情報保護學會論文誌
第14券 第4號, 2004. 8

Security Proof for a Leakage-Resilient Authenticated Key Establishment Protocol

SeongHan Shin[†] , Kazukuni Kobara[‡] , Hideki Imai[‡]
The University of Tokyo, Japan

ABSTRACT

At Asiacrypt 2003, Shin et al., have proposed a new class for Authenticated Key Establishment (AKE) protocol named Leakage-Resilient AKE (LR-AKE)[1]. The authenticity of LR-AKE is based on a user's password and his/her stored secrets in both client side and server side. In their LR-AKE protocol, no TRM (Tamper Resistant Modules) is required and leakage of the stored secrets from any side does not reveal any critical information on the password. This property is useful when the following situation is considered : (1) Stored secrets may leak out ; (2) A user communicates with a lot of servers ; (3) A user remembers only one password. The other AKE protocols, such as SSL/TLS and SSH (based on PKI), Password-Authenticated Key Exchange (PAKE) and Threshold-PAKE (T-PAKE), do not satisfy that property under the above-mentioned situation since their stored secrets (or, verification data on password) in either the client or the servers contain enough information to succeed in retrieving the relatively short password with off-line exhaustive search. As of now, the LR-AKE protocol is the currently known solution.

In this paper, we prove its security of the LR-AKE protocol in the standard model. Our security analysis shows that the LR-AKE protocol is provably secure under the assumptions that DDH (Decisional Diffie-Hellman) problem is hard and MACs are selectively unforgeable against partially chosen message attacks (which is a weaker notion than being existentially unforgeable against chosen message attacks).

Keywords : *passwords, on-line and off-line attacks, authenticated key establishment, leakage of stored secrets, DDH problem, standard model*

I. Introduction

The need for secure authentication and key establishment, which is an indispensible cryptographic primitive, can be satisfied in the 2-party setting (e.g., a client (a user) and a server) by an authenticated key establishment (AKE) protocol, at the end of which the two parties share a common session key to be used for subsequent cryptographic algorithms such as symmetric key cryptosystems and message authentication codes. In the literature, a lot of various AKE protocols [2,3] have been proposed up to now so that we introduce related AKE protocols using password briefly.

1.1 Related Works

The problem of off-line attacks in password only AKE protocols was first discussed by Bellovin and Merritt [4] that was very influential and became the basis for PAKE and Threshold-PAKE protocols.

접수일 : 2004년 1월 12일 ; 채택일 : 2004년 12월 5일
† shinsh@imailab.iis.u-tokyo.ac.jp
‡ {kobara,imai}@iis.u-tokyo.ac.jp

PAKE protocols are designed in the 2-party setting, where a relatively short password is shared between the two parties in advance, to generate a cryptographically secure session key. While the pre-shared password is short enough and exhaustible with off-line attacks, the generated session key is no longer exhaustible with them even if an adversary completely controls the communications. Up to now, a variety of studies on PAKE protocols have been appeared in the literature (complete references can be found at web-page of Phoenix Technologies Inc.[5]). In PAKE protocols, a client is required to keep in mind his/her password whereas the counterpart server should have its verification data that is used to verify the client's knowledge of the password. Consequently, if stored secrets (or, password verification data) in the server are leaked out, the client's password can be retrieved through off-line dictionary attacks, simply by verifying password candidates one by one using the verification data.

In order to prevent the leakage of stored secrets from a server, Threshold-PAKE[6,7] protocols have been proposed where a client's password or verification data (or verification function) is not stored in a single server but rather shared among a set of servers using a secret sharing scheme. Since only a certain threshold of servers can reconstruct the client's password or verification data in the authentication phase, the leakage of stored secrets from any number of servers smaller than the threshold doesn't help an adversary to perform off-line attacks. However, the client's password can be retrieved through off-line attacks if stored secrets from the threshold or more than the threshold of servers are leaked out.

As a consequence, the password only protocols (PAKE and Threshold-PAKE) have faced the following limitation : Any AKE protocol, whose authenticity depends only on password (and public information) cannot achieve its immunity against off-line attacks after the leakage of stored secrets (or, verification data) from server side.

Recently at Asiacrypt 2003, Shin et al., have proposed a new class for AKE protocol titled as "Leakage-Resilient Authenticated Key Establishment[1]" (we call LR-AKE for short) raising a more natural and interesting problem on password only AKE protocols : one client and multiple sever scenario where the client remembers only one password and each of servers provides an independent service. The authenticity of LR-AKE protocol is based on password and additional stored secrets in order to cope with the above problems. The protocol in the multiple server scenario provides immunity of client's password to the leakage of stored secrets from a client and servers, respectively, which achieves a much stronger level of security without using PKI (Public Key Infrastructure) and TRM (Tamper-Resistant Modules). That means, the client need not change his password, even if stored secrets are leaked out from either the client or servers. Having said this, carrying some (insecure) devices seems a small price to pay for more strengthened security in the LR-AKE protocol. Refer to the original paper[1] for more detailed discussion compared to SSH (Secure SHell)[8], SSL/TLS (Secure Socket Layer/Transport Layer Security)[9,10], PAKE and Threshold-PAKE protocols.

II. Review of LR-AKE[1]

Shin et al., have proposed a new class of AKE protocol (we call SKI for their proposal) provably secure in the standard model based on the difficulty of the DDH (Decisional Diffie-Hellman) problem[11] (see

Definition 2 and 3). Their protocol is the first LR-AKE one that has immunity to the leakage of stored secrets from not only a client but also servers, respectively.

The protocol stated in this paper is defined over a finite cyclic group $G=\langle g\rangle$ where $|G|=q$ and q is a large prime (or, a positive integer divisible by a large prime) with the assumption that G is a prime order subgroup over a finite field $\mathbb{F}_p$. That is, $G=\{g^i \bmod p: 0\leq i\leq q\}$ where p is a large prime number, q is a large prime divisor of $p-1$ and g is an integer such that $1<g<p-1$, $g^q\equiv 1$ and $g^i\neq 1$ for $0<i<q$. A generator of G is any element in G except 1. In the aftermath, all the subsequent arithmetic operations are performed in modulo p, unless otherwise stated. Let g and h be two generators of G so that its DLP (Discrete Logarithm Problem), i.e. calculating $a=\log_g h$, should be hard for each entity. Both g and h may be given as system parameters or chosen with an initialization phase between entities. And $A|B$ represents the concatenation of A and B.

2.1 The SKI Protocol

The SKI protocol[1] considers a situation where a client wants to use one password and secret values to produce cryptographically secure session keys with different $n-1$ servers at any time. The mutual authentication and key exchange protocol is described as follows.

