

Cryptanalysis of Some Client-to-Client Password-Authenticated Key Exchange Protocols

Tianjie Cao^{1,2}, Tao Quan¹, Bo Zhang¹

¹School of Computer, China University of Mining and Technology
Sanhuannanlu, Xuzhou, Jiangsu, 221116, China

²National Mobile Communications Research Laboratory, Southeast University
Sipailou No.2, Nanjing, Jiangsu, 210096, China

Email: tjcao@cumt.edu.cn, banaco@sina.com, cumtzhangbo@163.com

Abstract— Client-to-Client Password-Authenticated Key Exchange (C2C-PAKE) protocols allow two clients establish a common session key based on their passwords. In a secure C2C-PAKE protocol, there is no computationally bounded adversary learns anything about session keys shared between two clients. Especially a participating server should not learn anything about session keys. Server-compromise impersonation resilience is another desirable security property for a C2C-PAKE protocol. It means that compromising the password verifier of any client *A* should not enable outside adversary to share session key with *A*. Recently, Kwon and Lee proposed four C2C-PAKE protocols in the three-party setting, and Zhu et al. proposed a C2C-PAKE protocol in the cross-realm setting. All the proposed protocols are claimed to resist server compromise. However, in this paper, we show that Kwon and Lee's protocols and Zhu et al's protocol exist server compromise attacks, and a malicious server can mount man-in-the-middle attacks and can eavesdrop the communication between the two clients.

Index Terms— Information security, authentication, password, cryptanalysis

I. INTRODUCTION

Password-Authenticated Key Exchange (PAKE) enables two communication entities to authenticate each other and establish a session key via easily memorable passwords. In 1992, Bellare and Merritt [1] proposed a two-party encrypted key exchange protocol based on passwords. Since then, many two-party password-authenticated key exchange (2PAKE) protocols have been proposed.

2PAKE protocols are quite suitable for the client-server architecture. However, in large-scale client-client communication environments where a user wants to communicate with many other users, 2PAKE protocol is very inconvenient in key management that the user would need to remember large number of passwords. Gong, Lomas, Needham, and Saltzer [2] proposed a three-party password-based key transfer protocol using server's public key. Later, Steiner, Tsudik and Waider [3] proposed a three-party client-to-client PAKE (C2C-PAKE) protocol between two clients without server's

public key. The C2C-PAKE protocol is different from the 2PAKE protocol in the fact that the server is assumed besides two communicating clients. Both communicating clients should share respective passwords with the server rather than themselves.

To provide a cross-realm authentication, where clients from one environment (realm) wish to communicate with clients from other realms, Byun et al.[4] presented a four-party C2C-PAKE protocol in the cross-realm setting where two clients are in two different realms and hence two servers involved. The goal of their protocol is that these two clients can establish a common session key based on their passwords shared respectively with two servers.

It is desirable for C2C-PAKE protocols to possess the following security properties:

Key secrecy: There is no computationally bounded adversary learns anything about session keys shared between two honest clients. Especially a participating server should not learn anything about session keys shared between two honest clients [5]. In the cross-realm setting, even the two servers in two different realms sharing information with each other should not learn anything about session keys.

Server-compromise impersonation resilience: Compromising the password verifier of any client *A* should not enable outside adversary to share session key with *A*. There are two type server-compromise impersonation attacks in a C2C-PAKE protocol: (1). An adversary who steals the verifier of client *A* from the server can impersonate any other clients to communicate with *A* without performing dictionary attacks on the verifier of *A*. (2). An adversary who steals the verifier of client *A* from the server can impersonate client *A* to communicate with any other clients without performing dictionary attacks on the verifier of *A*.

Forward secrecy: If the passwords of some clients are compromised, the secrecy of previously established session keys should not be compromised.

Unknown key-share resilience: Client *A* should not be able to coerce into sharing a key with any client *C* when in fact he thinks that he is sharing the key with client *B*.

Off-line dictionary attack resilience: There is no successful adversary as follows: The adversary intercepts communication messages during a protocol execution and stores them locally, and then finds out the password off-line by verifying all candidate passwords via the captured information. In the cross-realm setting, one malicious server should not mount an off-line dictionary attack to obtain the password of a client who belongs to the other realm.

Undetectable on-line dictionary attack resilience: There is no successful adversary as follows: The adversary attempted to use a guessed password in an on-line transaction without being detected. He verified the correctness of his guess by using the response from server. If his guess failed, he starts a new transaction with server using another guessed password. A failed guess can not be logged. That is, a failed guess is never detected by the server and the client, and the adversary can legally and undetectably check the guessed password.

No key control: The secret session key between any two clients is determined by both users taking part in, and none of the two clients can influence the outcome of the secret session key, or enforce the session key to fall into a pre-determined interval.

In [5], Kwon and Lee studied password authenticated key agreement in the three-party setting and proposed four C2C-PAKE protocols, where the client holds password and the server holds the password verifier. They claimed that their three-party password authenticated key agreement protocols are resistant to server compromise and even the trusted third party could not be able to access the agreed key.

In [6], Kim et al. showed that Byun et al.'s C2C-PAKE protocol [4] is vulnerable to a Denning-Sacco-style attack where the adversary is an insider with knowledge of the password of a client in a different realm. Kim et al. also proposed an improved C2C-PAKE protocol. Yoon and Yoo [7] demonstrated that Kim et al.'s C2C-PAKE protocol is vulnerable to one-way man-in-the-middle attack and password-compromise impersonation attack. Yoon and Yoo also presented an enhancement to resolve these problems. In [8], Cao showed that Yoon and Yoo's protocol is vulnerable to server-compromise impersonation attacks. In [9], Xu proved that it is impossible to resist server-compromise attack for designing cross-realm C2C-PAKA protocols based sharing-password between client and server. Recently, Zhu et al. proposed a new C2C-PAKE protocol [10], where the client holds password and the server holds the password verifier.

