Efficient verifier-based password-authenticated key exchange in the three-party setting

Jeong Ok Kwon\textsuperscript{a}, Ik Rae Jeong\textsuperscript{b}, Kouichi Sakurai\textsuperscript{c}, Dong Hoon Lee\textsuperscript{a,*}

\textsuperscript{a} Graduate School of Information Security CIST, Korea university, 1, 5-Ka, Anam-dong Sangbuk-ku, Seoul, 136-701 Korea
\textsuperscript{b} ETRI (Electronics and Telecommunications Research Institute) 161 Gajeong-dong, Yuseong-Gu, Daejeon, 305-700 Korea
\textsuperscript{c} Department of Computer Science and Communication Engineering, Kyushu University 6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-0053 Japan

Received 14 August 2006; accepted 1 December 2006

Abstract

In the last few years, researchers have extensively studied the password-authenticated key exchange (PAKE) in the three-party setting. The fundamental security goal of PAKE is security against dictionary attacks. The protocols for verifier-based PAKE are additionally required to be secure against server compromise. Some verifier-based PAKE schemes in the three-party setting have been suggested to solve the server compromise problem. Unfortunately, the protocols are vulnerable to an off-line dictionary attack. In this paper, we present an efficient verifier-based PAKE protocol for three-parties that is secure against known-key attacks and provides forward secrecy. To the best of our knowledge, the proposed protocol is the first secure three-party verifier-based PAKE protocol in the literature.

Keywords: Cryptology; Password-authenticated key exchange; Verifier-based; Dictionary attack

Contents

1. Introduction
1.1. PAKE in the three-party setting
1.2. Dictionary attacks with respect to the three-party PAKE
1.3. Two models with respect to the server-aided PAKE
1.4. Key secrecy with respect to the server-aided PAKE
1.5. Desirable security goals of the three-party PAKE
2. Our work in relation to prior work
3. Preliminaries
3.1. Decisional Diffie-Hellman assumption
3.2. Pseudorandom functions
3.3. Message authentication codes
4. A verifier-based PAKE protocol in the three-party setting
4.1. The VB–3PAKE protocol
4.1.1. Public information
4.1.2. Initialization
4.1.3. Session key agreement phase
4.1.4. Completeness

© 2006 Elsevier B.V. All rights reserved.

Keywords: Cryptology; Password-authenticated key exchange; Verifier-based; Dictionary attack
1. Introduction

To communicate securely over an insecure public network, it is essential that secret session keys be securely exchanged. The shared session key may be subsequently used to achieve some cryptographic goals such as confidentiality or data integrity. In the public-key based and symmetric-key based key exchange protocols, a party has to keep long random secret keys. However, it is difficult for a person to memorize a long random string; thus, a party uses an additional storage device to keep the random string. On the other hand, password-authenticated key exchange (PAKE) protocols allow two or more specified parties to share a secret session key using only a human-memorable password. Hence, PAKE protocols do not require that each party hold some devices such as smart cards or hardware tokens. From this point of view, PAKE provides convenience and mobility. Protocols for PAKE can be used in several environments especially in networks where a security infrastructure like PKI (Public-Key Infrastructure) is not deployed. Because PAKE protocols provide a new and unique way to authenticate parties and derive high-quality cryptographic keys from low-grade passwords, PAKE has received significant attention.

1.1. PAKE in the three-party setting

Designing of secure and efficient PAKE protocols for two-party use, especially using the “client-server” model, has been extensively pursued in the last few years. Most of those designs assume that the two parties have a pre-shared password. However, a two-party PAKE protocol using the same password is not scalable in the sense that a user may want to communicate securely with many other users who have not shared the same password. If a user shares a password with the other user, the number of passwords that the user has to memorize linearly increases with the number of possible partners. PAKE, in the three-party setting, surmounts all the above mentioned problems. In this setting, each user only shares a password with a trusted server. The trusted server authenticates two users and helps the users with different passwords share a common session key. The main advantage of this solution is that it requires each user only to remember a single password with the trusted server. Consequently, three-party PAKE protocols can limit the number of passwords that each user must memorize. This seems to be a more practical scenario in the real world than two-party PAKE solutions. However, the server has to participate during the protocol run to help the two users share a session key.

