A novel multi-server authentication protocol

Yalin Chen [1] , Chun-Hui Huang [2] , *Jue-Sam Chou [3]

[1] Institute of information systems and applications, National Tsing Hua University

d949702@oz.nthu.edu.tw

[2] Department of Information Management, Nanhua University Chiayi 622 Taiwan, R.O.C

g6451519@mail.nhu.edu.tw

[3] Department of Information Management, Nanhua University Chiayi 622 Taiwan, R.O.C

*: corresponding author

jschou@mail.nhu.edu.tw

Tel: 886+ (0)5+272-1001 ext.56536

Abstract

Recently, Tsai and Hsiang *et al.* each proposed a multi-server authentication protocol. They claimed their protocols are secure and can withstand various attacks. However, after our analysis, we found some security loopholes in each protocol. We will first show the attacks on their schemes and then present ours. After the security analysis, we conclude that our scheme is the most secure one among all of the proposed protocols in multi-server environments nowadays.

Keywords: *multi-server, password authentication protocol, smart card, password change, key agreement*

1. Introduction

For password-based authentication protocols using smart cards are widely used in an open network. A two-party password authentication protocol for client-server architecture is therefore not sufficient when networks are getting larger and larger. Consequently, several multi-server protocols were proposed [1-15].

In 2003, Li *et al*. [5] proposed a multi-server protocol based on ElGamal digital signature and geometric transformations on an Euclidean plane. Unfortunately, their protocol is vulnerable and has been broken by Cao and Zhong [8]. In 2004 and 2005, Tsaur *et al*. [3, 4] proposed two multi-server schemes. However, both of their schemes are based on Lagrange interpolating polynomial which is computationally intensive. In 2006 and 2007, Cao *et al*. [9] and Hu *et al*. [7] each proposed an authentication scheme for a multi-server environment. Both of their schemes assume that all servers are trustworthy. Nevertheless, this assumption is not always true as stated in [1]. In 2008, Lee *et al*. [6] proposed an authenticated key agreement scheme for multi-server using mobile equipment. However, their scheme can not add a server freely. Because when a server is added, all users who want to login to this new server have to re-register at the registration center for getting a new smart card. This increases the

registration center's card-issue cost. Also, in 2008, Tsai [1] proposed an efficient multi-server authentication scheme. He claims that his protocol can withstand seven known attacks. Yet, after our analysis, we found that it is vulnerable to the server spoofing attack**.** Recently, in 2009, Liao *et al*. [2] proposed a secure dynamic ID scheme for multi-server environment. They claim that their protocol is secure. However, Hsiang and Shih [14] found their scheme suffers from both the server spoofing attack and the insider attack. Hence, they proposed an improvement on the protocol. We also illustrate the same weakness of Liao *et al*.'s scheme in eprint [15]. Yet, we found that their improvement is still insecure. It is vulnerable to the insider attack. In this paper, we will first show the attacks on [1] and [14], respectively. Then, we show our scheme and examine its security.

The remainder of this paper is organized as follows: In Section 2, we review both Tsai's and Hsiang-Shih's protocols. In Section 3, we demonstrate the vulnerabilities existing in their schemes, respectively. Then, we propose a novel protocol in Section 4 and analyze its security in Section 5. The discussions and comparisons are made in Section 6. Finally, a conclusion is given in Section 7.

2. Review of Tsai's and Hsiang-Shih's protocols

In this section, we review Tsai's protocol in Section 2.1 and Hsiang-Shih's protocol in Section 2.2, respectively. Before that, the notations used throughout this paper are first defined as follows.