2.1.1 Initialization

A client C registers each of the secret values generated by his password pw to the respective $n-1$ different server $S_i\,(1\leq i\leq n-1)$ using a secret sharing scheme[12]. For simplicity, we assign the servers consecutive integer $i\,(1\leq i\leq n-1)$ where S_i can be regarded as i-th server. First and foremost, the client picks a random polynomial $p(x)$ of degree $n-1$ with coefficients also randomly chosen in $(\mathbb{Z}/q\mathbb{Z})^*$:

$$p(x)=\sum_{j=0}^{n-1} a_j\cdot x^j \bmod q \qquad (1)$$

and sets $a_0=pw$. After computing the respective share $p(i)\,(1\leq i\leq n-1)$ with the above polynomial, he registers securely each of the secret values $h^{p(i)\cdot\lambda_i}$ to the corresponding server $S_i\,(1\leq i\leq n-1)$ as follows :

$$S_i \leftarrow h^{p(i)\cdot\lambda_i}, \quad \text{where } \lambda_i=\prod_{k=1,k\neq i}^{n+1}\frac{k}{k-i} \bmod q \qquad (2)$$

where λ_i is a Lagrange coefficient. Note that share $p(n)$, which is for the client, is never registered to any server. Then, the client stores the corresponding *secret values* $h_i\leftarrow h^{\sum_{l=1,l\neq i}^{n} p(l)\cdot\lambda_l}\,(1\leq i\leq n)$ to the server S_i on devices and remembers *his password* pw.

2.1.2 Secrecy Amplification

When client C wants to share a session key securely with one of the servers $S_i\,(1\leq i\leq n-1)$, he chooses a random number $r_1\leftarrow_R(\mathbb{Z}/q\mathbb{Z})^*$. Then, the client sends y_1 to server S_i, after calculating $y_1\leftarrow g^{r_1}\cdot h_i\cdot h^{-pw}$ using the corresponding secret value h_i to the server and his password pw. The server S_i also calculates $y_2\leftarrow g^{r_2}\cdot h^{p(i)\cdot\lambda_i}$ with a random number $r_2\leftarrow_R(\mathbb{Z}/q\mathbb{Z})^*$ and its secret value $h^{p(i)\cdot\lambda_i}$ registered by the client in the initialization phase, and then transmits it to the client. On both sides, the client's keying material becomes $km_c \leftarrow (y_2 \cdot h_i \cdot h^{-pw})^{r_1}$ and the server's one becomes $km_s \leftarrow (y_1 \cdot h^{p(i) \cdot \lambda_i})^{r_2}$. Only if the client uses the right password pw and the corresponding secret value h_i to server S_i *and* the server S_i uses the right secret value $h^{p(i) \cdot \lambda_i}$, both of them can share the same keying material that is obtained by Lagrange interpolation. An adversary cannot determine the correct password of the client through off-line attacks since she doesn't know the client's random number r_1 chosen at the time and the secret value h_i corresponding to sever S_i, both of which are required to narrow down the password pw.

2.1.3 Verification and Session-Key Generation

The client and the server calculate $v_1 \leftarrow MAC_{km_c}(Tag_c \mid y_1 \mid y_2)$ and $v_2 \leftarrow MAC_{km_s}(Tag_s \mid y_1 \mid y_2)$, respectively, using a MAC generation function $MAC_k(\cdot)$ with the keying materials as its key k. Both Tag_c and Tag_s are pre-determined distinct values, e.g., $Tag_c = (ID_c \mid ID_s \mid 00)$ and $Tag_s = (ID_c \mid ID_s \mid 01)$ where ID_c and ID_s are IDs of the client and the server respectively. After exchanging v_1 and v_2 each other, they can verify whether they share the same keying material or not by seeing $v_2 = MAC_{km_c}(Tag_s \mid y_1 \mid y_2)$ and $v_1 = MAC_{km_s}(Tag_c \mid y_1 \mid y_2)$ on both sides. If at least one of them does not hold, the corresponding entities wipe off all the temporal data including the keying materials, and then close the session. Otherwise they generate their session keys using the verified keying materials as follows : $sk_c \leftarrow MAC_{km_c}(Tag_{sk} \mid y_1 \mid y_2)$ and $sk_s \leftarrow MAC_{km_s}(Tag_{sk} \mid y_1 \mid y_2)$ where Tag_{sk} is a pre-determined distinct value from both Tag_c and Tag_s, e.g., $Tag_{sk} = (ID_c \mid ID_s \mid 11)$. The generated session keys are used for their subsequent cryptographic algorithms.

2.2 Procedure for Provable Security

Here is a general procedure to prove the security for a protocol in the standard model.

① Description of protocol - Section Ⅱ
② Security model, definitions and adversary's objective - Section Ⅲ
③ Security proof (analysis and reductions) - Section Ⅳ

Ⅲ. Security Model and Definitions

In order to consider a more advantageous situation for adversaries, we assume that they have access to the following oracles. The oracles provided to the adversaries were originally introduced by Bellare et al.,[13] (which was in turn based on the previous paper[14]) for PAKE protocols, but a little modified for our protocol. Notice that this means entities actually only communicate through the adversaries.

- **Execute oracle** : It accepts two IDs of entities, holding the respective secret values. Then it carries out a honest execution of the protocol between them, and outputs the corresponding transcript. This oracle ensures that adversaries are able to observe all the transcripts between any pair of entities including the target ones.
- **Send oracle** : It accepts an entity ID and a message that is a part of transcript. It acts as the entity, and

then outputs a completed transcript corresponding to them. This oracle ensures that adversaries are able to run a protocol with any entity impersonating its partner and obtain the corresponding transcripts.