1.2. Dictionary attacks with respect to the three-party PAKE

Comparisond to other security models, the most distinguishable characteristic of the PAKE security model is that the model must incorporate protection against dictionary attacks. Dictionary attacks are possible because of the low entropy of the password space. In practice, a password consists of 4 or 8 characters, such as a natural language phrase, to be easily memorized. The set of these probable passwords is small. As a consequence, there exists a relatively small dictionary. Usually dictionary attacks are classified into two classes: on-line and off-line dictionary attacks. In on-line dictionary attacks, an adversary attempts to use a guessed password by participating in a key exchange protocol. If the protocol run fails, the adversary initiates a new protocol run using another guessed password. These on-line attacks require the participation of the server. In off-line dictionary attacks, an adversary selects a password from a dictionary and verifies his guess in an off-line manner, i.e., the adversary uses only recorded transcripts from a successful run of the protocol. So these off-line attacks are undetectable.

On-line dictionary attacks are always possible, but these attacks cannot become a serious threat if the on-line attacks can be easily detected and thwarted by counting access failures. In the server-aided PAKE protocols, however, we must more carefully consider on-line dictionary attacks, because a malicious insider may launch on-line dictionary attacks indiscernibly using the server as a password verification oracle. If a failed guess cannot be detected and logged by the server, the attacks are called undetectable on-line dictionary attacks [4]. To better understand the power of these attacks, we present an example of undetectable on-line dictionary attacks on the existing three-party PAKE protocol, C2C-PAKE [6].

\[
\begin{align*}
\text{Alice} & \rightarrow \text{Server} : \text{ID}(A), \text{ID}(B), E_{pw_B}(g^r) \\
\text{Server} & \rightarrow \text{Bob} : \text{ID}(B), E_{pw_B}(g^s) \\
\text{Bob} & \rightarrow \text{Server} : \text{ID}(A), E_{pw_A}(g^t) \\
\text{Server} & \rightarrow \text{Alice} : \text{ID}(A), E_{pw_A}(g^x)
\end{align*}
\]

C2C-PAKE: Alice and Bob are users, ID(A) and ID(B) are identities, \(pw_A\) and \(pw_B\) are passwords of Alice and Bob, respectively, and \(E_X\) is a symmetric encryption with \(X\).

Suppose that the goal of a malicious user, Alice, is to discover the password of Bob. Alice selects a random number \(a \in \mathbb{Z}_p^*\) and sends (ID(A),ID(B), \(A1 = E_{pw_A}(g^a)\)) as its message to the server. Upon receiving this message, the server may send (ID(B), \(B2 = E_{pw_B}(g^{as})\)) to Bob. Alice intercepts and blocks this message. Alice selects a candidate password \(pw'_B\) from a dictionary of passwords and sends (ID(A), \(B1 = E_{pw'_B}(g^as)\)) as Bob’s message to the server. Upon receiving this message, the
server may send \((ID(A), A_2 = E_{pw_a}(D_{pw_a}(B_1)))\) to Alice. Alice compares \(D_{pw_a}(A_2)\) and \(D_{pw_b}(B_2)\). If they equal, Alice has guessed the correct password \(pw_A = pw_B\). Otherwise, Alice starts a new session with the server using another guessed password. This process continues for the remaining candidate passwords until the correct password is found. Notice that no participation of Bob is required and a failed guess of Alice is never noticed by the server and Bob. Alice participates in the protocol legally and unpredictably many times to get \(pw_B\). If this kind of attack succeeds on a PAKE protocol, an adversary is able to find the correct passwords of users and hence the attacker is able to access everything allowed to the honest users which breaks the overall security of the key exchange protocol. To prevent undetectable on-line attacks, a server-aided PAKE protocol must provide a method by which the server can distinguish an honest run of the protocol from a malicious one.

