Password-Based Key Exchange Protocols for Cross-Realm[*]

Lee, Young Sook[**]

Cross-Realm 환경에서 패스워드기반 키교환 프로토콜

이 영 숙

<Abstract>

Authentication and key exchange are fundamental for establishing secure communication channels over public insecure networks. Password-based protocols for authenticated key exchange are designed to work even when user authentication is done via the use of passwords drawn from a small known set of values. There have been many protocols proposed over the years for password authenticated key exchange in the three-party scenario, in which two clients attempt to establish a secret key interacting with one same authentication server. However, little has been done for password authenticated key exchange in the more general and realistic four-party setting, where two clients trying to establish a secret key are registered with different authentication servers. In fact, the recent protocol by Yeh and Sun seems to be the only password authenticated key exchange protocol in the four-party setting. But, the Yeh-Sun protocol adopts the so called "hybrid model", in which each client needs not only to remember a password shared with the server but also to store and manage the server's public key. In some sense, this hybrid approach obviates the reason for considering password authenticated protocols in the first place; it is difficult for humans to securely manage long cryptographic keys. In this work, we introduce a key agreement protocol and a key distribution protocol, respectively, that requires each client only to remember a password shared with its authentication server.

Key Words : Key exchange, Authentication, Password, Public key, Cross-realm

Ⅰ. 서론

Password-based protocols for authenticated key exchange are designed to work even when user authentication is done via the use of passwords drawn from a small known set of values. There have been many protocols proposed over the years for password authenticated key exchange in the three-party scenario, in which two clients attempt to establish a secret key

* This work was supported by Howon University in 2009.
** Department of Cyber Investigation Police, Howon Univresity

interacting with one same authentication server [1, 13, 15, 17, 19]. However, little has been done for password authenticated key exchange in the more general and realistic four-party setting, where two clients trying to establish a secret key are registered with different authentication servers [12, 21]. In fact, the recent protocol by Yeh and Sun seems to be the only password authenticated key exchange protocol in the four-party setting [20]. But, the Yeh-Sun protocol adopts the so called "hybrid model", in which each client needs not only to remember a password shared with the server but also to store and manage the server's public key [10]. In some sense, this hybrid approach obviates the reason for considering password authenticated protocols in the first place; it is difficult for humans to securely manage long cryptographic keys. In this work, we propose new protocols designed carefully for four-party password authenticated key exchange that requires each client only to remember a password shared with its authentication server.

From the viewpoint of the session key creation, key exchange protocols can be broadly subdivided into two categories: key distribution protocols and key agreement protocols. In key distribution protocols (e. g., "Star Based System" of [6]), the session key is created by one party and securely transmitted to other parties. In key agreement protocols, more than one party contributes information to generate the common session key. In this paper, we introduce a key agreement protocol [16] and a key distribution protocol, respectively.

The rest of this paper is organized as follows. We begin by setting up some notation and definitions in Section 2. We continue in Section 3 with the description of our proposed four-party password authenticated key agreement protocol [16] and analyze the security of the proposed protocol. Then in Section 4, we present the description and the security analysis of the proposed password authenticated key distribution protocol. Finally, we conclude this work in Section 5.

Ⅱ. Protocol Preliminaries

We begin with the requisite definitions. There are four entities involved in the protocol: two clients A and B, and two authentication servers SA and SB respectively of A and B. We denote by IDA, IDB, $IDSA$, and $IDSB$, the identities of A, B, SA, and SB, respectively.

2.1 Computational Diffie-Hellman (CDH) Assumption.

Let g be a fixed generator of the finite cyclic group Z_p^*. Informally, the CDH problem is to compute gab given ga and gb, where a and b were drawn at random from {1, ..., $|Z_p^*|$}. Roughly stated, Z_p^* is said to satisfy the CDH assumption if solving the CDH problem in Z_p^* is computationally infeasible for all probabilistic polynomial time algorithms.

2.2 Symmetric Encryption Scheme.

A symmetric encryption scheme is a triple of polynomial time algorithms $\Gamma = (K, \varepsilon, D)$ such that:

- The key generation algorithm K is a randomized algorithm that returns a key k Let Keys(Γ) be the set of all keys that have non-zero probability of being output of K.

- The encryption algorithm E takes as input a key $k \in Keys(\Gamma)$ and a plaintext $m \in \{0,1\}^*$. It returns a ciphertext $\varepsilon_k(m)$ of m under the key k. This algorithm might be randomized or stateful.

- The deterministic decryption algorithm D takes as input a key $k \in Keys(\Gamma)$ and a purported ciphertext $c \in \{0,1\}^*$. It returns $D_k(c)$, which is a plaintext $m \in \{0,1\}^*$ or a distinguished symbol $\perp$. The return value $\perp$ indicates that the given ciphertext c is invalid for the key k.