- **Reveal oracle** : It accepts both an entity ID and a session ID, and then reveals the corresponding session key. (This oracle does not reveal the session key of the challenge transcript.) Note that a session key might be revealed out since it is used outside of the protocol in various applications that might deal with the key insecurely (e.g. by using it as a key of very weak encryption algorithms). Reveal oracle simulates such a situation.
- **Leak oracle** : It accepts an entity ID (client's ID for formal security proof of SKI protocol), and then reveals the corresponding stored secrets. (This oracle does not reveal stored secrets of its partner at the same time.) Remind that AKE protocols that don't use password cannot maintain its security, given this oracle to adversaries. In case of leakage from servers which, of course, makes possible for adversaries to impersonate the victimized server as in the existing AKE protocols, but password of its partner (client) in the SKI protocol remains information-theoretically secure (refer to the original paper). Anyway, the situation where this oracle gives adversaries more strong power to obtain stored secrets of a client is as same as that of PAKE protocols for formal security proof. Note that a stored secret might be leaked out due to a bug of the system or physical limitations since it should be stored (all the time) in devices, such as smart cards or computers, that can be lost or stolen from a holder's carelessness. Leak oracle simulates such a situation.
- **Corrupt oracle** : This oracle is used to see whether the protocol satisfies forward secrecy[2], i.e. whether or not the disclosure of a long-lived secret (a password in SKI protocol) does compromise the secrecy of the session keys from earlier runs (even though that compromises the authenticity and thus the secrecy of new runs). It accepts an entity ID, and then reveals the corresponding password. This oracle can be used after transcripts related with the target password are generated.
- **Test_*sk* oracle** : This oracle is used to see whether adversaries can obtain some information on the challenge session key, by giving a hint on the session key to them. It accepts an entity ID in the challenge session, and then flips a coin $b \in \{0,1\}$. If $b=0$, it returns the corresponding session key. Otherwise, it returns a random one except the correct session key. This oracle can be used at most once per the challenge session.
- **Test_*km* oracle** : Because a session key is generated from a keying material, we prepare this oracle to see whether a strong secret can be generated in the secrecy amplification phase. This oracle accepts both an entity ID and a session ID, and then flips a coin $b \in \{0,1\}$. If $b=0$, it returns the corresponding keying material. Otherwise, it returns a random one except the correct keying material. Note that adversaries are not

2) This means that adversaries cannot learn any information about previously established session keys when making a query to the Corrupt oracle. Here, we do not deal with forward secrecy, but the security of SKI protocol can be proven in a very similar way of Section Ⅳ.

allowed to distinguish the obtained information from this oracle using $(Tag_c \mid y_1 \mid y_2)$ and v_1 or $(Tag_s \mid y_1 \mid y_2)$ and v_2 since it is given only to see whether a strong secret can be generated in the secrecy amplification phase.

Using the above oracles, adversaries are supposed to try to distinguish a session key given by Test_*sk* oracle. At first, we define the followings :

Definition 1 (Advantage) Let $\Pr[Win]$ denote the probability that an algorithm A can distinguish whether a given key is the correct session key or not. Then Adv_A^{ind}, the advantage of A distinguishing the session key, is given by

$$Adv_A^{ind} = 2 \cdot \Pr[Win] - 1 \qquad (3)$$

Definition 2 (DDH Problem) Given $g_b \in G$ and $d = (d_1, d_2, d_3) = (g_b^{x_1}, g_b^{x_2}, g_b^{x_3})$ where x_3 is either $x_1 \cdot x_2$ or not with probability 1/2, then decide whether $g_b^{x_3} = g_b^{x_1 \cdot x_2}$ or not.

Definition 3 (Probability of Solving DDH Problem) Let $\varepsilon_{ddh}(k_1, t)$ denote the probability that the DDH problem of size $k_1 = \log_2 q$ is solved in a polynomial time t with the best known algorithm.

Definition 2 and 3 indicate that a computationally-bounded adversary cannot solve the given DDH problem in a polynomial time.

The requirement for the MAC generation function in SKI protocol is that $\varepsilon_{mac}(k_2, t, i)$, given in the following Definition 4, can be negligibly small for the practical security parameter k_2 and i (that is a polynomial of k_2). That's the reason why if adversaries cannot forge a MAC corresponding to $(Tag_{sk} \mid y_1 \mid y_2)$ and km_c or km_s with significant probability, they cannot obtain any information of the session key. This requirement can be satisfied by using a universal one-way hash function[15] or by using a practical MAC generation function, such as HMAC-SHA-1[16] (and even KeyedMD5), since any effective algorithms have not been known so far to make $\varepsilon_{mac'}(k_2, t, i)$ non-negligible where $\varepsilon_{mac'}(k_2, t, i)$ given in Definition 5 is larger than or equal to $\varepsilon_{mac}(k_2, t, i)$.

Definition 4 (Selective UnForgeability of a MAC against Partially Chosen Message Attack) Let $\varepsilon_{mac}(k_2, t, i)$ denote the probability that a k_2 bit length MAC of a given message can be forged in a polynomial time t with the best known algorithm that are allowed to ask at most i (which is a polynomial of k_2) queries to the following MAC generation oracle (which is available in SKI protocol by abusing entities or using Execute, Send and Reveal oracles). The MAC generation oracle here accepts a message m, $entity \in \{server, client\}$, $target \in \{y, sk\}$ and a bijective function $f(\cdot)$ and then returns, for randomly chosen r_1 and r_2, $MAC_{f(km)}(Tag_s \mid m \mid g^{r_2})$ if $entity = server$ and $target = v$, $MAC_{f(km)}(Tag_c \mid g^{r_1} \mid m)$ if $entity = client$ and $target = v$, $MAC_{f(km)}(Tag_{sk} \mid m \mid g^{r_2})$ if $entity = server$ and $target = sk$ or $MAC_{f(km)}(Tag_{sk} \mid g^{r_1} \mid m)$ if $entity = client$ and $target = sk$, respectively. A MAC is said to be SUF-PCMA(Selectively UnForgeable against Partially Chosen Message Attacks) if $\varepsilon_{mac}(k_2, t, i)$ is negligibly small.

Definition 5 (Existential UnForgeability of a MAC against Chosen Message Attack) Let $\varepsilon_{mac'}(k_2, t, i)$ denote the probability that a new MAC-message pair for a k_2 bit length MAC can be generated in a polynomial time t with the best known algorithm that are allowed to ask at most i (which is a polynomial of k_2) queries to a MAC generation oracle, which accepts a message m and a bijective function $f(\cdot)$ and then returns $MAC_{f(km)}(m)$. A MAC is said to be EUF-CMA (Existentially UnForgeable against Chosen Message Attacks) if $\varepsilon_{mac'}(k_2, t, i)$ is negligibly small.

Ⅳ. Formal Security Proof

Under the following assumption, Theorem 1 is true.3) The intuitive interpretation of Theorem 1 is that if both N and G are large enough and both $\varepsilon_{mac}(k_2, t, q_{se}+2q_{ex}+q_{re}+2)$ and $\varepsilon_{ddh}(k_1, t)$ can be negligibly small for appropriate security parameters k_1 and k_2, the advantage for active adversaries (even if a client's secret values are given) can be bounded by a negligibly small value. Another interpretation can be provided as follows. On-line attacks for password in SKI protocol are possible, only if the client's secret values are given to adversaries. In other words, the SKI protocol can avoid even on-line attacks performed by adversaries as long as the client's secret values are not leaked out.

Assumption 1 (Password) Clients' passwords are chosen uniformly at random from a set of cardinality N.

3) Theorem 1 can be easily extended to the case where passwords are chosen non-uniformly since the uniformity assumption of passwords is just for simplicity.

