A preliminary version of this paper appeared in Proceedings of the 4th International Conference on Provable Security - ProvSec 2010, Lecture Notes in Computer Science vol. 6402, pp. 18-34, Springer-Verlag [1], under the title of “Identification Schemes of Proofs of Ability Secure against Concurrent Man-in-the-Middle Attacks”. This is a preprint of the full version.

Identification Schemes of Proofs of Malleability Secure against Concurrent Man-in-the-Middle Attacks

Hiroaki Anada and Seiko Arita

Institute of Information Security, Yokohama, Japan

hiroaki.anada@gmail.com, arita@iisec.ac.jp

December 1, 2010

Abstract

We provide identification schemes secure against concurrent man-in-the-middle attacks. For that purpose, we construct a series of four identification schemes. They are all *proofs of malleability* except the first prototype scheme. To define the notion, we firstly give a definition of non-malleable functions and malleability extractors. As a concrete example, we show that exponentiation functions are non-malleable functions with respect to the multiplication relation. By this non-malleability and a tag framework with algebraic trick, we are able to construct a tag-based scheme that is a proof of malleability, and that achieves the desired security based on the Gap Computational Diffie-Hellman Assumption. A generic method, the CHK transformation, is attractive to exit the tag framework, but the obtained scheme has somewhat long message length. The matter is resolved by the use of a target collision resistant hash function. This fourth scheme performs highly efficiently not only in message length but also in computational amount. Actually, it is shown that it performs better than the Cramer-Shoup-based ID scheme.

keywords: identification scheme, concurrent man-in-the-middle attack, proof of malleability, non-malleable function, malleability extractor, gap computational Diffie-Hellman assumption.

1 Introduction

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

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

However, even for such ID schemes, there is still a strong threat by the following active attack. Pretending a verifier, an adversary accesses a prover application (on a client PC, for instance), and invokes many clones of the application. Interacting with those clones, the adversary embeds some cheating trick in messages and collects information of the secret key from the responses of those clones. Afterwards, it tries to impersonate the prover against a true verifier (on a server, for instance) using those collected information. This situation is modeled as *two-phase concurrent attack* [7] in cryptography. If the adversary can access prover clones *during* trying impersonation, the attack is called *concurrent man-in-the-middle attack* and considered one of the strongest threat, especially in the Internet [7].

Historically, there have been two types of ID schemes. One is challenge-and-response type obtained easily from encryption schemes or signature schemes, and another is the Σ-protocol type [11] which is a kind of proofs of knowledge [18, 5] consisting of 3-round interaction. Most of known traditional ID schemes, such as the Schnorr Scheme [31] and the Guillou-Quisquater Scheme [19], are the Σ-protocol type because they are faster than challenge-and-response type. But what is problematic is that the security model is only against two-phase concurrent attacks. Moreover, the used assumptions are the one more type (the One More Discrete Log Assumption or the One More RSA Inversion Assumption [6, 7]), which are stronger than the ordinary assumptions.

After those traditional schemes, security against concurrent man-in-the-middle attacks is achieved by Katz [21] and Gennaro [17]. But for the Katz Scheme, the security model is with timing constraint, not against full concurrent man-in-the-middle attacks. Moreover, the protocol utilizes the so-called OR-Proof technique and is rather complicated. As for the Gennaro Scheme, a multi-trapdoor commitment is embedded in the protocol to remove those timing constraint. As a result, it needs some computation and is not so fast as challenge-and-response ID scheme obtained from the Cramer-Shoup Encryption Scheme [13], for example. In addition, the security of the Gennaro Scheme is also based on the strong type assumptions (the Strong Diffie-Hellman (SDH) Assumption or the Strong RSA Assumption).

One of the reason why it is difficult to construct an ID scheme secure against concurrent man-in-the-middle attacks is that we are rooted in the category of Σ-protocols. In the security proof, depending on the so-called special soundness property of Σ-protocols, we can construct a knowledge extractor employing any given adversary as subroutine. There the knowledge extractor rewinds the adversary and extracts the secret key (the Reset Lemma [7]). But in the concurrent man-in-the-middle composition, this rewinding strategy gives rise the difficulty. That is, large amount of computations are needed to do nested rewindings for the knowledge extractor to simulate concurrent prover clones, and eventually the security reduction becomes far from tight. Or, to cut off those computations in the security proof, some costly techniques are utilized in the protocols and strong assumptions are required in the security proofs as we have reviewed.

1.1 Our Contribution

Unlike those known ID schemes, our approach is neither a Σ-protocol nor a proof of knowledge. We take an approach of *a proof of malleability*, which is a new notion we propose in the paper. An ID scheme of a proof of malleability is a 5-tuple $(\mathtt{K}, \mathtt{P}, \mathtt{V}, f, \mathcal{R})$ where $(\mathtt{K}, \mathtt{P}, \mathtt{V})$ is a triple of probabilistic polynomial time (PPT) algorithms which represents an ID scheme, f is *a non-malleable function with respect to a relation* $\mathcal{R}$. Here we say that f is non-malleable with $\mathcal{R}$ if, for any given PPT algorithm $\mathcal{E}$ that receives function values $f(x_1), \ldots, f(x_n)$ as input, it is hard to output the related value $f(x)$ satisfying $x = \mathcal{R}(x_1, \ldots, x_n)$.

In proving the security of the ID scheme, we execute a proof of malleability. That is, employing any given adversary $\mathcal{A}$ as subroutine, we construct a PPT algorithm $\mathcal{E}$ that receives function values $f(x_1), \ldots, f(x_n)$ as input and outputs the related value $f(x)$, $x = \mathcal{R}(x_1, \ldots, x_n)$. This construction reduces the advantage of $\mathcal{A}$ to the advantage of $\mathcal{E}$. Here $\mathcal{E}$ is called *a malleability extractor against the non-malleability of f.*

We will pick up an exponentiation function $f(x) = g^x$ with values in a cyclic group of a prime order q as a concrete non-malleable function. That is, taking the multiplication relation $\mathcal{R}(x_1, x_2) = x_1x_2$, we get the non-malleability of f based on the Computational Diffie-Hellman (CDH) Assumption.

Using the concrete non-malleable function, we construct a series of four ID schemes step by step. We start from the first prototype scheme that consists of half the operation of Diffie-Hellman Key-Exchange. In the security proof, we need the Gap Discrete Log (Gap-DL) Assumption and the Knowledge-of-Exponent Assumption (KEA) only to get weak security, that is, the security against two-phase concurrent attacks.

We modify the first scheme to make it a proof of malleability by applying a tag framework. Actually, by using the tag framework, we are able to construct a malleability extractor against the non-malleability of the exponentiation function in the security proof, and hence we can reduce the security to the non-malleability. The tag framework also works to simulate concurrent prover clones in man-in-the-middle composition to get the security against concurrent man-in-the-middle attacks, where we owe the idea to the tag-based encryption scheme of Kiltz [23].

To leave the tag framework of the second scheme, the CHK transformation [12] is applied to get the third scheme. That is, tag is replaced by a one-time verification key of an employed strong one-time signature. The CHK transformation is generic and steady, but it brings a disadvantage that messages becomes somewhat long.

Fortunately, depending on the specific construction of the second scheme, we can employ a target collision resistant hash function [27, 30] instead of one-time signature to get the fourth scheme. As a result, it keeps messages as short as the second scheme.

Our schemes can be considered challenge-and-response ID schemes. Of course we can construct such ID schemes which are secure against concurrent man-in-the-middle attacks from EUF-CMA signature schemes or IND-CCA2 encryption schemes. To the best of our knowledge, the one obtained from the Cramer-Shoup Encryption Scheme [13, 32, 14] is the fastest in the standard model. In fact, the Cramer-Shoup-based ID scheme is faster than any other ID scheme secure against concurrent man-in-the-middle attacks, including the proof-of-knowledge-based ID schemes. We will see in Section 7 that our fourth scheme is faster than the Cramer-Shoup-based ID scheme.

As a remark, we point out that our schemes are secure against the reset attack (resettable, for short) defined by Bellare et al. [4]. More precisely, our schemes are prover-and-verifier-resettable. This is because that the prover is deterministic and that our schemes consists of 2-round. As is discussed by Yilek [36], resettable security is crucially helpful, for example, for virtual machine service in the cloud computing.

1.2 Related Works

Our first prototype scheme is similar to the scheme of Stinson and Wu [33, 34]. They proved it secure in the random oracle model based on the CDH Assumption and the KEA. Unlike theirs, we provide a security proof in the standard model. Although the assumptions for our first scheme are fairly strong (the Gap-DL Assumption and the KEA), we stress that the first scheme is a steppingstone towards the following schemes.

Concerning man-in-the-middle attacks, Katz [21] and Gennaro [17] employed proofs of knowledge. The Katz Scheme is a non-malleable proof of knowledge but its security model is with timing constraint. The Gennaro Scheme realized a concurrently non-malleable proof of knowledge. It utilizes a multi-trapdoor commitment scheme as a component, and as a result, is not as efficient as the Cramer-Shoup-based ID scheme. Moreover, the security proof is based on strong type assumptions (the SDH Assumption or the Strong RSA Assumption). Recently, Nishimaki-Fujisaki-Tanaka [28] succeeds in constructing a new multi-trapdoor commitment scheme whose security is based on a non-strong type, the RSA Assumption. It can be built-in to a Σ-protocol to get a concurrently non-malleable proof of knowledge based on the same assumption, but it is not as efficient as Gennaro's construction.

Concerning tight reduction to computational hardness assumptions, Arita and Kawashima [2] proposed an ID scheme whose security proof is based on tight reduction to the one more discrete log type assumption [6, 7] and the KEA. Our second, third and fourth schemes succeed in leaving such strong assumptions.

