Security Analysis of Password-Authenticated Key Agreement Protocols*

Kyung-Ah Shim[1] and Seung-Hyun Seo[2]

[1] Department of Mathematics
[2] Department of Computer Science and Engineering,
Ewha Womans University, Seoul, Korea
kashim@ewha.ac.kr, seosh@ewhain.net

Abstract. Recently, there have been proposed a number of password-authenticated key agreement protocols for two-party setting or three-party setting. In this paper, we show that recently proposed three password-authenticated key agreement protocols in [11, 12, 10] are insecure against several active attacks including a stolen-verifier attack, an off-line password guessing attack and impersonation attacks.

1 Introduction

Two entities, who only share a password, and who are communicating over an insecure network, want to authenticate each other and agree on a session key to be used for protecting their subsequent communication. This is called the *password-authenticated key exchange* problem. The first password-authenticated key exchange (PAKE) protocol, known as Encrypted Key Exchange (EKE), was suggested by Bellovin and Merritt [1]. By using a combination of symmetric and public-key cryptography, EKE resists dictionary attacks by giving a passive attacker insufficient information to verify a guessed password. Since it was invented, many password-authenticated key agreement protocols that promised increased security have been developed [2-4, 8, 9, 14-16].

In 1995, Steiner, Tsudik, and Waidner [15] extended two-party EKE protocol to three-party one (STW-3P-EKE), in which all clients share a password with a trusted server S only and in which S mediates between two communication parties to allow their mutual authentication. The three-party EKE protocol is particularly well-suited for large communication environments because it is inconvenient in key management that every two communication parties mutually share a secret. Unfortunately, Ding and Horster [7] showed that the STW-3P-EKE is not resistant to undetectable on-line password guessing attacks. Lin, Sun and Hwang [13] also pointed out that the STW-3P-EKE is not only vulnerable to undetectable on-line password guessing attacks but also vulnerable to off-line password guessing attacks. They proposed a new three-party EKE, in which the server holds a long-term and publicly known public key to prevent both off-line

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD).(KRF-2005-217-C00002).

and undetectable on-line password guessing attack. However, in their protocol, communication parties have to obtain and verify the public key of the server, a task which puts a high burden on the user. Later, there have been proposed several key agreement protocols for three-parties, in which two clients establish a common session key through a authentication server. Most of those protocols require to use server's public key to prevent password guessing attacks. However, the protocols may not be practical for some environments since clients need to verify and keep the server's public key. Recently, Lee *et al* [12] proposed a new efficient verifier-based key agreement protocol for three parties, which does not require server's public key. They argued that the protocol was secure against impersonation attacks and server compromise. In this paper, we show that the protocol is still insecure against a stolen-verifier attack and impersonation attacks. Also, Lee *et al* [11] proposed a two-party password-authenticated key agreement protocol PAKA and its verifier-based version PAKA-X. In the PAKA-X protocol, the client uses a plaintext of the password, while the server stores a verifier for the password. So the protocol does not allow an adversary who compromises the server to impersonate a client without actually running a dictionary attack on the password file. We will show that the PAKA-X protocol is insecure against a stolen-verifier attack and an off-line password-guessing attack.

At ICICS'02, Byun *et al* [5] proposed two password-authenticated key exchange protocol between clients with different passwords, so-called Client-to-Client Password-Authenticated Key Exchange (C2C-PAKE) protocol. One is for a cross-realm setting where two clients are in two different realms and hence there exist two servers involved, the other is for a single-server setting where two clients are in the same realm. The protocol being circulated for consideration at the 27th SC27/WG2. Subsequently, Chen [6] and Kim *et al* [10] showed that their protocol was insecure against a dictionary attack by a malicious server in a different realm and Denning-Sacco attacks mounted by insiders, respectively. And Kim *et al* [10] also proposed a modified protocol to resist these attacks. In this paper, we point out that the modified protocol is also insecure against impersonation attacks.

