ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Security for Authenticated Key Exchange Based on Non-Malleability

Hiroki Hada and Keisuke Tanaka

December 2004, C–207

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Security for Authenticated Key Exchange Based on Non-Malleability

Hiroki Hada * Keisuke Tanaka *

Abstract— This paper continues the study of password-based protocols for authenticated key exchange (AKE). In 2000, Bellare, Pointcheval, and Rogaway [2] proposed the formal model on AKE. In this paper, we propose the new security notions on AKE, based on the non-malleability of session keys. Then we prove that this security notion is equivalent to that proposed in [2]. Furthermore, we show that there is a protocol secure in the random oracle model not always secure in the standard model with collision-resistant hash functions.

Keywords: AKE, AKE-Secure, Non-Malleability, OMDHKE

1 Introduction

This paper continues the study of password-based protocols for authenticated key exchange (AKE). We consider the scenario in which there are two entities—a client A and a server B—where A holds a password pw and B holds a key related to this. The parties would like to engage in a conversation at the end of which each holds a session key, sk, which is known to nobody but the two of them. There can be an active adversary A whose capabilities include enumerating, off-line, the words in a dictionary D, this dictionary being rather likely to include pw.

This kind of attack was first suggested by Bellovin and Merritt [3], who also offered a protocol, Encrypted Key Exchange (EKE), and gave its informal security analysis. This kind of attack has become quite popular, and there are further papers suggesting solutions.

In 2000, Bellare, Pointcheval, and Rogaway [2] proposed the formal model on AKE. In addition to it, they proposed the protocol EKE2, which is essentially the pair of flows from the Bellovin and Merritt's Diffie-Hellman based Encrypted Key Exchange protocol [3]. They showed that EKE2 is secure, in the ideal-cipher model.

In this paper, we continue the study of passward-based protocols for AKE. Our work is inspired by the paper of Nagao, Manabe, and Okamoto [6].

First, we propose the new security notions on AKE, based on the non-malleability of session keys. For public-key cryptosystems there are several notions based on non-malleability. In this paper, we focus on those notions proposed by Dolev, Dwork, and Naor [5] and Bellare, Desai, Pointcheval, and Rogaway [1], and propose the security notions corresponding to them.

Consider the following situation in order to observe the necessity of non-malleability for AKE. Even if an adversary cannot obtain a session key sk of A and B, she may obtain another key sk' which is related to sk. Then when A uses the session key sk in the conversation with B in some protocol, the adversary may attack the protocol with the related key sk'. For example, we consider the situation that A and B distribute a session key sk with an AKE protocol, then A sends a ciphertext c encrypted with sk by the Vernam Cipher to B. If an adversary obtains a key sk' whose most significant bit is the same as that of sk, then the adversary can get the most significant bit of the plaintext of c. Therefore, we propose the security notions corresponding to the above situation.

Then, we prove that these security notions are equivalent to that proposed in [2]. Furthermore, we show that there is a protocol secure in the random oracle model not always secure in the standard model with collision-resistant hash functions.

2 Preliminaries

The two models for AKE, with and without the queries for corrupting the clients, was proposed in [2]. In this paper, in order to simplify the arguments on the proofs we employ the model without the query that corrupts the clients. We can extend out results to the model with this kind of query. In this section, we review the model without this kind of query.

Protocol participants. We fix a nonempty set ID of principals. Each principal is either a client or a server: ID is the union of the finite, disjoint, nonempty sets Client and Server. Each principal $U \in ID$ is named by a string, and that string has some fixed length. When $U \in ID$ appears in a protocol flow or as an argument to a function, we mean to the string which names the principal.

Long-lived keys. Each principal $A \in Client$ holds some password, pw_A. Each server $B \in Server$ holds a vector $pw_B = \langle pw_B[A] \rangle_{A \in Client}$ which contains an entry per client. Entry $pw_B[A]$ is called the transformed-password. In a protocol for the symmetric model $pw_A = pw_B[A]$; that is, the client and server share the same password. In a protocol for the asymmetric model, $pw_B[A]$ will typically be chosen so that it is hard to compute pw_A from A, B, and $pw_B[A]$. The password

* Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan. (hada1, keisuke)@is.titech.ac.jp
Supported in part by NTT Information Sharing Platform Laboratories and Grant-in-Aid for Scientific Research, Ministry of Education, Culture, Sports, Science, and Technology, 14780190, 16092206.

pw_A (and therefore the transformed password $pw_B[A]$) might be a poor one. Probably some human chose it himself, and then installed $pw_B[A]$ at the server. We call the pw_A and pw_B long-lived keys (LL-keys).