Even if there exists a little bit of redundancy information in the protocol messages, an adversary may perform an off-line dictionary attack by using the redundancy as a verifier for checking whether a guessed password is correct or not. We also have to consider insider attacks by a malicious user who attempts to perform an off-line dictionary attack on the other user’s password using his information. The main security goal of PAKE schemes is to restrict the adversaries so that they attempt detectable on-line dictionary attacks only. If a PAKE scheme is secure, then an adversary cannot obtain any advantage in guessing the passwords and the session keys of users through off-line dictionary attacks and undetectable on-line dictionary attacks.

### 1.3. Two models with respect to the server-aided PAKE

The server-aided PAKE protocols can be classified into two types of models according to the difference of knowledge of a shared password between each user and a server: symmetric and asymmetric (or verifier-based) models. In a symmetric model, each user and a server use the same knowledge related with a password to authenticate each other. In a verifier-based model, each user has a password; whereas a server has an image (called a verifier) of the password which is computed using a one-way function instead of the password itself.

If in a symmetric model, the server is compromised, an adversary with the server’s password file can immediately masquerade as a legitimate user by using the password in the password file (without executing any off-line dictionary attack). To better understand the damage of the server compromise in a symmetric model, consider the above described C2C-PAKE protocol in a symmetric model. In C2C-PAKE, it easy to see that if a server compromise occurs, an adversary who can access the compromised passwords, can immediately masquerade as a legitimate user, Alice or Bob, to the server since the adversary knows the password \(pw_A\) or \(pw_B\).

PAKE protocols in a verifier-based model are designed to limit the damage of a server compromise. In a verifier-based protocol, a server has verifiers of the passwords instead of the passwords themselves, so the server compromise does not directly reveal the passwords, themselves. The server compromise in a verifier-based model still allows off-line dictionary attacks against the verifiers of the passwords, but an attacker has to perform some additional computations to find out the passwords of the users. The time required to perform the computations may allow the server to acknowledge the server compromise to the users and to reduce the damage of this server compromise. Therefore, the main idea in designing secure verifier-based PAKE protocols is to force an adversary who gets a password file and wants to impersonate a user in the password file, in performing an off-line dictionary attack on the password file. The difficulty of off-line dictionary attacks on the password file depends on the difficulty of finding the original passwords from the verifiers.

### 1.4. Key secrecy with respect to the server-aided PAKE

One of the most basic security requirements of a key exchange protocol is key secrecy [2] which guarantees that no computationally bounded adversary can learn anything about the session keys shared between honest users by eavesdropping or sending messages of their choice to the users in the protocol. It is necessary that the key secrecy also be preserved against the server which behaves honestly but in a curious manner. That is, the server should not learn anything about the session keys of the users by eavesdropping. This is true even if the server helps two users establish a session key between them. This is the most basic notion of the key exchange.

The advantage of an adversary is measured by an experiment as follows: (1) an adversary \(A\) executes a protocol run between parties; (2) the adversary may insert a new message, modify a transmitted message, or delete a transmitted message; (3) the adversary selects a test session and is given a string; (4) the adversary determines whether the string is a session key or a random string. The advantage of the adversary, \(Adv_A\), is the difference of probabilities between the probability that the adversary outputs 1 when the string is a session key and the probability that the adversary outputs 1 when the string is a random string. A key exchange scheme provides key secrecy, if the advantage of any adversary is bounded by a negligible function \(\epsilon\).