1.3 Organization of the Paper

In the next section, we fix some notations. We briefly review the model of attacks on ID schemes, then we describe computational hardness assumptions which we need. In Section 3, we define non-malleable functions and the notion of a proof of malleability, and we show that exponentiation functions are non-malleable functions. In Section 4, we discuss the first prototype ID scheme. Our proposal ID schemes and their security are presented in Section 5 and 6. In Section 7, we evaluate the efficiency of our schemes comparing with the Cramer-Shoup-based ID scheme. In Section 8, we conclude our work.

2 Preliminaries

The empty string is denoted ϕ. The security parameter is denoted k. On input 1^k, a PPT algorithm $\mathtt{Grp}$ runs and outputs (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. $\mathtt{Grp}$ 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 input a outputs z we denote it as $z \leftarrow A(a)$. When A on input a and B on input b interact and B outputs z we denote it as $z \leftarrow \langle A(a), B(b)\rangle$. When A has oracle-access to O we denote it as A^O. When A does concurrent oracle-access to n oracles $O_1, \dots, O_n$ we denote it as $A^{O_1|\cdots|O_n}$. Here "concurrent" means that A accesses to oracles in arbitrarily interleaved order of messages.

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

2.1 ID Schemes

An *ID scheme* $\mathtt{ID}$ is a triple of PPT algorithms $(\mathtt{K}, \mathtt{P}, \mathtt{V})$. $\mathtt{K}$ is a key generator which outputs a pair of a public key and a matching secret key $(\mathtt{pk}, \mathtt{sk})$ on 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 interaction with $\mathtt{P}(\mathtt{sk})$ is TRUE with probability one. We say that $\mathtt{V}(\mathtt{pk})$ *accepts* if its boolean decision is TRUE.

2.2 Attacks on ID Schemes

The aim of an adversary $\mathcal{A}$ that attacks on 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.

Attacks on ID schemes are divided into two kinds. One is passive and another is active. We will concentrate on active attacks. Active attacks are divided into four patterns according to whether they are sequential or concurrent and whether they are two-phase or man-in-the-middle.

Firstly, a concurrent attack ([4, 7]) means that an adversary $\mathcal{A}(\mathtt{pk})$ interacts with polynomially many clones $\mathtt{P}_i(\mathtt{sk})$s of the prover $\mathtt{P}(\mathtt{sk})$ in arbitrarily interleaved order of messages. Here all prover clones $\mathtt{P}_i(\mathtt{sk})$s are given independent random tapes and independent inner states. A sequential attack is a special case that an adversary $\mathcal{A}(\mathtt{pk})$ interacts with the prover clone $\mathtt{P}(\mathtt{sk})$ arbitrary times, but with only one clone at a time. So concurrent attacks are stronger than sequential attacks.

Secondly, a two-phase attack ([4, 7]) means that an adversary $\mathcal{A}$ consists of two algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. In the first phase, learning phase, $\mathcal{A}_1$ starts with input $\mathtt{pk}$, interacts with prover clones $\mathtt{P}_i(\mathtt{sk})$s and outputs its inner state. In the second phase, impersonation phase, $\mathcal{A}_2$ starts on input the state, interacts with the verifier $\mathtt{V}(\mathtt{pk})$ and tries to make $\mathtt{V}(\mathtt{pk})$ accept. On the other hand, a man-in-the-middle attack means that an adversary $\mathcal{A}$ starts with input $\mathtt{pk}$, interacts with both $\mathtt{P}_i(\mathtt{sk})$s and $\mathtt{V}(\mathtt{pk})$ simultaneously in arbitrarily interleaved order of messages. So man-in-the-middle attacks are stronger than two-phase attacks.

As an experiment, impersonation by a two-phase concurrent adversary $\mathcal{A}$ (2pc adversary, for short) is described as follows.

$\mathbf{Exprmt}^{\text{imp-2pc}}_{\mathtt{ID},\mathcal{A}}(1^k)$

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

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

If decision = 1 then return WIN else return LOSE.

We define *imp-2pc advantage of* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *over* $\mathtt{ID}$ as:

$$\mathbf{Adv}^{\text{imp-2pc}}_{\mathtt{ID},\mathcal{A}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{imp-2pc}}_{\mathtt{ID},\mathcal{A}}(1^k) \text{ returns WIN}].$$

We say that $\mathtt{ID}$ is secure against two-phase concurrent attacks if, for any PPT algorithm $\mathcal{A}$, $\mathbf{Adv}^{\text{imp-2pc}}_{\mathtt{ID},\mathcal{A}}(k)$ is negligible in k.

As an experiment, impersonation by a concurrent man-in-the-middle adversary $\mathcal{A}$ (cmim adversary, for short) is described as follows.

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

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

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 $= 1 \wedge \pi \notin \Pi$ then return WIN

else return LOSE.

Note that man-in-the-middle adversary $\mathcal{A}$ is prohibited from relaying a transcript of a whole interaction with some prover clone. Denote the set of transcripts between $\mathtt{P}_i(\mathtt{sk})$s and $\mathcal{A}(\mathtt{pk})$ as Π and a transcript between $\mathcal{A}(\mathtt{pk})$ and $\mathtt{V}(\mathtt{pk})$ as π, then the constraint is described as $\pi \notin \Pi$. This is the standard and natural constraint to keep the attack meaningful.

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

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

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

2.3 Tag-Based ID Schemes

A tag-based ID scheme $\mathtt{TagID}$ works in the same way as an ordinary scheme $\mathtt{ID}$ except that a string *tag* $\mathtt{t}$ is a priori given to $\mathtt{P}$ and $\mathtt{V}$ by the first round. Note that the interaction depends on the given tag $\mathtt{t}$.

As for attacks on tag-based ID schemes, only the selective-tag attack is considered in this paper. That is, an attack on $\mathtt{TagID}$ by an adversary $\mathcal{A}$ is modeled in the same way as on $\mathtt{ID}$ except that an adversary

$\mathcal{A}$ designates a *target tag* $\mathtt{t}^*$ firstly, and then $\mathcal{A}$ gets a public key $\mathtt{pk}$. Before starting each interaction as a verifier, $\mathcal{A}$ provides a tag $\mathtt{t}_i (\neq \mathtt{t}^*)$ to each clone $\mathtt{P}_i(\mathtt{sk})$.

As an experiment, impersonation by a selective-tag imp-cmim adversary is described as follows.

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

$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{K}(1^k), \mathtt{t}^* \leftarrow \mathcal{A}(1^k)$

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 $= 1 \wedge (\mathtt{t}_i \neq \mathtt{t}^*, \forall i)$ then return Win

else return Lose.

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

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

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

2.4 Computational Hardness Assumptions

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

2.4.1 The Gap-CDH Assumption

A quadruple (g, X_1, X_2, X_3) of elements in G_q is called a Diffie-Hellman (DH) tuple if (g, X_1, X_2, X_3) is written as $(g, g^{x_1}, g^{x_2}, g^{x_1 x_2})$ for some elements x_1 and x_2 in $\mathbf{Z}_q$. A CDH problem instance is a triple $(g, X_1 = g^{x_1}, X_2 = g^{x_2})$, where the exponents x_1 and x_2 are hidden. The CDH oracle $\mathcal{CDH}$ is an oracle which, queried about a CDH problem instance (g, X_1, X_2), answers $X_3 = g^{x_1 x_2}$. A DDH problem instance is a quadruple (g, X_1, X_2, X_3). The DDH oracle $\mathcal{DDH}$ is an oracle which, queried about a DDH problem instance (g, X_1, X_2, X_3), answers a boolean decision whether (g, X_1, X_2, X_3) is a DH-tuple or not. A CDH problem solver is a PPT algorithm which, given a random CDH problem instance (g, X_1, X_2) as input, tries to return $X_3 = g^{x_1 x_2}$. A CDH problem solver $\mathcal{S}$ that is allowed to access $\mathcal{DDH}$ arbitrary times is called a Gap-CDH problem solver. We define the following experiment.

$\mathbf{Exprmt}^{\text{gap-cdh}}_{\mathtt{Grp},\mathcal{S}}(1^k)$

$(q, g) \leftarrow \mathtt{Grp}(1^k), x_1, x_2 \leftarrow \mathbf{Z}_q, X_1 := g^{x_1}, X_2 := g^{x_2}$

$X_3 \leftarrow \mathcal{S}^{\mathcal{DDH}}(g, X_1, X_2)$

If $X_3 = g^{x_1 x_2}$ then return Win else return Lose.

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

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

We say that the Gap-CDH Assumption [29] holds for $\mathtt{Grp}$ if, for any PPT algorithm $\mathcal{S}$, $\mathbf{Adv}^{\text{gap-cdh}}_{\mathtt{Grp},\mathcal{S}}(k)$ is negligible in k.

2.4.2 The Gap-DL Assumption

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

$\textbf{Exprmt}^{\text{gap-dl}}_{\texttt{Grp},\mathcal{S}}(1^k)$

$(q, g) \leftarrow \texttt{Grp}(1^k), x \leftarrow \mathbf{Z}_q, X := g^x$

$x^* \leftarrow \mathcal{S}^{\mathcal{CDH}}(g, X)$

If $g^{x^*} = X$ then return $\textsc{Win}$ else return $\textsc{Lose}$.

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

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

We say that the Gap-DL Assumption holds for $\texttt{Grp}$ if, for any PPT algorithm $\mathcal{S}$, $\textbf{Adv}^{\text{gap-dl}}_{\texttt{Grp},\mathcal{S}}(k)$ 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 [26].

2.4.3 The Knowledge-of-Exponent Assumption

Informally, the Knowledge-of-Exponent Assumption (KEA) [16, 8] says that, given a randomly chosen $h \in G_q$ as input, a PPT algorithm $\mathcal{H}$ can extend (g, h) to a DH-tuple (g, h, X, D) *only when $\mathcal{H}$ knows the exponent x of $X = g^x$*. The formal definition is as follows.