RC : the registration center
U_u : a legal user u
U_n : a legal malicious user n
S_j : a legal server j
E(P) : an attacker E who masquerades as a peer P.
SID_j : the identity of S_j
ID_u : the identity of U_u
PW_u : the password of U_u
x,y,r : RC's secret keys
Flag : a word string which is set to '*the first time login*', '*not the first time login*', '*for password change*' or '*accept*' used in our proposed protocol in Section 4.3.
g : the primitive element in a Galois field GF(p) where p is a large prime number.
H( ) : a collision-resistant one-way hash function
(*a*,*b*) : a string denotes that string *a* is concatenated with string *b*.
$\oplus$: a bitwise XOR operator
=> : a secure channel
$\rightarrow$: a common channel

2.1 Review of Tsai's protocol

Tsai's protocol assumes s servers exists in the system. At the beginning, RC computes and sends $H(SID_j,y)$ to each S_j, for j = 1 to s, with S_j keeping it secret, via a secure channel. Tsai's protocol contains four phases. They are: (1)registration phase, (2)login phase, (3)authentication of server and RC phase, and (4)authentication of server and user phase. We describe them as follows.

(1) Registration phase

In this phase (as shown in Fig. 1), U_u performs the following steps for obtaining a smart card from RC.

1. U_u freely chooses his ID_u and PW_u and calculates $H(PW_u)$. He then sends $\{ID_u, H(PW_u)\}$ to RC through a secure channel.
2. RC calculates $B=H(ID_u, x)\oplus H(PW_u)$ and issues U_u a smart card containing ID_u and B through a secure channel.

(2) Login phase

When U_u wants to login to S_j, he inserts his smart card and performs the following steps.

1. U_u keys his ID_u and PW_u and generates a random nonce Nc. He then computes $C_1 =(B\oplus H(PW_u))\oplus Nc = H(ID_u, x)\oplus Nc$.
2. U_u sends $\{ID_u, C_1\}$ to S_j.

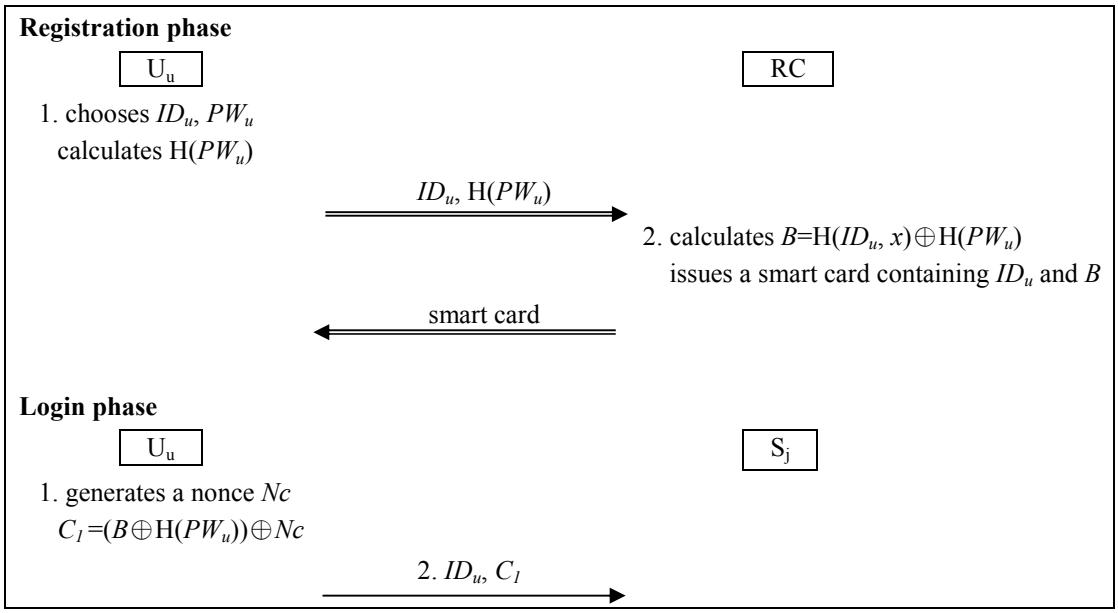


Fig. 1. Registration phase and login phase of Tsai's protocol

(3) Authentication of server and RC phase

In this phase (as shown in Fig. 2), after receiving message $\{ID_u, C_1\}$ from U_u, S_j runs the following steps to let himself be authenticated by RC, verify U_u's legitimacy, and negotiate the session key with U_u. Here, let the secret key shared between S_j and RC be $H(H(SID_j, y), Ns+1, N_{RC}+2)$, where Ns and N_{RC} are S_j's and RC's randomly