### 1.5. Desirable security goals of the three-party PAKE

The importance of the following attributes depends on the real applications. Forward Secrecy means that even with passwords of parties any adversary does not learn any information about previous session keys which are successfully established between honest parties without any interruption. A PAKE protocol is secure against known-key attacks if a compromise of multiple session keys for sessions other than the one that should be kept does not affect its key secrecy. This notion of security means that session keys are computationally independent from each other. A bit more formally, this security

---

1. A function \(\epsilon\) is negligible if for all polynomial functions \(f\), there exists an \(N\) such that \(\epsilon(n) < 1/f(n)\) for all \(n > N\).
protections against “Denning-Sacco” attacks [1] involving the compromise of multiple session keys (for sessions other than the one whose secrecy must be guaranteed). Security against known-key attacks implies that an adversary cannot gain the ability to perform off-line dictionary attacks on the passwords from the compromised session keys which are successfully established between honest parties.

2. Our work in relation to prior work

PAKE in the three-party setting has been extensively studied in the last few years [3–5,7,8,10,12,13]. Only the scheme in [10] among these studies is a verifier-based PAKE scheme in the three-party setting. In [9], a two-party PAKE scheme has been suggested. However, the protocol in [9] is not only vulnerable to three-party setting. In [9], a two-party PAKE scheme has been suggested. However, the protocol in [9] is not only vulnerable to the 3,2,2 requires the use of two verifiers, v1,2 and v2,2, to each user U, and the server but in fact v1,2 does not use anywhere. Thus the verifier v1,2 has to be eliminated in the protocol description.

3. Preliminaries

Before presenting our verifier-based PAKE protocol for three-party, we state preliminaries for the protocol.

3.1. Decisional Diffie-Hellman assumption

Let G = (g) be any finite cyclic group of prime order q. The DDH problem is defined as follows: given a triple (U, V, W), determine that the triple is a Diffie-Hellman triple (g^a, g^b, g^ab) or a random triple (g^a, g^b, g^c). The advantage of an algorithm A, Adv_{DDH}^\mathcal{A}(t), running in time t is ε, if

\[ |Pr[a, b \rightarrow Z_q : A(g^a, g^b, g^ab) = 1] - Pr[a, b \rightarrow Z_q : A(g^a, g^b, g^c) = 1]| \geq \epsilon. \]

We say the DDH assumption holds in G if no probabilistic polynomial time algorithm A can solve the DDH problem with non-negligible advantage. We let Adv_{DDH}^\mathcal{A}(t) denote the maximum advantage which is over all adversaries As running in time at most t.

3.2. Pseudorandom functions

Let F : Keys(\mathcal{F}) × D → R be a family of functions, and g: D → R a random function. A is an algorithm that takes an oracle access to a function and returns a bit. We consider two experiments:

\[
\text{Exp}_{\mathcal{F}, A}^{\text{prf}-1} = \begin{cases} \text{1} & \text{if } A(K) = g, \text{ and } A(G) = 1 \\ 0 & \text{otherwise} \end{cases},
\]

\[
\text{Exp}_{\mathcal{F}, A}^{\text{prf}-0} = \begin{cases} \text{1} & \text{if } A(K) \neq g, \text{ and } A(G) = 1 \\ 0 & \text{otherwise} \end{cases}.
\]

Let \mathcal{F} : Keys(\mathcal{F}) × D → R be a family of functions, and g: D → R a random function. A is an algorithm that takes an oracle access to a function and returns a bit. We consider two experiments:

\[
K \leftarrow \text{Keys(\mathcal{F})}, \quad g \leftarrow \text{Rand}^{0\rightarrow1},
\]

\[
d \leftarrow A^\mathcal{F}(1^\kappa),
\]

\[
return d
\]

\[
\text{The advantage of an adversary } \mathcal{A} \text{ is defined as follows:}
\]

\[
\text{Adv}_{\mathcal{F}, A}^{\text{prf}} = Pr[\text{Exp}_{\mathcal{F}, A}^{\text{prf}-1} = 1] - Pr[\text{Exp}_{\mathcal{F}, A}^{\text{prf}-0} = 1].
\]

The advantage function is defined as follows:

\[
\text{Adv}_{\mathcal{F}}(\kappa, q, \mu) = \max_{\mathcal{A}} \{\text{Adv}_{\mathcal{F}, A}^{\text{prf}}\},
\]

where \mathcal{A} is any adversary with time complexity T making at most q oracle queries and the sum of the length of these queries being at most \mu bits. The scheme \mathcal{F} is a secure pseudorandom function family if the advantage of any adversary \mathcal{A} with time complexity polynomial in \kappa is negligible.