We say, informally, that a symmetric encryption scheme Γ is secure if it ensures confidentiality of messages under chosen-ciphertext attack (CCA) and guarantees integrity of ciphertexts [18]. As shown in [2, 14], this combination of security properties implies indistinguishability under CCA which, in turn, is equivalent to non-malleability [9] under CCA.

2.3 Signature Scheme.

A digital signature scheme is a triple of algorithms $\Sigma = (\varsigma, S, V)$ such that:

- The probabilistic key generation algorithm G, on input a security parameter 1^l, outputs a pair of matching public and private keys (PK, SK).

- The signing algorithm S is a probabilistic polynomial time algorithm that, given as input a message m and a key pair (PK, SK), outputs a signature σ of m.

- The verification algorithm V is a polynomial time algorithm that on input (m, σ, PK), outputs 1 if σ is a valid signature of the message m with respect to PK, and 0 otherwise.

We say that a signature scheme Σ is secure if the probability of succeeding with an existential forgery under adaptive chosen message attack [11] is negligible for all probabilistic polynomial time attackers.

2.4 Initialization.

During some initialization phase, two servers SA and SB agree on the following public parameters: a large prime p, a generator g of Z_p^* satisfying the CDH assumption, a one-way hash function H, a secure symmetric encryption scheme $\Gamma = (K, \varepsilon, D)$, and a secure signature scheme $\Sigma = (\varsigma, S, V)$. In addition, public/private key pairs are generated for each server by running the key generation algorithm $\varsigma(1^l)$. We denote by (PK_X, SK_X) the public/private keys of the server X for $X \in \{SA, SB\}$. As part of the initialization, the client A (resp. B) registers with server SA (resp. SB), by choosing PW_A (resp. PW_B) and sending it to the server via a secure channel.

Ⅲ. A Four-party Password Authenticated Key Agreement Protocol(PAKA)

In this section we present a new four-party password authenticated key agreement protocol PAKA [16] in which two clients wishing to agree on a session key do not need to store any public key of their authentication server but only need to share a short, easy-to-remember password with the server. In describing the protocol PAKA, we will omit 'mod p' from expressions for notational simplicity. The PAKA protocol is outlined in <Fig. 1> and a more detailed description is as follows:

3.1 Description of PAKA Protocol

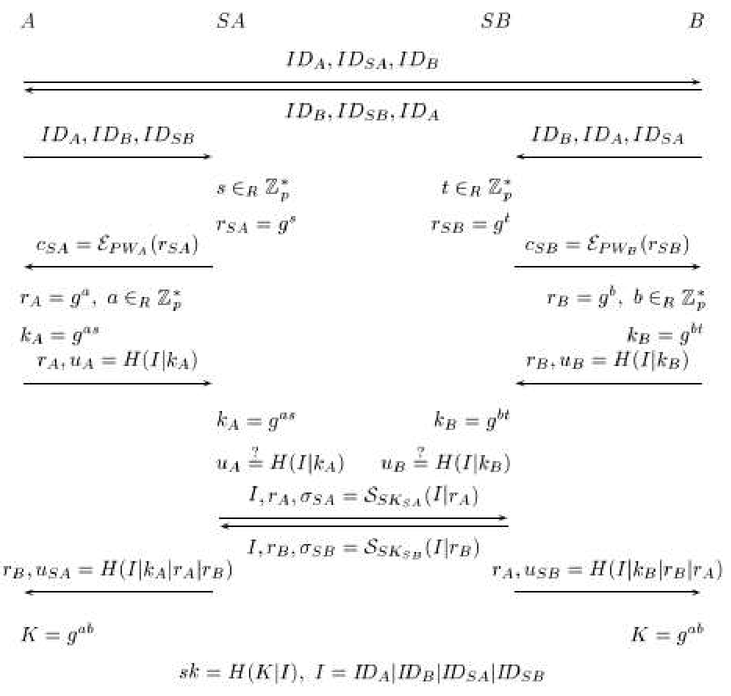
Step 1. Two clients A and B first need to inform each other of their respective authentication server. To this end, A sends to B the message $\langle ID_A, ID_{SA}, ID_B \rangle$ and B sends to A the message $\langle ID_B, ID_{SB}, ID_A \rangle$.

Step 2. The clients request the assistance of their respective server in establishing a session key between them. Client A (resp. B) does this by sending the message $\langle ID_A, ID_B, ID_{SB} \rangle$ (resp. $\langle ID_B, ID_A, ID_{SA} \rangle$) to the server SA (resp. SB).

Step 3. Server SA chooses a random number $s \in R$ Z_p^*, computes $r_{SA} = g^s$ and $c_{SA} = \varepsilon_{PW_A}(r_{SA})$, and sends the ciphertext c_{SA} to A. Similarly, server SB chooses a random number $t \in R$ Z_p^*, computes $r_{SB} = g^t$ and $c_{SB} = \varepsilon_{PW_B}(r_{SB})$, and sends the ciphertext c_{SB} to B.