Let W be any distribution taking some input. Let $\mathcal{H}$ and $\mathcal{H}'$ be any PPT algorithms taking input of the form (g, h, w). Here g is any fixed generator and h is a randomly chosen element in G_q. w is a string in $\{0, 1\}^*$ output by W called auxiliary input [10, 15]. We define the following experiment.

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

$(q, g) \leftarrow \texttt{Grp}(1^k), w \leftarrow W, a \leftarrow \mathbf{Z}_q, h := g^a$

$(X, D) \leftarrow \mathcal{H}(g, h, w), x' \leftarrow \mathcal{H}'(g, h, w)$

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

else return $\textsc{Lose}$.

Note that w is independent of h in the experiment. This independence is crucial ([10, 15]).

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

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

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

3 Non-Malleable Functions and ID Schemes

In this section, we define non-malleable functions. We show that an exponentiation function with values in G_q is a non-malleable function. Then we present a notion of proofs of malleability.

3.1 Non-Malleable Functions

Let f be a function from $\{0,1\}^k$ to $\{0,1\}^{l(k)}$, where $l(k)$ is a polynomially bounded function in k. We call a function $\mathcal{R} : (\{0,1\}^k)^n \to \{0,1\}^k$ a relation, and $x := \mathcal{R}(x_1, \ldots, x_n)$ the related element to $x_1, \ldots, x_n$. We say that *y is the related value of f to $(y_1, \ldots, y_n)$ with respect to (w.r.t., for short) $\mathcal{R}$* if the the following condition holds:

$$
\begin{aligned}
&\exists x, x_1, \ldots, x_n \in \{0,1\}^k, \\
&\quad y = f(x), y_1 = f(x_1), \ldots, y_n = f(x_n) \\
&\qquad \wedge\ x = \mathcal{R}(x_1, \ldots, x_n).
\end{aligned}
$$

Let $NMF(1^k)$ be a family of one-way functions. For any given PPT algorithm $\mathcal{E}$, we define the following experiment.

$$
\begin{aligned}
&\mathbf{Exprmt}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(1^k) \\
&\quad f \leftarrow NMF(1^k), x_1, \ldots, x_n \leftarrow \{0,1\}^k \\
&\quad y_1 := f(x_1), \ldots, y_n := f(x_n) \\
&\quad y \leftarrow \mathcal{E}(y_1, \ldots, y_n) \\
&\quad \text{If } y \text{ is the related value of } f \text{ to } (y_1, \ldots, y_n) \text{ w.r.t. } \mathcal{R} \\
&\qquad \text{then return } \textsc{Win} \text{ else return } \textsc{Lose}.
\end{aligned}
$$

We define *advantage of $\mathcal{E}$ over NMF in the game of non-malleability with respect to $\mathcal{R}$ ("nm-$\mathcal{R}$")* as:

$$
\mathbf{Adv}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(1^k) \text{ returns } \textsc{Win}].
$$

Definition 1 (Non-Malleable Functions) A one-way function family $NMF(1^k)$ is called *a non-malleable function family with respect to a relation $\mathcal{R}$* if, for any PPT algorithm $\mathcal{E}$, $\mathbf{Adv}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(k)$ is negligible in k. ($f \in NMF(1^k)$ is called *a non-malleable function with respect to a relation $\mathcal{R}$*.)

Next, we define non-malleable functions which is robust despite the presence of decision oracle $\mathcal{D}_{f,\mathcal{R}}$ below.

$$
\begin{aligned}
&\mathcal{D}_{f,\mathcal{R}}(y : y_1, \ldots, y_n) : \\
&\quad \text{If } y \text{ is the related value of } f \text{ to } (y_1, \ldots, y_n) \text{ w.r.t. } \mathcal{R} \\
&\qquad \text{then reply "}\textsc{True}\text{" else reply "}\textsc{False}\text{".}
\end{aligned}
$$

We define the following experiment.

$$
\begin{aligned}
&\mathbf{Exprmt}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(1^k) \\
&\quad f \leftarrow NMF(1^k), x_1, \ldots, x_n \leftarrow \{0,1\}^k \\
&\quad y_1 := f(x_1), \ldots, y_n := f(x_n) \\
&\quad y \leftarrow \mathcal{E}^{\mathcal{D}_{f,\mathcal{R}}}(y_1, \ldots, y_n) \\
&\quad \text{If } y \text{ is the related value of } f \text{ to } (y_1, \ldots, y_n) \text{ w.r.t. } \mathcal{R} \\
&\qquad \text{then return } \textsc{Win} \text{ else return } \textsc{Lose}.
\end{aligned}
$$

We define *advantage of $\mathcal{E}$ over NMF in the game of non-malleability with respect to $\mathcal{R}$ with decision oracle ("nm-$\mathcal{R}$-do")* as:

$$
\mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k) \stackrel{\text{def}}{=} \Pr[\mathbf{Exprmt}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(1^k) \text{ returns } \textsc{Win}].
$$

Definition 2 (Non-Malleable Functions withstanding Decision Oracle) A one-way function family $NMF(1^k)$ is called *a non-malleable function family with respect to a relation* $\mathcal{R}$ *withstanding the decision oracle* $\mathcal{D}_{f,\mathcal{R}}$ if, for any PPT algorithm $\mathcal{E}$, $\mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k)$ is negligible in k. ($f \in NMF(1^k)$ is called *a non-malleable function with respect to a relation* $\mathcal{R}$ *withstanding the decision oracle* $\mathcal{D}_{f,\mathcal{R}}$.)

It is easy to see that an exponentiation function is a non-malleable function based on the (Gap-)CDH Assumption.

Proposition 1 *Let* $\mathcal{R}$ *be the multiplication relation and let* f_λ *be an exponentiation function for* $\lambda = (q, g)$*;*

$$\mathcal{R} : \mathbf{Z}_q^2 \to \mathbf{Z}_q,\ (x_1, x_2) \mapsto x_1 x_2,$$
$$f_\lambda : \mathbf{Z}_q \to G_q,\ x \mapsto g^x,$$
$$\lambda \in \Lambda(1^k) = \{(q, g)\ ;\ (q, g) \leftarrow \mathtt{Grp}(1^k)\}.$$

If the Gap-CDH Assumption holds for $\mathtt{Grp}$*, then* $\{f_\lambda\}_{\lambda \in \Lambda(1^k)}$ *is a non-malleable function family with respect to the relation* $\mathcal{R}$ *withstanding the decision oracle* $\mathcal{D}_{f_\lambda,\mathcal{R}}$*.*

A proof is given in Appendix A.

3.2 ID Schemes of Proofs of Malleability

Our scenario is to build up a security proof by constructing a malleability extractor against a non-malleable function using any given adversary on the ID scheme.

Definition 3 (ID Schemes of Proofs of Malleability and Malleability Extractors) *An ID scheme of a proof of malleability* is 5-tuple $(\mathtt{K}, \mathtt{P}, \mathtt{V}, f, \mathcal{R})$, where $(\mathtt{K}, \mathtt{P}, \mathtt{V})$ is an ID scheme $\mathtt{ID}$ and f is a non-malleable function with respect to a relation $\mathcal{R}$ satisfying the following soundness condition. For any given PPT adversary $\mathcal{A}$ that attacks on $\mathtt{ID}$ in a game-$\mathcal{G}$, there exists a PPT algorithm $\mathcal{E}$ such that $\mathcal{E}$ wins in the experiment $\mathbf{Exprmt}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(1^k)$ with the advantage $\mathbf{Adv}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(k)$ satisfying the following inequality;

$$\mathbf{Adv}^{\text{nm-}\mathcal{R}}_{NMF,\mathcal{E}}(k) \geqslant \mathbf{Adv}^{\text{game-}\mathcal{G}}_{\mathtt{ID},\mathcal{A}}(k) - \varepsilon(k),$$

where $\varepsilon(k)$ is a negligible function in k. $\mathcal{E}$ is called *a malleability extractor against the non-malleability of* f. (When f is a non-malleable function withstanding the decision oracle $\mathcal{D}_{f,\mathcal{R}}$ and $\mathcal{E}$ accesses $\mathcal{D}_{f,\mathcal{R}}$, the game "nm-$\mathcal{R}$" is replaced with "nm-$\mathcal{R}$-do".)

Remark. In Definition 3, we require that $\mathcal{E}$ must not be expected polynomial time but *strictly* probabilistic polynomial time.

In Section 5 and 6, we pick up the exponentiation function f_λ and the multiplication relation $\mathcal{R}$, and apply the scenario to our ID schemes.

4 A Prototype ID Scheme Secure against Two-phase Concurrent Attacks

In this section, we construct and discuss a prototype ID scheme $\mathtt{IDproto}$. In the $\mathtt{IDproto}$, the verifier $\mathtt{V}$ checks whether or not the prover $\mathtt{P}$ has ability to complete Diffie-Hellman tuples.

4.1 A Prototype Scheme and Its Security

A prototype ID scheme $\mathtt{IDproto}$ consists of a triple $(\mathtt{K}, \mathtt{P}, \mathtt{V})$. The construction is as shown in the Fig.1. On input 1^k, a key generator $\mathtt{K}$ runs as follows. A group generator $\mathtt{Grp}$ outputs $\lambda = (q, g)$ on input 1^k (λ specifies an exponentiation function $f_\lambda(x) = g^x$). Then $\mathtt{K}$ chooses $x \in \mathbf{Z}_q$, computes $X = f_\lambda(x)$ and sets $\mathtt{pk} = (\lambda, X)$ and $\mathtt{sk} = (\lambda, x)$. Then $\mathtt{K}$ returns $(\mathtt{pk}, \mathtt{sk})$.

$\mathtt{P}$ and $\mathtt{V}$ interact as follows.

In the first round, $\mathtt{V}$ is given $\mathtt{pk}$ as input, chooses $a \in \mathbf{Z}_q$ at random and computes $h = g^a$. Then $\mathtt{V}$ sends h to $\mathtt{P}$.