Theorem 1 (Indistinguishability of *sk*) Suppose the following adversary A, which accepts a challenge transcript (that may be obtained by eavesdropping a protocol, impersonating a partner or intruding in the middle of the target entities), and then asks q_{ex}, q_{se}, q_{re} and q_{le} queries to the Execute, Send, Reveal, Leak oracles respectively, and finally is given sk_x by Test_*sk* oracle where sk_x is either the target session key or not with the probability of 1/2. Then Adv_A^{ind}, the advantage of adversary A to distinguish whether sk_x is the target session key or not in a polynomial time t, is upper bounded by

$$\begin{aligned} Adv_A^{ind} \le\ & \varepsilon_{mac}(k_2, t, q_{se}+2q_{ex}+q_{re}+2) \\ & +2(q_{se}+q_{ex}+1)\cdot\varepsilon_{ddh}(k_1, t) \\ & +\frac{2(q_{se}+1)}{N}+\frac{2(2q_{se}+q_{ex}+1)}{|G|} \end{aligned} \tag{4}$$

where both k_1 and k_2 are the security parameters.

Proof. Recall that *Win* is an event that A distinguishes sk_x correctly. *Win* happens either after an event *KmUnknown* occurs or after its compliment event $\overline{KmUnknown}$ occurs, where *KmUnknown* is an event that A does not obtain any significant information on the keying material *km* in the secrecy amplification phase and $\overline{KmUnknown}$ is an event that A obtains some significant information on the keying material *km* in the secrecy amplification phase. Thus $\Pr[Win]$ is upper bounded by

$$\begin{aligned} \Pr[Win] =\ & \Pr[Win \mid KmUnknown] \\ & \cdot \Pr[KmUnknown] \\ & + \Pr[Win \mid \overline{KmUnknown}] \\ & \cdot \Pr[\overline{KmUnknown}] \\ \le\ & \Pr[Win \mid KmUnknown] \\ & + \Pr[\overline{KmUnknown}] \end{aligned} \tag{5}$$

We evaluate $\Pr[Win \mid KmUnknown]$ first. Even if km is unknown, the following two adversaries A_{replay} and A_{mac} can distinguish sk_x. A_{mac} tries to forge a MAC of $(Tag_{sk} \mid y_1 \mid y_2)$, and then distinguish sk_x. A_{replay} tries to obtain at least one transcript coinciding with the challenge transcript using Send or Execute oracles, and then obtain the corresponding session key, which is the same as the challenge session key, using Reveal oracle.

Let $\Pr[Win_{A_{replay}}]$ and $\Pr[Win_{A_{mac}}]$ denote the probabilities of A_{replay} and A_{mac} being able to distinguish sk_x, respectively. $\Pr[Win_{A_{replay}}]$ is upper bounded by

$$\Pr[Win_{A_{replay}}] \le \frac{(q_{se}+q_{ex})}{|G|} \qquad (6)$$

since A_{replay} cannot control at least either (r_1 and h_i) or (r_2 and $h^{p(i)\cdot\lambda_i}$), but can obtain at most $(q_{se}+q_{ex})$ transcripts. The upper bound of $\Pr[Win_{A_{mac}}]$ is given by the following lemma.

► **Lemma 1** Suppose the probability that an adversary A_{mac} can forge a k_2 bit length MAC of a given message in a polynomial time t using i message-MAC pairs without knowing its key is $\varepsilon_{mac}(k_2, t, i)$. Then $\Pr[Win_{A_{mac}}]$, the probability of A_{mac} distinguishing a given session key without knowing its keying material, is upper bounded by $1/2+\varepsilon_{mac}(k_2,t,i)/2$.

Proof. The situation where A_{mac} tries to distinguish a session key can be divided into the following four cases according to whether a MAC (of the given message $(Tag_{sk} \mid y_1 \mid y_2)$ forged by A_{mac} is valid or not, and whether a key given by Test_*sk* oracle is correct or not, i.e., $b=0$ or $b=1$.

Let *MACValid* denote an event that the forged MAC is valid. The best strategy for A_{mac} to maximize the winning probability to distinguish the given key from Test_*sk* oracle is to return $b=0$ (with the probability 1) if the generated MAC coincides with the given key, and $b=1$ (with the probability 1) otherwise. Since A_{mac} can only know whether the generated MAC and the given key coincide each other or not, the probabilities they coincide and they do not are given by

$$\Pr[b=0, MACValid] + \Pr[b=1, \overline{MACValid}]\cdot\frac{1}{2^{k_2}-1} \qquad (7)$$

and

$$\Pr[b=0, \overline{MACValid}] + \Pr[b=1, MACValid] + \Pr[b=1, \overline{ForgeMAC}]\cdot\frac{2^{k_2}-2}{2^{k_2}-1} \qquad (8)$$

respectively where $\Pr[b=0, MACValid] > \Pr[b=1, \overline{MACValid}]\cdot\frac{1}{2^{k_2}-1}$ and $\Pr[b=0, \overline{MACValid}] > \Pr[b=1, MACValid] + \Pr[b=1, \overline{ForgeMAC}]\cdot\frac{2^{k_2}-2}{2^{k_2}-1}$ hold as long as $\varepsilon_{mac}(k_2,t,i) > \frac{1}{2^{k_2}}$. This gives the following probabilities.

$$\Pr[Win_{A_{mac}} \mid b=0, MACValid] = 1, \qquad (9)$$

$$\Pr[Win_{A_{mac}} \mid b=1, MACValid] = 1, \qquad (10)$$

$$\Pr[Win_{A_{mac}} \mid b=0, \overline{MACValid}] = 0, \qquad (11)$$

$$\Pr[Win_{A_{mac}} \mid b=1, \overline{MACValid}] = \frac{2^{Len(sk)}-2}{2^{Len(sk)}-1}. \qquad (12)$$

Thus, $\Pr[Win_{A_{mac}}]$ is upper bounded by

$$\begin{aligned}
&\Pr[Win_{A_{mac}}] \\
&= \Pr[Win_{A_{mac}} \mid b=0, MACValid] \\
&\quad \cdot \Pr[b=0] \cdot \Pr[MACValid] \\
&+ \Pr[Win_{A_{mac}} \mid b=1, MACValid] \\
&\quad \cdot \Pr[b=1] \cdot \Pr[MACValid] \\
&+ \Pr[Win_{A_{mac}} \mid b=0, \overline{MACValid}] \\
&\quad \cdot \Pr[b=0] \cdot \Pr[\overline{MACValid}] \\
&+ \Pr[Win_{A_{mac}} \mid b=1, \overline{MACValid}] \\
&\quad \cdot \Pr[b=1] \cdot \Pr[\overline{MACValid}] \\
&\ = \Pr[MACValid] \\
&+ \Pr[Win_{A_{mac}} \mid b=1, \overline{MACValid}] \\
&\quad \cdot \Pr[b=1] \cdot \Pr[\overline{MACValid}] \\
&\le \varepsilon_{mac}(k_2, t, i) \\
&+ \frac{2^{k_2}-2}{2^{k_2}-1} \cdot \frac{1}{2} \cdot 1 - \varepsilon_{mac}(k_2, t, i) \\
&\le \frac{1}{2} + \frac{\varepsilon_{mac}(k_2, t, i)}{2} \qquad (13)
\end{aligned}$$

