

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

$\mathbf{Exprmt}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(1^k)$

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k), (K^*, \psi^*) \leftarrow \mathtt{Enc}(\mathtt{pk})$

$\widehat{K^*} \leftarrow \mathcal{A}^{\mathcal{DEC}(\mathtt{sk},\cdot)}(\mathtt{pk}, \psi^*)$

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 against 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.

5.3 Tag-Based KEM

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 $\mathtt{t}$, called a *tag*, is a priori given to $\mathtt{Enc}$ and $\mathtt{Dec}$ as an input. A random string K and its ciphertext ψ depend on a given tag $\mathtt{t}$.

As for attacks on $\mathtt{tagKEM}$, 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 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 $\mathbf{Adv}^{\text{ow-cca2}}_{\mathcal{A},\mathtt{KEM}}(k)$.

5.4 A Generic Conversion from KEM to ID Scheme

Let $\mathtt{KEM} = (\mathtt{K},\mathtt{Enc},\mathtt{Dec})$ be a KEM. Then, an ID scheme $\mathtt{ID}$ is derived in a natural way as shown in Fig.5.1. The key generation algorithm is the same as that of $\mathtt{KEM}$.

The interaction proceeds as follows.

The first round (Challenge): given a public key $\mathtt{pk}$ as an input, the verifier $\mathtt{V}$ invokes the encapsulation algorithm $\mathtt{Enc}$ on $\mathtt{pk}$ and gets a return (K,ψ). $\mathtt{V}$ sends ψ to $\mathtt{P}$.

The second round (Response): given a secret key $\mathtt{sk}$ as an input and receiving ψ as an input message, the prover $\mathtt{P}$ invokes the decapsulation algorithm $\mathtt{Dec}$ on $(\mathtt{sk},\psi)$ and gets a return $\widehat{K}$. $\mathtt{P}$ sends $\widehat{K}$ to $\mathtt{V}$.

The final decision (Verification): receiving $\widehat{K}$ as an input message, the verifier $\mathtt{V}$ verifies whether $\widehat{K}$ is equal to K or not. If so, then $\mathtt{V}$ 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 non-hybrid encryption-based ID schemes.

Key Generation
– K: the same as that of KEM

Interaction
The 1st Round (Challenge)
– V: given pk as an input;
- Invoke Enc on pk: $(K, \psi) \leftarrow \mathtt{Enc}(\mathtt{pk})$
- Send ψ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)
– 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

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)
– V: receiving $\widehat{K}$ as an input message;
- If $\widehat{K} = K$ then return 1 else return 0

Figure 5.1: An ID Scheme ID=(K,P,V) Derived from a KEM KEM=(K,Enc,Dec).

Theorem 5.1 *If a key encapsulation mechanism KEM is one-way-CCA2 secure, then the derived ID scheme ID is cMiM secure. More precisely, for any PPT adversary $\mathcal{A}$ that attacks ID in the cMiM manner, there exists an PPT adversary $\mathcal{B}$ that attacks KEM in the one-way-CCA2 manner satisfying the following inequality:*

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

5.5 Proof of Theorem 5.1

Let KEM be a one-way-CCA2 secure KEM and ID be the derived 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.5.2.

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 $\mathtt{V}(\mathtt{pk})$ 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 $\mathtt{P}(\mathtt{sk})$, $\mathcal{B}$ checks whether or not ψ is equal to ψ^*. If so, then $\mathcal{B}$ puts $K = \perp$. Otherwise, $\mathcal{B}$ queries its decapsulation oracle $\mathcal{DEC}(\mathtt{sk}, \cdot)$ 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^*.

Given $(\mathtt{pk}, \psi^*)$ as an input;

Initial Setting
– Initialize its inner state
– Invoke $\mathcal{A}$ on $\mathtt{pk}$

Answering $\mathcal{A}$'s Queries
– In the case that $\mathcal{A}$ queries $\mathtt{V}(\mathtt{pk})$ for the challenge message
  • Send ψ^* to $\mathcal{A}$

– In the case that $\mathcal{A}$ sends ψ to a prover clone $\mathtt{P}(\mathtt{sk})$
  • 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 $\mathtt{V}(\mathtt{pk})$ as the response message
  • Return $\widehat{K^*}$ as the answer for ψ^*

Figure 5.2: A One-Way-CCA2 Adversary $\mathcal{B}$ Employing a cMiM Adversary $\mathcal{A}$ for the Proof of Theorem 5.1.

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 $\mathtt{P}$ 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 5.1 follows. (*Q.E.D.*)

5.6 Security Derivation from KEM to ID Scheme

In an analogous ways, we can show the following facts.

Theorem 5.2 *If a key encapsulation mechanism* `KEM` *is secure against passive attacks, then the derived ID scheme* `ID` *is secure against passive attacks. More precisely, for any PPT adversary* $\mathcal{A}$ *that attacks* `ID` *in the passive manner, there exists an PPT adversary* $\mathcal{B}$ *that attacks* `KEM` *in the passive manner satisfying the following inequality:*

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

Theorem 5.3 *If a key encapsulation mechanism* `KEM` *is one-way-CCA1 secure, then the derived ID scheme* `ID` *is secure against concurrent two-phase attacks. More precisely, for any PPT adversary* $\mathcal{A}$ *that attacks* `ID` *in the concurrent two-phase manner, there exists an PPT adversary* $\mathcal{B}$ *that attacks* `KEM` *in the one-way-CCA1 manner satisfying the following inequality:*

$$\mathbf{Adv}^{imp\text{-}ca}_{\mathcal{A},ID}(k) \leqslant \mathbf{Adv}^{ow\text{-}cca1}_{\mathcal{B},KEM}(k).$$

Table 5.1 shows the security derivation. We can view the Table 5.1 as a design principle to construct an ID scheme with a desired security.