In the second round, $\mathtt{P}$ is given $\mathtt{sk}$ as input and receives h as input message, computes $D = h^x$. Then $\mathtt{P}$ sends D to $\mathtt{V}$.

Finally, receiving D as input message, $\mathtt{V}$ verifies whether (g, h, X, D) is a DH-tuple. For this sake, $\mathtt{V}$ checks whether $D = X^a$ holds. If so, $\mathtt{V}$ returns 1 and otherwise 0.

Key Generation

– $\mathtt{K}$: given 1^k as input;

- $\lambda := (q, g) \leftarrow \mathtt{Grp}(1^k), x \leftarrow \mathbf{Z}_q, X := f_\lambda(x)$
- $\mathtt{pk} := (\lambda, X), \mathtt{sk} := (\lambda, x)$, return $(\mathtt{pk}, \mathtt{sk})$

Interaction

– $\mathtt{V}$: given $\mathtt{pk}$ as input;

- $a \leftarrow \mathbf{Z}_q, h := g^a$, send h to $\mathtt{P}$

– $\mathtt{P}$: given $\mathtt{sk}$ as input and receiving h as input message;

- $D := h^x$, send D to $\mathtt{V}$

– $\mathtt{V}$: receiving D as input message;

- If $D = X^a$ then return 1 else return 0

Figure 1: A Prototype ID Scheme $\mathtt{IDproto}$.

Theorem 1 *The ID scheme $\mathtt{IDproto}$ is secure against two-phase concurrent attacks based on the Gap-DL Assumption and the KEA for $\mathtt{Grp}$. More precisely, for any PPT two-phase concurrent adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT Gap-DL problem solver $\mathcal{S}$ and a PPT algorithm $\mathcal{H}$ for the KEA which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{imp-2pc}}_{\mathtt{IDproto},\mathcal{A}}(k) \leqslant \mathbf{Adv}^{\text{gap-dl}}_{\mathtt{Grp},\mathcal{S}}(k) + \mathbf{Adv}^{\text{kea}}_{\mathtt{Grp},\mathcal{H},\mathcal{H}'}(k).$$

4.2 Proof of Theorem 1

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

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

In the first phase $\mathcal{S}$ replies to $\mathcal{A}_1$'s queries as follows. In case that $\mathcal{A}_1$ sends h_i to the i-th prover clone $\mathtt{P}_i(\mathtt{sk})$, $\mathcal{S}$ queries its CDH oracle $\mathcal{CDH}$ for the answer of a CDH problem instance (g, X, h_i) and gets D_i. Then $\mathcal{S}$ sends D_i to $\mathcal{A}$. In case that $\mathcal{A}_1$ outputs its inner state st, $\mathcal{S}$ stops $\mathcal{A}_1$ and invokes $\mathcal{A}_2$ on st.

In the second phase $\mathcal{S}$ replies to $\mathcal{A}_2$'s query as follows. In case that $\mathcal{A}_2$ queries $\mathtt{V}(\mathtt{pk})$ for the first message by an empty string ϕ, $\mathcal{S}$ chooses $a^* \in \mathbf{Z}_q$ at random and computes $h^* = g^{a^*}$. Then $\mathcal{S}$ sends h^* to $\mathcal{A}_2$. In case that $\mathcal{A}_2$ sends D^* to $\mathtt{V}(\mathtt{pk})$, $\mathcal{S}$ invokes the KEA extractor $\mathcal{H}'$ on (g, h^*, st). Here $\mathcal{H}'$ is the one associated with the $\mathcal{H}$ below, which is essentially $\mathcal{A}_2$ itself.

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

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

When $\mathcal{H}'$ outputs x' $\mathcal{S}$ checks whether x' is actually the exponent for X. If so, $\mathcal{S}$ outputs $x^* = x'$ and otherwise a random element $x^* \in \mathbf{Z}_q$.

It is obvious that $\mathcal{S}$ simulates both concurrent $\mathtt{P}_i(\mathtt{sk})$s and $\mathtt{V}(\mathtt{pk})$ 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 discrete log of 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;

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

$\mathcal{A}$ wins if and only if $D^* = X^{a^*}$ holds. Therefore;

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

That means what we want.

$$\mathbf{Adv}^{\text{gap-dl}}_{\mathtt{Grp},\mathcal{S}}(k) \geqslant \mathbf{Adv}^{\text{imp-2pc}}_{\mathtt{IDproto},\mathcal{A}}(k) - \mathbf{Adv}^{\text{kea}}_{\mathtt{Grp},\mathcal{H},\mathcal{H}'}(k).$$

(*Q.E.D.*)

Given $\lambda = (q, g), X = g^x$ as input; **Initial Setting** – Initialize the inner state – $\mathtt{pk} := (\lambda, X)$, invoke $\mathcal{A}_1$ on $\mathtt{pk}$ **The First phase : Answering $\mathcal{A}_1$'s Queries** – In case that $\mathcal{A}_1$ sends h_i to $\mathtt{P}_i(\mathtt{sk})$; • $D_i \leftarrow \mathcal{CDH}(g, X, h_i)$, send D_i to $\mathcal{A}_1$ – In case that $\mathcal{A}_1$ outputs the inner state st; • Stop $\mathcal{A}_1$, invoke $\mathcal{A}_2$ on st **The Second phase : Answering $\mathcal{A}_2$'s Query** – In case that $\mathcal{A}_2$ queries $\mathtt{V}(\mathtt{pk})$ for the first message; • $a^* \leftarrow \mathbf{Z}_q, h^* := g^{a^*}$, send h^* to $\mathcal{A}_2$ – In case that $\mathcal{A}_2$ sends D^* to $\mathtt{V}(\mathtt{pk})$; • Invoke $\mathcal{H}'$ on (g, h^*, st) and get x' from $\mathcal{H}'$ If $g^{x'} = X$ then return $x^* := x'$ else return a random element $x^* \in \mathbf{Z}_q$

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

4.3 Discussion

Although the Gap-DL Assumption and the KEA are fairly strong assumptions, the fact that $\mathtt{IDproto}$ is proven secure against two-phase concurrent attacks is rather surprising, because it is obvious that

$\mathtt{IDproto}$ is insecure under man-in-the-middle attacks. To see it just recall the typical man-in-the-middle attack on the El Gamal Encryption Scheme.

Analogous phenomenon also occurs, for example, for the Schnorr ID scheme [7]. So it seems that the security against two-phase concurrent attacks is rather artificial and dose not match with real situations.

5 A Tag-Based ID Scheme Secure against Concurrent Man-in-the-Middle Attacks

In this section, we modify the prototype scheme $\mathtt{IDproto}$ to make it a proof of malleability by applying a tag framework. Actually, by using the tag framework, we construct a malleability extractor against the non-malleability of the exponentiation function in the security proof. The tag framework also works to simulate concurrent prover clones, where we owe the idea to the tag-based encryption scheme of Kiltz [23].

First of all, we note that we utilize hereafter an exponentiation function family $NMF(1^k) = \{f_\lambda\}_{\lambda \in \Lambda(1^k)}$ and the multiplication relation $\mathcal{R}(x_1, x_2) = x_1 x_2$, where $\Lambda(1^k)$ is the set $\{(q,g)\ ;\ (q,g) \leftarrow \mathtt{Grp}(1^k)\}$, f_λ is an exponentiation function $f_\lambda(x) = g^x$ with values in G_q.

5.1 A Tag-Based Scheme and Its Security

A tag-based ID scheme $\mathtt{tID}$ consists of a triple $(\mathtt{K}, \mathtt{P}, \mathtt{V})$. The construction is as shown in the Fig.3.

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

A string tag $\mathtt{t}$ is a priori given to $\mathtt{P}$ and $\mathtt{V}$ by the first round. In our construction, we set the tag $\mathtt{t}$ in $\mathbf{Z}_q$.

$\mathtt{P}$ and $\mathtt{V}$ interact as follows.

In the first round, $\mathtt{V}$ is given $\mathtt{pk}$ as input. $\mathtt{V}$ chooses $a \in \mathbf{Z}_q$ at random and computes $h = g^a$ and $d = (X^{\mathtt{t}} Y)^a$. Then $\mathtt{V}$ sends (h, d) to $\mathtt{P}$.

In the second round, $\mathtt{P}$ is given $\mathtt{sk}$ as input and receives (h, d) as input message. $\mathtt{P}$ verifies whether d is the related value of f_λ to $(X^{\mathtt{t}} Y, h)$ w.r.t. $\mathcal{R}$. For this sake, $\mathtt{P}$ checks whether $h^{\mathtt{t}x+y} = d$ holds. If it does not hold, then $\mathtt{P}$ puts $D = \perp$. Otherwise $\mathtt{P}$ computes $D = h^x$. Then $\mathtt{P}$ sends D to $\mathtt{V}$.

Finally, receiving D as input message, $\mathtt{V}$ verifies whether D is the related value of f_λ to (X, h) w.r.t. $\mathcal{R}$. For this sake, $\mathtt{V}$ checks whether $D = X^a$ holds. If so, $\mathtt{V}$ returns 1 and otherwise 0.