3.3. Message authentication codes

A message authentication code (MAC) algorithm consists of three algorithm, M = (KEY.G, MAC.G, MAC.V). KEY.G generates a key k_{mac}. Given k_{mac}, MAC.G computes a tag
\( \tau = MAC_{k_{mac}}(M) \) for a message \( M \). \( MAC.V \) verifies a message-tag pair using key \( k_{mac} \), and returns 1 if the tag is valid or 0 otherwise.

In defining the security of an MAC we use the standard definition of strong unforgeability under adaptive chosen-message attack. Namely, let \( M \) be an MAC scheme and \( A \) be an adversary, and consider the following experiment:

\[
Exp_{A, M}^{suf}(k) = \begin{cases} 
    1 & \text{if } MAC.V_{k_{mac}}(M, \tau) = 1 \text{ and oracle MAC.suf never returned } \tau \text{ on input } M \text{ then return } 1 \\
    0 & \text{else return } 0
\end{cases}
\]

The advantage of an adversary \( A \) is defined as: \( Adv_{A, M}^{suf}(k_{mac}) = Pr[Exp_{A, M}^{suf}(k) = 1] \). We say that \( M \) is strongly unforgeable (SUF-secure) if \( Adv_{A, M}^{suf}(k) \) is negligible for all ppt (probabilistic polynomial time) algorithms \( A \). When we are interested in a concrete security analysis, we drop the dependence on \( k \) and say that \( M \) is \((t, q, \epsilon)\)-SUF-secure if \( Adv_{A, M}^{suf} \leq \epsilon \) for all \( A \) running in time \( t \) and making at most \( q \) queries to its \( M \) oracle. (We remark that allowing \( N \) queries to an oracle MAC.V_{k_{mac}}(\cdot, \cdot) \) cannot increase the advantage of an adversary by more than a factor of \( N \).

4. A verifier-based PAKE protocol in the three-party setting

We now present our verifier-based protocol VB−3PAKE for PAKE in the three-party setting. In this paper, we assume that the parties can transmit messages simultaneously. That is, we assume a duplex channel which is useful in reducing the round complexity and/or the communication complexity. The Internet, for example, is a duplex channel. An example of an execution of VB−3PAKE is shown in Fig. 1.

\[
\begin{align*}
\langle \text{Public information: } & G, p, q, g_1, g_2, H, M, F \rangle \\
User A & \quad \text{Server } S \\
pw_A & \quad v_{A,1} = g_1^H(A||S||pw_A), v_{A,2} = g_1^H(S||v_{A,1}) \\
v_{B,1} = g_1^H(B||S||pw_B), v_{B,2} = g_1^H(S||v_{B,1}) \\
\end{align*}
\]

**Round 2**

\[
\begin{align*}
x_A & \in R \mathbb{Z}_q^* \\
y_A, y_B, z_A, z_B & \in R \mathbb{Z}_q^* \\
x_A = g_1^{x_A} \cdot v_{A,2} & \rightarrow \quad Y_A = g_1^{y_A} \cdot v_{A,1}^z, Z_A = g_1^{z_A} \cdot v_{A,2} \\
x_B & \in R \mathbb{Z}_q^* \\
y_B = g_1^{y_B} \cdot v_{B,1}^z, Z_B = g_1^{z_B} \cdot v_{B,2} & \rightarrow \\
\end{align*}
\]

**Round 3**

\[
\begin{align*}
k_A & = (g_1^{y_A})^{x_A} \\
k_B & = (g_1^{y_B})^{x_B} \\
k_A & = (g_1^{z_A})^{y_A}, k_B = (g_1^{z_B})^{y_B} \\
\tau_{A,S} & = MAC.G_{k_A}(A||S||X_A||Y_A\|Z_A) \rightarrow \quad \tau_{B,S} = MAC.G_{k_B}(B||S||X_B||Y_B\|Z_B)
\end{align*}
\]