Table 5.1: Security Derivation from a KEM to the Derived ID Scheme.

Security of a KEM against:	$\Longrightarrow$	Security of the Derived 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

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.

5.7 Discussion

The prover `P` in Fig.5.1 is deterministic. Therefore, the derived ID scheme `ID` is prover-resettable [5]. Moreover, `ID` is also verifier-resettable because `ID` consists of 2-round interaction.

Chapter 6

A Series of Concrete ID Schemes from KEMs

In this chapter, regarding the security derivation in the last chapter (in Table 5.1) as a design principle of ID schemes, we develop a series of concrete cMiM secure ID schemes by constructing one-way-CCA2 secure KEMs.

We start with El Gamal KEM (`EGKEM`) and prove it to be secure against non-adaptive chosen ciphertext attacks on one-wayness (one-way-CCA1) in the standard model based on strong assumptions (the Gap-DL assumption and KEA).

Then, we apply a tag framework with the algebraic trick of Boneh and Boyen to `EGKEM` to make it one-way-CCA2 secure, and obtain `tKEM`. Its security is obtained based on the Gap-CDH assumption.

Next, we apply the CHK transformation and a target collision resistant hash function to exit the tag framework.

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 akin to a "black box", instead we make the Twin DH technique compatible with the algebraic trick.

The ID schemes derived from our KEMs are secure against concurrent man-in-the-middle attacks. They show the highest performance in both computational amount and message length as compared with previously known ID schemes that are secure against concurrent man-in-the-middle attacks.

As a remark, we might say that our `KEM2` and `KEM3` are 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 these, we show the security of our KEM based on a weaker assumption, that is, the CDH assumption, rather than the hashed DH assumption; we thus obtain a weaker security, that is, the one-way-CCA2 security, rather than the IND-CCA2 security. In addition, the application of the Twin DH technique to KEM2 to get KEM3 is not akin to a "black box", instead we do so by making the Twin DH technique compatible with the algebraic trick of Boneh and Boyen [3, 27].

The achievements in this chapter are announced in ProvSec 2010 [1] and AfricaCrypt 2011 [2].

6.1 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 to be one-way-CCA1 secure. The derived ID scheme from EGKEM is similar to the scheme of Stinson and Wu [38, 39]. Unlike their security proof that was done in a random oracle model, we provide a security proof in the standard model.

6.1.1 El Gamal KEM and its Security

El Gamal KEM EGKEM consists of a triple $(\mathtt{K}, \mathtt{Enc}, \mathtt{Dec})$. The construction is shown in Fig.6.1.

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 ciphertext 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 6.1 *The key encapsulation mechanism EGKEM is one-way-CCA1 secure based on the Gap-DL assumption and KEA for Grp. More precisely, for any PPT*

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 $\widehat{K}$

Figure 6.1: El Gamal KEM: EGKEM.

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 KEA which satisfy the following tight reduction.*

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

6.1.2 Proof of Theorem 6.1

An outline is stated as follows. The CDH oracle enables the solver $\mathcal{S}$ to simulate the adversary $\mathcal{A}$'s decapsulation oracle perfectly. After that the KEA extractor works to extract the answer of a DL problem instance. (Note that the Gap-DL assumption and KEA are compatible.)

Let us proceed in detail. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be as in Theorem 6.1. Using $\mathcal{A}$ as a subroutine, we construct a Gap-DL problem solver $\mathcal{S}$. The construction is illustrated in Fig.6.2.

$\mathcal{S}$ 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 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.

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

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:

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

Then, we do the following deformation;

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

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^*

Figure 6.2: A Gap-DL Problem Solver $\mathcal{S}$ for the Proof of Theorem 6.1.

6.1.3 ID Scheme Derived from EGKEM

Fig.6.3 shows the ID scheme derived from EGKEM.

6.1.4 Discussion

It is well-known that El Gamal KEM EGKEM is one-way-PA secure based on the CDH assumption. In contrast, Theorem 6.1 says a stronger fact though the assumption is a stronger one.

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 $(\mathtt{pk}, \mathtt{sk})$

Interaction
The 1st Round (Challenge)
– V: given pk as an input;
- $a \leftarrow \mathbf{Z}_q, K := X^a, h := g^a, \psi := h$
- Send ψ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)
– P: given sk as an input and receiving $\psi = h$ as an input message;
- $\widehat{K} := h^x$
- Send $\widehat{K}$ to V

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)
– V: receiving $\widehat{K}$ as an input message;
- If $\widehat{K} = K$ then return 1 else return 0

Figure 6.3: An ID Scheme Derived from EGKEM.

6.2 A Tag-Based One-Way-CCA2 Secure KEM

In this section, owing the idea to the tag-based encryption scheme of Kiltz [27], we apply a tag framework to El Gamal KEM EGKEM to make it one-way-CCA2 secure.

6.2.1 A Tag-Based KEM and its Security

A tag-based KEM tKEM consists of a triple $(\mathtt{K}, \mathtt{Enc}, \mathtt{Dec})$. The construction is shown in Fig.6.4.