A_{mac} can obtain at most $q_{se} + 2q_{ex} + q_{re}$ message-MAC pairs using Send, Execute, Reveal oracles, and at most 2 message-MAC pairs from a challenge transcript. Thus

$$i = q_{se} + 2q_{ex} + q_{re} + 2 \qquad (14)$$

By substituting (14) for (13) and summing up (6) and (33), we can obtain

$$\begin{aligned}
&\Pr[Win \mid KmUnknown] \le \frac{(q_{se}+q_{ex})}{|G|} + \frac{1}{2} \qquad (15) \\
&\quad + \frac{\varepsilon_{mac}(k_2, t, q_{se}+2q_{ex}+q_{re}+2)}{2}
\end{aligned}$$

Next we evaluate $\Pr[\overline{KmUnknown}]$, the probability of A being able to obtain some information on the keying material *km* in the secrecy amplification phase. In the secrecy amplification phase, A can obtain g, h, y_1, y_2 (and pre-images of either y_1 or y_2 by impersonating the corresponding entity) in a situation whether secret value h_i is given to the adversary using Leak oracle or not. The obtained data can be classified into the following twelve cases according to whether the secret value h_i is given or not *and* whether or not the password of the client and a randomly chosen password coincide with each other *and* whether or not the adversary knows the preimage of either g^{r_1} or g^{r_2}.

First, we think of the following cases where secret value h_i is not given to the adversary.

Case 1 : Password of the client pw and a random password pw' are different, i.e. $pw \neq pw'$, and the adversary knows the pre-image of neither g^{r_1} nor g^{r_2}.

Case 2 : Password of the client pw and a random password pw' are the same, i.e. $pw = pw'$, and the adversary knows the pre-image of neither g^{r_1} nor g^{r_2}.

Case 3 : Password of the client pw and a random password pw' are different, i.e. $pw \neq pw'$, and the adversary knows the pre-image of g^{r_2}.

Case 4 : Password of the client pw and a random password pw' are different, i.e. $pw \neq pw'$, and the adversary knows the pre-image of g^{r_1}.

Case 5: Password of the client pw and a random password pw' are the same, i.e. $pw = pw'$, and the adversary knows the pre-image of g^{r_2}.

Case 6 : Password of the client pw and a random password pw' are the same, i.e. $pw = pw'$, and the adversary knows the pre-image of g^{r_1}.

Even though Case 6 is more advantageous for A than any other five cases, it happens only when A inputs the correct password pw and the correct secret value h_i while impersonating the client on-line. Rather than guessing the correct password pw and the correct secret value h_i, A can try a random value for $h^{-p(i)\cdot\lambda_i}$

$$h_i \cdot h^{-pw} = h^{-p(i)\cdot\lambda_i} \tag{16}$$

The probability of distinguishing the keying material of the server in this case is bounded by $(q_{se}+1)/|G|$ since A can try at most $q_{se}+1$ random values impersonating the client on-line where q_{se} random values are tried using Send oracle and 1 using the challenge session. The other cases happen with more high probabilities. For example, Case 1 and 2 happen when an adversary eavesdrops a session or sends modified values of ever used y_1 or y_2, i.e. sends $y_1 \cdot g^{j_1} \cdot h^{j_2}$ or $y_2 \cdot g^{j_1} \cdot h^{j_2}$ for $j_1, j_2 \in (\mathbb{Z}/q\mathbb{Z})^*$ to the corresponding entity. Case 3 and 5 happen when an adversary impersonates the server. Meanwhile, Case 4 happens when an adversary generates $g^{r_1} \cdot h_i' \cdot h^{-pw'}$ from its pre-images and sends y_1 to the server.

While Case 1 to 5 happen with high probability, distinguishing the keying material in these cases is as hard as or harder than solving DDH problem. Lemma 2 shows that distinguishing it in Case 2 is as hard as or harder than solving DDH problem.

► **Lemma 2** Suppose there exists an algorithm A_1, which accepts a challenge transcript g, h, y_1 and y_2 between the entities holding the respective secret value, and then is given a hint km_x from Test_*km* oracle where km_x is either equal to the keying material of the target entity, i.e. km_c or km_s, or not with the probability of 1/2. A_1 finally distinguishes whether km_x is the correct keying material or not in at most τ steps and with the advantage of ε. Then one can construct an algorithm B_1, which runs in τ' steps and solves a given DDH problem with the advantage of ε' where $\varepsilon' = \varepsilon$, $\tau' = \tau + Poly(k_1)$ and $Poly(k_1)$ is a polynomial of the security parameter $k_1 = \log_2 q$.

Proof. B_1 can be constructed as follows. At first B_1 receives a DDH set g_b and $d = (d_1, d_2, d_3) = (g_b^{x_1}, g_b^{x_2}, g_b^{x_3})$. B_1 chooses a random password $pw' = pw$, a random generator $h \in G$ and a random value h_i'. After computing the corresponding value $(h^{p(i)\cdot\lambda_i})' = (h_i' \cdot h^{-pw'})^{-1}$, B_1 gives $g := g_b$, h, $y_1 := d_1 \cdot h_i' \cdot h^{-pw'}$, $y_2 := d_2 \cdot (h^{p(i)\cdot\lambda_i})'$ and $km_x := d_3$ to A_1. If the answer of A_1 is $km_x = km_c$ (which also means $km_x = km_s$), B_1 returns $d_3 = g_b^{x_1 \cdot x_2}$. Otherwise it returns $d_3 \neq g_b^{x_1 x_2}$.

B_1 can solve the DDH problem with the same advantage as ε since $d_3 = g_b^{x_1 \cdot x_2}$ holds with probability 1 if $km_x = km_c = km_s$. The number of steps required for B_1 is mainly consumed in the calculation of $h_i' \cdot h^{-pw'}$ and $(h^{p(i)\cdot\lambda_i})'$ which ends in polynomial steps of $k_1 = \log_2 q$. Thus $\tau' = \tau + Poly(k_1)$.

Lemma 3 shows that distinguishing the keying material of the client (while impersonating the server) in Case 3 is as hard as or harder than solving DDH problem.