Step 4. After receiving c_{SA}, client A recovers r_{SA} by decrypting c_{SA}, i. e., $r_{SA} = DPWA(c_{SA})$, and chooses a random number $a \in R$ Z_p^*. Given r_{SA} and a, client A computes the one time key k_A to be shared with the server SA as

$$k_A = g^{as} = (r_{SA})^a.$$



A
SA
SB
B
ID_A, ID_{SA}, ID_B
ID_B, ID_{SB}, ID_A
ID_A, ID_B, ID_{SB}
ID_B, ID_A, ID_{SA}
$s \in_R \mathbb{Z}_p^*$
$r_{SA} = g^s$
$t \in_R \mathbb{Z}_p^*$
$r_{SB} = g^t$
$c_{SA} = \mathcal{E}_{PW_A}(r_{SA})$
$c_{SB} = \mathcal{E}_{PW_B}(r_{SB})$
$r_A = g^a,\ a \in_R \mathbb{Z}_p^*$
$k_A = g^{as}$
$r_A, u_A = H(I|k_A)$
$r_B = g^b,\ b \in_R \mathbb{Z}_p^*$
$k_B = g^{bt}$
$r_B, u_B = H(I|k_B)$
$k_A = g^{as}$
$k_B = g^{bt}$
$u_A \overset{?}{=} H(I|k_A)$
$u_B \overset{?}{=} H(I|k_B)$
$I, r_A, \sigma_{SA} = \mathcal{S}_{SK_{SA}}(I|r_A)$
$I, r_B, \sigma_{SB} = \mathcal{S}_{SK_{SB}}(I|r_B)$
$r_B, u_{SA} = H(I|k_A|r_A|r_B)$
$r_A, u_{SB} = H(I|k_B|r_B|r_A)$
$K = g^{ab}$
$K = g^{ab}$
$sk = H(K|I),\ I = ID_A|ID_B|ID_{SA}|ID_{SB}$

<Fig. 1> Four-party password authenticated key agreement protocol

Additionally, A computes $r_A = g^a$ and $u_A = H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_A)$. Then A sends the message $<r_A\ u_A>$ to the server SA.

Meanwhile, client B, having received c_{SB}, computes $r_{SB} = D_{PW_B}(c_{SB})$ and chooses a random number $b \in R\ Z_p^*$. From r_{SB} and b, B computes the one time key k_B to be shared with SB as

$$k_B = g^{bt} = (r_{SB})^b;$$

B also computes $r_B = g^b$ and $u_B = (ID_A/ID_B/ID_{SA}/ID_{SB}/k_B)$ and then sends $<r_B, u_B>$ to SB.

Step 5. Upon receiving $<r_A, u_A>$, server SA first computes the one time key $k_A = g^{as}$ shared with A and then verifies that u_A from A equals the hash value $H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_A)$If the verification fails, SA stops executing the protocol; otherwise, SA believes that client A is genuine. SA then sends the message $<ID_A, ID_B, ID_{SA}, ID_{SB}, r_A, \sigma_{SA}>$ to the server SB, where σ_{SA} is the signature on $ID_A/ID_B/ID_{SA}/ID_{SB}/r_A$ generated by the singing algorithm S using the private key SK_{SA}, namely,

$$\sigma_{SA} = S_{SK_{SA}}(ID_A|ID_B|ID_{SA}|ID_{SB}|r_A).$$

Similarly, upon receiving $<r_B\ u_B>$, server SB computes the one time key $k_B = g^{bt}$ shared with B and verifies that u_B from B equals $H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_B)$. If the verification fails, SB aborts the protocol; otherwise, SB believes client B as authentic. Then SB sends the message $<ID_A, ID_B, ID_{SA}, ID_{SB}, r_B, \sigma_{SB}>$ to SA, where σ_{SB} is the signature on $ID_A/ID_B/ID_{SA}/ID_{SB}/r_B$ generated by S using the private key SK_{SB}, namely,

$$\sigma_{SB} = S_{SK_{SB}}(ID_A|ID_B|ID_{SA}|ID_{SB}|r_B).$$

Step 6. After receiving $<ID_A, ID_B, ID_{SA}, ID_{SB}, r_B, \sigma_{SB}>$, SA first verifies the signature σ_{SB} using the public key PK_{SB}. SA halts immediately if the verification fails. Otherwise, SA computes

$$u_{SA} = H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_A|\mathrm{r}_A|\mathrm{r}_A)$$

and sends $<r_B\ u_{SA}>$ to client A.

The server SB, upon receiving $<ID_A, ID_B, ID_{SA}, ID_{SB}, r_A, \sigma_{SA}>$, verifies the signature σ_{SA}, and if correct, computes

$$u_{SB} = H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_A|\mathrm{r}_A|\mathrm{r}_A)$$

and sends $<r_A, u_{SB}>$ to B.