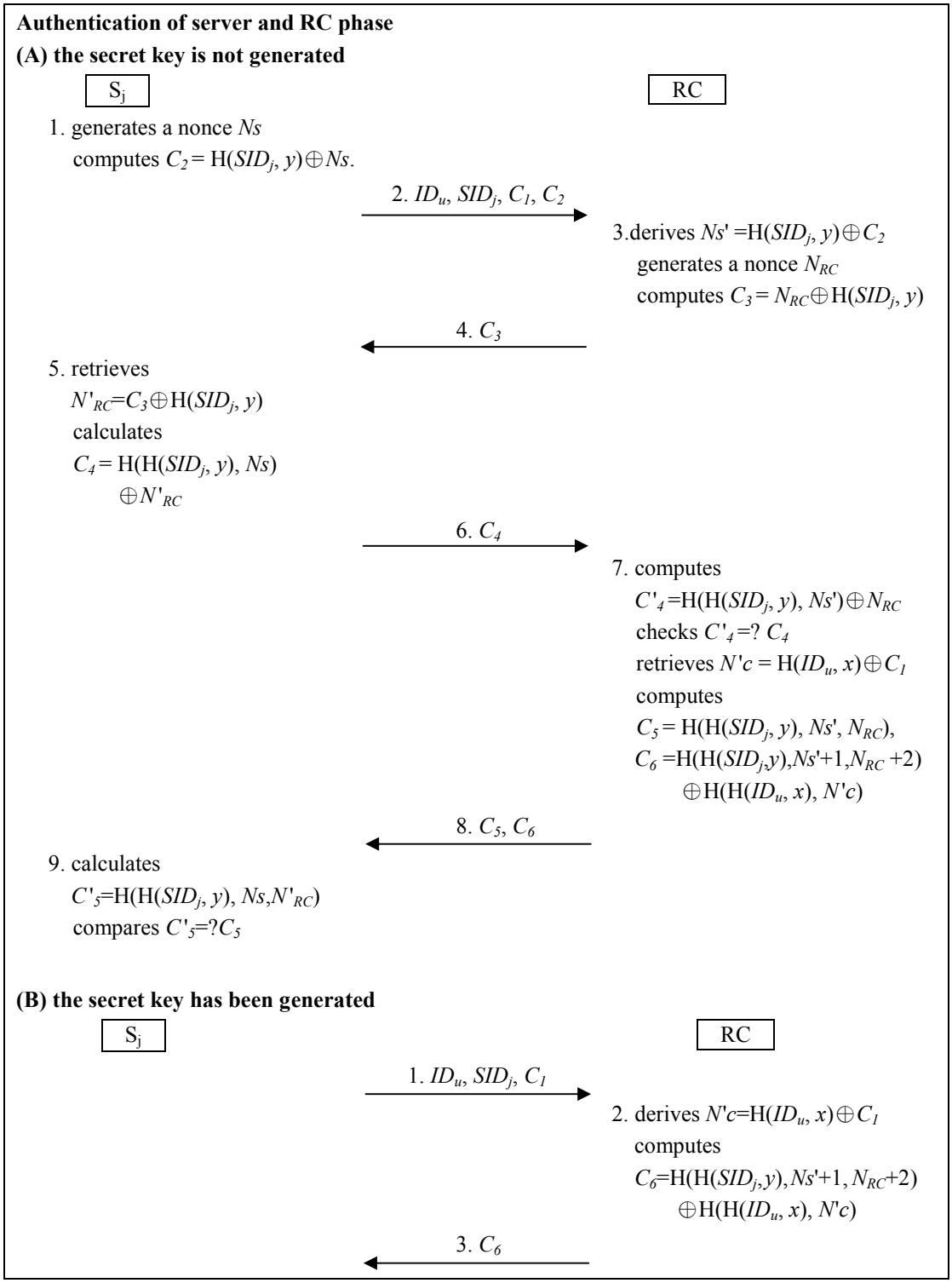


Fig. 2. Authentication of server and RC phase of Tsai's protocol

chosen nonces respectively. To reduce the computational cost, this phase is divided into two scenarios: (A) the secret key is not generated, and (B) the secret key has been generated. We describe them below.