Theorem 2 *The tag-based ID scheme $\mathtt{tID}$ is secure against selective-tag concurrent man-in-the-middle attacks based on the non-malleability of an exponentiation function family $NMF(1^k) = \{f_\lambda\}_{\lambda \in \Lambda(1^k)}$. More precisely, for any PPT selective-tag concurrent man-in-the-middle adversary $\mathcal{A}$, there exists a PPT malleability extractor $\mathcal{E}$ against the non-malleability of f_λ which satisfies the following tight reduction.*

$$\mathbf{Adv}^{\text{stag-imp-cmim}}_{\mathtt{tID},\mathcal{A}}(k) \leqslant \mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k).$$

Corollary *The tag-based ID scheme $\mathtt{tID}$ is secure against selective-tag concurrent man-in-the-middle attacks based on the Gap-CDH Assumption.*

Proof. By Proposition 1 and Theorem 2. (*Q.E.D.*)

Key Generation
– K: given 1^k as input;
 • $\lambda := (q, g) \leftarrow \mathtt{Grp}(1^k), x, y \leftarrow \mathbf{Z}_q, X := f_\lambda(x), Y := f_\lambda(y)$
 • $\mathtt{pk} := (\lambda, X, Y), \mathtt{sk} := (\lambda, x, y)$, return $(\mathtt{pk}, \mathtt{sk})$
Tag-Receiving
– P and V receive a tag $\mathtt{t} \in \mathbf{Z}_q$ by the first round
Interaction
– V: given pk as input;
 • $a \leftarrow \mathbf{Z}_q, h := g^a, d := (X^{\mathtt{t}} Y)^a$, send (h, d) to P
– P: given sk as input and receiving (h, d) as input message;
 • If $h^{\mathtt{t}x+y} \neq d$ then $D := \perp$ else $D := h^x$, send D to V
– V: receiving D as input message;
 • If $D = X^a$ then return 1 else return 0

Figure 3: A Tag-Based ID Scheme tID.

5.2 Proof of Theorem 2

Let $\mathcal{A}$ be as in Theorem 2. Using $\mathcal{A}$ as subroutine, we construct a malleability extractor $\mathcal{E}$ against the non-malleability of f_λ. The construction is illustrated in Fig.4.

$\mathcal{E}$ is given $\lambda = (q, g)$ and function values $X_1 = f_\lambda(x_1), X_2 = f_\lambda(x_2)$ as input, where x_1 and x_2 are random and hidden. $\mathcal{E}$ initializes its inner state. $\mathcal{E}$ invokes $\mathcal{A}$ on input 1^k and gets a target tag $\mathtt{t}^*$ from $\mathcal{A}$. $\mathcal{E}$ chooses $r \in \mathbf{Z}_q$ at random and computes $Y = X_1^{-\mathtt{t}^*} g^r$. $\mathcal{E}$ sets $\mathtt{pk} = (\lambda, X, Y)$ and inputs pk into $\mathcal{A}$. Note that pk is correctly distributed. Note also that $\mathcal{E}$ knows neither x_1 nor y, where y is the discrete log of Y;

$$y = \log_g(Y) = -\mathtt{t}^* x_1 + r.$$

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

In case that $\mathcal{A}$ queries V(pk) for the first message by ϕ, $\mathcal{E}$ chooses $a^* \in \mathbf{Z}_q$ at random and computes $h^* = X_2 g^{a^*}$ and $d^* = (h^*)^r$. Then $\mathcal{E}$ sends (h^*, d^*) to $\mathcal{A}$ (Call this case $\mathscr{V}$).

In case that $\mathcal{A}$ gives a tag $\mathtt{t}_i$ and sends (h_i, d_i) to the i-th prover clone $\mathtt{P}_i(\mathtt{sk})$, $\mathcal{E}$ verifies whether d_i is the related value of f_λ to $(X_1^{\mathtt{t}_i} Y, h_i)$ w.r.t. $\mathcal{R}$. For this sake, $\mathcal{E}$ queries its decision oracle $\mathcal{D}_{f_\lambda, \mathcal{R}}$. If the answer is "False", then $\mathcal{E}$ puts $D_i = \perp$. Otherwise $\mathcal{E}$ computes $D_i = (d_i / h_i^r)^{1/(\mathtt{t}_i - \mathtt{t}^*)}$ (Call this case $\mathscr{P}$). Then $\mathcal{E}$ sends D_i to $\mathcal{A}$. Note that, in the selective-tag model, $\mathcal{A}$ is prohibited from using $\mathtt{t}^*$ as $\mathtt{t}_i$ (that is, $\mathtt{t}^* \neq \mathtt{t}_i$ for any i).

In case that $\mathcal{A}$ sends D^* to V(pk), $\mathcal{E}$ verifies whether D^* is the related value of f_λ to (X_1, h^*) w.r.t. $\mathcal{R}$. For this sake, $\mathcal{E}$ queries $\mathcal{D}_{f_\lambda, \mathcal{R}}$. If the answer is "True", then $\mathcal{E}$ returns $X_3 = D^* / X_1^{a^*}$. Otherwise, $\mathcal{E}$ returns a random element $X_3 \in G_q$.

The view of $\mathcal{A}$ in $\mathcal{E}$ is the same as the real view, as we see below.

In the case $\mathscr{V}$, $\mathcal{E}$ simulates V(pk) perfectly. This is because the distribution of (h^*, d^*) is equal to that of the real (h, d). To see it, note that $x_2 + a^*$ is substituted for a;

$$h^* = g^{x_2 + a^*}, \; d^* = (g^{x_2 + a^*})^r = (g^r)^{x_2 + a^*} = (X_1^{\mathtt{t}^*} Y)^{x_2 + a^*}.$$

In the case $\mathscr{P}$, $\mathcal{E}$ simulates concurrent $\mathtt{P}_i(\mathtt{sk})$s perfectly. This is because $D_i = (d_i / h_i^r)^{1/(\mathtt{t}_i - \mathtt{t}^*)}$ is equal to $h_i^{x_1}$ by the following equalities.

$$d_i / h_i^r = h_i^{\mathtt{t}_i x_1 + y - r} = h_i^{(\mathtt{t}_i - \mathtt{t}^*) x_1 + (\mathtt{t}^* x_1 + y - r)} = h_i^{(\mathtt{t}_i - \mathtt{t}^*) x_1}.$$

Now we evaluate the advantage of $\mathcal{E}$. When $\mathcal{A}$ wins, D^* is the related value of f_λ to (X_1, h^*) w.r.t. $\mathcal{R}$, so the followings hold.

$$D^* = f_\lambda(\mathcal{R}(x_1, x_2 + a^*)) = g^{x_1(x_2 + a^*)} = g^{x_1 x_2 + x_1 a^*}.$$

Hence the output X_3 is equal to $D^*/X_1^{a^*} = g^{x_1 x_2} = f_\lambda(\mathcal{R}(x_1, x_2))$. That is, X_3 is the related value of f_λ to (X_1, X_2) w.r.t. $\mathcal{R}$. This means that $\mathcal{E}$ wins. Therefore the probability that $\mathcal{E}$ wins is lower bounded by the probability that $\mathcal{A}$ wins;

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

Hence we get what we want;

$$\mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k) \geqslant \mathbf{Adv}^{\text{stag-imp-cmim}}_{\mathtt{tID},\mathcal{A}}(k). \quad (Q.E.D.)$$

Given $\lambda = (q, g), X_1 = f_\lambda(x_1), X_2 = f_\lambda(x_2)$ as input;
Initial Setting
– Initialize the inner state
– invoke $\mathcal{A}$ on input 1^k, get a target tag $\mathtt{t}^*$ from $\mathcal{A}$
– $r \leftarrow \mathbf{Z}_q, Y := X_1^{-\mathtt{t}^*} g^r$, $\mathtt{pk} := (\lambda, X_1, Y)$, input $\mathtt{pk}$ into $\mathcal{A}$
Answering $\mathcal{A}$'s Queries
– In case that $\mathcal{A}$ queries $\mathtt{V}(\mathtt{pk})$ for the first message (the case $\mathscr{V}$);
- $a^* \leftarrow \mathbf{Z}_q, h^* := X_2 g^{a^*}, d^* := (h^*)^r$, send (h^*, d^*) to $\mathcal{A}$

– In case that $\mathcal{A}$ gives $\mathtt{t}_i$ and sends (h_i, d_i) to $\mathtt{P}_i(\mathtt{sk})$;
- If $\mathcal{D}_{f_\lambda,\mathcal{R}}(d_i : X_1^{\mathtt{t}_i} Y, h_i) \neq 1$ then $D_i := \perp$
- else $D_i := (d_i/h_i^r)^{1/(\mathtt{t}_i - \mathtt{t}^*)}$ (the case $\mathscr{P}$)
- Send D_i to $\mathcal{A}$

– In case that $\mathcal{A}$ sends D^* to $\mathtt{V}(\mathtt{pk})$;
- If $\mathcal{D}_{f_\lambda,\mathcal{R}}(D^* : X_1, h^*) = 1$ then return $X_3 := D^*/X_1^{a^*}$
- else return a random element $X_3 \in G_q$

Figure 4: A Malleability Extractor $\mathcal{E}$ for the Proof of Theorem 2.

5.3 Discussion

By virtue of the tag framework with algebraic trick [23], we were able to construct the malleability extractor $\mathcal{E}$. In fact, $\mathcal{E}$ constructs a public key $\mathtt{pk}$ using a function value $X_1 = f_\lambda(x_1)$, and $\mathcal{E}$ simulates concurrent prover clones ($\mathtt{P}_i(\mathtt{sk})$s) perfectly by the algebraic trick. Moreover, simulating the verifier ($\mathtt{V}(\mathtt{pk})$) perfectly, $\mathcal{E}$ embeds another value $X_2 = f_\lambda(x_2)$ in a challenge message by the algebraic trick. Once the malleability extractor $\mathcal{E}$ gets a valid response from the adversary $\mathcal{A}$, $\mathcal{E}$ succeeds in forging the related value $X_3 = f_\lambda(\mathcal{R}(x_1, x_2))$.

6 ID Schemes Secure against Concurrent Man-in-the-Middle Attacks

In this section, to exit the tag framework, we apply two methods. The generic method is the CHK transformation [12]. Another method is employing a target collision resistant hash function [27, 30] depending on the specific structure of the tag-based scheme $\mathtt{tID}$.

6.1 A Scheme with a One-Time Signature and Its Security

Firstly, we describe an ID scheme with a one-time signature $\mathtt{ID1}$. Along the technique of CHK transformation, we replace the tag $\mathtt{t}$ by a one-time verification key $\mathtt{vk}$ of a strong one-time signature.