**Round 4**

\[
\begin{align*}
MAC.V_{k_A}(\tau_{A,S}) & \overset{?}{=} 1; MAC.V_{k_B}(\tau_{B,S}) \overset{?}{=} 1 \\
S_A & = (g_1^{x_B})^{x_A}; \tau_{S,A} = MAC.G_{k_A}(A||B\|S) \\
S_B & = (g_1^{x_A})^{x_B}; \tau_{S,B} = MAC.G_{k_B}(B||A\|S_B) \\
\end{align*}
\]

**Key computation**

\[
\begin{align*}
MAC.V_{k_A}(\tau_{S,A}) & \overset{?}{=} 1 \quad MAC.V_{k_B}(\tau_{S,B}) \overset{?}{=} 1 \\
K_A & = (S_A)^{x_A} \quad K_B = (S_B)^{x_B} \\
sk_A & = FA_{k_A}(A||S||B) \\
sk_B & = FA_{k_B}(A||S||B)
\end{align*}
\]

Fig. 1. An example of an execution of VB−3PAKE.
4.1. The VB-3PAKE protocol

4.1.1. Public information

Let $G$ be a finite cyclic group of order $q$ for which the decisional Diffie-Hellman problem is hard. The terms, $g_1$ and $g_2$, are generators of $G$, where $g_1$ and $g_2$ must be generated so that their discrete logarithmic relation cannot be known. Let $M$ a strongly unforgeable MAC algorithm and let $H: \{0, 1\}^* \to \{0, 1\}^l$ be a hash function. Let $F$ be a secure pseudorandom function family.

4.1.2. Initialization

We assume that each user $A$ and the server $S$ have shared the verifiers $v_{A,1} = g_{1}^{H(A||S||pw_A)} \mod p$ and $v_{A,2} = g_{2}^{H(A||S||pw_A)} \mod p$ for password $pw_A$ and the public information.

4.1.3. Session key agreement phase

Assume $A$ wants to establish a session key with $B$.

Round 1 The initiator $A$ broadcasts a session key with $B$.

Round 2

1. $A$ chooses a random number $x_{A} \in \mathbb{Z}_{q}^{*}$, computes $X_{A} = g_{1}^{x_{A}} \cdot v_{A,2} \mod p$, and sends $(A, X_{A})$ to $S$.
2. $B$ analogously computes $X_{B} = g_{2}^{x_{B}} \cdot v_{B,2} \mod p$, and sends $(B, X_{B})$ to $S$.
3. $S$ selects random numbers $y_{A}, y_{B} \in \mathbb{Z}_{q}^{*}$, computes $Z_{A} = g_{1}^{y_{A}} \cdot v_{A,2} \mod p$ and $Y_{A} = g_{1}^{y_{A}} \cdot v_{A,1} \mod p$, and sends $(S, Y_{A}, Z_{A})$ to $A$. (Analogously $S$ sends $(S, Y_{B}, Z_{B})$ to $B$.)

Round 3

1. Upon receiving $(S, Y_{A}, Z_{A})$, $A$ computes $T_{A} = (Z_{A} / v_{A,2})^{H(A||S||pw_A)} \mod p$ and $k_{A} = (Y_{A} / T_{A})^{x_{A}} \mod p$.
2. Upon receiving $(S, Y_{B}, Z_{B})$, $B$ analogously computes $k_{B} = (Y_{B} / T_{B})^{x_{B}} \mod p$.
3. Upon receiving $(A, X_{A})$, $S$ computes $k_{A} = (X_{A} / v_{A,2})^{x_{A}} \mod p$. (Analogously $S$ computes $k_{B} = (X_{B} / v_{B,2})^{x_{B}} \mod p$.)
4. $A$ computes $\tau_{A,S} = MAC.G_{k_{A}}(A||S||X_{A}||Y_{A}||Z_{A})$ and sends it to $S$. ($B$ analogously computes $\tau_{B,S}$ and sends it to $S$.)