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 ciphertext 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 holds, then Dec computes the decapsulation $\widehat{K} = h^x$. Otherwise, Dec puts $\widehat{K} = \perp$. Dec returns $\widehat{K}$.

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 $\widehat{K}$

Figure 6.4: A Tag-Based One-Way-CCA2 KEM: tKEM.

The security of tKEM is stated as follows.

Theorem 6.2 *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.*

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

6.2.2 Proof of Theorem 6.2

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. Moreover, the algebraic trick of the tag framework 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 response.

Let us proceed in detail. Let $\mathcal{A}$ be as in Theorem 6.2. Using $\mathcal{A}$ as a subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$. The construction is illustrated in Fig.6.5.

$\mathcal{S}$ 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 $h^* = W g^{a^*}, d^* = (h^*)^u$ and $\psi^* = (h^*, d^*)$. $\mathcal{S}$ inputs $\mathtt{pk}$ and ψ^* into $\mathcal{A}$.

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;

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

$\mathcal{S}$ 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.

First, 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;

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

Second, in the case SIMDEC, $\mathcal{S}$ can simulate the decapsulation 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; $\quad \mathbf{Adv}^{\text{gap-cdh}}_{\mathcal{S},\mathtt{Grp}}(k) \geqslant \mathbf{Adv}^{\text{stag-ow-cca2}}_{\mathcal{A},\mathtt{tKEM}}(k)$. $(Q.E.D.)$

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: $\mathtt{t}^* \leftarrow \mathcal{A}(1^k)$
– $u \leftarrow \mathbf{Z}_q, X := V, Y := X^{-\mathtt{t}^*} g^u$, $\mathtt{pk} := (q, g, X, Y)$
– $a^* \in \mathbf{Z}_q, h^* = W g^{a^*}, d^* = (h^*)^u, \psi^* = (h^*, d^*)$
– 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 $Z := \widehat{K^*}/X^{a^*}$

Figure 6.5: A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 6.2.

6.2.3 ID Scheme Derived from tKEM

Fig.6.6 shows the ID scheme derived from tKEM.

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

Tag-Receiving
– P and V are given a tag $\mathtt{t} \in \mathbf{Z}_q$

Interaction
The 1st Round (Challenge)
– V: given pk as an input;
- $a \leftarrow \mathbf{Z}_q, h := g^a$
- $d := (X^{\mathtt{t}}Y)^a, K := X^a, \psi = (h, d)$
- Send ψ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)
– P: given sk as an input and receiving $\psi = (h, d)$ as an input message;
- If $h^{\mathtt{t}x+y} \neq d_1$ then $\widehat{K} :=\perp$ else $\widehat{K} := h^x$
- Send $\widehat{K}$ to V

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)
– V: receiving $\widehat{K}$ as an input message;
- If $\widehat{K} = K$ then return 1 else return 0

Figure 6.6: An ID Scheme Derived from tKEM.

6.2.4 Discussion

We note that the selective tag model for a security proof is compatible with the conversion. That is, a KEM is secure in the selective tag model, then the derived ID scheme is secure in the selective tag model.

6.3 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.

6.3.1 A KEM by the CHK transformation and its Security

A KEM KEM1 by the CHK transformation of the tag-based KEM tKEM consists of a triple $(\mathtt{K}, \mathtt{Enc}, \mathtt{Dec})$. The construction is shown in Fig.6.7.

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 $\mathbf{Z}_q$, 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 of a strong one-time signature $\mathtt{OTS} = (\mathtt{SGK}, \mathtt{Sign}, \mathtt{Vrfy})$. 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 we take a hash value $H(\mathtt{vk})$ of vk before the exponentiation so that $H(\mathtt{vk}) < q$ as an integer, though we do not write it explicitly). 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 ciphertext is ψ. Enc returns the pair (K, ψ).

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 calculates $\widehat{K} = h^x$. Dec returns $\widehat{K}$.

The security of KEM1 is stated as follows.

Theorem 6.3 *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 OTS employed. More precisely, for any PPT adversary $\mathcal{A}$ there exist a PPT Gap-CDH problem solver $\mathcal{S}$ and a PPT forger $\mathcal{F}$ on OTS which satisfy the following*

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 $(\mathtt{pk}, \mathtt{sk})$

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 $\widehat{K}$

Figure 6.7: A One-Way-CCA2 KEM by the CHK Transformation: KEM1.

tight reduction.

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

6.3.2 Proof of Theorem 6.3

An outline is stated as follows. The proof goes almost the same way as the proof of Theorem 6.2 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.

Let us proceed in detail. Let $\mathcal{A}$ be as in Theorem 6.3. Using $\mathcal{A}$ as a subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$. The construction is illustrated in Fig.6.8.

$\mathcal{S}$ 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. Invoking 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 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.

First, 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;

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

Second, 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;

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

That is;

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

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

Given (q, g, V, W) as an input;

Initial Setting
– Initialize inner state
– $(\mathtt{vk}^*, \mathtt{sgk}^*) \leftarrow \mathtt{SGK}(1^k)$
– $u \leftarrow \mathbf{Z}_q, X := V, Y := X^{-\mathtt{vk}^*} g^u$, $\mathtt{pk} := (q, g, X, Y)$
– $a^* \leftarrow \mathbf{Z}_q, h^* := W g^{a^*}, d^* := (h^*)^u$, $\sigma^* \leftarrow \mathtt{Sign}_{\mathtt{sgk}^*}((h^*, d^*))$
– $\psi^* := (\mathtt{vk}^*, (h^*, d^*), \sigma^*)$, 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 $Z := \widehat{K}^* / X^{a^*}$

Figure 6.8: A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 6.3.