Figure 1 specifies how a protocol is run. It is in Initialization that pw_A and pw_B arise: everybody's LL-key is determined by running a LL-key generator, PW. A simple possibility for PW is that the password for client A is determined by $pw_A \leftarrow_R PW_A$, for some finite set PW_A, and $pw_B[A]$ is set to pw_A. Notice that, in Figure 1, PW takes a superscript h, which is chosen from space Ω. This lets PW's behavior depend on an idealized hash function. Different LL-key generators can be used to capture other settings, like a public-key one.

Executing the protocol. Formally, a protocol is just a probabilistic algorithm taking strings to strings. This algorithm determines how instances of the principals behave in response to signals (messages) from their environment. It is the adversary who sends these signals. As with the LL-key generator, P may depend on h.

Adversary $\mathcal{A}$ is a probabilistic algorithm with a distinguished query tape. Queries written on this tape are answered as specified in Figure 1. The following description may clarify what is happening.

During the execution there may be running many instances of each principal $U \in ID$. We call instance i of principal U an oracle, and we denote it Π^i_U. Each instance of a principal might be embodied as a process (running on some machine) which is controlled by that principal.

A client-instance speaks first, producing some first message, *Flow1*. A server-instance responds with a message of its own, *Flow2*, intended for the client-instance which sent *Flow1*. This process is intended to continue for some fixed number of flows, until both instances have terminated. By that time each instance should have accepted, holding a particular session key (SK), session id (SID), and partner id (PID). Let us describe these more fully.

At any point in time an oracle may accept. When an oracle accepts it holds a session key sk, a session id sid, and a partner id pid. Think of these values as having been written on a write-only tape. The SK is what the instance was aiming to get. It can be used to protect an ensuing conversation. The SID is an identifier which can be used to uniquely name the ensuing session. The PID names the principal with which the instance believes it has just exchanged a key. The SID and PID aren't secret—indeed we will hand them to the adversary—but the SK certainly is. A client-instance and a server-instance can accept at most once.

The adversary A can make queries to any instance: she has an endless supply of Π^i_U oracles ($U \in ID$ and $i \in N$). There are all together five types of queries that A can make. The responses to these queries are specified in Figure 1. We now explain the capability that each kind of query captures.

(1) `Send` (U, i, M) — This sends message M to oracle Π^i_U. The oracle computes what the protocol says to, and sends back the response. Should the oracle accept, this fact, as well as the SID and PID, will be made visible to the adversary. Should the oracle terminate, this too will be made visible to the adversary. To initiate the protocol with client A trying to enter into an exchange with server B the adversary should send message $M = B$ to an unused instance of A. A Send-query models the real-world possibility of an adversary A causing an instance to come into existence, for that instance to receive communications fabricated by A, and for that instance to respond in the manner prescribed by the protocol.

(2) `Reveal` (U, i) — If oracle Π^i_U has accepted, holding some session key sk, then this query returns sk to the adversary. This query models the idea that loss of a session key shouldn't be damaging to other sessions. A session key might be lost for a variety of reasons, including hacking, cryptanalysis, and the prescribed-release of that session key when the session is torn down.

(3) `Execute` (A, i, B, j) — Assuming that client oracle Π^i_A and server oracle Π^j_B have not been used, this call carries out an honest execution of the protocol between these oracles, returning a transcript of that execution. This query may at first seem useless since, using `Send` queries, the adversary already has the ability to carry out an honest execution between two oracles. Yet the query is essential for properly dealing with dictionary attacks. In modeling such attacks the adversary should be granted access to plenty of honest executions, since collecting these involves just passive eavesdropping. The adversary is comparatively constrained in its ability to actively manipulate flows to the principals, since bogus flows can be auditied and punitive measures taken should there be too many.

(4) `Test` (U, i) — If Π^i_U has accepted, holding a session key sk, then the following happens. A coin b is flipped. If it lands $b = 0$, then sk is returned to the adversary. If it lands $b = 1$, then a random session key, drawn from the distribution from which session keys are supposed to be drawn, is returned. This type of query is only used to measure adversarial success—it does not correspond to any actual adversarial ability. You should think of the adversary asking this query just once.

(5) `Oracle` (M) — Finally, we give the adversary oracle access to a function h, which is selected at random from some probability space Ω. As already remarked, not only the adversary, but the protocol and the LL-key generator may depend on h. The choice of Ω determines if we are woking in the standard model, random oracle model. See the discussion below.

Standard model, random oracle model. Figure 1 refers to probability space Ω. We consider two possibilities for Ω, giving rise to two different models of computation. In the standard model Ω is the distribution which puts all the probability mass on one function: the constant function which returns the empty-string, ε, for any query M. So in the standard model, all mention of h can be ignored. Fix a finite set of strings $\mathcal{C}$. In the random oracle model choosing a random function from Ω means choosing a random function h from $\{0,1\}^*$ to $\mathcal{C}$. This models the use of a

cryptographic hash function which is so good that, for purposes of analysis, one prefers to think of it as a public random function.