Since the CHK transformation is an well known technique, we only denote the feature of ID1 giving the construction in Fig.5, security statement in Theorem 3, and the construction of a malleability extractor $\mathcal{E}$ in Fig.6. The definition of a strong one-time signature OTS and advantage $\mathbf{Adv}^{\text{euf-cma}}_{\text{OTS},\mathcal{F}}(k)$ of a PPT forger $\mathcal{F}$ over OTS are in Appendix B.

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

Interaction
– V: given pk as input;
- $(\mathtt{vk}, \mathtt{sgk}) \leftarrow \mathtt{SGK}(1^k)$, $a \leftarrow \mathbf{Z}_q$
- $h := g^a, d := (X^{\mathtt{vk}}Y)^a, \sigma \leftarrow \mathtt{Sign}_{\mathtt{sgk}}((h, d))$
- Send $\mathtt{vk}, (h, d), \sigma$ to P

– P: given sk as input and receiving $\mathtt{vk}, (h, d), \sigma$ as input message;
- If $\mathtt{Vrfy}_{\mathtt{vk}}((h, d), \sigma) \neq 1$ or $h^{(\mathtt{vk})x+y} \neq d$ then $D := \perp$ else $D := h^x$
- Send D to V

– V: receiving D as input message;
- If $D = X^a$ then return 1 else return 0

Figure 5: An ID Scheme ID1.

Theorem 3 *The ID scheme ID1 is secure against concurrent man-in-the-middle attacks based on the non-malleability of an exponentiation function family* $NMF(1^k) = \{f_\lambda\}_{\lambda \in \Lambda(1^k)}$ *and the one-time security in the strong sense of a one-time signature OTS. More precisely, for any PPT concurrent man-in-the-middle adversary* $\mathcal{A}$*, there exist a PPT malleability extractor* $\mathcal{E}$ *against the non-malleability of* f_λ *and a PPT forger* $\mathcal{F}$ *on OTS which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{imp-cmim}}_{\text{ID1},\mathcal{A}}(k) \leqslant \mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k) + \mathbf{Adv}^{\text{euf-cma}}_{\text{OTS},\mathcal{F}}(k).$$

Corollary *The ID scheme ID1 is secure against concurrent man-in-the-middle attacks based on the Gap-CDH Assumption and the one-time security in the strong sense of an employed one-time signature.*

Proof. By Proposition 1 and Theorem 3. (*Q.E.D.*)

6.2 A Scheme with a Target Collision Resistance Hash Function and Its Security

Secondly, we describe an ID scheme with a TCR hash function ID2. We replace the tag t by a TCR hash function value τ at $h = g^a$. We need target collision resistance to apply the algebraic trick to all but a negligible case. The definition of a TCR hash function family $Hfam(1^k) = \{H_\mu\}_{\mu \in Hkey(1^k)}$ and advantage $\mathbf{Adv}^{\text{tcr}}_{Hfam,\mathcal{CF}}(k)$ of a PPT collision finder $\mathcal{CF}$ over $Hfam$ are in Appendix C.

An ID scheme with a TCR hash function ID2 consists of a triple (K, P, V). The construction is as shown in the Fig.7.

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

P and V interact as follows.

In the first round, V is given pk as input. V chooses $a \in \mathbf{Z}_q$ at random and computes $h = g^a$. Then V computes the hash value $\tau \leftarrow H_\mu(h)$ and computes $d = (X^\tau Y)^a$. V sends (h, d) to P.

Given $\lambda = (q, g), X_1 = f_\lambda(x_1), X_2 = f_\lambda(x_2)$ as input;
Initial Setting
– Initialize inner state
– $(\mathtt{vk}^*, \mathtt{sgk}^*) \leftarrow \mathtt{SGK}(1^k)$
– $r \leftarrow \mathbf{Z}_q, Y := X_1^{-\mathtt{vk}^*} g^r$, $\mathtt{pk} := (\lambda, X_1, Y)$, invoke $\mathcal{A}$ on $\mathtt{pk}$
Answering $\mathcal{A}$'s Queries
– In case that $\mathcal{A}$ queries $\mathtt{V}(\mathtt{pk})$ for the first message (the case $\mathscr{V}$);
- $a^* \leftarrow \mathbf{Z}_q, h^* := X_2 g^{a^*}, d^* := (h^*)^r, \sigma^* \leftarrow \mathtt{Sign}_{\mathtt{sgk}^*}((h^*, d^*))$
- Send $\mathtt{vk}^*, (h^*, d^*), \sigma^*$ to $\mathcal{A}$

– In case that $\mathcal{A}$ sends $\mathtt{vk}_i, (h_i, d_i), \sigma_i$ to $\mathtt{P}_i(\mathtt{sk})$;
- If $\mathtt{Vrfy}_{\mathtt{vk}_i}((h_i, d_i), \sigma_i) \neq 1$ or $\mathcal{D}_{f_\lambda,\mathcal{R}}(d_i : X_1^{\mathtt{vk}_i} Y, h_i) \neq 1$
  then $D_i :=\perp$
- else
  If $\mathtt{vk}_i \neq \mathtt{vk}^*$ then $D_i := (d_i/h_i^r)^{1/(\mathtt{vk}_i - \mathtt{vk}^*)}$ (the case $\mathscr{P}$)
  else abort (the case Abort)
- Send D_i to $\mathcal{A}$

– In case that $\mathcal{A}$ sends D^* to $\mathtt{V}(\mathtt{pk})$;
- If $\mathcal{D}_{f_\lambda,\mathcal{R}}(D^* : X_1, h^*) = 1$ then return $X_3 := D^*/X_1^{a^*}$
- else return a random element $X_3 \in G_q$

Figure 6: A Malleability Extractor $\mathcal{E}$ for the Proof of Theorem 3.

In the second round, $\mathtt{P}$ is given $\mathtt{sk}$ as input and receives (h, d) as input message. $\mathtt{P}$ computes the hash value $\tau \leftarrow H_\mu(h)$. Then $\mathtt{P}$ verifies whether d is the related value of f_λ to $(X^\tau Y, h)$ w.r.t. $\mathcal{R}$. For this sake, $\mathtt{P}$ checks whether $h^{\tau x + y} = d$ holds. If it does not hold, then $\mathtt{P}$ puts $D =\perp$. Otherwise $\mathtt{P}$ computes $D = h^x$. $\mathtt{P}$ sends D to $\mathtt{V}$.

Finally, receiving D as input message, $\mathtt{V}$ verifies whether D is the related value of f_λ to (X, h) w.r.t. $\mathcal{R}$. For this sake, $\mathtt{V}$ checks whether $D = X^a$ holds. If so, $\mathtt{V}$ returns 1 and otherwise 0.

Theorem 4 *The ID scheme $\mathtt{ID2}$ is secure against concurrent man-in-the-middle attacks based on the non-malleability of an exponentiation function family $NMF(1^k) = \{f_\lambda\}_{\lambda \in \Lambda(1^k)}$ and the target collision resistance of a hash function family $Hfam(1^k) = \{H_\mu\}_{\mu \in Hkey(1^k)}$. More precisely, for any PPT concurrent man-in-the-middle adversary $\mathcal{A}$, there exist a PPT malleability extractor $\mathcal{E}$ against the non-malleability of f_λ and a PPT collision-finder $\mathcal{CF}$ on Hfam which satisfy the following tight reduction.*

$$\mathbf{Adv}^{\text{imp-cmim}}_{\mathtt{ID2},\mathcal{A}}(k) \leqslant \mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k) + \mathbf{Adv}^{\text{tcr}}_{Hfam,\mathcal{CF}}(k).$$

Corollary *The ID scheme $\mathtt{ID2}$ is secure against concurrent man-in-the-middle attacks based on the Gap-CDH Assumption and the target collision resistance of an employed hash function family.*

Proof. By Proposition 1 and Theorem 4. (*Q.E.D.*)

6.3 Proof of Theorem 4

Let $\mathcal{A}$ be as in Theorem 4. Using $\mathcal{A}$ as subroutine, we construct a malleability extractor $\mathcal{E}$ against the non-malleability of f_λ. The construction is illustrated in Fig.8.

$\mathcal{E}$ is given $\lambda = (q, g)$ and function values $X_1 = f_\lambda(x_1), X_2 = f_\lambda(x_2)$ as input, where x_1 and x_2 are random and hidden. $\mathcal{E}$ initializes its inner state. $\mathcal{E}$ chooses $a^* \in \mathbf{Z}_q$ at random and computes $h^* = X_2 g^{a^*}$. Then $\mathcal{E}$ chooses μ from $Hkey(1^k)$ and computes $\tau^* \leftarrow H_\mu(h^*)$. $\mathcal{E}$ chooses $r \in \mathbf{Z}_q$ at random, and computes $Y = X_1^{-\tau^*} g^r$ and $d^* = (h^*)^r$. $\mathcal{E}$ sets $\mathtt{pk} = (\lambda, X_1, Y)$ and invokes $\mathcal{A}$ on input $\mathtt{pk}$. Note that $\mathtt{pk}$ is correctly

Key Generation
– $\mathtt{K}$: given 1^k as input;
  • $\lambda := (q,g) \leftarrow \mathtt{Grp}(1^k), x, y \leftarrow \mathbf{Z}_q$
  • $X := f_\lambda(x), Y := f_\lambda(y), \mu \leftarrow Hkey(1^k)$
  • $\mathtt{pk} := (\lambda, X, Y, \mu), \mathtt{sk} := (\lambda, x, y, \mu)$, return $(\mathtt{pk}, \mathtt{sk})$
Interaction
– $\mathtt{V}$: given $\mathtt{pk}$ as input;
  • $a \leftarrow \mathbf{Z}_q, h := g^a, \tau \leftarrow H_\mu(h), d := (X^\tau Y)^a$
  • Send (h,d) to $\mathtt{P}$
– $\mathtt{P}$: given $\mathtt{sk}$ as input and receiving (h,d) as input message;
  • $\tau \leftarrow H_\mu(h)$
  • If $h^{\tau x+y} \neq d$ then $D := \perp$ else $D := h^x$
  • Send D to $\mathtt{V}$
– $\mathtt{V}$: receiving D as input message;
  • If $D = X^a$ then return 1 else return 0

Figure 7: An ID Scheme $\mathtt{ID2}$.

distributed. Note also that $\mathcal{S}$ knows neither x_1 nor y, where y is the discrete log of Y;

$$y = \log_g(Y) = -\tau^* x_1 + r.$$

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

In case that $\mathcal{A}$ queries $\mathtt{V}(\mathtt{pk})$ for the first message by ϕ, $\mathcal{E}$ sends (h^*, d^*) to $\mathcal{A}$ (Call this case $\mathscr{V}$).