The rest of the paper is organized as follows. The next section presents the attacks on the PAKA-X protocol. In section 3, we point out the Lee *et al*'s PAKE for three-party is insecure against a stolen-verifier attack. In section 4, we show that the modified C2C-PAKE protocol is insecure against partition attacks and impersonation attacks. Concluding remarks are given in section 5.

2 Cryptanalysis of the PAKA-X Protocol

Lee *et al* [11] proposed a password-based authenticated key agreement protocol, PAKA and its verifier-based version, PAKA-X. In this section, we show that the PAKA-X protocol is insecure against a stolen-verifier attack and an off-line password guessing attack. First, we review the PAKA-X protocol.

2.1 The PAKA-X Protocol

Let g be a generator of $\mathbb{Z}_p^*$, where p is a large prime and h a collision-resistant one-way hash function. Henceforth, we will omit the operation 'mod p' for simplicity. We assume that there is an initialization in which the client, Alice chooses a memorable password, π, computes a verifier $\nu = g^{h(Id_A,Id_B,\pi)}$ and then sends ν to the server, Bob, over a secure channel. Bob stores (Id_A, ν), where Id_A indicates an identifier of Alice. To enhance the efficiency of the protocol, $\nu = g^{h(Id_A,Id_B,\pi)}$ and $h(Id_A, Id_B, \pi)^{-1}$ can be precomputed by Alice before the protocol runs. The PAKA-X protocol is as follows;

1. Alice computes $X_A = g^a \oplus \nu$ by choosing $a \in_R \mathbb{Z}_p^*$ and then sends $\{Id_A, X_A\}$ to Bob.
2. After receiving the message, Bob retrieves ν from a password file, computes $X_B = \nu^b \oplus \nu$ by choosing $b \in_R \mathbb{Z}_p^*$ and then sends X_B to Alice. While waiting for a message from Alice, Bob computes $K_B = (X_A \oplus \nu)^b = g^{ab}$ and $V_A' = h(Id_A, X_B, K_B)$ and $V_B = h(Id_B, X_A, K_B)$, in sequence.
3. After receiving the message from Bob, Alice computes
$$K_A = (X_B \oplus \nu)^{ah(Id_A,id_B,\pi)^{-1}} = g^{ab}$$
and $V_A = h(Id_A, X_B, K_A)$ and then sends V_A to Bob. While waiting for a message from Bob, Alice computes $V_B' = h(Id_B, X_A, K_A)$.
4. On the receipt the message V_A, Bob checks whether $V_A = V_A'$ holds or not. If it holds, Bob is convinced that K_A is validated, and then sends V_B to Alice.
5. On the receipt the message V_B, Alice checks whether $V_B = V_B'$ holds or not. If it holds, Alice is convinced that K_B is validated.
6. Finally, Alice and Bob compute the common session key $K = h(K_A) = h(K_B) = h(g^{ab})$.

2.2 Attacks on the PAKA-X Protocol

Lee *et al* [11] argue that the PAKA-X is secure against sever compromise, i.e., an attacker who steals password file from the server cannot use that information directly to impersonate the client. Now, we show that the PAKA-X protocol is still vulnerable to server compromise, i.e., a stolen-verifier attack. If the host's password file is captured and then an adversary learns the value of the verifier ν, it should still not allow the adversary to impersonate the user without an expensive dictionary search. Then we say that the protocol is secure against stolen-verifier attack or server compromise attack.

• Stolen-Verifier Attack on the PAKA-X Protocol
Suppose that an adversary E has captured A's verifier ν and wishes to impersonate A to B. $E(A)$ represents E impersonating A.

1. First, the adversary E chooses $a \in_R \mathbb{Z}_p^*$ and computes $X_A = \nu^a \oplus \nu$ from ν. Then E starts a protocol run sending $\{Id_A, X_A\}$ to B impersonating A.