In this paper, we find that Kwon and Lee's protocols and Zhu et al.'s protocol are still vulnerable to server-compromise impersonation attacks and a malicious server can eavesdrop the communication between the two clients.

II. CRYPTANALYSIS OF KWON AND LEE'S C2C - PAKE PROTOCOLS

A. Description of Kwon and Lee's Protocols

In [5], Kwon and Lee proposed four C2C-PAKE protocols. The conceptual flows of the proposed protocol are depicted in Fig. 1. In Fig. 1, A and B have already agreed on running a protocol to establish their secure channel, and then connect to S separately. This is a *separate mode*. *Relay mode* and *ticket mode* can be derived from basis separate mode. From practice, Kwon and Lee also proposed an alternate mode. In this paper, we only analyze the protocol of separate mode. The same attacks also exist in other modes.

We assume two parties A and B try to establish a secure channel between them, while the trusted third party S corresponds to a password authentication server. For example, A and B are both clients in the domain served by S . Respective passwords are denoted by pw_A and pw_B .

Let κ be a general security parameter (say 160 bits) and l be a special security parameter for public keys (1024 or 2048 bits). A , B , and S should agree on the algebraic parameters related to Diffie-Hellman key agreement such as p , q , and g . As for the prime $p = rq+1$, p and q are secure primes. Let g be a generator of order q in Z_p^* and G_q be the group generated by g . Let us omit 'mod p ' from the expressions that are obvious in Z_p^* . We assume $h(\cdot)$, $f(\cdot)$, $k(\cdot)$, and $kdf(\cdot)$ mean strong one-way hash functions having κ -bit output. For distinguishing the functionality, we denote them by each different name and also we define $h_i(\cdot)$, $f_i(\cdot)$, and $k_i(\cdot)$ where $h_i(\cdot) = h(i, \cdot)$, $f_i(\cdot) = f(i, \cdot)$, and $k_i(\cdot) = k(i, \cdot)$ for integer i . Finally E_j and D_j are respectively encryption and decryption functions for symmetric key j .

A and B register to S by choosing passwords pw_A and pw_B , respectively, in a secure way. S stores $[A, \pi, \nu]$, and $[B, \varpi, \mu]$ in its storage, after computing $\pi = h_0(A, pw_A)$, $\mu = h_1(A, pw_A)$, $\nu = g^\mu$, $\varpi = f_0(B, pw_B)$, $w = f_1(B, pw_B)$, and $\mu = g^w$. In order to hide each plain password from S , it is considerable that A and B , respectively, compute $[A, \pi, \nu]$ and $[B, \varpi, \mu]$ and sends them to S securely for registration.

Let us follow the message flows for A in Fig. 1 and abbreviate those of B since B may perform similar steps.

In the first step, A sends message 1 to S by computing $\pi = h_0(A, pw_A)$, $\mu = h_1(A, pw_A)$, and $X^* = E_\pi(X)$ where $X = g^x$ for random element x .

$$1. A \rightarrow S: A, B, X^*$$

Upon receiving message 1, S may choose random elements a , b , and r , and look up the user profiles for obtaining $[A, \pi, \nu]$ and $[B, \varpi, \mu]$. S can then recover X by decrypting X^* , and compute $V = \nu^r$. Subsequently, S computes $e = h_2(X^*, V)$, and $\alpha = (Xg^e)^a$. S then computes $\chi = X^r$ as well. S computes $\chi^* = E_\alpha(\chi)$ where $\alpha = kdf(\alpha)$. Subsequently S may respond with message 3 by computing $H_A^3 = h_3(\psi, X^*, V, \chi^*, \alpha)$ where $\psi = (A, B, S)$.

$$3. S \rightarrow A: V, \chi^*, H_A^3$$

After receiving message 3, A computes $e' = h_2(X^*, V)$, and $\alpha' = V^{\mu^{-1}(x+e')}$. If $H_A^3 \neq h_3(\psi, X^*, V, \chi^*, \alpha)$, A may abort this protocol. Otherwise, A computes $H_A^4 = h_4(\psi, X^*, V, \chi^*, \alpha)$ and responds with message 5. A also computes $\chi = D_{\alpha'}^{-1}(\chi^*)$ where $\bar{\alpha} = kdf(\alpha)$, and sends message 7 to B .

$$5. A \rightarrow S: H_A^4$$

$$7. A \rightarrow B: \chi$$

Upon receiving message 5, S may log the failed result and abort this protocol if $H_A^4 \neq h_4(\psi, X^*, V, \chi^*, \alpha)$.

B may receive χ from A after running steps 2, 4, and 6 in the similar way. B then computes $\gamma = D_{\beta'}^{-1}(\chi^*)$ where

$\bar{\beta} = kdf(\beta)$. Upon receiving message 7, B can compute

$K_B = \gamma^y$ and $H_B = k_5(\psi, \chi, \gamma, K_B)$, and responds with message 8.

$$8. B \rightarrow A: \gamma, H_B$$

After receiving message 8, A computes $K_A = \gamma^x$, and may abort if $H_B \neq k_5(\psi, \chi, \gamma, K_A)$. Unless B is aborted, A may compute $H_A = k_6(\psi, \chi, \gamma, K_A)$ and responds with message 9.

$$9. A \rightarrow B: H_A$$

Finally B may abort if $H_A \neq k_6(\psi, \chi, \gamma, K_B)$. Thus, if the protocol is not aborted, A and B could agree on $K_A = K_B = g^{xyr}$ and establish their communication channel.