PARTNERING USING SIDS. Fix a protocol P, adversary $\mathcal{A}$, LL-key generator PW, and session-key space $\mathcal{SK}$. Run P in the manner specified above. In this execution, we say that oracles Π^i_U and $\Pi^{i'}_{U'}$ are partnered (and each oracle is said to be a partner of the other) if both oracles accept, holding (sk, sid, pid) and (sk', sid', pid') respectively, and the following hold:

1. $sid = sid'$, $sk = sk'$, $pid = U'$, and $pid' = U$.
2. $(U, U') \in Client \times Server$, or $(U, U') \in Server \times Client$.
3. No oracle besides Π^i_U and $\Pi^{i'}_{U'}$ accepts with a PID of pid.

The above definition of partnering is quite strict. For two oracles to be partners with one another they should have the same SID and the same SK, one should be a client and the other a server, each should think itself partnered with the other, and, finally, no third oracle should have the same SID. Thus an oracle that has accepted will have a single partner, if it has any partner at all.

FRESHNESS. Once again, run a protocol with its adversary. Suppose that the adversary made exactly one Test query, and it was to Π^i_U. Intuitively, the oracle Π^i_U should be considered unfresh if the adversary may know the SK contained within it. In Figure 2 we define a notion of freshness. Here is the notation used in that figure. We say "$\mathsf{RevealTo}(U, i)$" is true iff there was, at some point in time, a query $\mathsf{Reveal}(U, i)$. We say "$\mathsf{RevealToPartnerOf}(U, i)$" is true iff there was, at some point in time, a query $\mathsf{Reveal}(U', i')$ and $\Pi^{i'}_{U'}$ is a partner to Π^i_U.

In order to define the security on authentication of AKE protocols, we use the following definition.

Definition 1. The goal of adversary is unilateral authentication of either A or S wherein the adversary impersonates a party. We denote by $\mathbf{Succ}^{\mathrm{A-auth}}_P(A)$ (resp. $\mathbf{Succ}^{\mathrm{S-Auth}}_P(A)$) the probability that A successfully impersonates an A instance (resp. an S instance) in an execution of P, which means that S (resp. A) agrees on a key, while the latter is shared with no instance of A (resp. S).

In order to define the security on session keys of AKE protocols, we use the following definition.

Definition 2. Let $A = (A_1, A_2)$ be an adversary. For $t \in N$ let,

$$\mathbf{Adv}^{\mathrm{AKE}}_P(A) = 2\Pr[\mathbf{Exp}^{\mathrm{AKE}}_P(A) = 1] - 1$$

where

```
Algorithm Initialization() {
  h ←_R Ω
  ⟨pw_A, pw_B⟩_{A∈Client,B∈Server} ←_R PW^h()
  for i ∈ N and U ∈ ID do
    state^i_U ← ready
    acc^i_U ← term^i_U ← used^i_U ← false
    sid^i_U ← pid^i_U ← sk^i_U ← false
}

Algorithm Send(U, i, M) {
  used^i_U ← true
  if term^i_U = true then return invalid
  ⟨msg-out, acc, term^i_U, sid, pid, sk, state^i_U⟩
    ← P^h(⟨U, pw_U, state^i_U, M⟩)
  if (acc = true and ¬acc^i_U = true)
    then sid^i_U ← sid; pid^i_U ← pid;
         sk^i_U ← sk; acc^i_U ← true
  return ⟨msg-out, sid, pid, acc, term^i_U⟩
}

Algorithm Reveal(U, i) {
  return sk^i_U
}

Algorithm Execute(A, i, B, j) {
  if A ∉ Client or B ∉ Server
      or used^i_A = true or used^j_B = true
    then return invalid
  msg-in ← B
  for t ← 1 to ∞ do
    ⟨msg-out, sid, pid, acc, term_A⟩
      ←_R Send(A, i, msg-in)
    α_t ← ⟨msg-out, sid, pid, acc, term_A⟩
    if term_A and term_B
      then return ⟨α_1, β_1, α_2, β_2, ..., α_t⟩
    ⟨msg-out, sid, pid, acc, term_B⟩
      ←_R Send(B, j, msg-in)
    β_t ← ⟨msg-out, sid, pid, acc, term_B⟩
    if term_A and term_B
      then return ⟨α_1, β_1, α_2, β_2, ..., α_t, β_t⟩
}

Algorithm Oracle(M) {
  return h(M)
}
```

Figure 1: Available oracles

```
Algorithm Fresh(U, i) {
  if RevealTo(U, i) or RevealToPartnerOf(U, i)
    then return 0
  return 1
}
```

Figure 2: Fresh

$\mathbf{Exp}_P^{\mathrm{AKE}}(A)$ {
  Initialization()
  $(U, i, s) \leftarrow A_1^{\mathcal{O}}()$
  $b \leftarrow_R \{0, 1\}$
  if $b = 1$
    then $sk \leftarrow \mathtt{Reveal}(U, i)$
    else $sk \leftarrow_R \mathcal{SK}$
  $d \leftarrow A_2^{\mathcal{O}}(sk, s)$
  return 1 iff
    $\mathtt{Fresh}(U, i) = 1$ and $b = d$ and $acc_U^i = \mathtt{true}$
}

and

$$\mathcal{O} = \{\mathtt{Send}, \mathtt{Reveal}, \mathtt{Execute}\}.$$

We say that the AKE protocol P is (t, ε)-AKE secure, if for all adversaries A with the running time t, $\mathbf{Adv}_P^{\mathrm{AKE}}(A)$ is smaller than ε.