Claim 6.1 *The probability that the case* ABORT *occurs is negligible in* k.

Proof of Claim 6.1. 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;

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

Combining the two equalities, we get

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

The right hand side is negligible in k by the assumption in Theorem 6.3. (*Q.E.D.*)

6.3.3 ID Scheme Derived from KEM1

Fig.6.9 shows the ID scheme derived from KEM1.

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 $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Challenge)

– V: given pk as an input;

- $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$
- $a \leftarrow \mathbf{Z}_q, K := X^a, h := g^a, d := (X^{\mathtt{vk}}Y)^a, \sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}((h, d))$
- $\psi := (\mathtt{vk}, (h, d), \sigma)$
- Send ψ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)

– P: given sk as an input and receiving $\psi = (\mathtt{vk}, (h, d), \sigma)$ as an input message;

- $\tau \leftarrow H_\kappa(h)$
- If $\mathtt{Vrfy}_{\mathtt{vk}}((h, d), \sigma) \neq 1$ or $h^{(\mathtt{vk})x+y} \neq d$ then $\widehat{K} :=\perp$ else $\widehat{K} := h^x$
- Send $\widehat{K}$ to V

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)

– V: receiving $\widehat{K}$ as an input message;

- If $\widehat{K} = K$ then return 1 else return 0

Figure 6.9: An ID Scheme Derived from KEM1.

6.3.4 Discussion

When we use a one-time signature such that $|\mathtt{vk}| \geqslant |q|$, we should take a hash value $H(\mathtt{vk})$ by a hash function $H : \{0,1\}^* \to \mathbf{Z}_q$ for computations in $\mathbf{Z}_q$. In this setting, the security statement in Theorem 6.3 needs a target collision resistance for $\{H\}$ to make a collision case (: the case that $\mathtt{vk} \neq \mathtt{vk}^*$ and $H(\mathtt{vk}) = H(\mathtt{vk}^*)$) negligible.

6.4 A One-Way-CCA2 Secure KEM with a TCR 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 hash function value τ.

6.4.1 A KEM with a TCR Hash Function and its Security

A KEM `KEM2` with TCR hash function consists of a triple $(\mathtt{K}, \mathtt{Enc}, \mathtt{Dec})$. The construction is shown in Fig.6.10.

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 (`pk`, `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 ciphertext 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` calculates $\widehat{K} = h^x$. `Dec` returns $\widehat{K}$.

The security of `KEM2` is stated as follows.

Theorem 6.4 *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 $Hfam(1^k) = \{H_\kappa\}_{\kappa \in Hkey(1^k)}$ employed. More precisely, for any PPT adversary $\mathcal{A}$ there exist a PPT Gap-CDH problem solver $\mathcal{S}$ and a PPT collision-*

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 $\widehat{K}$

Figure 6.10: A One-Way-CCA2 KEM with a TCR Hash Function: KEM2.

finder $\mathcal{CF}$ on Hfam which satisfy the following tight reduction.

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

6.4.2 Proof of Theorem 6.4

An outline is stated as follows. The proof goes almost the same way as the proof of Theorem 6.2 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 6.4. Using $\mathcal{A}$ as a subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$. The construction is illustrated in Fig.6.11.

$\mathcal{S}$ 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.

First, 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;

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

Second, 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 $\Pr[\text{ABORT}]$ is negligible in k.

Given (q, g, V, W) as an input;

Initial Setting
– Initialize its inner state
– $a^* \leftarrow \mathbf{Z}_q, h^* := W g^{a^*}$
– $\kappa \leftarrow \mathit{Hkey}(1^k), \tau^* \leftarrow H_\kappa(h^*)$
– $u \leftarrow \mathbf{Z}_q, X := V, Y := X^{-\tau^*} g^u, d^* = (h^*)^u$
– $\mathtt{pk} := (q, g, X, Y, \kappa), \psi^* := (h^*, d^*)$
– Invoke $\mathcal{A}$ on $\mathtt{pk}$ 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 $\psi = (h, d)$;
  • $\tau \leftarrow H_\kappa(h)$
  • 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 $Z := \widehat{K}^*/X^{a^*}$

Figure 6.11: A Gap-CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 6.4.

Claim 6.2 *The probability that* ABORT *occurs is negligible in* k.

Proof of Claim 6.2. 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:

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

Combining the two equalities, we get

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

The right hand side is negligible in k by the assumption in Theorem 6.4. (*Q.E.D.*)

6.4.3 ID Scheme Derived from KEM2

Fig.6.12 shows the ID scheme derived from KEM2.

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

Interaction
The 1st Round (Challenge)
– V: given pk as an input;
- $a \leftarrow \mathbf{Z}_q, h := g^a, \tau \leftarrow H_\kappa(h)$
- $d := (X^\tau Y)^a, K := X^a, \psi = (h, d)$
- Send ψ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)
– P: given sk as an input and receiving $\psi = (h, d)$ as an input message;
- $\tau \leftarrow H_\kappa(h)$
- If $h^{\tau x+y} \neq d_1$ then $\widehat{K} :=\perp$ else $\widehat{K} := h^x$
- Send $\widehat{K}$ to V

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)
– V: receiving $\widehat{K}$ as an input message;
- If $\widehat{K} = K$ then return 1 else return 0

Figure 6.12: An ID Scheme Derived from KEM2.