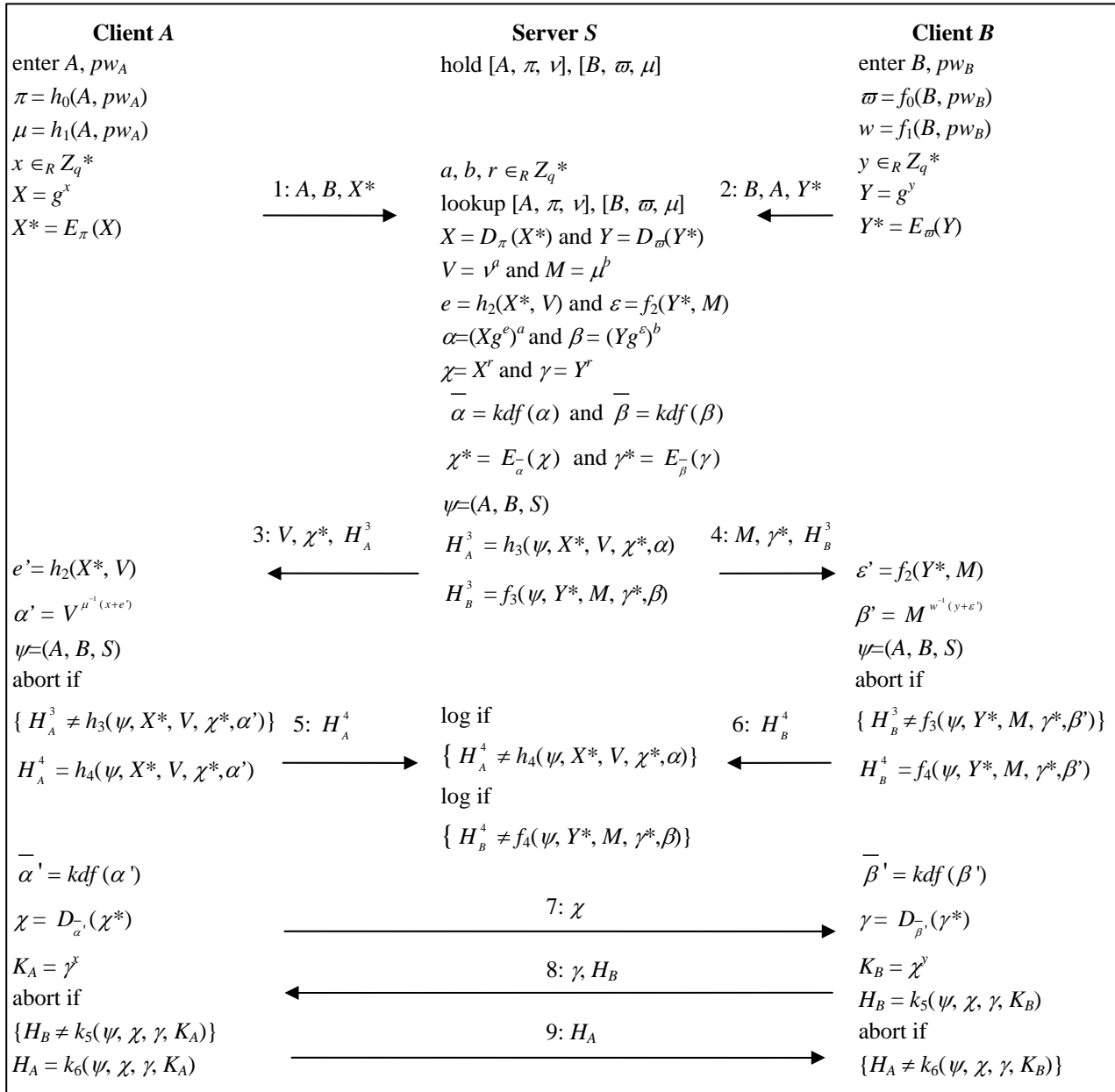


Figure 1. Kwon and Lee's C2C -PAKE protocol

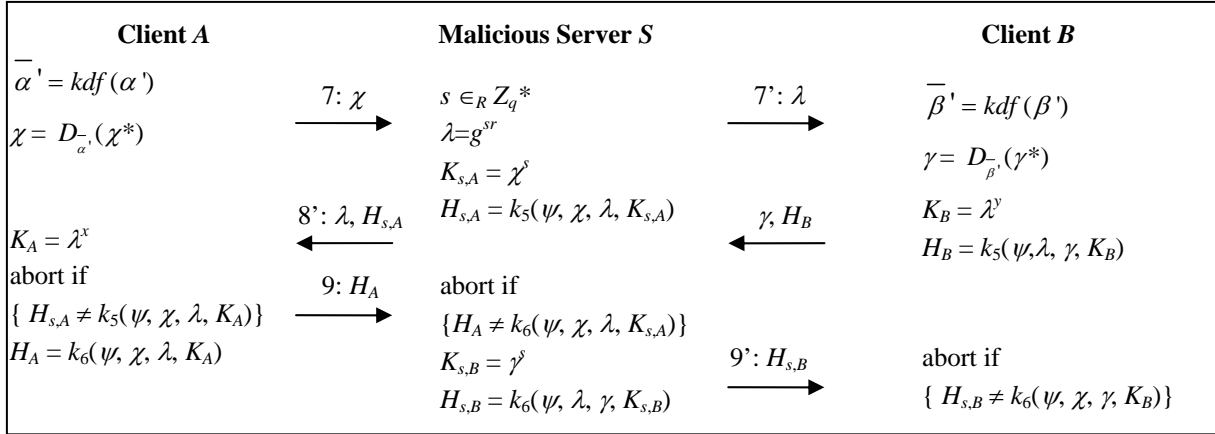


Figure 2. Malicious server's eavesdrop attack

B. Cryptanalysis of Kwon and Lee's Protocols

1) Malicious server's eavesdrop attack

If the server S is a malicious one, it can perform a man-in-the-middle attack between client A and B and then eavesdrop the communication as in the following:

After receiving message 3, A computes $\chi = D_{\alpha'}^-(\chi^*)$, and sends χ to B . The malicious server S intercepts it. S chooses $s \in_R Z_q^*$, computes $\lambda = g^{sr}$, $K_{s,A} = \chi^s$ and $H_{s,A} = k_5(\psi, \chi, \lambda, K_{s,A})$, and sends λ to B .

$$7. A \rightarrow S(B): \chi$$

$$7'. S(A) \rightarrow B: \lambda$$

B then computes $\gamma = D_{\beta'}^-(\gamma^*)$ where $\bar{\beta}' = kdf(\beta)$.