3 New Security Notions

In this section we define two security notions based on non-malleability. For public-key cryptosystems there are several notions based on non-malleability. In this paper, we focus on this proposed by Dolev, Dwork, and Naor [5] and Bellare, Desai, Pointcheval, and Rogaway [1].

Consider the following situation. Even if an adversary cannot get a session key sk of A and B, she may get another key sk' which is related to sk. Then when A uses the session key sk in the communication with B in some protocol, the adversary may attack the protocol with the related key sk'. Therefore we propose the security notions corresponded to the above situation.

We propose the security notion NM1 corresponding to the non-malleability proposed in [5].

Definition 3. Let $A = (A_1, A_2)$ be an adversary and $S = (S_1, S_2)$ be a simulator and $R : (x, y, s) \to \{0, 1\}$ be a relation. For $t \in N$, define

$$\mathbf{Adv}_P^{\mathrm{NM1}}(R, A, S) = \Pr[\mathbf{Exp}_P^{\mathrm{NM1}}(R, A) = 1] - \Pr[\mathbf{Exp}_P^{\mathrm{NM1}}(R, S) = 1],$$

where

$\mathbf{Exp}_P^{\mathrm{NM1}}(R, A)$ {
  Initialization()
  $(U, i, s_1) \leftarrow A_1^{\mathcal{O}}()$
  $sk \leftarrow \mathtt{Reveal}(U, i)$
  $(sk', s_2) \leftarrow A_2^{\mathcal{O}}(s_1)$
  return 1 iff $\mathtt{Fresh}(U, i) = 1$
    and $R(sk, sk', s_2) = 1$ and $acc_U^i = \mathtt{true}$
}

$\mathbf{Exp}_P^{\mathrm{NM1}}(R, S)$ {
  Initialization()
  $(U, i, s_1) \leftarrow S_1^{\mathcal{O}}()$
  $sk \leftarrow_R \mathcal{SK}$
  $(sk', s_2) \leftarrow S_2^{\mathcal{O}}(s_1)$
  return 1 iff $\mathtt{Fresh}(U, i) = 1$
    and $R(sk, sk', s_2) = 1$ and $acc_U^i = \mathtt{true}$
}

and

$$\mathcal{O} = \{\mathtt{Send}, \mathtt{Reveal}, \mathtt{Execute}\}.$$

We say that the AKE protocol P is (t, ε)-NM1 Secure, if for all adversaries A with the running time t and for all relations R, there exists a simulator S such that $\mathbf{Adv}_P^{\mathrm{NM1}}(R, A, S) < \varepsilon$.

We next propose the security notion NM2 corresponding to the non-malleability in [1].

Definition 4. Let $A = (A_1, A_2)$ be an adversary. For $t \in N$, define

$$\mathbf{Adv}_P^{\mathrm{NM2}}(A) = \Pr[\mathbf{Exp}_P^{\mathrm{NM2}}(A) = 1] - \Pr[\widetilde{\mathbf{Exp}}_P^{\mathrm{NM2}}(A) = 1],$$

where

$\mathbf{Exp}_P^{\mathrm{NM2}}(A)$ {
  Initialization()
  $(U, i, s) \leftarrow A_1^{\mathcal{O}}()$
  $sk \leftarrow \mathtt{Reveal}(U, i)$
  $(R, sk') \leftarrow A_2^{\mathcal{O}}(s)$
  return 1 iff $\mathtt{Fresh}(U, i) = 1$
    and $R(sk, sk') = 1$ and $acc_U^i = \mathtt{true}$
}

$\widetilde{\mathbf{Exp}}_P^{\mathrm{NM2}}(A)$ {
  Initialization()
  $(U, i, s) \leftarrow A_1^{\mathcal{O}}()$
  $\widetilde{sk} \leftarrow_R \mathcal{SK}$
  $(R, sk') \leftarrow A_2^{\mathcal{O}}(s)$
  return 1 iff $\mathtt{Fresh}(U, i) = 1$
    and $R(\widetilde{sk}, sk') = 1$ and $acc_U^i = \mathtt{true}$
}

and

$$\mathcal{O} = \{\mathtt{Send}, \mathtt{Reveal}, \mathtt{Execute}\}.$$

We say that the AKE protocol P is (t, ε)-NM2 Secure, if for all adversaries A with the running time t, $\mathbf{Adv}_P^{\mathrm{NM2}}(A) < \varepsilon$.