In case that $\mathcal{A}$ sends (h_i, d_i) to the i-th prover clone $\mathtt{P}_i(\mathtt{sk})$, $\mathcal{E}$ computes $\tau_i \leftarrow H_\mu(h_i)$. $\mathcal{E}$ verifies whether d_i is the related value of f_λ to $(X_1^{\tau_i} Y, h_i)$ w.r.t. $\mathcal{R}$. For this sake, $\mathcal{E}$ queries its decision oracle $\mathcal{D}_{f_\lambda, \mathcal{R}}$. If the answer is "False", then $\mathcal{E}$ puts $D_i = \perp$. Otherwise, if $\tau_i \neq \tau^*$, then $\mathcal{E}$ computes $D_i = (d_i/h_i^r)^{1/(\tau_i - \tau^*)}$ (Call this case $\mathscr{P}$). If $\tau_i = \tau^*$, then $\mathcal{E}$ aborts (Call this case Abort). Then $\mathcal{E}$ sends D_i to $\mathcal{A}$ except the case Abort.

In case that $\mathcal{A}$ sends D^* to $\mathtt{V}(\mathtt{pk})$, $\mathcal{E}$ verifies whether D^* is the related value of f_λ to (X_1, h^*) w.r.t. $\mathcal{R}$. For this sake, $\mathcal{E}$ queries $\mathcal{D}_{f_\lambda, \mathcal{R}}$. If the answer is "True", then $\mathcal{E}$ returns $X_3 = D^*/X_1^{a^*}$. Otherwise, $\mathcal{E}$ returns a random element $X_3 \in G_q$.

The view of $\mathcal{A}$ in $\mathcal{E}$ is the same as the real view until the case Abort happens, as we see below.

In the case $\mathscr{V}$, $\mathcal{E}$ simulates $\mathtt{V}(\mathtt{pk})$ perfectly. This is because the distribution of (h^*, d^*) is equal to that of the real (h,d). To see it, note that $x_2 + a^*$ is substituted for a;

$$h^* = g^{x_2 + a^*}, \; d^* = (g^{x_2+a^*})^r = (g^r)^{x_2+a^*} = (X_1^{\tau^*} Y)^{x_2+a^*}.$$

In the case $\mathscr{P}$, $\mathcal{E}$ simulates concurrent $\mathtt{P}_i(\mathtt{sk})$s perfectly. This is because $D_i = (d_i/h_i^r)^{1/(\tau_i - \tau^*)}$ is equal to $h_i^{x_1}$ by the following equalities.

$$d_i/h_i^r = h_i^{\tau_i x_1 + y - r} = h_i^{(\tau_i - \tau^*)x_1 + (\tau^* x_1 + y - r)} = h_i^{(\tau_i - \tau^*)x_1}.$$

Now we evaluate the advantage of $\mathcal{E}$. When $\mathcal{A}$ wins, D^* is the related value of f_λ to (X_1, h^*) w.r.t. $\mathcal{R}$, so the followings hold.

$$D^* = f_\lambda(\mathcal{R}(x_1, x_2 + a^*)) = g^{x_1(x_2+a^*)} = g^{x_1 x_2 + x_1 a^*}.$$

Hence the output X_3 is equal to $D^*/X_1^{a^*} = g^{x_1 x_2} = f_\lambda(\mathcal{R}(x_1, x_2))$. That is, X_3 is the related value of f_λ to (X_1, X_2) w.r.t. $\mathcal{R}$. This means that $\mathcal{E}$ wins. Therefore the probability that $\mathcal{E}$ wins is lower bounded by the probability that $\mathcal{A}$ wins and Abort does not happen.

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

Hence we get the following inequality.

$$\mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k) \geqslant \mathbf{Adv}^{\text{imp-cmim}}_{\mathtt{ID2},\mathcal{A}}(k) - \Pr[\textsc{Abort}].$$

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

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

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

In case that $\mathcal{A}$ queries $\mathtt{V}(\mathtt{pk})$ for the first message, $\mathcal{CF}$ sends (h^*, d^*) to $\mathcal{A}$.

In case that $\mathcal{A}$ sends (h_i, d_i) to the i-th prover clone $\mathtt{P}_i(\mathtt{sk})$, $\mathcal{CF}$ computes $\tau_i \leftarrow H_\mu(h_i)$ and verifies whether d_i is the related value of f_λ to $(X^{\tau_i}Y, h_i)$ w.r.t. $\mathcal{R}$. $\mathcal{CF}$ can check this in the same way as the real prover does because $\mathcal{CF}$ has the secret key $\mathtt{sk}$. If d_i is not so, $\mathcal{CF}$ sets $D_i = \perp$. Otherwise, if $\tau_i \neq \tau^*$, then $\mathcal{CF}$ sends $D_i = h_i^x$ to $\mathcal{A}$. If $\tau_i = \tau^*$, then $\mathcal{CF}$ outputs h_i 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 $\mathcal{E}$ until the case Abort or the case Collision happens. So we have;

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

Notice that the case Collision implies the followings;

$$\begin{cases} d_i \text{ is the related value of } f_\lambda \text{ to } (X^{\tau_i}Y, h_i) \text{ w.r.t. } \mathcal{R} \\ \text{and} \\ d^* \text{ is the related value of } f_\lambda \text{ to } (X^{\tau^*}Y, h^*) \text{ w.r.t. } \mathcal{R} \\ \text{and} \\ \tau_i = \tau^*. \end{cases}$$

If in addition to the above conditions h_i were equal to h^*, then d_i would be equal to d^*. This means that the transcript of a whole interaction with $\mathtt{P}_i(\mathtt{sk})$ would be relayed by $\mathcal{A}$, which is ruled out by the definition of man-in-the-middle attack. Hence it must hold that

$$h_i \neq h^*.$$

So in the case Collision, $\mathcal{CF}$ succeeds in obtaining a target collision. That is;

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

Combining the two equalities, we get

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

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

Given $\lambda = (q, g), X_1 = f_\lambda(x_1), X_2 = f_\lambda(x_2)$ as input;
Initial Setting
– Initialize the inner state
– $a^* \leftarrow \mathbf{Z}_q, h^* := X_2 g^{a^*}$
– $\mu \leftarrow Hkey(1^k), \tau^* \leftarrow H_\mu(h^*)$
– $r \leftarrow \mathbf{Z}_q, Y := X_1^{-\tau^*} g^r, d^* = (h^*)^r$
– $\mathtt{pk} := (\lambda, X_1, Y, \mu)$, invoke $\mathcal{A}$ on $\mathtt{pk}$
Answering $\mathcal{A}$'s Queries
– In case that $\mathcal{A}$ queries $\mathtt{V}(\mathtt{pk})$ for the first message (the case $\mathscr{V}$);
- Send (h^*, d^*) to $\mathcal{A}$

– In case that $\mathcal{A}$ sends (h_i, d_i) to $\mathtt{P}_i(\mathtt{sk})$;
- $\tau_i \leftarrow H_\mu(h_i)$
- If $\mathcal{D}_{f_\lambda,\mathcal{R}}(d_i : X_1^{\tau_i} Y, h_i) \neq 1$ then $D_i :=\perp$
- else
  If $\tau_i \neq \tau^*$ then $D_i := (d_i/h_i^r)^{1/(\tau_i - \tau^*)}$ (the case $\mathscr{P}$)
  else abort (the case Abort)
- Send D_i to $\mathcal{A}$

– In case that $\mathcal{A}$ sends D^* to $\mathtt{V}(\mathtt{pk})$;
- If $\mathcal{D}_{f_\lambda,\mathcal{R}}(D^* : X_1, h^*) = 1$ then return $X_3 := D^*/X_1^{a^*}$
- else return a random element $X_3 \in G_q$

Figure 8: A Malleability Extractor $\mathcal{E}$ for the Proof of Theorem 4.

6.4 Discussion

If it were a disadvantage for ID1, it would be the length of the maximum size message $(\mathtt{vk}, (h, d), \sigma)$. Fortunately, using the specific structure of tID, we can replace the tag by a TCR hash function value to get ID2, in which the message length is kept the same as that of tID.

We point out that the provers in ID1 and ID2 are deterministic. Therefore, ID1 and ID2 are prover-resettable [4]. Moreover, they are also verifier-resettable because they consists of 2-round interaction.

7 Efficiency Comparison

In this section, we evaluate the efficiency of our schemes comparing with other ID schemes secure against concurrent man-in-the-middle attacks in the standard model. It turns out that our fourth scheme is faster than the Cramer-Shoup-based ID scheme.

Comparable schemes are divided into three categories. The first category is proofs of knowledge, the second category is challenge-and-response ID schemes obtained from EUF-CMA signature schemes, and the third category is the ones obtained from IND-CCA2 encryption schemes. Note that we are considering schemes whose security proofs are in the standard model.