Upon receiving message 7', B can compute $K_B = \lambda^y$ and $H_B = k_5(\psi, \lambda, \gamma, K_B)$, and responds with message 8. S intercepts it and sends λ and $H_{s,A}$ to A .

$$8. B \rightarrow S(A): \gamma, H_B$$

$$8'. S(B) \rightarrow A: \lambda, H_{s,A}$$

After receiving message 8', A computes $K_A = \lambda^x$, and may abort if $H_{s,A} \neq k_5(\psi, \chi, \lambda, K_A)$. A computes $H_A = k_6(\psi, \chi, \lambda, K_A)$ and responds with message 9. S intercepts it and aborts if $H_A \neq k_6(\psi, \chi, \lambda, K_{s,A})$. After that, A and S could agree on $K_A = K_{s,A} = g^{xsr}$ and establish their secure channel. S computes $K_{s,B} = \gamma^s$ and $H_{s,B} = k_6(\psi, \lambda, \gamma, K_{s,B})$, then sends $H_{s,B}$ to B .

$$9. A \rightarrow S(B): H_A$$

$$9'. S(A) \rightarrow B: H_{s,B}$$

Finally B may abort if $H_{s,B} \neq k_6(\psi, \chi, \gamma, K_B)$. Thus, if the protocol is not aborted, S and B could agree on $K_{s,B} = K_B = g^{syr}$ and establish their secure channel. The attack scenario is outlined in Fig. 2.

2) Impersonation-of-responder attack

Once client A 's password verifier $[A, \pi, \nu]$ has stolen by the adversary Malice, for example, the server is compromise. Malice can impersonate client B to communicate with A by performing the following steps. (See Fig.3)

When client A wants to communicate with B , A sends message 1 to S by computing $\pi = h_0(A, pw_A)$, $\mu = h_1(A, pw_A)$, and $X^* = E_{\pi}(X)$ where $X = g^x$ for random element x . Malice intercepts it.

$$1. A \rightarrow \text{Malice (S)}: A, B, X^*$$

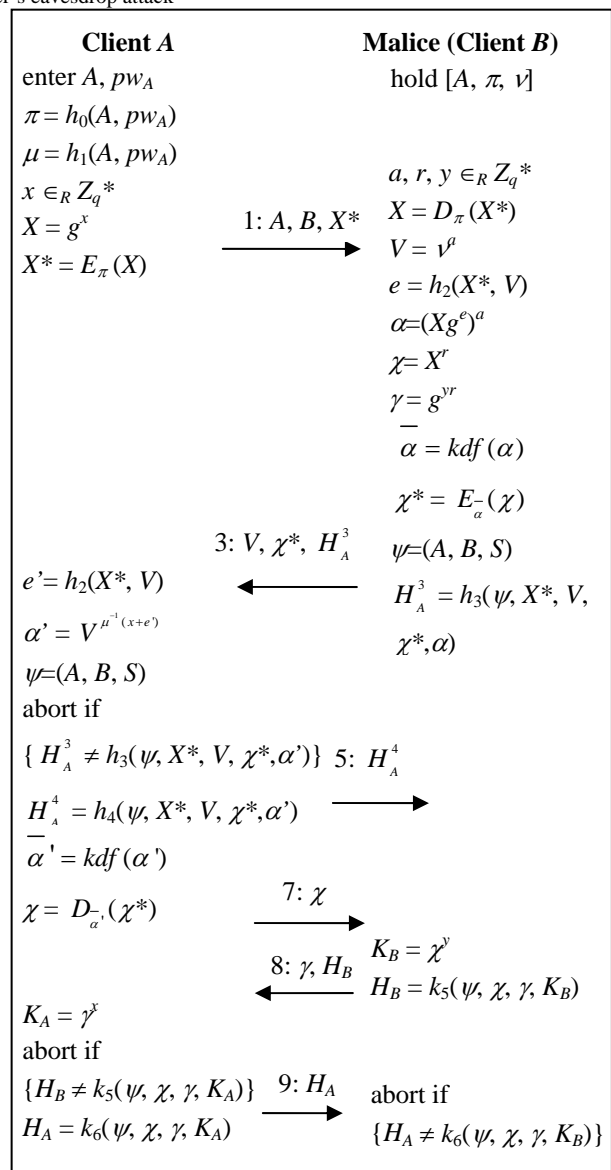


Figure 3. Impersonation-of-responder attack

Upon receiving message 1, Malice poses as S and chooses random elements a, r , and y . Malice can then recover X by decrypting X^* using the compromised verifier π , and compute $V = \nu^r$. Subsequently, Malice computes $e = h_2(X^*, V)$, and $\alpha = (Xg^e)^a$. Malice then

computes $\chi = X^r$ and $\gamma = g^{yr}$. Malice computes $\chi^* = E_{\alpha}(\chi)$ where $\alpha = kdf(\alpha)$. Subsequently Malice may respond with message 3 by computing $H_A^3 = h_3(\psi, X^*, V, \chi^*, \alpha)$ where $\psi = (A, B, S)$.

3. Malice (S)→A: V, χ^*, H_A^3

After receiving message 3, A computes $e' = h_2(X^*, V)$, and $\alpha' = V^{u^{-1}(x+e')}$. If $H_A^3 \neq h_3(\psi, X^*, V, \chi^*, \alpha)$, A may abort this protocol. Otherwise, A computes $H_A^4 = h_4(\psi, X^*, V, \chi^*, \alpha)$ and responds with message 5. A also computes $\chi = D_{\alpha}(\chi^*)$ where $\alpha = kdf(\alpha)$, and sends message 7 to B. Malice intercepts message 5 and 7.