Step 7. The client A checks whether u_{SA} from SA equals $H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_A|\mathrm{r}_A|\mathrm{r}_A)$. If this is untrue, A aborts the protocol. Otherwise, A computes the common secret value K as

$$K = g^{ab} = r_B^a$$

Similarly, client B verifies that u_{SB} from SB equals $H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_A|\mathrm{r}_B|\mathrm{r}_A)$, and if the verification succeeds, computes the common secret value K as

$$K = g^{ab} = r_A^b$$

Finally, the clients compute their session key sk as

$$sk = H(K/ID_A/ID_B/ID_{SA}/ID_{SB}).$$

3.2 Security Analysis on PAKA Protocol

In this preliminary version of the paper, we only provide a heuristic security analysis of the PAKA protocol, considering a variety of attacks and security properties.

3.2.1 Off-line Password Guessing Attack

In this attack, an attacker may try to guess a password and then to check the correctness of the guessed password off-line. If his guess fails, the attacker tries again with another password, until he find the proper one. In the PAKA protocol, the only information related to passwords is $c_{SA} = \varepsilon_{PW_A}(g^s)$ and $c_{SB} = \varepsilon_{PW_B}(g^t)$, but because s and t are chosen randomly, these values does not help the attacker to verify directly the correctness of the guessed passwords. Thus, off-line password guessing attacks would be unsuccessful against the PAKA protocol.

3.2.2 Explicit Authentication.

Another stronger kind of security goal for a key exchange protocol to achieve is explicit authentication, the property obtained when both implicit authentication and key confirmation hold. It is straightforward to see that PAKA does not achieve explicit authentication; that is, the client A (resp. B) does not know whether the client B (resp. A) has successfully computed a matching session key. However, it is easy to transform any key exchange protocol P with implicit authentication into a protocol P' providing explicit authentication by using standard techniques [3, 5].

The transformation works as follows. Suppose that in protocol P, two clients A and B ended up with computing their session key sk_A and sk_B, respectively. In protocol P', client A sends one additional flow $auth_A = H(sk_A/1)$ to B and similarly, client B sends $auth_B \overset{?}{=} H(sk_B/2)$ to client A. Upon receiving $auth_B$, client A checks the equality $auth_B \overset{?}{=} H(sk_A/2)$. If they are equal, then A computes the final session key sk' as $sk' = H(sk_A/0)$. Otherwise, A aborts the protocol.

Likewise, client B, after receiving $auth_A$, verifies that $auth_A$ equals $H(sk_B/1)$. If so, then B computes the final session key sk' as $sk' = H(sk_B/0)$. Otherwise, B aborts the protocol. This procedure of adding explicit authentication is outlined in Fig. 2.

Ⅳ. A Four-Party Password Authenticated Key Distribution Protocol (PAKD)

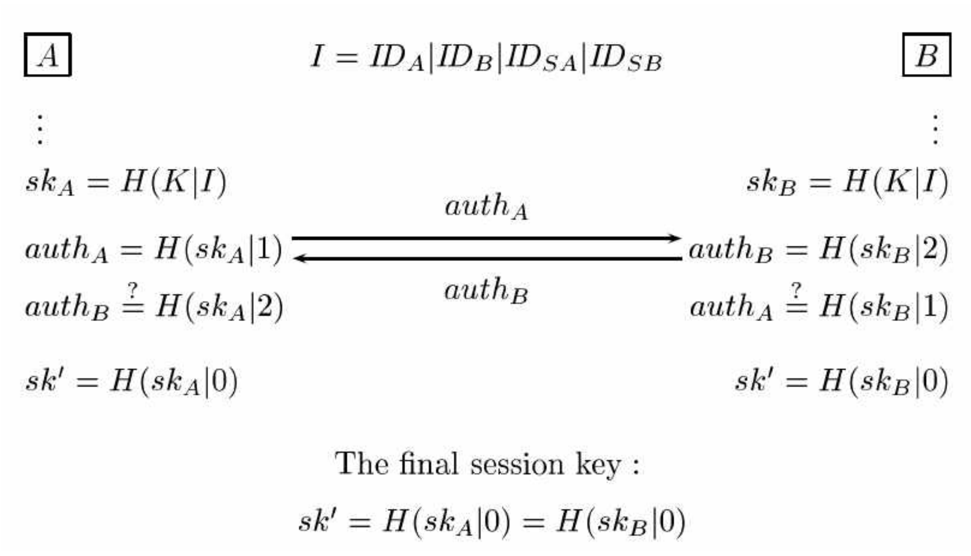
In this section we present a new four-party password authenticated key distribution protocol PAKD in which the session key is created by two servers and securely transmitted to these two clients. In describing the PAKD protocol, we will omit 'mod p' from expressions for notational simplicity. The PAKD protocol is outlined in Fig. 3 and a more detailed description is as follows:

4.1 Description of PAKD Protocol

Step 1. Two clients A and B first need to inform each other of their respective authentication server. To this end, A sends to B the message $<ID_A, ID_{SA}, ID_B>$ and B sends to A the message $<ID_B, ID_{SB}, ID_A>$.