6.4.4 Discussion

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

6.5 A One-Way-CCA2 Secure KEM by the Twin DH 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, 27] introducing necessary random variables.

6.5.1 A KEM by the Twin DH Technique and its Security

A KEM KEM3 by the Twin Diffie-Hellman technique consists of a triple (K, Enc, Dec). The construction is shown in Fig.6.13.

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 the ciphertext 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 $\widehat{K} = h^{x_1}$. Note that (g, X_1, h, K) is a DH tuple. Finally, Dec returns $\widehat{K}$.

The security of KEM3 is stated as follows.

Theorem 6.5 *The key encapsulation mechanism KEM3 is one-way-CCA2 secure based on the CDH assumption and the target collision resistance of a hash func-*

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 $\widehat{K}$

Figure 6.13: A One-Way-CCA2 KEM by the Twin DH Technique: KEM3.

tion family $Hfam(1^k) = \{H_\kappa\}_{\kappa \in Hkey(1^k)}$ *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.*

$$\mathbf{Adv}^{ow\text{-}cca2}_{\mathcal{A},KEM3}(k) \leqslant \frac{q_{dec}}{q} + \mathbf{Adv}^{cdh}_{\mathcal{S},Grp}(k) + \mathbf{Adv}^{tcr}_{\mathcal{CF},Hfam}(k).$$

6.5.2 Proof of Theorem 6.5

An outline is stated as follows. The proof goes almost the same way as the proof of Theorem 6.2 except that we have to treat an argument for the trapdoor test (2.1) which decides whether a given tuple is a twin DH tuple or not.

Let us proceed in detail. Let $\mathcal{A}$ be as in Theorem 6.5. Using $\mathcal{A}$ as a subroutine, we construct a PPT CDH problem solver $\mathcal{S}$ as shown in Fig.6.14, 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 $h^* = W g^{a^*}$. 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 $\mathtt{pk}$ 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.

$$y_i = -\tau^* x_i + u_i, \quad i = 1, 2. \tag{6.1}$$

$\mathcal{S}$ 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.

First, 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;

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

Second, $\mathcal{S}$ 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.

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 $\mathtt{pk}$ 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 $\psi = (h, d_1, d_2)$
  • 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 $Z := \widehat{K^*}/X_1^{a^*}$

Figure 6.14: A CDH Problem Solver $\mathcal{S}$ for the Proof of Theorem 6.5.

Claim 6.3 *$(g, X_1^\tau Y_1, X_2^\tau Y_2, h, d_1, d_2)$ 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 6.3. 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.*)

Claim 6.4 *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 6.4. 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 2.1. (*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 6.4, 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;

$$\Pr[\bigvee_{i=1}^{q_{dec}} \text{OVERLOOK}_i] \leqslant \frac{q_{dec}}{q}. \tag{6.2}$$

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 (6.1).

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, $\textsc{Overlook}_i$ never occurs for $i = 1, \ldots, q_{dec}$ and Abort does not happen:

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

Using the inequality (6.2), we get:

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

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

Claim 6.5 *The probability that* $(\bigwedge_{i=1}^{q_{dec}} (\neg \textsc{Overlook}_i)) \wedge \textsc{Abort}$ *occurs is negligible in* k.

Proof of Claim 6.5. 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:

$$\Pr[(\bigwedge_{i=1}^{q_{dec}} (\neg \text{Overlook}_i)) \wedge \text{Abort}] \leqslant \Pr[\text{Collision}]. \tag{6.3}$$

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:

$$\Pr[\text{Collision}] = \mathbf{Adv}^{\text{tcr}}_{\mathcal{CF}, Hfam}(k). \tag{6.4}$$

Combining (6.3) and (6.4), we get

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

The right hand side is negligible in k by the assumption in Theorem 6.5. (*Q.E.D.*)

6.5.3 ID Scheme Derived from KEM3

Fig.6.15 shows the ID scheme derived from KEM3.

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 $(\mathtt{pk}, \mathtt{sk})$

Interaction
The 1st Round (Challenge)
– V: given pk as an input;
- $a \leftarrow \mathbf{Z}_q, h := g^a, \tau \leftarrow H_\kappa(h)$
- $d_1 := (X_1^\tau Y_1)^a, d_2 := (X_2^\tau Y_2)^a, K := X_1^a, \psi = (h, d_1, d_2)$
- Send ψ to P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)
– P: 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 $h^{\tau x_2 + y_2} \neq d_2$ then $\widehat{K} :=\perp$ else $\widehat{K} := h^{x_1}$
- Send $\widehat{K}$ to V

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)
– V: receiving $\widehat{K}$ as an input message;
- If $\widehat{K} = K$ then return 1 else return 0

Figure 6.15: An ID Scheme Derived from KEM3.

6.5.4 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 $\widetilde{\mathtt{KEM3}}$.

In $\widetilde{\mathtt{KEM3}}$, 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 $\mathbf{Adv}^{\mathrm{cr}}_{\mathcal{CF}',Hfam}(k)$ below means collision resistance).

Corollary 6.1 *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.*

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

The ID scheme derived from KEM$\tilde{3}$ is shown in Fig.6.16. By the above tuning, the maximum message length reduces to two elements $2G$ of Grp plus one hash value h of H_κ.