Round 4

1. Upon receiving $\tau_{A,S}$, $S$ computes $MAC.V_{k_{A}}(\tau_{A,S})$. $S$ terminates if MAC.V returns 0 or moves to the next phase otherwise. ($S$ analogously checks the validity of $\tau_{B,S}$ using $k_{B}$.)
2. $S$ selects a random number $x_{S} \in \mathbb{Z}_{q}^{*}$, computes $S_{A} = g_{1}^{x_{S} \cdot k_{A}} \mod p$ and $\tau_{S,A} = MAC.G_{k_{S}}(A||S||B||S_{A})$, and sends $(S_{A}, \tau_{S,A})$ to $A$.
3. $S$ computes $S_{B} = g_{1}^{x_{S} \cdot k_{B}}$ and mod $p$ and $\tau_{S,B} = MAC.G_{k_{S}}(A||S||B||S_{B})$, and sends $(S_{B}, \tau_{S,B})$ to $B$.

Key computation

Upon receiving $(S_{A}, \tau_{S,A})$, $A$ terminates if $MAC.V_{k_{A}}(\tau_{S,A})$ returns 0 or computes $K_{A} = (S_{A})^{x_{S}} \mod p$ and the session key $sk_{A} = F_{K_{A}}(A||S||B)$ otherwise. ($B$ analogously computes $K_{B} = (S_{B})^{x_{S}} \mod p$ and $sk_{B} = F_{K_{B}}(A||S||B)$).

4.1.4. Completeness

In an honest execution of the protocol, both users calculate an identical session key as $sk = F_{K}(A||S||B)$, where $K = g_{1}^{x_{S} \cdot k_{S}^{x_{S}} \cdot k_{B}^{x_{S}}} \mod p$.

4.2. Efficiency and security analysis

4.2.1. Efficiency analysis

We analyze the computational, communicational, and round complexities of VB-3PAKE, which need for deployment in a practical circumstance. VB-3PAKE requires four rounds and four modular exponentiations per each user. It also requires a single message in $G$ and a single MAC message that each user sends, which are reasonable costs for practical use.

4.2.2. Security analysis

We examine the security against conventional attacks. We consider an offline attacker $E$ that has full control over the network and is not a member of the protocol. We also consider two malicious inside attackers: $A$ (or $B$) is a legal user of the protocol and $S$ is the legal (trusted) server that behaves honestly but in a curious manner. They are all probabilistic polynomial Turing machines.

(1) VB-3PAKE is secure against off-line dictionary attacks via the difficulty of the decisional Diffie-Hellman (DDH) problem. That is, in order to successfully find users’ correct passwords, $E$ has to solve the DDH problem. $A$ also has to solve the DDH problem to find $B$’s password. However, it is believed that there is no ppt algorithm that can solve the DDH problem.

(2) VB-3PAKE is designed to be secure against undetectable on-line dictionary attacks since $E$ and $A$ cannot use the server as a password verification oracle. The server verifies that the user really knows the password by using MAC.V in Round 3. If the MAC verification is failed, the server will notice whose password is being a target of undetectable on-line dictionary attacks and it is at a crisis. If the number of failing tries exceeds a predefined threshold, the server reacts and informs the target user to stop any further use of the password and to change the password into a new one. To generate a valid message-tag pair, there are only three ways: an adversary successfully guesses a correct password at once or after a small number of guesses (but it is generally very low with respect to the size of the password space), solves the DDH problem or breaks the MAC algorithm.

VB-3PAKE is secure against detectable on-line dictionary attacks since the users are able to detect a failed guess. If the MAC verification in the key computation phase is failed, a user will notice that his/her password is being a target of on-line dictionary attacks and it is at a crisis. After a small amount of detection of failures the user may stop any further use of the password and change the password into a new one.