5. A→Malice (S): H_A^4

7. A→Malice (B): χ

Upon receiving message 7, Malice computes $K_B = \chi^y$ and $H_B = k_5(\psi, \chi, \gamma, K_B)$, and responds with message 8.

8. Malice (B)→A: γ, H_B

After receiving message 8, A computes $K_A = \gamma^x$, and may abort if $H_B \neq k_5(\psi, \chi, \gamma, K_A)$. Unless B is aborted, A may compute $H_A = k_6(\psi, \chi, \gamma, K_A)$ and responds with message 9. Malice intercepts it.

9. A→Malice (B): H_A

Finally, A and Malice could agree on $K_A = K_B = g^{xyr}$ and establish their secure channel.

3) Impersonation-of- initiator attack

Once client B's password verifier $[B, \varpi, \mu]$ has stolen by the adversary Malice. Malice can impersonate client A to communicate with B by performing the following steps. (See Fig.4)

First, Malice poses as client A and requests to communicate with B. B sends message 2 to S by computing $\varpi = f_0(B, pw_B)$, $w = f_1(B, pw_B)$, and $Y^* = E_{\varpi}(Y)$ where $Y = g^y$ for random element y. Malice intercepts it.

2. B→Malice (S): B, A, Y^*

Upon receiving message 1, Malice may choose random elements x, b , and r . Malice can then recover Y by decrypting Y^* , and compute $M = \mu^b$. Subsequently, Malice computes $\varepsilon = f_2(Y^*, M)$ and $\beta = (Yg^{\varepsilon})^b$. Malice then computes $\chi = g^{xr}$ and $\gamma = Y^r$. Malice computes $\gamma^* = E_{\beta}(\gamma)$ where $\beta = kdf(\beta)$. Subsequently Malice responds with message 4 by computing $H_B^3 = f_3(\psi, Y^*, M, \gamma^*, \beta)$ where $\psi = (A, B, S)$.

4. Malice (S)→B: M, γ^*, H_B^3

After receiving message 3, B computes $\varepsilon' = f_2(Y^*, M)$, and $\beta' = M^{w^{-1}(y+\varepsilon')}$. If $H_B^3 \neq f_3(\psi, Y^*, M, \gamma^*, \beta')$, B may abort this protocol. Otherwise, B computes $H_B^4 = f_4(\psi, Y^*, M, \gamma^*, \beta')$ and responds with message 6. Malice intercepts it.

6. B→Malice (S): H_B^4

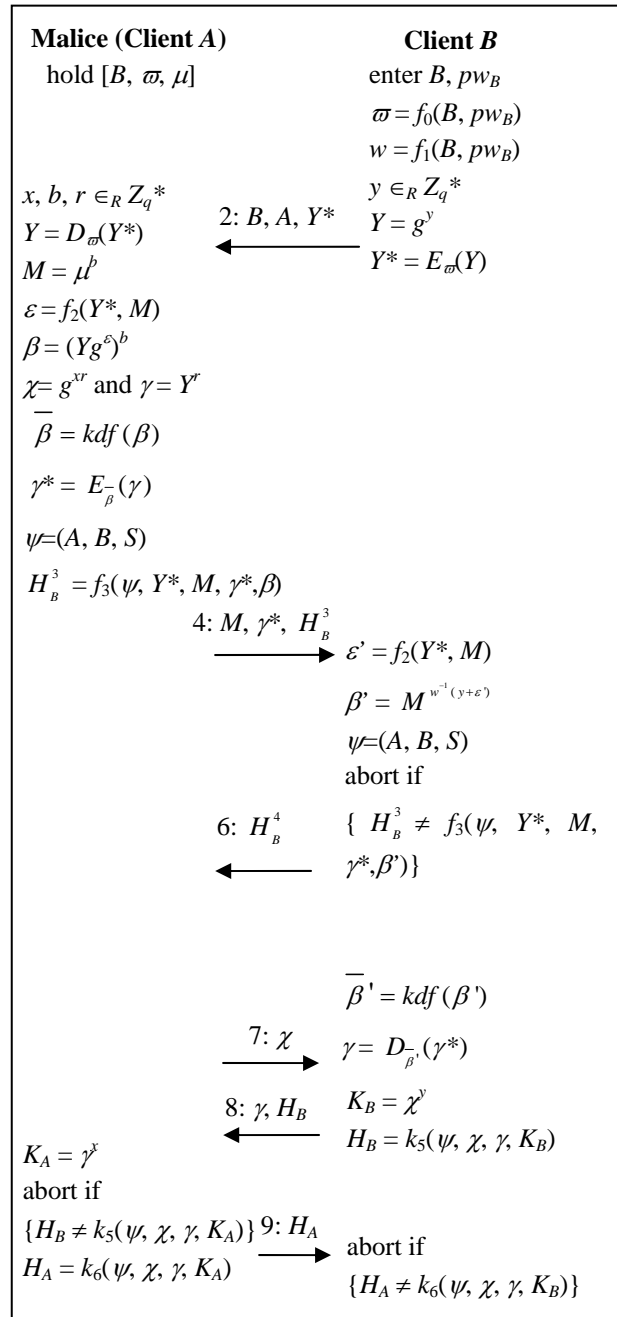


Figure 4. Impersonation-of-initiator attack

Malice sends message 7 to B.

7. Malice (A)→B: χ

B may receive χ from A after running steps 2, 4, and 6.

B then computes $\gamma = D_{\beta'}(\gamma^*)$ where $\beta' = kdf(\beta')$. Upon receiving message 7, B can compute $K_B = \chi^y$ and $H_B = k_5(\psi, \chi, \gamma, K_B)$, and responds with message 8. Malice intercepts it.

8. B→Malice (A): γ, H_B

After receiving message 8, Malice computes $K_A = \gamma^x$, and may abort if $H_B \neq k_5(\psi, \chi, \gamma, K_A)$. Unless B is aborted, A may compute $H_A = k_6(\psi, \chi, \gamma, K_A)$ and responds with message 9.