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 $(\mathtt{pk}, \mathtt{sk})$

Interaction

The 1st Round (Challenge)

– V: 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 P

$$\mathtt{P} \xleftarrow{\psi} \mathtt{V}$$

The 2nd Round (Response)

– P: 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 V

$$\mathtt{P} \xrightarrow{\widehat{K}} \mathtt{V}$$

The Final Decision (Verification)

– V: receiving $\widehat{K}$ as an input message;

- If $\widehat{K} = K$ then return 1 else return 0

Figure 6.16: An ID Scheme Derived from KEM3̃.

Chapter 7

Efficiency Comparison

In this chapter, 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 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 KEM3̃ shows the highest performance in both.

7.1 Four Categories of Comparable ID schemes

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 `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 `HKKEM`) is the most comparable.

7.2 Comparison Results

Table 7.2 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 7.2, we compare computational amount by counting the number of exponentiations in a prover `P` and a verifier `V`. We also compare the maximum message length.

In Computational Amount column, notation (s, t) or (s, t, u) means that the prover `P` or the verifier `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 `P` and $T_1(\mathtt{V})$ for a verifier `V` 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 7.2, 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 [1].

T_1 and T_2 in Computational Amount column of Table 7.2 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.

In some applications, computational amount of a prover is more important than that of a verifier because of limited computational resource (in a smartcard, for instance). The ID scheme from our KEM2 achieves $T_1(\mathtt{P}) = T_2(\mathtt{P}) = 2.00$, which is the smallest amount among ID schemes secure against cMiM attacks.

It is noteworthy that our ID schemes are resettably secure for both a prover and a verifier, which is crucially helpful for smartcards service [5] and virtual machine service in the Cloud Computing [41].

[1] For example, (referring to [9],) in the case that G_q is a 160-bit elliptic curve group with $k = |q| = 160$, an group exponentiation with the crypto++ library is about 3300 hashes for an unoptimized implementation. On the other hand, the Lamport signature needs $2k = 320 \approx 0.10 \times 3300$ for a key generation and $k = 160 \approx 0.05 \times 3300$ for a verification.

Table 7.1: 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) (and (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.
SchID	OMDL	-	ca
GenID	DL&sRSA	-	cMiM
EGKEM	GDL&KEA	OW-CCA1	ca
CSKEM	DDH	IND-CCA2	cMiM
HKKEM	CDH	OW-CCA2	cMiM
Our tKEM	GCDH(stag)	OW-CCA2	cMiM
Our KEM1	GCDH	OW-CCA2	cMiM
Our KEM2	GCDH	OW-CCA2	cMiM
Our KEM3	CDH	OW-CCA2	cMiM
Our KEM$\tilde{3}$	CDH	OW-CCA2	cMiM

ID Sch. or KEM	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	(1,0)	1.00	1.00	(0,1,0)	2.00	1.75	3.00	2.75	G
GenID	(3,1)+kg	5.10	4.85	($\frac{1}{2}$,2,0)+vf	4.55	4.05	9.65	8.90	$G + \mathtt{vk}$
EGKEM	(1,0)	1.00	1.00	(2,0,0)	2.00	2.00	3.00	3.00	G
CSKEM	(3,0)	3.00	3.00	(3,1,0)	5.00	4.75	8.00	7.75	$3G$
HKKEM	(3,0)	3.00	3.00	(2,0,2)	8.00	5.75	11.00	8.75	$3G$
Our tKEM	(2,0)	2.00	2.00	(2,1,0)	4.00	3.75	6.00	5.75	$2G$
Our KEM1	(2,0)+vf	2.05	2.05	(2,1,0)+kg	4.10	3.85	6.15	5.90	$2G + \mathtt{vk} + \sigma$
Our KEM2	(2,0)	2.00	2.00	(2,1,0)	4.00	3.75	6.00	5.75	$2G$
Our KEM3	(3,0)	3.00	3.00	(2,2,0)	6.00	5.50	9.00	8.50	$3G$
Our KEM$\tilde{3}$	(3,0)	3.00	3.00	(2,2,0)	6.00	5.50	9.00	8.50	$2G + h$

Chapter 8

Conclusions

We tackled the problem of constructing efficient identification schemes that prevent concurrent man-in-the-middle attacks. Our choice was not Σ-protocols but challenge-and-response ID schemes from key encapsulation mechanisms. The strategy "ID schemes from KEMs" succeeded in the sense that we got efficient cMiM secure ID schemes by constructing one-way-CCA2 KEMs. There are two reasons why they are efficient: the first reason is that 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). The second reason is that a KEM only need to be one-way-CCA2 secure rather than to be IND-CCA2 secure for the derived ID scheme to be cMiM secure.

Concretely, we obtained a series of ID schemes by constructing KEMs. We started from El Gamal KEM (`EGKEM`) and proved it to be one-way-CCA1 secure based on the Gap-DL assumption and KEA. Then, we applied a tag framework with the algebraic trick of Boneh and Boyen to `EGKEM` to make it one-way-CCA2 secure (`tKEM`). The security is based on the Gap-CDH assumption. Next, we applied the CHK transformation and a target collision resistant hash function to exit the tag framework (`KEM1` and `KEM2`, respectively). Finally, as it is better to rely on the CDH assumption rather than the Gap-CDH assumption, we applied the Twin DH technique of Cash, Kiltz and Shoup (`KEM3` and `KEM3̃`).

Our ID schemes are cMiM secure, and show the highest performance in both computational amount and message length as compared with previously known ID schemes secure against cMiM attacks. Our challenge-and-response ID schemes exceeded Σ-protocols as far as preventing cMiM attacks.