Step 2. The clients request the assistance of their respective server in establishing a session key between them. Client A (resp. B) does this by sending the message $<ID_A, ID_B, ID_{SB}>$ (resp. $<ID_B, ID_A, ID_{SA}>$) to the server SA (resp. SB).

Step 3. Server SA chooses a random number $s \in R$ Z_p^*, computes $r_{SA} = g^s$ and $c_{SA} = \varepsilon_{PW_A}(r_{SA})$, and sends the ciphertext c_{SA} to A. Similarly, server SB chooses a



<Fig. 2> Adding Explicit Authentication

random number $t \in_R Z_p^*$, computes $r_{SB} = g^t$ and $c_{SB} = \varepsilon_{PW_B}(r_{SB})$, and sends the ciphertext c_{SB} to B.

Step 4. After receiving c_{SA}, client A recovers r_{SA} by decrypting c_{SA}, i. e., $r_{SA} = D_{PW_A}(c_{SA})$, and chooses a random number $a \in_R Z_p^*$. Given r_{SA} and a, client A computes the one time key k_A to be shared with the server SA as

$$k_A = g^{as} = (r_{SA})^a.$$

Additionally, A computes $r_A = g^a$ and $u_A = H(ID_A|ID_B|ID_{SA}|ID_{SB}|k_A)$. Then A sends the message $<r_A, u_A>$ to the server SA.

Meanwhile, client B, having received c_{SB}, computes $r_{SB} = D_{PW_B}(c_{SB})$ and chooses a random number $b \in_R Z_p^*$. From r_{SB} and b, B computes the one time key k_B to be shared with SB as

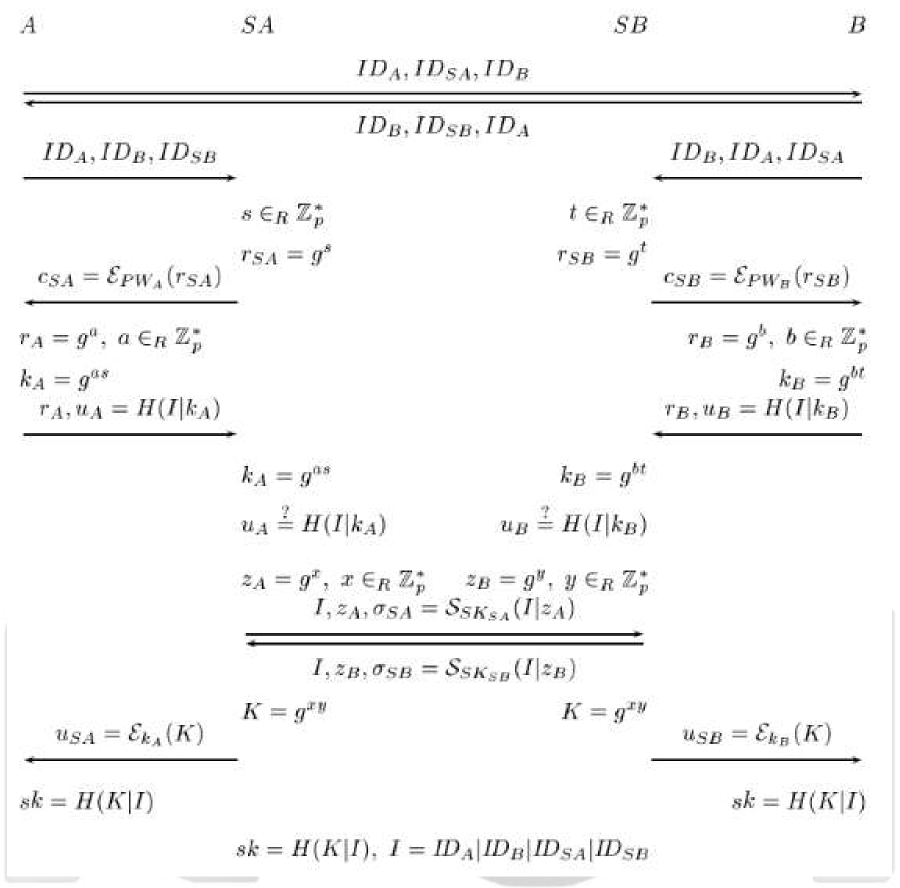
$$k_B = g^{bt} = (r_{SB})^b.$$

B also computes $r_B = g^b$ and $u_B = H(ID_A|ID_B|ID_{SA}|ID_{SB}|k_B)$ and then sends $<r_B, u_B>$ to SB.