4 Relations among the Notions of Security

In this section, we prove that these security notions proposed in this paper and that proposed in [2] are equivalent. In order to prove this, we will show the three theorems below. We denote $A \Rightarrow B$ that if protocol P is any AKE protocol meeting notion of security A then P also meets notion of security B.

Theorem 5. (t, ε)-NM2 Secure $\Rightarrow$ $(t - \alpha, \varepsilon)$-NM1 Secure.

Proof. We assume that a protocol P is not (t', ε)-NM1 Secure. That is, for some relation R, there exists an adversary $A = (A_1, A_2)$ attacking the NM1 Security of P such that $\mathbf{Adv}_P^{\mathrm{NM1}}(R, A, S) > \varepsilon$ for any simulator $S = (S_1, S_2)$. Then we construct the adversary $B = (B_1, B_2)$ attacking P in the sense of NM2 Security as follows.

Algorithm $B_1^{\mathcal{O}}()$ {	Algorithm $B_2^{\mathcal{O}}(s_1)$ {
$(U, i, s) \leftarrow A_1^{\mathcal{O}}()$	$(sk', s_2) \leftarrow A_2^{\mathcal{O}}(s_1)$
return (U, i, s)	Define $R'(x, y)$ as $R(x, y, s_2)$
}	return (R', sk')
	}

and the simulator $S = (S_1, S_2)$ as follows.

Algorithm $S_1^{\mathcal{O}}()$ {	Algorithm $S_2^{\mathcal{O}}(s_1)$ {
$(U, i, s) \leftarrow A_1^{\mathcal{O}}()$	$(sk', s_2) \leftarrow \mathcal{A}_2^{\mathcal{O}}(s_1)$
return (U, i, s)	return (sk', s_2)
}	}

It is easy to see that the running time of B is within a constant factor of that of A. Now we claim that $\mathbf{Adv}_P^{\mathrm{NM2}}(B) = \mathbf{Adv}_P^{\mathrm{NM1}}(R, A)$.

$\mathbf{Exp}_P^{\mathrm{NM2}}(A) = 1$ if and only if $\mathbf{Exp}_P^{\mathrm{NM1}}(A) = 1$, that is, $\Pr[\mathbf{Exp}_P^{\mathrm{NM1}}(A) = 1] = \Pr[\mathbf{Exp}_P^{\mathrm{NM2}}(A) = 1]$. Algorithm $\mathbf{Exp}_P^{\mathrm{NM2}}(S)$ is the same as $\widetilde{\mathbf{Exp}}_P^{\mathrm{NM1}}(A)$. □

Theorem 6. (t, ε)-NM1 Secure $\Rightarrow$ $(t - \alpha, \varepsilon)$-AKE Secure.

Proof. We prove that the AKE protocol P is not NM1 Secure under the assumption that P is not AKE secure. Let $A = (A_1, A_2)$ be the adversary attacking P in the sense of AKE security. We show that there exists another adversary $B = (B_1, B_2)$ attacking P in the sense of NM1 Security that succeeds with the same probability. The running time of B is almost the same as that of A. The adversary $B = (B_1, B_2)$ is as follows:

Algorithm $B_1^{\mathcal{O}}()$ {	Algorithm $B_2^{\mathcal{O}}(s_1)$ {
$(U, i, s) \leftarrow A_1^{\mathcal{O}}()$	$sk' \leftarrow_R \mathcal{SK}$
return (U, i, s)	return (sk', s_1)
}	}

and the relation $R(sk, sk', s_2)$ is as follows.

$R(sk, sk', s_2)$ {
  $d \leftarrow A_2^{\mathcal{O}}(sk, s_2)$
  return d
}

Then,

$$\Pr[\mathbf{Exp}_P^{\mathrm{NM1}}(R, A) = 1] = \Pr[\mathbf{Exp}_P^{\mathrm{AKE}}(A) = 1 \mid b = 1],$$

$$\Pr[\mathbf{Exp}_P^{\mathrm{NM1}}(R, S) = 1] = \Pr[\mathbf{Exp}_P^{\mathrm{AKE}}(A) = 0 \mid b = 0].$$

Therefore,

$$\begin{aligned}\mathbf{Adv}_P^{\mathrm{NM1}}(R, A, S) &= \Pr[\mathbf{Exp}_P^{\mathrm{AKE}}(A) = 1 \mid b = 1] \\ &\quad - (1 - \Pr[\mathbf{Exp}_P^{\mathrm{AKE}}(A) = 1] \mid b = 0) \\ &= 2\Pr[\mathbf{Exp}_P^{\mathrm{NM1}}(A) = 1] - 1 \\ &= \mathbf{Adv}_P^{\mathrm{AKE}}(A).\end{aligned}$$

□

Theorem 7. (t, ε)-AKE Secure $\Rightarrow$ $(t - \alpha, 2\varepsilon)$-NM2 Secure.