Next directions are: factoring-based and lattice-based constructions, or a signature-based construction. Another important themes are identity-based identification scheme and attribute-based identification scheme.

Bibliography

[1] H. Anada, S. Arita, *"Identification Schemes of Proofs of Ability Secure against Concurrent Man-in-the-Middle Attacks"*. In Proc. of *ProvSec 2010*, Malacca, Malaysia, Oct. 13-15, 2010, Lecture Notes in Computer Science, vol. 6402, pp. 18-34, Springer-Verlag, Heidelberg, Germany.

[2] H. Anada, S. Arita, *"Identification Schemes from Key Encapsulation Mechanisms"*. In Proc. of *AFRICACRYPT 2011*, Dakar, Senegal, July. 5-7, 2011, Lecture Notes in Computer Science, vol. 6737, pp. 59-76, Springer-Verlag, Heidelberg, Germany.

[3] D. Boneh, X. Boyen, *Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles*. In Proc. of *EUROCRYPT 2004*, Interlaken, Switzerland, May 2-6, 2004, Lecture Notes in Computer Science, vol. 3027, pp. 223-238, Springer-Verlag, Heidelberg, Germany.

[4] D. Boneh, X. Boyen, *"Short Signatures without Random Oracles"*. In Proc. of *EUROCRYPT 2004*, Interlaken, Switzerland, May 2-6, 2004, Lecture Notes in Computer Science, vol. 3027, pp. 56-73, Springer-Verlag, Heidelberg, Germany.

[5] M. Bellare, M. Fischlin, S. Goldwasser, S. Micali, *"Identification Protocols Secure against Reset Attacks"*. In Proc. of *EUROCRYPT 2001*, Innsbruck, Austria, May 6-10, 2001, Lecture Notes in Computer Science, vol. 2045, pp. 495-511, Springer-Verlag, Heidelberg, Germany.

[6] M. Bellare, O. Goldreich, *"On Defining Proofs of Knowledge"*. In Proc. of *CRYPTO '92*, Santa Barbara, CA, USA, Aug. 16-20, 1992, Lecture Notes in Computer Science, vol. 740, pp. 390-420, Springer-Verlag, Heidelberg, Germany.

[7] M. Bellare, A. Palacio, *"GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks"*. In Proc. of *CRYPTO 2002*, Santa Barbara, CA, USA, Aug. 18-22, 2002, Lecture Notes in Computer Science, vol. 2442, pp. 162-177, Springer-Verlag, Heidelberg, Germany.

[8] M. Bellare, A. Palacio, *"The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols"*. In Proc. of *CRYPTO 2004*, Santa Barbara, CA, USA, Aug. 15-19, 2004, Lecture Notes in Computer Science, vol. 3152, pp. 273-289, Springer-Verlag, Heidelberg, Germany.

[9] M. Bellare, S. Shoup, *"Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir without Random Oracles"*. In Proc. of *PKC 2007*, Beijing, China, April 16-20, 2007, Lecture Notes in Computer Science, vol. 4450, pp. 201-216, Springer-Verlag, Heidelberg, Germany.

[10] R. Canetti, R. R. Dakdouk, *"Extractable Perfectly One-way Functions"*. In Proc. of *ICALP 2008*, Reykjavik, Iceland, July 7-11, 2008, Lecture Notes in Computer Science, vol. 5126, pp. 449-460, Springer-Verlag, Heidelberg, Germany.

[11] R. Cramer, I. Damgård, J. B. Nielsen, *"Multiparty Computation from Threshold Homomorphic Encryption"*. In Proc. of *EUROCRYPT 2001*, Innsbruck, Austria, May 6-10, 2001, Lecture Notes in Computer Science, vol. 2045, pp. 280-300, Springer-Verlag, Heidelberg, Germany.

[12] R. Canetti, S. Halevi, J. Katz, *"Chosen-Ciphertext Security from Identity-Based Encryption"*. In Proc. of *EUROCRYPT 2004*, Interlaken, Switzerland, May 2-6, 2004, Lecture Notes in Computer Science, vol. 3027, pp. 207-222, Springer-Verlag, Heidelberg, Germany.

[13] D. Cash, E. Kiltz, V. Shoup, *"The Twin Diffie-Hellman Problem and Applications"*. In Proc. of *EUROCRYPT 2008*, Istanbul, Turkey, April 13-17, 2008, Lecture Notes in Computer Science, 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, V. Shoup, *"A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack"*. In Proc. of *CRYPTO '98*, Santa Barbara, CA, USA, Aug. 23-27, 1998, Lecture Notes in Computer Science, vol. 1462, pp. 13-25, Springer-Verlag, Heidelberg, Germany.

[15] R. Cramer, V. Shoup, *"Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack"*. SIAM Journal on Computing, vol. 33, num. 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"*. In Proc. of *CRYPTO '91*, Santa Barbara, CA, USA, Aug. 11-15, 1991, Lecture Notes in Computer Science, 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"*. In Proc. of *CRYPTO '84*, Santa Barbara, California, USA, August 19-22, 1984, Lecture Notes in Computer Science, vol. 196, pp. 10-18, Springer-Verlag, Heidelberg, Germany.

[19] E. Fujisaki, *"New Constructions of Efficient Simulation-Sound Commitments Using Encryption"*. In Proc. of *The 2011 Symposium on Cryptography and Information Security, SCIS 2011*, Kokura, Japan, January 25-28, 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"*. In Proc. of *CRYPTO 2004*, Santa Barbara, CA, USA, Aug. 15-19, 2004, Lecture Notes in Computer Science, vol. 3152, pp. 220-236, Springer-Verlag, Heidelberg, Germany.