2. After receiving the message, B retrieves A's verifier ν from a password file. He chooses a random b, computes $X_B = \nu^b \oplus \nu$ and sends it to $E(A)$. Then B computes $K_B = (X_A \oplus \nu)^b = (\nu^a \oplus \nu \oplus \nu)^b = \nu^{ab}$, and $V_B = h(Id_B, X_A, K_B)$. While waiting for a message from $E(A)$, B computes $V'_A = h(Id_A, X_B, K_B)$.
3. On the receipt the message, $E(A)$ can obtain K_A (which is the same as K_B) by computing $(X_B \oplus \nu)^a = (\nu^b \oplus \nu \oplus \nu)^a = \nu^{ab} = K_A$ from X_B, ν and a. And then $E(A)$ computes $V_A = h(Id_A, X_B, K_A)$ and sends it to B.
4. After receiving the message V_A, B checks whether $V_A = V'_A$ holds or not. The equation holds since $K_A = K_B$. Then B is convinced that K_A is validated and sends $V_B = h(Id_B, X_A, K_B)$ to A. Next, B computes the session key $K = h(K_B) = h(\nu^{ab})$.
5. After receiving the message V_B, $E(A)$ computes $V'_B = h(Id_B, X_A, K_A)$ and checks $V_B = V'_B$ holds or not. Finally, E succeeds to impersonate A to B as well as the knowledge of the session key $K = h(K_A) = h(\nu^{ab})$.

By using the verifier, an adversary can compute the first transmitted message X_A to confine the shared secret to a predictable value from the message computed by the legitimate user B. In other words, by fabricating the message from the compromised verifier, the adversary can impersonate A and compute the shared secret, $K_A = K_B = \nu^{ab}$ established between $E(A)$ and B without the knowledge of the password π and an expensive dictionary search. Therefore, the PAKA-X is insecure against the stolen-verifier attack unlike their claim in [11].

We also show that the PAKA-X protocol is insecure against an off-line password guessing attack. The attack on the PAKA-X is mounted as follows;

• Off-line Password Guessing Attack on the PAKA-X Protocol
When A starts a protocol run by sending a message $\{Id_A, X_A = g^a \oplus \nu\}$ to B, an attacker $E(B)$ intercepts it and sends $X_B = 0$ to A impersonating B. On the receipt the message, A computes

$$K_A = (X_B \oplus \nu)^{ah(A,B,\pi)^{-1}} = \nu^{ah(A,B,\pi)^{-1}} = g^a, \quad V_A = h(Id_A, X_B, K_A)$$

and then sends V_A to $E(B)$. After receiving the message, E stores it and stops the protocol run.

$$\begin{array}{llcll} (1.1) & A & \longrightarrow E(B): & Id_A, & X_A = g^a \oplus \nu \\ (1.2) & E(B) & \longrightarrow \quad A \quad : & & X_B = 0 \\ (1.3) & A & \longrightarrow E(B): & \multicolumn{2}{l}{V_A = h(Id_A, X_B, K_A)} \\ (1.4) & E(B) & \longrightarrow \quad A \quad : & & \textit{stop.} \end{array}$$

Finally, E executes an off-line password guessing attack and then finds the password π iterating upon all possible choices of π;

1. Pick a candidate π'.
2. Compute $\nu' = g^{h(Id_A, Id_B, \pi')}$ and $K'_A = X_A \oplus \nu' = (g^a \oplus \nu) \oplus \nu'$ from the recorded message X_A.
3. Compare V_A with $h(Id_A, X_B, K'_A)$.

Since $V_A = h(Id_A, X_B, K_A) = h(Id_A, X_B, g^a)$, a match in the last step indicates correct guess of the password. Therefore, an attacker succeeds to guess the valid password π.