► **Lemma 3** Suppose there exists an algorithm A_2, which accepts g, h, y_1, y_2, r_2, pw' and km_x where g, h, y_1 and y_2 are a challenge transcript between the entities holding the respective secret value, r_2 is the pre-image of g^{r_2}, pw' is a random password, and km_x is a hint km_x given by Test_km oracle which is either km_c or not with the probability of 1/2. A_2 finally distinguishes whether $km_x = km_c$ or not in at most τ steps and with the advantage of ε. Then one can construct an algorithm B_2, which runs in τ' steps and solves a given DDH problem with the advantage of ε' where $\varepsilon' = \varepsilon$, $\tau' = \tau + Poly(k_1)$ and $Poly(k_1)$ is a polynomial of the security parameter $k_1 = \log_2 q$.

Proof. B_2 can be constructed as follows. At first B_2 receives a DDH set g_b and $d = (d_1, d_2, d_3) = (g_b^{x_1}, g_b^{x_2}, g_b^{x_3})$. B_2 chooses a random number $r_2 \in (\mathbb{Z}/q\mathbb{Z})^*$, a distinct password pw' from pw ($pw' \neq pw$) and a random value h_i'. After computing the corresponding value $(h^{p(i)\cdot\lambda_i})' = (h_i' \cdot h^{-pw'})^{-1}$, B_2 gives A_2 $g := g_b$, $h := d_2$, $y_1 := d_1 \cdot h_i' \cdot h^{-pw'}$, $y_2 := g^{r_2} \cdot (h^{p(i)\cdot\lambda_i})'$, r_2, pw' and $km_x := d_1^{r_2} \cdot d_3^{(pw-pw')}$. If the answer of A_2 is $km_x = km_c$, B_2 returns $d_3 = g_b^{x_1 \cdot x_2}$. Otherwise it returns $d_3 \neq g_b^{x_1 x_2}$.

B_2 can solve the DDH problem with the same advantage as ε since

$$km_x = d_1^{r_2} \cdot d_3^{(pw-pw')}, \tag{17}$$

$$\begin{aligned} km_c &= (g^{r_2} \cdot (h^{p(i)\cdot\lambda_i})' \cdot h_i' \cdot h^{-pw'})^{x_1} \\ &= (g^{r_2} \cdot h^{pw} \cdot h^{-pw'})^{x_1} \\ &= d_1^{r_2} \cdot g_b^{x_2(pw-pw')x_1} \end{aligned} \tag{18}$$

and $d_3 = g_b^{x_1 \cdot x_2}$ hold if $km_x = km_c$. The number of steps required for B_2 is mainly consumed in the calculation of $h_i' \cdot h^{-pw'}$ and km_x which ends in polynomial steps of $k_1 = \log_2 q$. Thus $\tau' = \tau + Poly(k_1)$.

Lemma 4 shows that distinguishing the keying material of the server (while impersonating the client) in Case 4 is as hard as or harder than solving DDH problem.

► **Lemma 4** Suppose there exists an algorithm A_3, which accepts g, h, y_1, y_2, r_1, pw' and km_x where g, h, y_1 and y_2 are a challenge transcript between the entities holding the respective secret value, r_1 and pw' are the pre-image of $g^{r_1} \cdot h^{-pw'}$, and km_x is a hint km_x given by Test_km oracle which is either km_s or not with the probability of 1/2. A_3 finally distinguishes whether $km_x = km_s$ or not in at most τ steps and with the advantage of ε. Then one can construct an algorithm B_3, which runs in τ' steps and solves a given DDH problem with the advantage of ε' where $\varepsilon' = \varepsilon$, $\tau' = \tau + Poly(k_1)$ and $Poly(k_1)$ is a polynomial of the security parameter $k_1 = \log_2 q$.

Proof. B_3 can be constructed as follows. At first B_3 receives a DDH set g_b and

$d=(d_1,d_2,d_3)=(g_b^{x_1},g_b^{x_2},g_b^{x_3})$. B_3 chooses a random number $r_1 \in (\mathbb{Z}/q\mathbb{Z})^*$, a distinct password pw' from pw ($pw' \neq pw$) and a random value h_i'. After computing the corresponding value $(h^{p(i)\cdot\lambda_i})'=(h_i'\cdot h^{-pw'})^{-1}$, B_3 gives A_3 $g:=g_b$, $h:=d_1$, $y_1:=g^{r_1}\cdot h_i'\cdot h^{-pw'}$, $y_2:=d_2\cdot(h^{p(i)\cdot\lambda_i})'$, r_1, pw' and $km_x:=d_2^{r_1}\cdot d_3^{(pw-pw')}$. If the answer of A_3 is $km_x=km_s$, B_3 returns $d_3=g_b^{x_1\cdot x_2}$. Otherwise it returns $d_3 \neq g_b^{x_1x_2}$.

B_3 can solve the DDH problem with the same advantage as ϵ since

$$km_x=d_2^{r_1}\cdot d_3^{(pw-pw')} \tag{19}$$

$$\begin{aligned} km_s&=(g^{r_1}\cdot h_i'\cdot h^{-pw'}\cdot(h^{p(i)\cdot\lambda_i})')^{x_2}\\ &=(g^{r_1}\cdot h^{pw}\cdot h^{-pw'})^{x_2}\\ &=d_2^{r_1}\cdot g_b^{x_1(pw-pw')x_2} \end{aligned} \tag{20}$$

and $d_3=g_b^{x_1\cdot x_2}$ hold if $km_x=km_s$. The number of steps required for B_3 is mainly consumed in the calculation of y_1, $(h^{p(i)\cdot\lambda_i})'$ and km_x which ends in polynomial steps of $k_1=\log_2 q$. Thus $\tau'=\tau+Poly(k_1)$.

Distinguishing the keying material of the target entity in Case 1 is as hard as or harder than those of Case 3 and Case 4 since the pre-image of neither g^{r_1} nor g^{r_2} is given to the adversary in Case 1. The corresponding proof can be obtained simply by removing r_2 (r_1) from the inputs of B_2 (B_3) in the proof of Lemma 3 (Lemma 4), respectively.

Lemma 5 shows that distinguishing the keying material of the client (while impersonating the server) in Case 5 is as hard as or harder than solving DDH problem.

► **Lemma 5** Suppose there exists an algorithm A_4, which accepts g, h, y_1, y_2, r_2, pw' and km_x where g, h, y_1 and y_2 are a challenge transcript between the entities holding the respective secret value, r_2 is the pre-image of g^{r_2}, pw' is a random password, and km_x is a hint km_x given by Test_km oracle which is either km_c or not with the probability of 1/2. A_4 finally distinguishes whether $km_x=km_c$ or not in at most τ steps and with the advantage of ε. Then one can construct an algorithm B_4, which runs in τ' steps and solves a given DDH problem with the advantage of ε' where $\varepsilon'=\varepsilon$, $\tau'=\tau+Poly(k_1)$ and $Poly(k_1)$ is a polynomial of the security parameter $k_1=\log_2 q$.