[21] S. Goldwasser, S. Micali, C. Rackoff, *"The Knowledge Complexity of Interactive Proof Systems"*. SIAM Journal on Computing, vol. 18, num. 1, pp. 186-208, Feb. 1989.

[22] L. Guillou, J. J. Quisquater, *"A Paradoxical Identity-Based Signature Scheme Resulting from Zero-Knowledge"*. In Proc. of *CRYPTO '88*, Santa Barbara, CA, USA, Aug. 21-25, 1988, Lecture Notes in Computer Science, vol. 403, pp. 216-231, Springer-Verlag, Heidelberg, Germany.

[23] J. Herranz, D. Hofheinz, E. Kiltz, *"The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure"*. Cryptology ePrint Archive, 2006/207, http://eprint.iacr.org/

[24] G. Hanaoka, K. Kurosawa, *"Efficient Chosen Ciphertext Secure Public Key Encryption under the Computational Diffie-Hellman Assumption"*. In Proc. of *ASIACRYPT 2008*, Melbourne, Australia, December 7-11, 2008, Lecture Notes in Computer Science, 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"*. In Proc. of *EUROCRYPT 2003*, Warsaw, Poland, May 4-8, 2003, Lecture Notes in Computer Science, vol. 2656, pp. 211-228, Springer-Verlag, Heidelberg, Germany.

[27] E. Kiltz, *"Chosen-Ciphertext Security from Tag-Based Encryption"*. In Proc. of *TCC 2006*, New York, NY, USA, March 4-7, 2006, Lecture Notes in Computer Science, vol. 3876, pp. 581-600, Springer-Verlag, Heidelberg, Germany.

[28] E. Kiltz, *"Chosen-Ciphertext Security from Hashed Gap Diffie-Hellman"*. In Proc. of *PKC 2007*, New York, NY, USA, March 4-7, 2006, Lecture Notes in Computer Science, vol. 3876, pp. 581-600, Springer-Verlag, Heidelberg, Germany.

[29] E. Kiltz, personal communication at ProvSec 2010, Malacca (2010)

[30] K. Kurosawa, Y. Desmedt, *"A New Paradigm of Hybrid Encryption Scheme"*. In Proc. of *CRYPTO 2004*, Santa Barbara, CA, USA, Aug. 15-19, 2004, Lecture Notes in Computer Science, 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, S. Wolf, *"Lower Bounds on Generic Algorithms in Groups"*. In Proc. of *EUROCRYPT '98*, Espoo, Finland, May 31-June 4, 1998, Lecture Notes in Computer Science, vol. 1403, pp. 72-84, Springer-Verlag, Heidelberg, Germany.

[33] M. Naor, M. Yung, *"Universal One-Way Hash Functions and their Cryptographic Applications"*. In Proc. of *the 21st Symposium on Theory of Computing*, Seattle, Washington, USA, May 14-17, 1989, pp. 33-43, Association for Computing Machinery, New York, USA.

[34] T. Okamoto, D. Pointcheval, *"The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes"*. In Proc. of *PKC 2001*, Cheju Island, Korea, February 13-15, 2001, Lecture Notes in Computer Science, vol. 1992, pp. 104-118, Springer-Verlag, Heidelberg, Germany.

[35] D. Pointcheval, *"Chosen-Ciphertext Security for Any One-Way Cryptosystem"*. In Proc. of *PKC 2000*, Melbourne, Victoria, Australia, January 18-20, 2000, Lecture Notes in Computer Science, vol. 1751, pp. 129-146, Springer-Verlag, Heidelberg, Germany.

[36] J. Rompel, *"One-Way Functions are Necessary and Sufficient for Secure Signatures"*. In Proc. of *the 22nd Annual Symposium on Theory of Computing*, Baltimore, MD, USA, May 13-17, 1990, pp.387-384, Association for Computing Machinery, New York, USA.

[37] C. P. Schnorr, *"Efficient Identification and Signatures for Smart Cards"*. In Proc. of *CRYPTO '89*, Santa Barbara, CA, USA, Aug. 20-24, 1989, Lecture Notes in Computer Science, vol. 435, pp. 239-252, Springer-Verlag, Heidelberg, Germany.

[38] D. R. Stinson, J. Wu, *"An Efficient and Secure Two-flow Zero-Knowledge Identification Protocol"*. Journal of Mathematical Cryptology, vol. 1, issue 3, pp. 201-220, Aug. 2007.

[39] J. Wu, 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"*. In Proc. of *EUROCRYPT 2005*, Aarhus, Denmark, May 22-26, 2005, Lecture Notes in Computer Science, vol. 3494, pp. 114-127, Springer-Verlag, Heidelberg, Germany.

[41] S. Yilek, *"Resettable Public-Key Encryption: How to Encrypt on a Virtual Machine"*. In Proc. of *the Cryptographers' Track at the RSA Conference 2010*, San Francisco, CA, USA, March 1-5, 2010, Lecture Notes in Computer Science, vol. 5985, pp. 41-56, Springer-Verlag, Heidelberg, Germany.

[42] S. M. Yen, C. S. Laih, A. K. Lenstra, *"Multi-exponentiation"*. In IEE Proc. *Computers and Digital Techniques*, vol. 141, issue. 6, pp. 325-326, Nov. 1994.