3 Cryptanalysis of the LKY Protocol

3.1 The LKY Protocol

We review the Lee *et al*'s verifier-based key agreement protocol (called the LKY protocol)[12] for three parties without server's public key. We assume that each client uses a memorable password, while the server stores corresponding verifiers instead of plaintext-equivalent passwords to resist to server compromise. Let p be a large prime and g a generator of $\mathbb{Z}_p^*$. Let $h(\cdot)$ be a collision-resistant one-way hash function. Assume that two clients, A and B, want to agree a common session key though a authentication server, called AS. For registering for AS, A and B, respectively, choose passwords π_A and π_B, compute verifiers $v_A = g^{t_A}$, $t_A = h(A, S, \pi_A)$ and $v_B = g^{t_B}$, $t_B = h(B, S, \pi_B)$, and then send v_A and v_B to AS over a secure channel. AS stores v_A and v_B in a password file. The protocol runs as follows;

1. A computes $X_A = g^a$ by choosing a random $a \in \mathbb{Z}_p^*$ and sends $\{A, X_A\}$ to B.
2. After receiving the message from A, B choose a random $b \in \mathbb{Z}_p^*$, computes $X_B = g^b$ and sends $\{A, X_A, B, X_B\}$ to AS. B also sends X_B to A.
3. After receiving the message from B, AS retrieves v_A and v_B from a password file. Then AS chooses $c, d \in \mathbb{Z}_p^*$ computes $X_{SA} = (v_A)^c \oplus v_A$ and $X_{SB} = (v_B)^d \oplus v_B$ and sends X_{SA} and X_{SB} to A and B, respectively. While waiting for messages from A and B, AS computes $K_{SA} = (X_A)^c = g^{ac}$ and $K_{SB} = (X_B)^d = g^{bd}$.
4. After receiving the messages from AS and B, A computes $K_{AS} = (X_{SA} \oplus v_A)^{t_A^{-1}a} = g^{ac}$, $V_{AS} = h(A, B, S, X_A, X_B, X_{SA}, K_{AS})$ and sends V_{AS} to AS. Similarly, after receiving the message from AS, B computes $K_{BS} = (X_{SB} \oplus v_B)^{t_B^{-1}b} = g^{bd}$, $V_{BS} = h(B, A, S, X_B, X_A, X_{SB}, K_{BS})$ and sends V_{BS} to AS.
5. After receiving the messages from A and B, AS checks whether $V_{AS} = h(A, B, S, X_A, X_B, X_{SA}, K_{SA})$ and $V_{BS} = h(B, A, S, X_B, X_A, X_{SB}, K_{SB})$ hold or not. If they hold, AS is convinced that A and B are validated. Then AS computes $V_{SA} = h(S, A, B, X_A, X_B, K_{SA})$, $V_{SB} = h(S, B, A, X_B, X_A, K_{SB})$ and sends V_{SA} and V_{SB} to A and B, respectively.
6. After receiving the message from AS, A checks whether $V_{SA} = h(S, A, B, X_A, X_B, K_{AS})$ holds or not. If it holds, A is convinced that both B and AS are validated. Similarly, B checks whether $V_{SB} = h(S, B, A, X_B, X_A, K_{BS})$ holds or not. If it holds, B is convinced that both A and AS are validated. Finally, A and B compute $K_{AB} = (X_B)^a = g^{ab}$ and $K_{BA} = (X_A)^b = g^{ab}$, respectively, and then compute a common session key $K = h(A, B, S, g^{ab})$.

3.2 Stolen-Verifier Attack on the LKY Protocol

Now, we point out that the LKY protocol is insecure against a stolen-verifier attack.

• Stolen-Verifier Attack on the LKY Protocol
Suppose that an adversary E has captured the verifiers v_A and v_B of A and B, respectively, and wishes to impersonate both A and B to AS, simultaneously.