9. Malice (A)→B: H_A

Finally B may abort if $H_A \neq k_6(\psi, \chi, \gamma, K_B)$. Thus, if the protocol is not aborted, Malice and B could agree on $K_A = K_B = g^{xyr}$ and establish their secure channel.

III. CRYPTANALYSIS OF ZHU ET AL.'S C2C-PAKE PROTOCOL

A. Description of Zhu et al.'s C2C-PAKE Protocol

We assume that user A and the KDC_A have shared the verifiers $v_{A,1} = g_1^{H(A||KDC_A||pw_A)}$ and $v_{A,2} = g_2^{H(A||KDC_A||pw_A)}$ for password pw_A and the public information. User B and the KDC_B have also shared the verifiers $v_{B,1} = g_1^{H(B||KDC_B||pw_B)}$ and $v_{B,2} = g_2^{H(B||KDC_B||pw_B)}$ for password pw_B and the public information.

A message authentication code (MAC) algorithm consists of three algorithm, $M = (\text{KEY.G}, \text{MAC.G}, \text{MAC.V})$. KEY.G generates a key k_{mac} . Given k_{mac} , MAC.G computes a tag $\tau = \text{MAC.G}_{k_{mac}}(M)$ 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.

Assume A wants to establish a session key with B . The protocol is shown in Fig. 5.

Round 1

The initiator A broadcasts (A, B, KDC_A, KDC_B) .

Round 2

(1) A chooses a random number $x_A \in Z_q^*$, computes

$$X_A = g_1^{x_A} v_{A,2}, \text{ and sends } (A, X_A) \text{ to } KDC_A.$$

(2) B analogously computes $X_B = g_1^{x_B} v_{B,2}$, and sends (B, X_B) to KDC_B .

(3) KDC_A selects random numbers $y_A, z_A \in Z_q^*$, computes $Y_A = g_1^{y_A} (v_{A,1})^{z_A}$ and $Z_A = g_1^{z_A} v_{A,2}$, and sends $(KDC_A; Y_A; Z_A)$ to A . (Analogously KDC_B sends $(KDC_B; Y_B; Z_B)$ to B .)

Round 3

(1) Upon receiving $(KDC_A; Y_A; Z_A)$, A computes $T_A = (Z_A / v_{A,2})^{H(A||KDC_A||pw_A)}$ and $k_A = (Y_A / T_A)^{x_A}$.

(2) Upon receiving $(KDC_B; Y_B; Z_B)$, B analogously computes $k_B = (Y_B / T_B)^{x_B}$.

(3) Upon receiving (A, X_A) , KDC_A computes $k_A = (X_A / v_{A,2})^{y_A}$. (Analogously KDC_B computes $k_B = (X_B / v_{B,2})^{y_B}$.)

(4) A computes $\tau_{A,KDC_A} = \text{MAC.G}_{k_A}(A || KDC_A || X_A || Y_A || Z_A)$ and sends it to KDC_A . (B analogously computes τ_{B,KDC_B} and sends it to KDC_B .)

Round 4

(1) Upon receiving τ_{A,KDC_A} , KDC_A computes $\text{MAC.V}_{k_A}(\tau_{A,KDC_A})$. KDC_A terminates if MAC.V returns 0 or moves to the next phase otherwise. (KDC_B analogously checks the validity of τ_{B,KDC_B} using k_B .)

(2) KDC_A chooses a random number $a \in Z_q^*$, computes $M_A = E_K(KDC_A || g_1^a || g_1^{x_A})$ using K and sends M_A to KDC_B . (KDC_B analogously computes $M_B = E_K(KDC_B || g_1^b || g_1^{x_B})$ and sends M_B to KDC_A .)

Round 5

(1) Upon receiving M_B , KDC_A decrypts M_B by using pre-distributed key K . Then KDC_A computes $c = (g_1^b)^a$, $S_A = (g_1^{x_B})^c$ and $\tau_{KDC_A,A} = \text{MAC.G}_{k_A}(A || KDC_A || B || S_A)$ and sends them to A .

(2) Upon receiving M_A , KDC_B decrypts M_A by using pre-distributed key K . Then KDC_B computes $c = (g_1^a)^b$, $S_B = (g_1^{x_A})^c$ and $\tau_{KDC_B,B} = \text{MAC.G}_{k_B}(B || KDC_B || A || S_B)$ and sends them to B .

Key computation

Upon receiving $(S_A, \tau_{KDC_A,A})$, A terminates if $\text{MAC.V}_{k_A}(\tau_{KDC_A,A})$ returns 0 or computes $K_A = (S_A)^{x_A}$ and the session key $sk_A = F_{K_A}(A || KDC_A || KDC_B || B)$. (B analogously computes $K_B = (S_B)^{x_B}$ and $sk_B = F_{K_B}(A || KDC_A || KDC_B || B)$.)

B. Cryptanalysis of Zhu's Protocol

1) Malicious server's eavesdrop attack

If the server KDC_A is a malicious one, it can perform a man-in-the-middle attack between client A and B in Round 4 and then eavesdrop the communication as in the following:

Round 4

(1) Upon receiving τ_{A,KDC_A} , KDC_A computes $\text{MAC.V}_{k_A}(\tau_{A,KDC_A})$. KDC_A terminates if MAC.V returns 0 or moves to the next phase otherwise. (KDC_B analogously checks the validity of τ_{B,KDC_B} using k_B .)

(2) KDC_A chooses a random number $a, r \in Z_q^*$, computes $M_A = E_K(KDC_A || g_1^a || g_1^r)$ using K and sends M_A to KDC_B . KDC_B analogously computes $M_B = E_K(KDC_B || g_1^b || g_1^{x_B})$ and sends M_B to KDC_A .

Round 5

(1) Upon receiving M_B , KDC_A decrypts M_B by using pre-distributed key K . Then KDC_A computes $c = (g_1^b)^a$, $S_A = g_1^r$ and $\tau_{KDC_A,A} = \text{MAC.G}_{k_A}(A || KDC_A || B || S_A)$ and sends them to A .