Step 5. Upon receiving $<r_A, u_A>$, server SA first computes the one time key $k_A = g^{as}$ shared with A and then verifies that u_A from A equals the hash value $H(ID_A|ID_B|ID_{SA}|ID_{SB}|k_A)$. If the verification fails, SA stops executing the protocol; otherwise, SA believes that client A is genuine. SA chooses a random number $x \in_R Z_p^*$, computes $z_A = g^x$, and then sends the message $<ID_A, ID_B, ID_{SA}, ID_{SB}, z_A, \sigma_{SA}>$ to the server SB, where σ_{SA} is the signature on $ID_A|ID_B|ID_{SA}|ID_{SB}|z_A$ generated by the singing algorithm S using the private key SK_{SA}, namely,

$$\sigma_{SA} = S_{SK_{SA}}(ID_A|ID_B|ID_{SA}|ID_{SB}|z_A).$$

Similarly, after receiving $<r_B, u_B>$, server SB computes the one time key $k_B = g^{bt}$ shared with B and


A
SA
SB
B
ID_A, ID_{SA}, ID_B
ID_B, ID_{SB}, ID_A
ID_A, ID_B, ID_{SB}
ID_B, ID_A, ID_{SA}
$s \in_R \mathbb{Z}_p^*$
$r_{SA} = g^s$
$t \in_R \mathbb{Z}_p^*$
$r_{SB} = g^t$
$c_{SA} = \mathcal{E}_{PW_A}(r_{SA})$
$c_{SB} = \mathcal{E}_{PW_B}(r_{SB})$
$r_A = g^a,\ a \in_R \mathbb{Z}_p^*$
$k_A = g^{as}$
$r_A, u_A = H(I|k_A)$
$r_B = g^b,\ b \in_R \mathbb{Z}_p^*$
$k_B = g^{bt}$
$r_B, u_B = H(I|k_B)$
$k_A = g^{as}$
$u_A \overset{?}{=} H(I|k_A)$
$k_B = g^{bt}$
$u_B \overset{?}{=} H(I|k_B)$
$z_A = g^x,\ x \in_R \mathbb{Z}_p^*$
$z_B = g^y,\ y \in_R \mathbb{Z}_p^*$
$I, z_A, \sigma_{SA} = \mathcal{S}_{SK_{SA}}(I|z_A)$
$I, z_B, \sigma_{SB} = \mathcal{S}_{SK_{SB}}(I|z_B)$
$K = g^{xy}$
$K = g^{xy}$
$u_{SA} = \mathcal{E}_{k_A}(K)$
$u_{SB} = \mathcal{E}_{k_B}(K)$
$sk = H(K|I)$
$sk = H(K|I)$
$sk = H(K|I),\ I = ID_A|ID_B|ID_{SA}|ID_{SB}$

<Fig. 3> Four-party password authentication key distribution protocol

verifies that u_B from B equals $H(ID_A|ID_B|ID_{SA}|ID_{SB}|k_B)$. If the verification fails, *SB* aborts the protocol; otherwise, *SB* believes client *B* as authentic. Then *SB* chooses a random number $y \in RZ_p^*$, computes $z_B = g^y$ and sends the message $<ID_A, ID_B, ID_{SA}, ID_{SB}, z_B, \sigma_{SB}>$ *to SA*, where σ_{SB} is the signature on $ID_A|ID_B|ID_{SA}|ID_{SB}|z_B$ generated by S using the private key SK_{SB}, namely,

$$\sigma_{SB} = S_{SK_{SB}}(ID_A|ID_B|ID_{SA}|ID_{SB}|z_B).$$

Step 6. After receiving $<ID_A, ID_B, ID_{SA}, ID_{SB}, z_B, \sigma_{SB}>$, *SA* first verifies the signature σ_{SB} using the public key PK_{SB}. *SA* halts immediately if the verification fails. Otherwise, SA computes

$$K = g^{xy} = (z_B)^x$$
$$u_{SA} = \varepsilon_{k_A}(K)$$

and sends $<u_{SA}>$ *to client A*.

The server *SB*, upon receiving $<ID_A, ID_B, ID_{SA}, ID_{SB}, z_A, \sigma_{SA}>$, verifies the signature σ_{SA}, and if correct, computes

$$K = g^{xy} = (z_A)^y$$
$$u_{SB} = \varepsilon_{k_B}(K)$$

and sends <u_{SB}> to client B.

Step 7. Upon receiving <u_{SB}>, client A recovers K by decrypting u_{SA}, i. e., $K = D_{k_A}(u_{SA})$.

Similarly, after receiving <u_{SA}>, client B recovers K by decrypting u_{SB}, i. e., $K = D_{k_B}(u_{SB})$. Finally, the clients compute their session key sk as

$$sk = H(K/ID_A/ID_B/ID_{SA}/ID_{SB}).$$

4.2 Security Analysis on PAKD Protocol

In this preliminary version of the paper, we only provide a heuristic security analysis of the proposed protocol, considering a variety of attacks and security properties; a rigorous proof of security in a formal communication model will be given in the full version of this paper.

4.2.1 Off-line Password Guessing Attack.

In the proposed protocol, the only information related to passwords is $c_{SA} = \varepsilon_{PW_A}(g^s)$ and $c_{SB} = \varepsilon_{PW_B}(g^t)$, but because s and t are chosen randomly, these values does not help the attacker to verify directly the correctness of the guessed passwords. Thus, off-line password guessing attacks would be unsuccessful against the proposed protocol.