Proof. We prove that the AKE protocol P is not AKE secure under the assumption that P is not NM2 Secure. Let $A = (A_1, A_2)$ be the adversary attacking P in the sense of NM2 Security. We show that there exists another adversary $B = (B_1, B_2)$ attacking P in the sense of AKE security that succeeds with the same probability. The running time of B is almost the same as that of A. The adversary $B = (B_1, B_2)$ is as follows:

Algorithm $B_1^{\mathcal{O}}()$ {	Algorithm $B_2^{\mathcal{O}}(sk, s)$ {
$(U, i, s) \leftarrow A_1^{\mathcal{O}}()$	$(R, sk') \leftarrow A_2^{\mathcal{O}}()$
return (U, i, s)	if $R(sk, sk') = 1$
}	then $g \leftarrow 1$
	else $g \leftarrow_R \{0, 1\}$
	return g
	}

In this situation,

$$\begin{aligned}\Pr[\mathbf{Exp}_P^{\mathrm{AKE}}(A) = 1] &= \Pr[b = 1] \times \Pr[g = 1 \mid b = 1] \\ &\quad + \Pr[b = 0] \times \Pr[g = 0 \mid b = 0] \\ &= \frac{1}{2}\Big\{ \Pr[\mathbf{Exp}_P^{\mathrm{NM2}}(B) = 1] \times 1 \\ &\qquad + (1 - \Pr[\mathbf{Exp}_P^{\mathrm{NM2}}(B) = 1]) \times \frac{1}{2}\Big\} \\ &\qquad + \frac{1}{2}\Big\{ \Pr[\widetilde{\mathbf{Exp}}_P^{\mathrm{NM2}}(B) = 1] \times 0 \\ &\qquad + (1 - \Pr[\widetilde{\mathbf{Exp}}_P^{\mathrm{NM2}}(B) = 1]) \times \frac{1}{2}\Big\} \\ &= \frac{1}{4}(\Pr[\mathbf{Exp}_{P,B}^{\mathrm{NM2}}(z) = 1] - \Pr[\widetilde{\mathbf{Exp}}_{P,B}^{\mathrm{NM2}}(z) = 1]) + \frac{1}{2}.\end{aligned}$$

Therefore,

$$\begin{aligned}&2\Pr[\mathbf{Exp}_P^{\mathrm{AKE}}(A) = 1] - 1 \\ &\quad = \frac{1}{2}(\Pr[\mathbf{Exp}_P^{\mathrm{NM2}}(B) = 1] - \Pr[\widetilde{\mathbf{Exp}}_P^{\mathrm{NM2}}(B) = 1]).\end{aligned}$$

Thus,

$$\mathbf{Adv}_P^{\mathrm{AKE}}(A) = \frac{1}{2}\mathbf{Adv}_P^{\mathrm{NM2}}(A).$$

If there exists the NM2 adversary $A = (A_1, A_2)$ that succeeds with the probability ε and running time t, there is the AKE adversary that succeeds with the same probability and running time. □

5 Relaxing the Random Oracle Model in the One Masked Diffie Hellman Key Exchange Protocol

5.1 One Masked Diffie Hellman Key Exchange

We introduce the One Masked Diffie Hellman Key Exchange (OMDHKE) protocol described in Figure 3 and mention the security of that protocol in the random oracle model. The arithmetic is in a finite cyclic group $\mathbb{G} = \langle g \rangle$ of order a ℓ-bit prime number q, where the operation is denoted multiplicatively. We also denote by $\mathbb{G}^*$ the subset $\mathbb{G} \setminus \{1\}$ of the generators of $\mathbb{G}$. Hash functions from $\{0,1\}^*$ to $\{0,1\}^{\ell_i}$ are denoted $\mathcal{H}_i$, for $i = 0, 1$. While $\mathcal{G}$ denotes a full-domain hash function from $\{0,1\}^*$ into $\mathbb{G}$. As illustrated on Figure 3, the protocol runs between two parties C and S, and the session-key space $\mathcal{SK}$ associated to this protocol is $\{0,1\}^{\ell_0}$ equipped with a uniform distribution. The parties initially share a low-quality string pw, the password, drawn from the dictionary Password according to the distribution $\mathcal{D}_{pw}$. In the following, we use the notation $\mathcal{D}_{pw}(q)$ for the probability to be in the most probable set of q passwords:

$$\mathcal{D}_{pw}(q) = \max_{P \subseteq \mathsf{Password}} \left\{ \Pr_{pw \in \mathcal{D}_{pw}} [pw \in P \mid \#P \leq q] \right\}.$$

The security of this protocol is based on the Computational Diffie Hellman Assumption. It is described as follows.