(2) Upon receiving M_A , KDC_B decrypts M_A by using pre-distributed key K . Then KDC_B computes $c = (g_1^a)^b$, $S_B = (g_1^r)^c$ and $\tau_{KDC_B,B} = \text{MAC.G}_{k_B}(B || KDC_B || A || S_B)$ and sends them to B . KDC_A

Key computation

Upon receiving $(S_A, \tau_{KDC_A,A})$, A terminates if $\text{MAC.V}_{k_A}(\tau_{KDC_A,A})$ returns 0 or computes $K_A = (S_A)^{x_A}$ and the session key $sk_A = F_{K_A}(A || KDC_A || KDC_B || B)$. B

analogously computes $K_B = (S_B)^{x_B}$ and $sk_B = F_{K_B}(A \parallel KDC_A \parallel KDC_B \parallel B)$. KDC_A computes $K_{M,A} = (X_A / v_{A,2})^r$, $sk_{M,A} = F_{K_{M,A}}(A \parallel KDC_A \parallel KDC_B \parallel B)$, $K_{M,B} = (g_1^{x_B})^{cr}$ and $sk_{M,B} = F_{K_{M,B}}(A \parallel KDC_A \parallel KDC_B \parallel B)$. KDC_A and A could agree on $sk_{M,A} = sk_A$ because $K_{M,A} = K_A$

$= g_1^{x_A r}$ and establish their secure channel. KDC_A and B could agree on $sk_{M,B} = sk_B$ because $K_{M,B} = K_B = g_1^{x_B cr}$ and establish their secure channel.

The attack scenario is outlined in Fig. 6

Public information: $G, p, q, g_1, g_2, H, M, F$			
Client A	KDC_A	KDC_B	Client B
pw_A	$v_{A,1} = g_1^{H(A \parallel KDC_A \parallel pw_A)}$ $v_{A,2} = g_2^{H(A \parallel KDC_A \parallel pw_A)}$	$v_{B,1} = g_1^{H(B \parallel KDC_B \parallel pw_B)}$ $v_{B,2} = g_2^{H(B \parallel KDC_B \parallel pw_B)}$	pw_B
Round 2 $x_A \in Z_q^*$ $X_A = g_1^{x_A} v_{A,2}$	$y_A, z_A \in Z_q^*$ $Y_A = g_1^{y_A} (v_{A,1})^{z_A}$ $Z_A = g_1^{z_A} v_{A,2}$	$y_B, z_B \in Z_q^*$ $Y_B = g_1^{y_B} (v_{B,1})^{z_B}$ $Z_B = g_1^{z_B} v_{B,2}$	$x_B \in Z_q^*$ $X_B = g_1^{x_B} v_{B,2}$
Round 3 $k_A = (g_1^{y_A})^{x_A}$ $\tau_{A,KDC_A} = \text{MAC.G}_{k_A}(A \parallel KDC_A \parallel X_A \parallel Y_A \parallel Z_A)$	$k_A = (g_1^{x_A})^{y_A}$	$k_B = (g_1^{y_B})^{x_B}$	$k_B = (g_1^{y_B})^{x_B}$ $\tau_{B,KDC_B} = \text{MAC.G}_{k_B}(B \parallel KDC_B \parallel X_B \parallel Y_B \parallel Z_B)$
Round 4	$\text{MAC.V}_{k_A}(\tau_{A,KDC_A}) ?= 1$ $a \in Z_q^*$ $M_A = E_K(KDC_A \parallel g_1^a \parallel g_1^{x_A})$	$\text{MAC.V}_{k_B}(\tau_{B,KDC_B}) ?= 1$ $b \in Z_q^*$ $M_B = E_K(KDC_B \parallel g_1^b \parallel g_1^{x_B})$	
Round 5	$c = (g_1^b)^a, S_A = (g_1^{x_A})^c$ $\tau_{KDC_A,A} = \text{MAC.G}_{k_A}(A \parallel KDC_A \parallel B \parallel S_A)$	$c = (g_1^a)^b, S_B = (g_1^{x_B})^c$ $\tau_{KDC_B,B} = \text{MAC.G}_{k_B}(B \parallel KDC_B \parallel A \parallel S_B)$	
Key computation	$\text{MAC.V}_{k_A}(\tau_{KDC_A,A}) ?= 1$ $K_A = (S_A)^{x_A}$ $sk_A = F_{K_A}(A \parallel KDC_A \parallel KDC_B \parallel B)$	$\text{MAC.V}_{k_B}(\tau_{KDC_B,B}) ?= 1$ $K_B = (S_B)^{x_B}$ $sk_B = F_{K_B}(A \parallel KDC_A \parallel KDC_B \parallel B)$	

Figure 5. Zhu et al's C2C-PAKE Protocol

Client A	Malicious KDC_A	KDC_B	Client B
Round 4	$\text{MAC.V}_{k_A}(\tau_{A,KDC_A}) ?= 1$ $a, r \in Z_q^*$ $M_A = E_K(KDC_A \parallel g_1^a \parallel g_1^r)$	$\text{MAC.V}_{k_B}(\tau_{B,KDC_B}) ?= 1$ $b \in Z_q^*$ $M_B = E_K(KDC_B \parallel g_1^b \parallel g_1^{x_B})$	
Round 5	$c = (g_1^a)^b, S_A = g_1^r$ $\tau_{KDC_A,A} = \text{MAC.G}_{k_A}(A \parallel KDC_A \parallel B \parallel S_A)$	$c = (g_1^a)^b, S_B = (g_1^r)^c$ $\tau_{KDC_B,B} = \text{MAC.G}_{k_B}(B \parallel KDC_B \parallel A \parallel S_B)$	
Key computation	$\text{MAC.V}_{k_A}(\tau_{KDC_A,A}) ?= 1$ $K_{M,A} = (X_A / v_{A,2})^r$ $sk_{M,A} = F_{K_{M,A}}(A \parallel KDC_A \parallel KDC_B \parallel B)$ $K_{M,B} = (g_1^{x_B})^{cr}$ $sk_{M,B} = F_{K_{M,B}}(A \parallel KDC_A \parallel KDC_B \parallel B)$	$\text{MAC.V}_{k_B}(\tau_{KDC_B,B}) ?= 1$ $K_B = (S_B)^{x_B}$ $sk_B = F_{K_B}(A \parallel KDC_A \parallel KDC_B \parallel B)$	

Figure 6. Malicious server's eavesdrop attack

2) Impersonation-of-responder attack

Once client A's password verifier $v_{A,1}$ and $v_{A,2}$ have stolen by the adversary Malice. Malice can impersonate client B to communicate with A by performing the following steps.