1. First, the adversary E sets $X_A = v_A$ and $X_B = v_B$ and then starts a protocol run by sending $\{A, X_A, B, X_B\}$ to AS impersonating B.
2. After receiving the message from B, AS retrieves v_A and v_B from a password table. Then AS chooses $c, d \in \mathbb{Z}_p^*$, computes $X_{SA} = (v_A)^c \oplus v_A$ and $X_{SB} = (v_B)^d \oplus v_B$ and sends X_{SA} and X_{SB} to A and B, respectively. While waiting for messages from A and B, AS computes $K_{SA} = (X_A)^c = v_A^c$ and $K_{SB} = (X_B)^d = v_B^d$.
3. After receiving the messages from AS, from known value v_A, E computes $K_{AS} = X_{SA} \oplus v_A = v_A^c$, $V_{AS} = h(A, B, S, X_A, X_B, X_{SA}, K_{AS})$ and sends V_{AS} to AS impersonating A. Similarly, E computes $K_{BS} = X_{SB} \oplus v_B = v_B^d$, $V_{BS} = h(B, A, S, X_B, X_A, X_{SB}, K_{BS})$ and sends V_{BS} to AS impersonating B.
4. After receiving the messages from A and B, AS checks whether $V_{AS} = h(A, B, S, X_A, X_B, X_{SA}, K_{SA})$ and $V_{BS} = h(B, A, S, X_B, X_A, X_{SB}, K_{SB})$ hold or not. The equations hold since $K_{AS} = K_{SA} = v_A^c$ and $K_{BS} = K_{SB} = v_B^d$ as intended by the adversary. Thus, AS is convinced that A and B are validated. Then, AS computes $V_{SA} = h(S, A, B, X_A, X_B, K_{SA})$, $V_{SB} = h(S, B, A, X_B, X_A, K_{SB})$ and sends V_{SA} and V_{SB} to A and B, respectively.
5. After receiving the message from AS, E checks whether $V_{SA} = h(S, A, B, X_A, X_B, K_{AS})$ and $V_{SB} = h(S, B, A, X_B, X_A, K_{BS})$ hold or not. Finally, E succeeds to impersonate both A and B to AS, simultaneously.

In above attack, by fabricating the messages and impersonating both A and B, the adversary can compute the shared secrets, K_{AS} and K_{BS} established between A and AS and B and AS, respectively, without the knowledge of genuine passwords π_A and π_B, equivalently, t_A and t_B.

4 Cryptanalysis of the Modified C2C-PAKE Protocol

4.1 The Modified C2C-PAKE Protocol

We first review the modified C2C-PAKE protocol [10] in a cross-realm setting. The following notation is used throughout this section.

• A, B	Two clients in two different realms.
• ID_A, ID_B	Identities of A and B, respectively.
• KDC_A, KDC_B	Key Distribution Centers which store passwords of A and B, resp.

• K	A symmetric key shared between KDC_A and KDC_B.
• $E_K(\cdot)$	A symmetric encryption under the symmetric key K.
• $E_{pwa}(\cdot)/D_{pwa}(\cdot)$	A symmetric encryption/decryption under the password pwa.

The modified C2C-PAKE protocol runs as follows;

[**Protocol Initialization**]

1. Let p and q be sufficiently large primes such that $q|p-1$, and let $\mathbb{G} =< g >$ be a subgroup of $\mathbb{F}_p^*$ with order q. Let g be a generator of $\mathbb{G}$. Let H_i $(i = 1, 2)$ be cryptographic hash functions. The public system parameters are $< p, q, g, H_1, H_2 >$. The system parameters are shared by all protocol participants.
2. A chooses a password pwa, then transfers it to KDC_A through a secure channel. B also transfers pwb to KDC_B, similarly. KDC_A and KDC_B store $(ID_A,\ pwa)$ and $(ID_B,\ pwb)$, respectively, in their database.