(A) the secret key is not generated.

1. S_j generates a random nonce Ns and computes $C_2 = H(SID_j, y) \oplus Ns$.
2. S_j sends $\{ID_u, SID_j, C_1, C_2\}$ to RC.
3. RC derives $Ns' = H(SID_j, y) \oplus C_2$. He then generates a random nonce N_{RC} and computes $C_3 = N_{RC} \oplus H(SID_j, y)$.
4. RC sends $\{C_3\}$ to S_j.
5. After receiving the message from RC, S_j retrieves $N'_{RC} = C_3 \oplus H(SID_j, y)$ and calculates $C_4 = H(H(SID_j, y), Ns) \oplus N'_{RC}$.
6. S_j sends $\{C_4\}$ to RC.
7. RC computes $C'_4 = H(H(SID_j, y), Ns') \oplus N_{RC}$ and checks to see if C'_4 is equal to the received C_4. If so, S_j is authentic. He then retrieves $N'c = H(ID_u, x) \oplus C_1$ and computes $C_5 = H(H(SID_j, y), Ns', N_{RC})$, $C_6 = H(H(SID_j, y), Ns'+1, N_{RC}+2) \oplus H(H(ID_u, x), N'c)$.
8. RC sends $\{C_5, C_6\}$ to S_j.
9. After receiving the message from RC, S_j calculates $C'_5 = H(H(SID_j, y), Ns, N'_{RC})$ and compares it with the received C_5. If they are equal, RC is authentic. Both S_j and RC will store the common secret key $Auth_{S\text{-}RC} = H(H(SID_j, y), Ns+1, N'_{RC}+2)$ for next execution of server and RC authentication to reduce the computational cost in the verifier table.

(B) the secret key has been generated.

1. S_j sends $\{ID_u, SID_j, C_1\}$ to RC.
2. RC derives $N'c = H(ID_u, x) \oplus C_1$ and uses his $Auth_{S\text{-}RC}$ to compute $C_6 = H(H(SID_j, y), Ns'+1, N_{RC}+2) \oplus H(H(ID_u, x), N'c)$.
3. RC sends $\{C_6\}$ to S_j.

(4) Authentication of server and user phase

After the authentication of server and RC phase, S_j and U_u perform the following steps for mutual authentication (as shown in Fig. 3).

1. S_j generates a random nonce N_{SU} and uses his $Auth_{S\text{-}RC}$ to compute $C_7 = C_6 \oplus H(H(SID_j, y), Ns+1, N'_{RC}+2) = H(H(ID_u, x), N'c)$. He then calculates $C_8 = C_1 \oplus C_7$, $V_2 = C_7 \oplus N_{SU}$, and $C_9 = H(C_7, N_{SU}) \oplus C_8$.
2. S_j sends $\{V_2, C_9\}$ to U_u.
3. After receiving the message, U_u computes $C'_7 = H(H(ID_u, x), Nc)$, retrieves $N'_{SU} =$