Round 1

The initiator A sends (A, B, KDC_A, KDC_B) to client B, KDC_A and KDC_B . Malice intercepts it.

Round 2

(1) A chooses a random number $x_A \in Z_q^*$, computes

$X_A = g_1^{x_A} v_{A,2}$, and sends (A, X_A) to KDC_A . Malice intercepts it.

(2) Malice selects random numbers $x_B, y_A, z_A \in Z_q^*$, computes $Y_A = g_1^{y_A} (v_{A,1})^{z_A}$ and $Z_A = g_1^{z_A} v_{A,2}$, and sends $(KDC_A; Y_A; Z_A)$ to A.

Round 3

(1) Upon receiving $(KDC_A; Y_A; Z_A)$, A computes $T_A = (Z_A / v_{A,2})^{H(A \| KDC_A \| pw_A)}$ and $k_A = (Y_A / T_A)^{x_A}$.

(3) Upon receiving (A, X_A) , Malice computes $k_A = (X_A / v_{A,2})^{y_A}$.

(4) A computes $\tau_{A,KDC_A} = \text{MAC}_{G_{k_A}}(A \| KDC_A \| X_A \| Y_A \| Z_A)$ and sends it to KDC_A . Malice intercepts it.

Round 5

Malice computes $S_A = g_1^{x_B}$ and $\tau_{KDC_A, A} = \text{MAC}_{G_{k_A}}(A \| KDC_A \| B \| S_A)$ and sends them to A.

Key computation

Upon receiving $(S_A, \tau_{KDC_A, A})$, A terminates if $\text{MAC}_{V_{k_A}}(\tau_{KDC_A, A})$ returns 0 or computes $K_A = (S_A)^{x_A}$ and the session key $sk_A = F_{K_A}(A \| KDC_A \| KDC_B \| B)$. Malice computes $K_B = (X_A / v_{A,2})^{x_B}$ and the session key $sk_B = F_{K_B}(A \| KDC_A \| KDC_B \| B)$.

Once client B's password verifier has stolen by the adversary Malice. Malice can also impersonate client A to communicate with B by performing the similar steps.

IV. CONCLUSION

Client-to-Client Password-Authenticated Key Exchange (C2C-PAKE) protocols allow two clients establish a common session key based on their passwords. Such protocols are attractive for their simplicity and convenience and have received much interest in the research community. Recently, Kwon and Lee proposed four C2C-PAKE protocols in the three-party setting, and Zhu et al. proposed a C2C-PAKE protocol in the cross-realm setting. All the proposed protocols are claimed to resist server compromise. However, in this paper, we have shown that Kwon and Lee's protocols and Zhu et al's protocol exist server compromise attacks, and a malicious server can mount

man-in-the-middle attacks and can eavesdrop the communication between the two clients. Now, how to design a C2C-PAKE protocols that resist against all known attacks without requiring any server's public key is still an open problem.

ACKNOWLEDGMENT

This work is supported by the Jiangsu Provincial Natural Science Foundation of China (BK2007035), the open research fund of National Mobile Communications Research Laboratory, Southeast University (W200817) and the Science and Technology Foundation of CUMT (0D080309).

REFERENCES

- [1] S. M. Bellovin and M. Merrit, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 72-84, 1992.
- [2] L. Gong, M. Lomas, R. Needham and J. Saltzer, "Protecting poorly chosen secrets from guessing attacks," *IEEE Journal on Selected Areas in Communications*, 11(5), pp. 648-656, 1993.
- [3] M. Steiner, G. Tsudik and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM Operating Systems Review*, vol. 29 (3), pp. 22-30, 1995.
- [4] J.W. Byun, I.R. Jeong, D.H. Lee and C.S. Park, "Password-authenticated key exchange between clients with different passwords," in *Proceedings of ICICS 2002*, LNCS 2513, pp. 134-146, 2002.
- [5] T. Kwon and D. H. Lee, "Three-party password authenticated key agreement resistant to server compromise", *WISA 2006*, LNCS 4298, pp. 312-323, 2007.
- [6] J. Kim, S. Kim, J. Kwak and D. Won, "Cryptanalysis and improvements of password authenticated key exchange scheme between clients with different passwords," in *Proceedings of ICCSA 2004*, LNCS3044, pp. 895-902, 2004.
- [7] E.-J. Yoon and K.-Y. Yoo, "A secure password-authenticated key exchange between clients with different passwords," in *Proceedings of APWeb Workshops 2006*, LNCS 3842, pp. 659-663, 2006.
- [8] T. Cao and Y. Zhang, "Cryptanalysis of two password-authenticated key exchange protocols between clients with different passwords," *International Mathematical Forum*, vol. 2(11), pp. 525-532, 2007.
- [9] J. Xu, Z. Zhang and D. Feng, "Analysis and improvement of client-to-client password-authenticated key exchange protocols," in *Proceedings of Security Protocols*, SKLOIS, pp. 119-124, 2007.
- [10] H. Zhu, T. Liu, J. Liu and G. Chang, "An efficient client-to-client password-authenticated key exchange resilient to server compromise," *13th IEEE International Symposium on Pacific Rim Dependable Computing*, pp. 405-408, 2007.