4.2.2 Undetectable On-Line Password Guessing Attack.

At the highest level of security threat to password authenticated key exchange protocols are undetectable on-line password guessing attacks [8] where an attacker tries to check the correctness of a guessed password in an on-line transaction with the server, i. e., in a fake execution of the protocol; if his guess fails, he starts a new transaction with the server using another guessed password. Indeed, the possibility of an undetectable on-line password guessing attack in the three-or- more-party setting represents a qualitative difference from the two-party setting where such attack is not a concern. However, this attack is meaningful only when the server is unable to distinguish an honest request from a malicious one, since a failed guess should not be detected and logged by the server.

In our protocol, the server is the first who issues a challenge and the client is the first who replies with an answer to some challenge. It is mainly due to this ordering that the protocol is secure against undetectable on-line password guessing attacks. Suppose that an attacker A', posing as A, decrypts c_{SA} by guessing a password, computes $r_{A'}$, $k_{A'}$, and $u_{A'} = H(ID_A/ID_B/ID_{SA}/ID_{SB}/k_{A'})$ by choosing his own random a', and sends the fake message <$r_{A'}$, $u_{A'}$> to server SA. Then, the server SA, upon receiving $r_{A'}$ and $u_{A'}$ from A', should be easily able to detect a failed guess since the protocol specification mandates SA to check the correctness of $u_{A'}$. Note that the attacker cannot send a correct u_A without knowing a correct k_A which in turn only can be computed if the guessed password is correct. Hence, the proposed protocol can resist undetectable on-line password guessing attacks.

4.2.3 Insider Attack.

Suppose one client, say A, tries to learn the password of the other client B during execution of the protocol. Of course, A should not be able to have this ability through a protocol run and this is still an important security concern to be addressed. As mentioned earlier in this section, the only information related to the B's password is $c_{SB} = \varepsilon_{PW_B}(r_{SB})$, where

r_{SB} is computed as $r_{SB} = g^t$ and t is a random value chosen by the server SB. The actual value r_{SB} itself is never included in any message sent in the protocol and t is a secret information only known to SB. This means that the malicious insider A has no advantage over outside attackers in learning B's password. In other words, A's privileged information — PW_A, r_{SA}, and k_A — gives A no help in learning B's password. Therefore, our protocol is also secure against insider attacks.

4.2.4 Implicit Key Authentication.

Given $z_A = g^x$ and $z_B = g^y$, the secret value $K = g^{xy}$ cannot be computed, since no polynomial algorithm has been found to solve the computational Diffie-Hellman problem. Thus, if the random numbers x and y are unknown, then the session key sk cannot be computed since H is a one-way hash function. Hence, the secrecy of the session key is guaranteed based on the computational Diffe-Hellman assumption in the random oracle model [4].

4.2.5 Perfect Forward Secrecy.

The proposed protocol also achieves perfect forward secrecy [7] since the long-term information of the protocol participants is used for implicit key authentication only, and not for hiding the session key. Even if an attacker obtains the clients' passwords and the servers' signing private keys, he cannot get any of secret values $K = g^{xy}$ computed in previous sessions since x and y chosen respectively by SA (resp. SB) are unknown. Because the session key in this protocol is computed as $sk = H(K|ID_A|ID_B|ID_{SA}|ID_{SB})$, he cannot compute it without knowing the secret value K. Hence, our protocol provides perfect forward secrecy.

Table 1. Comparison of security properties between PAKA and PKAD

Security Property	PAKA	PAKD
Off-line password guessing attack	satisfy	satisfy
Undectable on-line password guessing attack	satisfy	satisfy
Insider attack	satisfy	satisfy
Perfect forward secrecy	satisfy	satisfy
Known key security	satisfy	satisfy
Implicit key authentication	satisfy	satisfy

4.2.6 Known Key Security.

In our protocol, the session keys generated in different sessions are independent since the ephemeral secret exponents x and y are chosen independently at random from session to session. Thus, the proposed protocol still achieves its goal in the face of an attacker who has learned some other session keys.

V. Conclusion

We considered the problem of password authenticated key exchange in the inter domain distributed computing environment, where the authentication server of one client is different from that of another client, hence four parties — two clients and two servers — involved in key exchange. The work of Yeh and Sun [20] is probably the first and only one that deals with the problem. But, the protocols by Yeh and Sun exhibit an undesirable feature that each client needs not only to remember a password shared with the server but also to store and manage the server's public key. In some sense, this disadvantage obviates the reason for considering password authenticated protocols in the first place; it is difficult for humans to securely manage long cryptographic keys. The contribution of this paper was to propose the four-party password authenticated key exchange protocol that requires each client only to remember a password shared with its authentication server. We, in this preliminary version of the paper, showed heuristically that the proposed protocol achieves all the intended security goals against a variety of attacks. We leave a rigorous security proof for the protocol in a formal communication model as a future work.