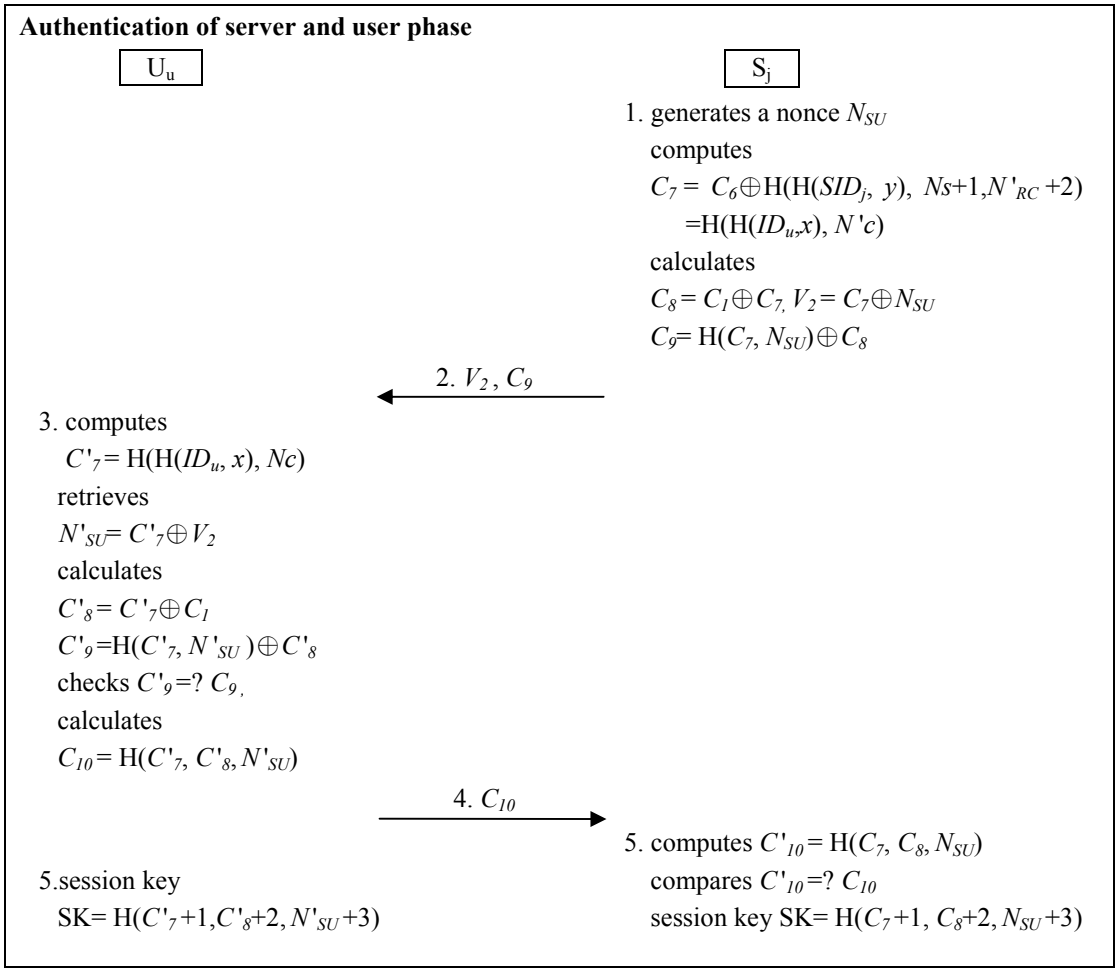


Fig. 3. Authentication of server and user phase of Tsai's protocol

$C'_7 \oplus V_2$, and calculates $C'_8 = C'_7 \oplus C_1$, $C'_9 = H(C'_7, N'_{SU}) \oplus C'_8$. He then checks to see if the newly computed C'_9 is equal to the received C_9. If so, S_j is authentic. U_u then calculate $C_{10} = H(C'_7, C'_8, N'_{SU})$.

4. U_u sends $\{C_{10}\}$ to S_j.
5. After receiving $\{C_{10}\}$, S_j computes $C'_{10} = H(C_7, C_8, N_{SU})$ and checks to see if C'_{10} is equal to the received C_{10}. If so, U_u is authentic. They then have the same session key SK= $H(C'_7+1, C'_8+2, N'_{SU}+3)$ and SK= $H(C_7+1, C_8+2, N_{SU}+3)$, respectively.

2.2 Review of Hsiang-Shih's protocol

In this section, we review Hsiang-Shih's protocol. Their protocol consists of four phases: (1) registration phase, (2) login phase, (3) mutual verification and session key agreement phase, and (4) password change phase. In their protocol, RC first computes and sends $H(SID_j, y)$ to legal servers S_j, for j=1 to w. (Assume that there are w servers in the system.) We describe their protocol as follows and also depict it in Figure 4.

(1) Registration phase

In this phase, U_u performs the following steps to register at RC for obtaining a smart card so that he can access the resources of all servers.