Proof. B_4 can be constructed as follows. At first B_4 receives a DDH set g_b and $d=(d_1,d_2,d_3)=(g_b^{x_1},g_b^{x_2},g_b^{x_3})$. B_4 chooses a random number $r_2 \in (\mathbb{Z}/q\mathbb{Z})^*$, a random password $pw'=pw$, a random generator $h \in G$ and a random value h_i'. After computing the corresponding value $(h^{p(i)\cdot\lambda_i})'=(h_i'\cdot h^{-pw'})^{-1}$, B_4 gives A_4 $g:=g_b$, h, $y_1:=d_1\cdot h_i'\cdot h^{-pw'}$, $y_2:=g^{r_2}\cdot(h^{p(i)\cdot\lambda_i})'$, r_2, pw' and $km_x:=d_1^{r_2}\cdot d_3$. If the answer of A_4 is $km_x=km_c$, B_4 returns $d_3=g_b^{x_1\cdot x_2}$. Otherwise it returns $d_3 \neq g_b^{x_1x_2}$.

B_4 can solve the DDH problem with the same advantage as ϵ since

$$km_x=d_1^{r_2}\cdot d_3, \tag{21}$$

$$\begin{aligned} km_c&=(g^{r_2}\cdot d_2\cdot(h^{p(i)\cdot\lambda_i})'\cdot h_i'\cdot h^{-pw'})^{x_1}\\ &=(g^{r_2}\cdot d_2)^{x_1}=d_1^{r_2}\cdot g_b^{x_2\cdot x_1} \end{aligned} \tag{22}$$

and $d_3 = g_b^{x_1 \cdot x_2}$ hold if $km_x = km_c$. The number of steps required for B_4 is mainly consumed in the calculation of $h_i' \cdot h^{-pw''}$, $g^{r_2} \cdot (h^{p(i) \cdot \lambda_i})'$ and $d_1^{r_2}$ which ends in polynomial steps of $k_1 = \log_2 q$. Thus $\tau' = \tau + Poly(k_1)$.

Second, we think of the following cases where secret value h_i is given to the adversary by using Leak oracle.

Case 7 : Password of the client pw and a random password pw' are different, i.e. $pw \neq pw'$, and the adversary knows the pre-image of neither g^{r_1} nor g^{r_2}.

Case 8 : Password of the client pw and a random password pw' are the same, i.e. $pw = pw'$, and the adversary knows the pre-image of neither g^{r_1} nor g^{r_2}.

Case 9 : Password of the client pw and a random password pw' are different, i.e. $pw \neq pw'$, and the adversary knows the pre-image of g^{r_2}.

Case 10 : Password of the client pw and a random password pw' are different, i.e. $pw \neq pw'$, and the adversary knows the pre-image of . g^{r_1}

Case 11 : Password of the client pw and a random password pw' are the same, i.e. $pw = pw'$, and the adversary knows the pre-image of g^{r_2}.

Case 12 : Password of the client pw and a random password pw' are the same, i.e. $pw = pw'$, and the adversary knows the pre-image of g^{r_1}.

Even though Case 12 is the most advantageous for A among all the cases, it happens only when A inputs the correct password pw while impersonating the client on-line. The probability of distinguishing the keying material of the server in this case is bounded by $(q_{se}+1)/N$ since A can try at most $q_{se}+1$ passwords impersonating the client on-line where q_{se} passwords are tried using Send oracle and 1 using the challenge session. Case 7 to 11 happen in the same way of Case 1 to 5, respectively, with more high probabilities.

While Case 7 to 11 happen with high probability, distinguishing the keying material in these cases is as hard as or harder than solving DDH problem. Lemma 6 shows that distinguishing it in Case 8 is as hard as or harder than solving DDH problem.

► **Lemma 6** Suppose there exists an algorithm A_5, which accepts g, h, y_1, y_2, h_i and km_x where g, h, y_1 and y_2 are a challenge transcript between the entities holding the respective secret value and h_i is the client's secret value, and then is given a hint km_x from Test_km oracle where km_x is either equal to the keying material of the target entity, i.e. km_c or km_s, or not with the probability of 1/2. A_5 finally distinguishes whether km_x is the correct keying material or not in at most τ steps and with the advantage of ε. Then one can construct an algorithm B_5, which runs in τ' steps and solves a given DDH problem with the advantage of ε' where $\varepsilon' = \varepsilon$, $\tau' = \tau + Poly(k_1)$ and $Poly(k_1)$ is a polynomial of the security parameter $k_1 = \log_2 q$.

Proof. B_5 can be constructed as follows. At first B_5 receives a DDH set g_b and $d=(d_1,d_2,d_3)=(g_b^{x_1},g_b^{x_2},g_b^{x_3})$. B_5 chooses a random password $pw'=pw$, a random generator $h \in G$ and a random value h_i'. After computing the corresponding value $(h^{p(i)\cdot\lambda_i})'=(h_i'\cdot h^{-pw'})^{-1}$, B_5 gives $g:=g_b$, h, $y_1:=d_1\cdot h_i'\cdot h^{-pw'}$, $y_2:=d_2\cdot(h^{p(i)\cdot\lambda_i})'$, $h_i=h_i'$ and $km_x:=d_3$ to A_5. If the answer of A_5 is $km_x=km_c$ (which also means $km_x=km_s$), B_5 returns $d_3=g_b^{x_1\cdot x_2}$. Otherwise it returns $d_3 \neq g_b^{x_1 x_2}$.

B_5 can solve the DDH problem with the same advantage as ε since $d_3=g_b^{x_1\cdot x_2}$ holds with probability 1 if $km_x=km_c=km_s$. The number of steps required for B_5 is mainly consumed in the calculation of $h_i'\cdot h^{-pw'}$ and $(h^{p(i)\cdot\lambda_i})'$ which ends in polynomial steps of $k_1=\log_2 q$. Thus $\tau'=\tau+Poly(k_1)$.

The rest proofs of Case 7, 9, 10 and 11 can be obtained simply by adding $h_i=h_i'$ to the inputs of each algorithm in the proof of Case 1, 3, 4 and 5 respectively.