[**Ticket Issuing Stage**]

1. First, A chooses a random $x \in \mathbb{Z}_p^*$ and sends $\{E_{pwa}(g^x),\ ID_B\}$ to KDC_A.
2. On the receipt of the message from A, KDC_A obtains g^x by decrypting $E_{pwa}(g^x)$ under A's password pwa. Next, KDC_A selects a random $r \in \mathbb{Z}_p^*$ and issues $Ticket_B = E_K(g^{xr}, g^r, ID_A, ID_B, L)$, where K is a symmetric key shared between KDC_A and KDC_B and L is a lifetime of $Ticket_B$. Then KDC_A sends $\{Ticket_B,\ ID_A,\ ID_B,\ L\}$ to A.

[**Mutual Authentication and Key Exchange Stage**]

1. Upon receiving $Ticket_B$ from KDC_A, A forwards $Ticket_B$ to B with ID_A.
2. B chooses a random $y \in \mathbb{Z}_p^*$ and computes $E_{pwb}(g^y)$. Then B sends $\{E_{pwb}(g^y),\ ID_A,\ ID_B,\ Ticket_B\}$ to KDC_B.
3. After receiving the message from B, KDC_B obtains g^{xr} and g^r by decrypting $Ticket_B$ with K, selects a random $r' \in \mathbb{Z}_p^*$ and computes $(g^{xr})^{r'}$ and $(g^r)^{r'}$. Next, KDC_B sends $\{g^{xrr'},\ g^{rr'}\}$ to B.
4. When B receives the message, B computes $cs = H_1(g^{xyrr'})$ from $g^{xrr'}$ and y, chooses a random $a \in \mathbb{Z}_p^*$ and computes $E_{cs}(g^a)$ and $g^{rr'y}$. Finally, B sends $\{E_{cs}(g^a),\ g^{rr'y}\}$ to A.
5. On the receipt of the message, A computes $cs = H_1(g^{xyrr'})$ from $g^{yrr'}$ and x and then obtains g^a by decrypting $E_{cs}(g^a)$ under cs. Next, A chooses a random $b \in \mathbb{Z}_p^*$ and computes the session key $sk = H_2(g^{ab})$ and $E_{cs}(g^b)$. Then A sends $\{E_{sk}(g^a),\ E_{cs}(g^b)\}$ to B for the session key confirmation.
6. By decrypting $E_{cs}(g^b)$ with cs, B gets g^b and computes $sk = H_2(g^{ab})$. Then B verifies g^a by decrypting $E_{sk}(g^a)$ with sk and sends $E_{sk}(g^b)$ to A.
7. After receiving the message, A verifies g^b by decrypting $E_{sk}(g^b)$ with sk. Finally, A (resp., B) authenticates B (resp., A)and share the session key.

Kim *et al* [10] argued that the modified C2C-PAKE protocol is secure against all kinds of attacks considered in [5] including the Denning-Sacco attack and Chen's dictionary attacks [6]. However, we will show that the modified C2C-PAKE protocol is totally broken by an active adversary without the knowledge of any secret information such as past session keys and the shared symmetric key.

4.2 Attacks on the Modified C2C-PAKE Protocol

Now, we show that the modified protocol is insecure against partition attacks and two types of impersonation attacks.

• Partition Attacks
Both Byun *et al*'s original protocol [5] and the modified C2C-PAKE protocol [10] use g as a generator of $\mathbb{G}$, where $\mathbb{G}$ is a subgroup of $\mathbb{F}_p^*$ of order q. However, it is known that the protocols with such a generator are insecure against partition attacks [9]. Thus, the C2C-PAKE and the modified C2C-PAKE protocols are also insecure the attacks. However, the attacks can be easily prevented if g is taken as a primitive element of $\mathbb{F}_p^*$, i.e., a generator of $\mathbb{F}_p^*$.

• Impersonation Attack I on the Modified C2C-PAKE Protocol
Suppose that an adversary E wants to impersonate A to B on the protocol.