Definition 8. A (t, ε)-$\mathsf{CDH}_{g,\mathbb{G}}$ attacker, in a finite cyclic group $\mathbb{G}$ of prime order q with g as a generator, is a probabilistic machine Δ running in time t such that its success probability $\mathbf{Succ}^{\mathrm{cdh}}_{g,\mathbb{G}}(\Delta)$, given random elements g^x and g^y to output g^{xy}, is greater than ε. As usual, we denote by $\mathbf{Succ}^{\mathrm{cdh}}_{g,\mathbb{G}}(t)$ the maximal success probability over every adversaries running within time t. The CDH-Assumption states that $\mathbf{Succ}^{\mathrm{cdh}}_{g,\mathbb{G}}(t) \leq \varepsilon$ for any t/ε not too large.

About the security of the OMDHKE protocol in the random oracle model, Bresson, Chevassut, and Pointcheval [4] proved following proposition. They claimed that the OMDHKE protocol is secure in the sense of the security notions of [2] in the random oracle model.

Proposition 9 (Bresson, Chevassut, and Pointcheval). Consider the OMDHKE protocol in the random oracle model, over a group of prime order q, where Password is a dictionary equipped with the distribution $\mathcal{D}_{pw}$. For any adversary A within time bound t, with less than q_s active interactions with the parties (Send-queries) and q_p passive eavesdroppings (Execute-queries), and asking q_g and q_h hash queries to the random oracles $\mathcal{G}$ and any $\mathcal{H}_i$, respectively,

$$\mathbf{Adv}^{\mathrm{AKE}}_{\mathrm{omdhke}}(A) \leq \frac{2q_s}{2^{\ell_1}} + 12 \times \mathcal{D}_{pw}(q_s) + 12q_h^2 \times \mathbf{Succ}^{\mathrm{cdh}}_{g,\mathbb{G}}(t + 2\tau_e) + \frac{2Q^2}{q}$$

$$\mathbf{Succ}^{\mathrm{S-auth}}_{\mathrm{omdhke}}(A) \leq \frac{q_s}{2^{\ell_1}} + 3 \times \mathcal{D}_{pw}(q_s) + 3q_h^2 \times \mathbf{Succ}^{\mathrm{cdh}}_{g,\mathbb{G}}(t + 3\tau_e) + \frac{Q^2}{2q}$$

where $Q = q_p + q_s + q_g$ and τ_e denotes the computational time for an exponentiation in $\mathbb{G}$.

5.2 Non AKE-Secure in the Standard Model with Collision-Resistant Functions

In section 5.1, we stated that the OMDHKE protocol in the random oracle model is secure. In this section, we show that the OMDHKE protocol is not secure in the standard model with collision-resistant functions.

Theorem 10. The OMDHKE protocol is not NM2 secure in the standard model with collision-resistant functions.

Proof. We denote $\mathcal{F}_1$ and $\mathcal{F}_2$ family of collision-resistant functions. For each function $F_i \in \mathcal{F}_i$ $(i = 1, 2)$, $F_i : \{0,1\}^* \to \{0,1\}^{\ell/2}$, define $\mathcal{F}$ such that each function $F : \{0,1\}^* \to \{0,1\}^{\ell}$ is as follows,

$$\mathcal{F} = \left\{ F(x) \;\middle|\; F(x) = \begin{cases} F_1(x) & (Y < \frac{q}{2}) \\ F_2(x) + \frac{\ell}{2} & (Y \geq \frac{q}{2}), \end{cases} (F_1, F_2) \in \mathcal{F}_1 \times \mathcal{F}_2 \right\},$$

where $x = C \| X^* \| S \| Y \| PW \| K$. Then $\mathcal{F}$ is also family of collision-resistant function. Otherwise, there is an adversary which can find $x_1,\ x_2$ such that $F(x_1) = F(x_2)$ $(x_1 \neq x_2)$. However, $\{F_1(x) \mid x \in \{0,1\}^*\} \wedge \{F_2(x) + \frac{\ell}{2} \mid x \in \{0,1\}^*\} = \emptyset$ so she can find $x_1,\ x_2$ $(x_1 \neq x_2)$ such that $F_1(x_1) = F_1(x_2)$ or $F_2(x_1) = F_2(x_2)$. The adversary $A = (A_1, A_2)$ is as follows.

```
Algorithm A_1^O() {
  ((C, X*), (S, Y, Auth_S)) ← Execute(C, 0, S, 0)
  s ← {C, S, X*, Y}
  return (C, 0, s)
}

Algorithm A_2^O(s) where s = {C, S, X*, Y} {
  Send(C, 1, S); (S, Y') ← Send(S, 1, X*)
  sk' ← Reveal(S, 1)
  if (Y < q/2 and Y' < q/2) or (Y ≥ q/2 and Y' ≥ q/2)
    then define R(a, b) =
      { 1  ((a < ℓ/2 and b < ℓ/2) or (a ≥ ℓ/2 and b ≥ ℓ/2))
      { 0  (otherwise)
    else define R(a, b) =
      { 1  ((a < ℓ/2 and b ≥ ℓ/2) or (a ≥ ℓ/2 and b < ℓ/2))
      { 0  (otherwise)
  return (R, sk')
}
```