Reference

[1] M. Abdalla, P. -A. Fouque, and D. Pointcheval, Password-based authenticated key exchange in the three-party setting, In Proceedings of 8th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2005), LNCS 3386, 2005, pp. 65-84.

[2] M. Bellare and C. Namprempre, Authenticated encryption: Relations among notions and analysis of the generic composition paradigm, Advances in Cryptology - Asiacrypt 2000, LNCS 1976, 2000, pp. 531-545.

[3] M. Bellare, D. Pointcheval, and P. Rogaway, Authenticated key exchange secure against dictionary attacks, Advances in Cryptology - Eurocrypt 2000, LNCS 1807, 2000, pp. 139-155.

[4] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing e±cient protocols, In Proceedings of the 1st ACM Conferences on Computer and Communications Security (CCS 1993), 1993, pp. 62-73.

[5] E. Bresson, O. Chevassut, D. Pointcheval, and J. -J. Quisquater, Provably authenticated group Diffie-Hellman key exchange, In Proceedings of the 8th ACM conference on Computer and Communications Security (CCS 2001), 2001, pp. 255-264.

[6] M. Burmester and Y. Desmedt, “A secure and efficient conference key distribution system,” In Advances in Cryptology Eurocrypt′94, LNCS 950, 1995, pp. 275-286.

[7] W. Di±e, P. Oorschot, and M. Wiener, Authentication and authenticated key exchanges, Designs, Codes, and Cryptography, vol. 2, no. 2, 1992, pp. 107-125.

[8] Y. Ding and P. Horster, Undectectable on-line password guessing attacks, ACM SIGOPS Operating Systems Review, vol. 29, no. 4, 1995, pp. 77-86.

[9] D. Dolev, C. Dwork, and M. Naor, Nonmalleable cryptography, SIAM Journal on Computing, vol. 30, no. 2, 2000, pp. 391-437.

[10] IEEE P1363. 2: Password-based public-key cryptography, http://grouper.ieee.org/groups/1363/passwdPK/

[11] S. Goldwasser, S. Micali, and R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, SIAM Journal of Computing, vol. 17, no. 2, 1988, pp. 281-308.

[12] Y. Gang, F. Dengguo, and H. Xiaoxi, Improved Client-to-Client Password-Authenticated Key Exchange Protocol, The Second International Conference on Availability, Reliability and Security (ARES′07), 2007, pp. 564-574.

[13] L. Gong, M. -L. Lomas, R. -M. Needham, and J. -H. Saltzer, Protecting poorly chosen secrets from guessing attacks, IEEE Journal on Selected Areas

in Communications, vol. 11, no. 5, 1993, pp. 648-656.

[14] J. Katz and M. Yung, Unforgeable encryption and adaptively secure modes of operation, In Proceedings of 7th International Workshop on Fast Software Encryption (FSE 2000), LNCS 1978, 2000, pp. 284-299.

[15] T. Kwon, M. Kang, S. Jung, and J. Song, An improvement of the password-based authentication protocol (K1P) on security against replay attacks, IEICE Transactions on Communications, vol. E82-B, no. 7, 1999, pp. 991-997.

[16] Y. Lee, J. Nam, and D. Won, An Inter-domain key Agreement Protocol Using Weak Passwords, LNCS, vol. 3982, 2006, pp. 517-526.

[17] C. -L. Lin, H. -M. Sun, and T. Hwang, Three-party encrypted key exchange: attacks and a solution, ACM SIGOPS Operating Systems Review, vol. 34, no. 4, 2000, pp. 12-20.

[18] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, OCB: A block-cipher mode of operation for efficient authenticated encryption, In Proceedings of the 8th ACM conference on Computer and Communications Security (CCS 2001), 2001, pp. 196-205.

[19] M. Steiner, G. Tsudik, and M. Waidner, Refinement and extension of encrpyted key exchange, ACM SIGOPS Operating Systems Review, vol. 29, no. 3, 1995, pp. 22-30.

[20] H. -T. Yeh, and H. -M. Sun, Password authenticated key exchange protocols among diverse network domains, Computers and Electrical Engineering, vol. 31, no. 3, 2005, pp. 175-189.

[21] F. Zhou, X. Liu, and G. Chang, Cryptanalysis and Improvement of EC2C-PAKA Protocol in Cross-Realm, 2008 IFIP International Conference on Network and Parallel Computing, 2008, pp. 82-87.

■ 저자소개 ■

이 영 숙
Lee, Young Sook

2009년 3월~현재 호원대학교 사이버수사경찰학부 전임 강사
2008년 8월 성균관대학교 컴퓨터공학과(공학박사)
2005년 2월 성균관대학교 정보보호학과(공학석사)
1987년 2월 성균관대학교 정보공학과(공학사)

관심분야 : 암호프로토콜, 암호이론, 네트워크 보안
E-mail : ysooklee@howon.ac.kr

논문접수일 : 2009년 11월 6일
수 정 일 : 2009년 11월 27일
게재확정일 : 2009년 12월 5일