[**Ticket Issuing Stage**]

1. An adversary $E(A)$ chooses a random $k \in \mathbb{Z}_p^*$ and sends $\{k,\ ID_A,\ ID_B\}$ to KDC_A impersonating A.
2. On the receipt of the message from $E(A)$, KDC_A thinks that the stage is initiated by A. Then KDC_A obtains $D_{pwa}(k)$ by decrypting k under pwa, selects a random $r \in \mathbb{Z}_p^*$ and generates $Ticket_B = E_K(D_{pwa}(k)^r, g^r, ID_A, L)$. Next, KDC_A sends $\{Ticket_B,\ ID_A,\ ID_B\}$ to $E(A)$.

[**Mutual Authentication and Key Exchange Stage**]

1. After receiving $Ticket_B$ from KDC_A, $E(A)$ sends $\{Ticket_B,\ ID_A\}$ to B impersonating A.
2. Then B computes $E_{pwb}(g^y)$ and sends $\{E_{pwb}(g^y), ID_A, ID_B, Ticket_B\}$ to KDC_B.
3. E intercepts the message from B, chooses random $k', k'' \in \mathbb{Z}_p^*$ and computes $g^{k'}$ and $g^{k'k''}$. Then $E(KDC_B)$ sends $\{g^{k'}, g^{k'k''}\}$ to B impersonating KDC_B.
4. After receiving $\{g^{k'}, g^{k'k''}\}$ from $E(KDC_B)$, B thinks that it is sent from KDC_B. B computes $cs = H_1(g^{k'y})$ from $g^{k'}$ and computes $E_{cs}(g^a)$ and $g^{k'k''y}$. Then, B sends $\{E_{cs}(g^a),\ g^{k'k''y}\}$ to $E(A)$.
5. On the receipt of the message, $E(A)$ can obtain $g^{k'y}$ from $g^{k'k''y}$ by computing $(g^{k'k''y})^{k''^{-1}}$. Next, E computes $cs = H_1(g^{k'y})$ and then recover g^a from $E_{cs}(g^a)$. Next, $E(A)$ chooses a random $b \in \mathbb{Z}_p^*$ and computes a session key $sk = H_2(g^{ab})$ and $E_{cs}(g^b)$. Then $E(A)$ sends $\{E_{sk}(g^a),\ E_{cs}(g^b)\}$ to B.
6. After receiving $E_{sk}(g^a)$ and $E_{cs}(g^b)$, B first obtains g^b by decrypting $E_{cs}(g^b)$ and computes $sk = H_2(g^{ab})$ and then verifies g^a from $E_{sk}(g^a)$. B computes $E_{sk}(g^b)$ and sends it to $E(A)$.
7. Finally, E succeeds to impersonate both A and KDC_B to B as well as the knowledge of the session key $sk = H_2(g^{ab})$.

• **Impersonation Attack II on the Modified C2C-PAKE Protocol**
Another impersonation attack can be mounted on the protocol without performing the **Ticket Issuing Stage**.

1. First, without issuing $Ticket_B$, $E(A)$ chooses a random $k \in \mathbb{Z}_p^*$ and sends $\{k, ID_A\}$ to B impersonating A.
2. After receiving the message, B believes that k is a $Ticket_B$ issued by KDC_A. Because there is no way to confirm that it is issued by KDC_A. Then B computes $E_{pwd}(g^y)$ and sends $\{E_{pwb}(g^y), ID_A, ID_B, k\}$ to KDC_B.
3. Intercepting the message sent from B, E chooses $k', k'' \in \mathbb{Z}_p^*$, computes $g^{k'}$ and $g^{k'k''}$ and sends them to B impersonating KDC_B.
4. The remainder of this attack is the same as the impersonation attack I. Finally, E succeeds to impersonate A to B as well as the knowledge of the session key without issuing $Ticket_B$.