1. Chooses PW_u, a random number b_u, and computes $H(b_u \oplus PW_u)$. He then sends $\{ID_u, H(b_u, PW_u)\}$ to RC through a secure channel.
2. RC computes $T_u=H(ID_u, x)$, $V_u=T_u \oplus H(ID_u, H(b_u \oplus PW_u))$, $A_u=H(H(b_u \oplus PW_u), r) \oplus H(x \oplus r)$, $B_u= A_u \oplus H(b_u \oplus PW_u)$, $R_u= H(H(b_u \oplus PW_u), r)$, and $H_u=H(T_u)$. He then stores V_u, B_u, H_u and R_u to the smart card and issues the card to U_u through a secure channel.
3. U_u enters b_u to his card and the smart card now contains b_u, V_u, B_u, H_u and R_u.

(2) Login phase

When U_u wants to login to S_j, he inserts his smart card and performs the following steps.

1. U_u keys his ID_u, PW_u and SID_j to the smart card. The smart card computes $T_u'=V_u \oplus H(ID_u, H(b_u \oplus PW_u))$, $H_u'=H(T_u')$, and checks to see if H_u stored is equal to the computed H_u'. If so, smart card knows U_u is the real card holder. It then generates a random nonce N_u and calculates $A_i'=B_u \oplus H(b_u \oplus PW_u)$, $CID_u= H(b_u \oplus PW_u) \oplus H(T_u', A_u', N_u)$, $P_{uj}= T_u' \oplus H(A_u', N_u, SID_j)$, $Q_u=H(B_u, A_u', N_u)$, $D_u=R_u \oplus SID_j \oplus N_u$, and $C_0=H(A_u', N_u+1, SID_j)$.
2. U_u sends $\{CID_u, P_{uj}, Q_u, D_u, C_0, N_u\}$ to S_j.

(3) Mutual verification and session key agreement phase

After receiving the login message from U_u, S_j executes the following steps together with U_u to authenticate each other and compute a common session key.

1. S_j generates a random nonce N_{jr} and calculates $M_{jr}= H(SID_j, y) \oplus N_{jr}$.
2. S_j sends $\{M_{jr}, SID_j, D_u, C_0, N_u\}$ to RC.
3. RC computes $N_{jr}^*=M_{jr} \oplus H(SID_j, y)$, $R_u^*=D_u \oplus SID_j \oplus N_u$, $A_u^*=R_u^* \oplus H(x \oplus r)$, $C_o^*= H(A_u^*, N_u+1, SID_j)$ and checks to see if C_o^* is equal to the received C_0. If so, S_j is authentic. RC then generates a random nonce N_{rj} and calculates $C_1 = H(N_{jr}^*, H(SID_j, y), N_{rj})$, $C_2 = A_u^* \oplus H(H(SID_j, y), N_{jr}^*)$.
4. RC sends $\{C_1, C_2, N_{rj}\}$ to S_j.
5. After receiving the message from RC, S_j computes $H(N_{jr}, H(SID_j, y), N_{rj})$ and checks to see if it is equal to the received C_1. If so, RC is authentic. S_j then calculates $A_u''= C_2 \oplus H(H(SID_j, y), N_{rj})$, $T_u''= P_{uj} \oplus H(A_u'', N_u, SID_j)$, $h_u=CID_u \oplus H(T_u'', A_u'', N_u)$ and $B_u''= A_u'' \oplus h_u$. Then he computes $H(B_u'', A_u'', N_u)$ and compares it with Q_i received in the login phase. If they are equal, the login request is accepted. He proceeds to generate a random nonce N_j and calculate