(2) If the DDH assumption holds and $F$ is a secure pseudo random function family, VB-3PAKE provides key secrecy. Suppose to the contrary that $E$ or $S$ can determine whether the string is a session key or a random string with a non-negligible probability. This means that the adversary has solved the DDH problem or has broken the pseudo randomness of the pseudo
random function family $\mathcal{F}$. This is a contradiction to the fact that the DDH assumption holds and $\mathcal{F}$ is a secure pseudo random function family.

(3) If the DDH assumption holds, $\text{VB-3PAKE}$ provides forward secrecy. That is, even if $\text{pw}_A$, $(v_{A,1}, v_{A,2})$ and $\text{pw}_B$, $(v_{B,1}, v_{B,2})$ are compromised, $E$ cannot break the key secrecy of previously established session keys because we are not able to solve the DDH problem.

(4) We say a protocol is secure against server compromise attacks if an adversary who steals the password file from the server, still cannot impersonate a legitimate user without performing dictionary attacks on the password file (of course, a server compromise still allows dictionary attacks). $\text{VB-3PAKE}$ is secure against server compromise attacks. Even if the password file of the server is compromised, $E$ posing as a user $A$ to $S$ cannot compute $g^x_A$ from $YA$ unless the adversary successfully guesses a correct password at once or after a small number of guesses. However, this probability is generally very low according to the size of the password space. Thus $E$ can not pass the MAC verification in Round 3. In order to successfully pass the MAC verification, $E$ has to execute an off-line dictionary attack on the compromised password file.

(5) $\text{VB-3PAKE}$ is secure against known-key attacks since a session key is constructed by ephemeral random numbers. That is, $x_A$, $x_B$ and $x_S$ are randomly and independently selected in each session. Thus, the compromised session keys are not helpful for an adversary in guessing other unknown session keys. $A$ may gain the ability to perform the off-line dictionary attacks on $\text{pw}_A$ from the compromised session keys, if one can break the strong unforgeability of the underline MAC algorithm. $E$ may gain the ability to perform the off-line dictionary attacks on the honest users’ passwords, if one can break the strong unforgeability of the underline MAC algorithm and the pseudo randomness of the pseudo random function family $\mathcal{F}$.

Acknowledgement

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IIITA (Institute of Information Technology Advancement) (IIITA-2006-(C1090-0603-0025)).

References


Jeong Ok Kwon received the B.S. degree in Computer Science from Dongduk Woman’s University, Korea in 2000. She received the M.S. and Ph.D. degrees in Information Security from Korea University, Korea, in 2004 and 2007, respectively. Her research interests include cryptography and information security.

Ik Rae Jeong received the B.S. and M.S. degrees in Computer Science from Korea University, Korea, in 1998 and 2000, respectively. He received the Ph.D. degree in Information Security from Korea University in 2004. He is currently a senior engineer at ETRI (Electronics and Telecommunications Research Institute) in Korea. His research interests include cryptography and theoretical computer science.

Kouchi Sakurai received the B.S. degree in Mathematics from the Faculty of Science, Kyushu University, Japan and the M.S. degree in Applied Science from the Faculty of Engineering, Kyushu University in 1986 and 1988, respectively. He had been engaged in the research and development of cryptography and information security at the Computer and Information Systems Laboratory at Mitsubishi Electric Corporation from 1988 to 1994. He received the Ph.D. degree in engineering from the Faculty of Engineering, Kyushu University in 1993. Since 1994 he has been working for the Department of Computer Science of Kyushu University as an associate professor, and now he is a full professor since 2002. His current research interests are in cryptography and information security. Dr. Sakurai is a member of the Information Processing Society of Japan, the Mathematical Society of Japan, ACM and the International Association for Cryptologic Research.
Dong Hoon Lee received the B.S. degree in Economics from Korea University, Korea, in 1984. He received the M.S. and Ph.D. degrees in Computer Science from the University of Oklahoma, USA, in 1988 and 1992, respectively. He is currently a full professor at the Korea University. His research interests include cryptography and information security.