From the above discussion and Definition 3, the probability that one can obtain some information on the keying material from one transcript in Case 1 to 5 and Case 7 to 11 is upper bounded by $\varepsilon_{ddh}(k_1,t)$. In total, A can obtain at most $q_{se}+q_{ex}+1$ transcripts where $q_{se}+q_{ex}$ transcripts can be obtained using Send and Execute oracles, and 1 from a challenge transcript. Thus the probability of A being able to obtain some information on the challenge keying material in Case 1 to 5 and Case 7 to 11 is upper bounded by $(q_{se}+q_{ex}+1)\cdot\varepsilon_{ddh}(k_1,t)$. And then the probability of A being able to obtain it in the secrecy amplification phase is upper bounded by

$$\begin{aligned}&\Pr[\overline{KmUnknown}]\\&\le\frac{(q_{se}+1)}{|G|}+\frac{(q_{se}+1)}{N}\\&+(q_{se}+q_{ex}+1)\cdot\varepsilon_{ddh}(k_1,t)\end{aligned}\qquad(23)$$

By substituting (15) and (23) for (5), the upper bound of $\Pr[Win]$ is given by

$$\begin{aligned}&\Pr[Win]\\&\le\frac{(2q_{se}+q_{ex}+1)}{|G|}+\frac{1}{2}+\frac{(q_{se}+1)}{N}\\&+\frac{\varepsilon_{mac}(k_2,t,q_{se}+2q_{ex}+q_{re}+2)}{2}\\&+(q_{se}+q_{ex}+1)\cdot\varepsilon_{ddh}(k_1,t)\end{aligned}\qquad(24)$$

(4) can be obtained by substituting (24) for (3).

V. Conclusion

In this paper, we have proved its security of the LR-AKE protocol in the standard model. Our security analysis shows that the LR-AKE protocol is provably secure under the assumptions that DDH (Decisional Diffie-Hellman) problem is hard and MACs are selectively unforgeable against partially chosen message attacks (which is a weaker notion than being existentially unforgeable against chosen message attacks). Future works include the tight security proof of LR-AKE protocol and to design a more efficient AKE protocol in aspects of computational costs and memory size required mainly on client.

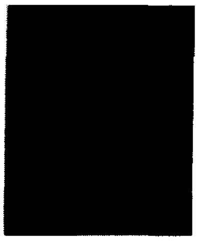
Reference

[1] S. H. Shin, K. Kobara, and H. Imai, "Leakage-Resilient Authenticated Key Establishment Protocols", In Proc. of ASIACRYPT 2003, LNCS 2894, pp. 155-172, Springer-Verlag, 2003.

[2] IEEE Std 1363-2000, "IEEE Standard

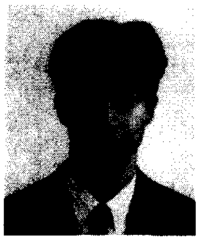
Specifications for Public Key Cryptography", Main Document, pp. 53-57, IEEE, August 29, 2000.
[3] IEEE P1363.2, "Standard Specifications for Password-based Public Key Cryptographic Techniques", Draft version 12, December 9, 2003.
[4] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange : Password-based Protocols Secure against Dictioinary Attacks", In Proc. of IEEE Symposium on Security and Privacy, pp. 72-84, 1992.
[5] Phoenix Technologies Inc., "Research Papers on Strong Password Authentication", available at http://www.integritysciences.com/links.html.
[6] P. MacKenzie, T. Shrimpton, and M. Jakobsson, "Threshold Password-Authenticated Key Exchange", In Proc. of CRYPTO 2002, LNCS 2442, pp. 385-400, Springer-Verlag, 2002.
[7] M. D. Raimondo and R. Gennaro, "Provably Secure Threshold Password-Authenticated Key Exchange", In Proc. of EUROCRYPT 2003, LNCS 2656, pp. 507-523, Springer-Verlag, 2003.
[8] IETF (Internet Engineering Task Force), "Secure Shell (secsh) Charter", available at http://www.ietf.org/html.charters/secsh-charter.html.
[9] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., 1996, available at http://wp.netscape.com/eng/ssl3/.
[10] IETF (Internet Engineering Task Force), "Transport Layer Security (tls) Charter", availabel at http://www.ietf.org/html.charters/tls-charter.html.
[11] D. Boneh, "The Decision Diffie-Hellman Problem", In Proc. of the Third Algorithmic Number Theory Symposium, 1998.
[12] A. Shamir, "How to Share a Secret", In Proc. of Communications of the ACM, Vol. 22(11), pp. 612-613, 1979.
[13] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks", In Proc. of EUROCRYPT 2000, LNCS 1807, pp. 139-155, Springer-Verlag, 2000.
[14] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution", In Proc. of CRYPTO '93, LNCS 773, pp. 232-249, Springer-Verlag, 1993.
[15] M. Naor and M. Yung, "Universal One-Way Hash Functions and Their Cryptographic Applications", In Proc. of STOC '98, pp. 33-43, 1998.
[16] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC : Keyed-Hashing for Message Authentication", IETF RFC 2104, 1997, available at http://www.ietf.org/rfc /rfc2104.txt.

〈著者紹介〉



SeongHan Shin Student Member

Feb. 2000 : B.S. degree in computer science from Pukyong National University, Korea
Feb. 2002 : M.S. degree in computer science from Pukyong National University, Korea
Oct. 2002~Present : Ph.D. candidate in information and communication engineering, The University of Tokyo, Japan
〈Research interests〉 information security, cryptography



Kazukuni Kobara

Mar. 1992 : B.E. degree in electrical engineering from Yamaguchi University, Japan
Mar. 1994 : M.E. degree in computer science and system engineering from Yamaguchi University, Japan
1994~Present : Research associate in the Institute of Industrial Science, The University of Tokyo, Japan
Mar. 2003 : Ph.D. degree in engineering from the University of Tokyo, Japan
2000~Present : Member of CRYPTREC
2003 : Vice president of WLAN security committee
〈Research interests〉 information security, cryptography

Hideki Imai

The B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Tokyo, Japan

1971~1992 : Professor of Yokohama National University, Japan
1992~Present : Full professor in the Inistitute of Industrial Science of the University of Tokyo, Japan
1999 : Honor doctor degree by Soonchunhyang University, Korea
2002 : Doctor honoris causa from the University of Toulon Var, France
He chaired several committees of scientific societies and many international conferences such as IEEE-ITW, IEEE-ISIT, AAECC, PKC, FSE and WPMC. He also was on the board of IEICE (1992~94, 1996~99), the IEEE Information Theory Society (IT-SOC, 1993~1998, 2000~present), International Association of Cryptologic Research (IACR, 1992~1994), Japan Sociery of Security Management (1988~present), and the Society of Information Theory and Its Applications (SITA, 1981~1997). He served as the present of SITA (1997), IEICE Engineering Sciences Society (1998~1999), a vice president of IT-SOC (2002~present), and as the chairman of CRYPTREC (Cryptography Techniques Research and Evaluation Committee of Japan, 2000~present).
〈Research interests〉 information theory, coding theory, cryptography, spread spectrum systems and their applications