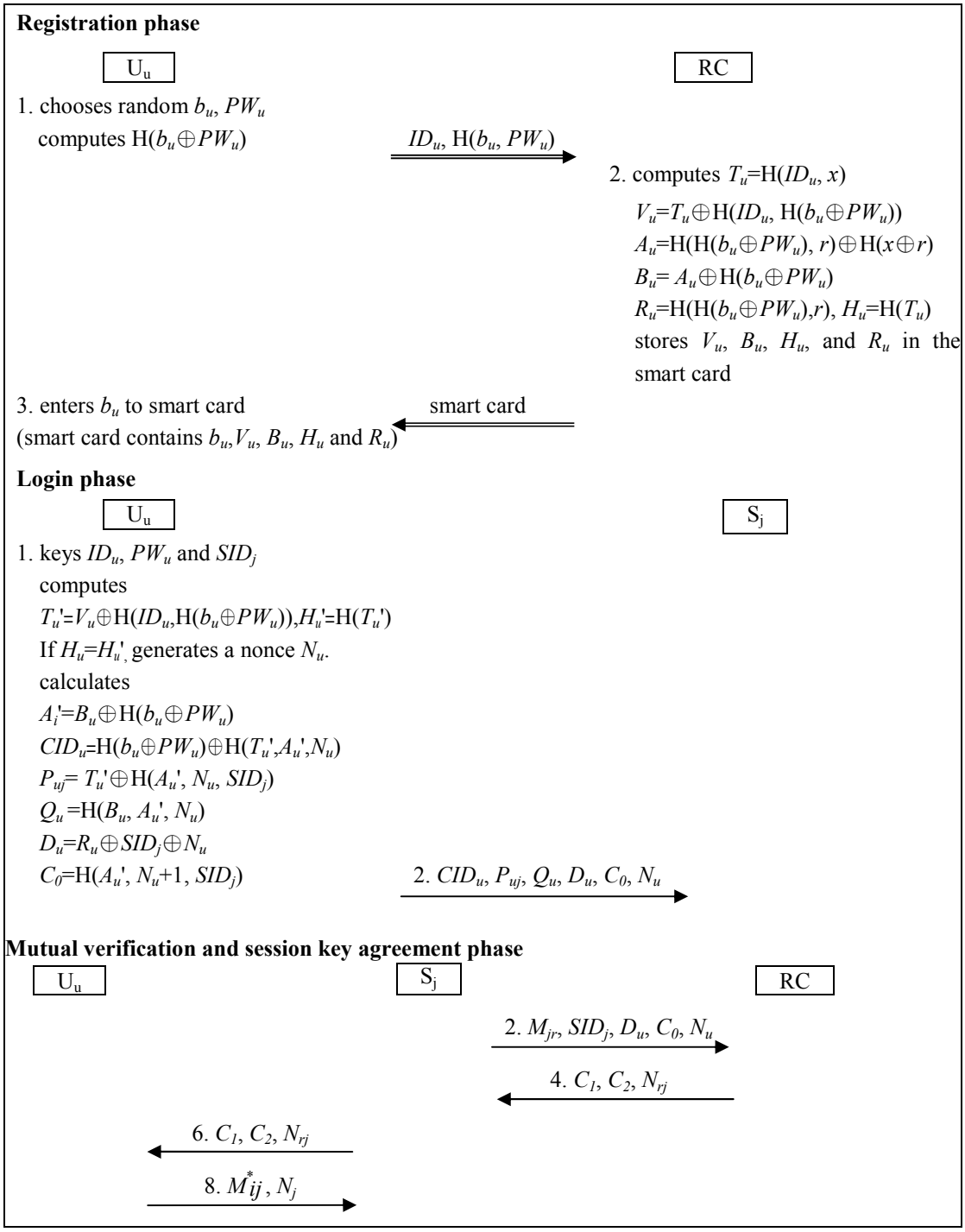


Fig. 4. Hsiang-Shih's protocol

$M_{ij}=H(B''_u, N_i, A''_u, SID_j)$.

6. S_j sends $\{M_{ij}, N_j\}$ to U_u.
7. U_u computes $H(B_u, N_i, A_u, SID_j)$ and checks to see if it is equal to the received M_{ij}. If it is, S_j is authentic. He then calculates $M^*_{ij}=H(B_u, N_j, A_u, SID_j)$.
8. U_u sends $\{M^*_{ij}\}$ to S_j.
9. S_j computes $H(B''_u, N_j, A''_u, SID_j)$ and checks to see if it is equal to the received M^*_{ij}.