The weaknesses of the protocol against the impersonation attacks are due to the fact that; i) in the **Ticket Issuing Stage**, anyone can obtain a valid ticket of A from KDC_A, because there is no device to verify whether the sender is A, i.e., there is no way to verify the received message is a valid encryption under A's password pwa. ii) Similarly, in the **Mutual Authentication and Key Exchange Stage**, B cannot assure that, in the step 3 in the modified C2C-PAKE protocol, the received message is sent from KDC_B, because, unlike the original C2C-PAKE protocol, KDC_B does not use B's password or any secret information between KDC_B and B to generate the messages $g^{xrr'}$ and $g^{rr'}$ for B, and so it is impossible to verify the message is generated by a genuine KDC_B. Thus, in the modified C2C-PAKE protocol, KDC_A and KDC_B do not perform the key distribution centers' role as required.

5 Conclusion

We have shown that the PAKA-X and LKY protocol are insecure against the stolen-verifier attacks. We have shown that the modified C2C-PAKE protocol is insecure against the partition attack and impersonation attacks. These results imply that the protocol is insecure against active adversaries without the knowledge of any secret information such as past session keys and the shared symmetric key.

References

1. S. M. Bellovin and M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, Proc. of the 1992 IEEE Computer Society Conference on Research in security and Privacy, pp. 72-84, 1992.
2. S. M. Bellovin and M. Merritt, Augmented encrypted key exchange: Password-based protocols secure against dictionary attacks and password file compromise, Technical report, AT&T Bell Laboratories, 1994.

3. M. Bellare, D. Pointcheval and P. Rogaway, Authenticated Key Exchange Secure Against Dictionary Attacks, Advances in Cryptography-Eurocrypt'00, LNCS 1807, Springer-Verlag, pp. 139-155, 2000.
4. V. Boyko, P. MacKenzie, and S. Patel, Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman, Advances in Cryptography-Eurocrypt'00, LNCS 1807, Springer-Verlag, pp. 156-171, 2000.
5. J. Byun, I. Jeong, D. Lee and C. Park, Password-authenticated key exchange between clients with different passwords, ICICS'02, LNCS 2513, Springer-Verlag, pp. 134-146, 2004.
6. L. Chen, A weakness of the password-authenticated key exchange between clients with different passwords scheme, The documnet was being circulated for consideration at the 27th SC27/WG2 meeting in Paris, France, 2003-10-20/24(2003).
7. Y. Ding and P. Horster, Undetectable on-line password guessing attacks, ACM Operating Systems Review, vol. 29(4), 1995, pp. 77-86.
8. D. Jablon, Extended password methods immune to dictionary attack, Proc. of the WETICE'97 Enterprise Security Workshop, Cambridge, MA, June 1997.
9. D. Jablon, Strong password-only authenticated key exchange, Computer Communication Review, 26(5), pp. 5-26, 1996.
10. J. Kim, S. Kim, J. Kwak and D. Won, Cryptanalysis and improvment of password-authenticated key exchange between clients with different passwords, ICcsa'04, LNCS 3043, Springer-Verlag, pp. 895-902, 2002.
11. S-W Lee, W-H Kim, H-S Kim, and K-Y Yoo, Efficient password-based authenticated key agreement protocol, ICCSA'04, LNCS 3046, Springer-Verlag, pp. 617-626, 2004.
12. S-W Lee, H-S Kim and K-Y Yoo, Efficient verifier-based key agreement protocol for three parties without server's public key, Applied Mathematics and Computation, in Press.
13. C.-L. Lin, H.-M. Sun and T. Hwang, Three-party encrypted key exchange: attacks and a solution, ACM Operating Systems Review, vol. 34(4), 2000, pp. 12-20.
14. P. MacKenzie and R. Swaminathan, Secure Network Authentication with Password Identification, 1999 Submission to IEEE P1363a.
15. M. Steiner, G. Tsudik and M. Waidner, Refinement and extension of encrypted key exchange, ACM Operating System review, 29(3), July, 1995.
16. T. Wu, The secure remote password protocol, Internet Society Symposium on Network and Distribute System Security, pp. 97-111, 1998.