In the first category, to the best of our knowledge, the Gennaro Scheme is the most efficient but is no more efficient than the Cramer-Shoup-based ID scheme [13, 32, 14]. Moreover, the Gennaro Scheme needs 3-round but the Cramer-Shoup-based ID scheme needs only 2-round. As for the second category, all the known signature schemes in the standard model, including the Short Signature [3] and the Water's Signature [35], are far more inefficient than the Cramer-Shoup-based ID scheme. And finally, in the third category, the Cramer-Shoup-based ID scheme is the most efficient.

Therefore, we compare our schemes with the Cramer-Shoup-based ID scheme. Note that the Cramer-Shoup key encapsulation mechanism (KEM) [32, 14] is also usable as an ID scheme because the KEM is IND-CCA2 secure. Hence we compare the ID scheme obtained from the Cramer-Shoup Encryption Scheme (CS, for short) and the ID scheme obtained from the Cramer-Shoup KEM (CS-KEM, for short).

We remark that the Kurosawa-Desmedt Encryption Scheme [24] is not comparable because the KEM part of it is not CCA2 secure [20].

Table 1 shows the comparison of ID1 and ID2 with the CS and the CS-KEM.

Table 1: Efficiency Comparison. et. means an element in $\mathbf{Z}_q$, g-et. means an element in G_q.

Scheme	Assump.	Max. Msg. Length	Exponentiation	
			V	P
CS	DDH	4 g-et.	5	3
CS-KEM	DDH	3 g-et.	5	3
ID1	Gap-CDH	1 et. + 2 g-et. + $O(k^2)$	4	2
ID2	Gap-CDH	2 g-et.	4	2

We are estimating computational amount by counting the number of exponentiation. As in Table 1, ID2 is the fastest and is faster than the CS and the CS-KEM in one exponentiation in verifier and prover, respectively,

As for the maximum message length, which in fact is the message in the first round, ID2 is also the shortest and is shorter than the CS-KEM in 1 group element (and is shorter than the CS in 2 group elements). The maximum message length of ID1 is somewhat long. It amounts to a several kilo byte because of signature components, which appears as the term $O(k^2)$ in Table 1. Here we estimated it considering the case of the Lamport One-Time Signature [25].

8 Conclusion

We gave a definition of non-malleable functions and malleability extractors. Using these notions, we defined ID schemes of proofs of malleability. As a concrete example, we showed that exponentiation functions are non-malleable functions with respect to the multiplication relation. By this non-malleability and the tag framework with algebraic trick, we were able to construct a tag-based ID scheme that is a proof of malleability. This tag-based scheme achieved the security against concurrent man-in-the-middle attacks.

A generic method, the CHK transformation, was attractive to exit the tag framework, but the message length became somewhat long. Fortunately we were able to resolve the matter by using a target collision resistant hash function. This fourth scheme performs highly efficiently not only in message length but also in computational amount. Actually, it was shown that it performs better than the Cramer-Shoup-based ID scheme.

It is an interesting problem to find a non-malleable function in the RSA setting, to construct a malleability extractor by some technique, and to build up an ID scheme based on a proof of malleability.

Acknowledgements The authors appreciate valuable comments by anonymous reviewers of ProvSec 2010.

References

[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, Berlin, Germany.

[2] S. Arita, N. Kawashima, *"An Identification Scheme with Tight Reduction"*. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E90-A, issue 9, pp. 1949-1955, Sept. 2007.

[3] 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, Berlin, Germany.

[4] 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, Berlin, Germany.

[5] 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, Berlin, Germany.

[6] M. Bellare, C. Namprempre, D. Pointcheval, M. Semanko, *"The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme"*. In Proc. of *Financial Cryptography 2001*, Grand Cayman, British West Indies, Feb. 19-22, 2001, Lecture Notes in Computer Science, vol. 2339, pp. 319-338, Springer-Verlag, Berlin, 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, Berlin, 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, Berlin, Germany.

[9] M. Bellare, P. Rogaway, *"Collision-Resistant Hashing: Towards Making UOWHFs Practical"*. In Proc. of *CRYPTO '97*, Santa Barbara, CA, USA, Aug. 17-21, 1997, Lecture Notes in Computer Science, vol. 1294, pp. 470-484, Springer-Verlag, Berlin, 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, Berlin, 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, Berlin, 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, Berlin, Germany.

[13] 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, Berlin, Germany.

[14] 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.

[15] R. R. Dakdouk, *“Theory and Application of Extractable Functions”*. Doctor of Philosophy Dissertation, Yale University, New Haven, CT, USA, 2009.

[16] 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, Berlin, Germany.

[17] 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, Berlin, Germany.

[18] 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.

[19] 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, Berlin, Germany.

[20] 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/

[21] J. Katz, *“Efficient Cryptographic Protocols Preventing “Man-in-the-Middle” Attacks”*. Doctor of Philosophy Dissertation, Columbia University, New York, NY, USA, 2002.

[22] 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, Berlin, Germany.

[23] 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, Berlin, Germany.

[24] 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, Berlin, Germany.

[25] L. Lamport, *“Constructing Digital Signatures from a One-Way Function”*. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.

[26] 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, Berlin, Germany.

[27] 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.

[28] R. Nishimaki, E. Fujisaki, K. Tanaka, *“A Multi-trapdoor Commitment Scheme from the RSA Assumption”*. In Proc. of *ACISP 2010*, Sydney, Australia, July 5-7, 2010, Lecture Notes in Computer Science, vol. 6168, pp. 182-199, Springer-Verlag, Berlin, Germany.

[29] 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, Berlin, Germany.

[30] 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.

[31] 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, Berlin, Germany.

[32] V. Shoup, *"Using Hash Functions as a Hedge against Chosen Ciphertext Attack"*. In Proc. of *EUROCRYPT 2000*, Bruges, Belgium, May 14-18, 2000, Lecture Notes in Computer Science, vol. 1807, pp. 275-288, Springer-Verlag, Berlin, Germany.

[33] 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.

[34] J. Wu, D. R. Stinson, *"An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption"*. Cryptology ePrint Archive, 2007/479, http://eprint.iacr.org/

[35] B. Waters, *"Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions"*. In Proc. of *CRYPTO 2009*, Santa Barbara, CA, USA, Aug. 16-20, 2009, Lecture Notes in Computer Science, vol. 5677, pp. 619-636, Springer-Verlag, Berlin, Germany.

[36] 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, Berlin, Germany.

A A Proof of the Proposition 1

Let $\mathcal{E}$ be any given PPT algorithm for the non-malleability game "nm-$\mathcal{R}$-do". Employing $\mathcal{E}$ as subroutine, we construct a Gap-CDH problem solver $\mathcal{S}$ as follows. Let $\lambda = (q, g)$ be an output of $\mathtt{Grp}(1^k)$. For $x_1, x_2 \leftarrow \mathbf{Z}_q$, put $X_1 = g^{x_1} = f_\lambda(x_1), X_2 = g^{x_2} = f_\lambda(x_2)$. $\mathcal{S}$ is given q and (g, X_1, X_2) as input. $\mathcal{S}$ invokes $\mathcal{E}$ on input λ and (X_1, X_2). In case that $\mathcal{E}$ queries its decision oracle $\mathcal{D}_{f_\lambda,\mathcal{R}}$ whether X'_3 is the related value of f_λ to (X'_1, X'_2) w.r.t. $\mathcal{R}$, $\mathcal{S}$ queries its DDH oracle $\mathcal{DDH}$ about (g, X'_1, X'_2, X'_3). If the answer is "True", then $\mathcal{S}$ replies "True" to $\mathcal{E}$. Otherwise $\mathcal{S}$ replies "False" to $\mathcal{E}$. In case that $\mathcal{E}$ outputs X_3, $\mathcal{S}$ queries its DDH oracle $\mathcal{DDH}$ about (g, X_1, X_2, X_3). If the answer is "True", then $\mathcal{S}$ outputs X_3. Otherwise $\mathcal{S}$ outputs a random element in G_q.

We evaluate the advantages. If $\mathcal{E}$ wins, then X_3 is the related value of f_λ to (X_1, X_2) w.r.t. $\mathcal{R}$. That is, $X_3 = f_\lambda(\mathcal{R}(x_1, x_2)) = g^{x_1 x_2}$. This means that $\mathcal{S}$ wins. So we get

$$\mathbf{Adv}^{\text{gap-cdh}}_{\mathtt{Grp},\mathcal{S}}(k) \geqslant \mathbf{Adv}^{\text{nm-}\mathcal{R}\text{-do}}_{NMF,\mathcal{E}}(k).$$

The left-hand-side is negligible in k by the assumption of the proposition, so the right-hand-side is, too. (*Q.E.D.*)

B One-Time Signatures

A *one-time signature* $\mathtt{OTS}$ is a triple of PPT algorithms $(\mathtt{SGK}, \mathtt{Sign}, \mathtt{Vrfy})$. $\mathtt{SGK}$ is a signing key generator which outputs a pair of a verification key and a matching signing key $(\mathtt{vk}, \mathtt{sgk})$ on 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 (EUF-CMA) by any PPT forger $\mathcal{F}$. The following experiment is for the strong version.

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

Then we define *advantage of $\mathcal{F}$ over $\mathtt{OTS}$ in the game of existential unforgery in the strong sense against chosen message attack* as follows.

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

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

One-time signatures can be constructed, for example, based on the existence of a one-way function ([25]).

C Target Collision Resistant Hash Functions

Target collision resistant (TCR) hash functions [27, 30] are treated as a family. Let us denote a function family as $Hfam(1^k) = \{H_\mu\}_{\mu \in Hkey(1^k)}$. Here $Hkey(1^k)$ is a hash key space, $\mu \in Hkey(1^k)$ is a hash key and H_μ is a function from $\{0,1\}^*$ to $\{0,1\}^k$. We may assume that H_μ is from $\{0,1\}^*$ to $\mathbf{Z}_q$, where q is a prime of length k.

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

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

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

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

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

TCR hash function families can be constructed based on the existence of a one-way function [27, 30].