The above algorithms use a few oracle queries

$$q_s = 2,\ q_p = 1,\ q_r = 1,$$

where q_r is the number of the Reveal queries. Those algorithms run in constant time and always return correct answers, that is, $\mathbf{Adv}^{\mathrm{NM2}}_{P}(A) = 1$. □

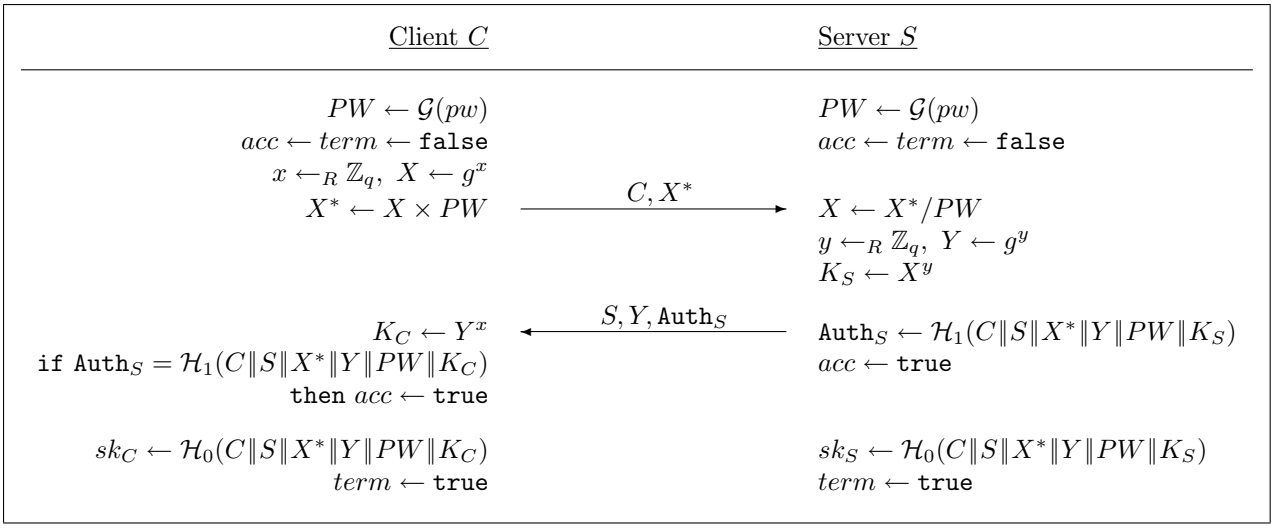

Figure 3: OMDHKE

6 Conclusion

In this paper, we have proposed two new security notions based on non-malleability which is often employed in public-key cryptosystems. We have proved these are equivalent to the security notion proposed in [2]. We have shown that there is a protocol secure in the random oracle model not always secure in the standard model with collision-resistant hash functions.

References

[1] Bellare, M., Desai, A., Pointcheval, D., and Rogaway, P. Relations Among Notions of Security for Public-key Encryption Schemes. In *Advances in Cryptology – CRYPTO '98* (Santa Barbara, California, USA, August 1998), H. Krawczyk, Ed., vol. 1462 of *LNCS*, Springer-Verlag.

[2] Bellare, M., Pointcheval, D., and Rogaway, P. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology – EUROCRYPT 2000* (Bruges, Belgium, May 2000), B. Preneel, Ed., vol. 1807 of *LNCS*, Springer-Verlag.

[3] Bellovin, S., and Merritt, M. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. In *Proc. of the Symposium on Security and Privacy* (1992), pp. 72–84.

[4] Bresson, E., Chevassut, O., and Pointcheval, D. New Security Results on Encrypted Key Exchange. In *Public Key Cryptography – PKC 2004* (Singapore, March 2004), F. Bao, R. H. Deng, and J. Zhou, Eds., vol. 2947 of *LNCS*, Springer-Verlag.

[5] Dolev, D., Dwork, C., and Naor, M. Non-Malleable Cryptography. In *Proceedings of the 23rd Annual Symposium on Theory of Computing* (1991), pp. 542–552.

[6] Nagao, W., Manabe, Y., and Okamoto, T. On the Security of Hybrid Public-Key Encryption. In *Symposium on Cryptography and Information Security – SCIS* (2004), pp. 35–40.