10. After finishing mutual authentication, U_u and S_j can compute the common session key SK= H(B_u, A_u, N_u, N_j, SID_j) and SK= H(B''_u, A''_u, N_u, N_j, SID_j), respectively.

(4) Password change phase

When U_u wants to change his password from PW_u to PW_u^{new}, he executes the following steps.

1. Keys his ID_u, PW_u.
2. The smart card computes $T_u'=V_u \oplus H(ID_u, H(b_u \oplus W_u))$, $H_u'=H(T_u')$ and checks to see if H_u stored in the smart card is equal to the computed H_u'. If so, U_u is the real card holder.
3. The smart card allows U_u to submit a new password PW_{unew}.
4. The smart card computes $V_{unew}=T_{unew} \oplus H(ID_u, H(b_u \oplus PW_{unew}))$, $B_{unew}= B_u \oplus H(b_u \oplus PW_u) \oplus H(b_u \oplus PW_{unew})$ and replaces V_u, B_u with V_{unew}, B_{unew}, respectively.

3. Security loopholes in Tsai's and Hsiang-Shih's protocols

In this section, we will show that Tsai's protocol suffers server spoofing attacks in both scenarios and Hsiang-Shih's protocol suffers user impersonation attack and off-line password guessing attack when the smart card is lost. We demonstrate the security loopholes of both schemes in Section 3.1 and Section 3.2, respectively.

3.1 Server spoofing attack by an insider server on Tsai's protocol

Assume that S_i is a legal server registered at RC. He also has his H(SID_i, y) and keeps it secret. He can then masquerade as a legal server to cheat a remote user. This is because in the authentication of server and user phase, a user doesn't examine if the message is indeed sent from the correct server. In the following, we present server spoofing attacks on the two scenarios, (A) and (B), and also illustrate them in Figure 5 and 6, respectively.

(A) the secret key is not generated.

For this case, we describe the attack as follows and also illustrate it in Figure 5.

1. When U_u wants to communicate with S_j, he starts the protocol and sends {ID_u, C_1} to S_j who S_i masquerades.
2. S_i generates a nonce Ns, computes $C_2 = H(SID_i, y) \oplus Ns$, and sends {$ID_u$, SID_i, C_1, C_2} to RC. Then, for the subsequent transmitted messages, C_3, C_4, C_5 and C_6 (except C_6) between RC and S_i for authenticating each other are independent on U_u's secrecy H(H(ID_u, x), Nc) (as depicted in scenario (A) of Fig. 2), RC and S_i will be doomed to achieve mutual authentication successfully.

3. RC and S_i then negotiate to establish the common secret key $Auth_{S\text{-}RC}$=H(H(SID_i, y), Ns+1, N'_{RC}+2)=H(H(SID_i, y), Ns'+1, N_{RC}+2) in the phase of server and RC authentication. Then, S_i and U_u perform the authentication of server and user phase.
4. S_i generates a random nonce N_{SU} and uses his $Auth_{S\text{-}RC}$ to compute $C_7 = C_6 \oplus Auth_{S\text{-}RC}$ =H(H(ID_u, x), $N'c$). He then calculates $C_8 = C_1 \oplus C_7$, $V_2 = C_7 \oplus N_{SU}$, and C_9= H(C_7, N_{SU})$\oplus C_8$.
5. S_i sends {V_2 , C_9} to U_u.
6. After receiving the message, U_u computes C'_7= H(H(ID_u, x), Nc), retrieves N'_{SU}= $C'_7 \oplus V_2$, and calculates $C'_8 = C'_7 \oplus C_1$, C'_9= H(C'_7, N'_{SU})$\oplus C'_8$. He then checks to see if C'_9 is equal to the received C_9. If so, U_u confirms that the message is sent from the sender who had received his C_1 in the login phase. S_i disguising himself as S_j is thus regarded as being authentic by U_u. U_u then calculates C_{10}= H(C'_7, C'_8, N'_{SU}).
7. U_u sends {C_{10}} to S_i.
8. S_i computes C'_{10}= H(C_7, C_8, N_{SU}) and checks to see if C'_{10} is equal to the received

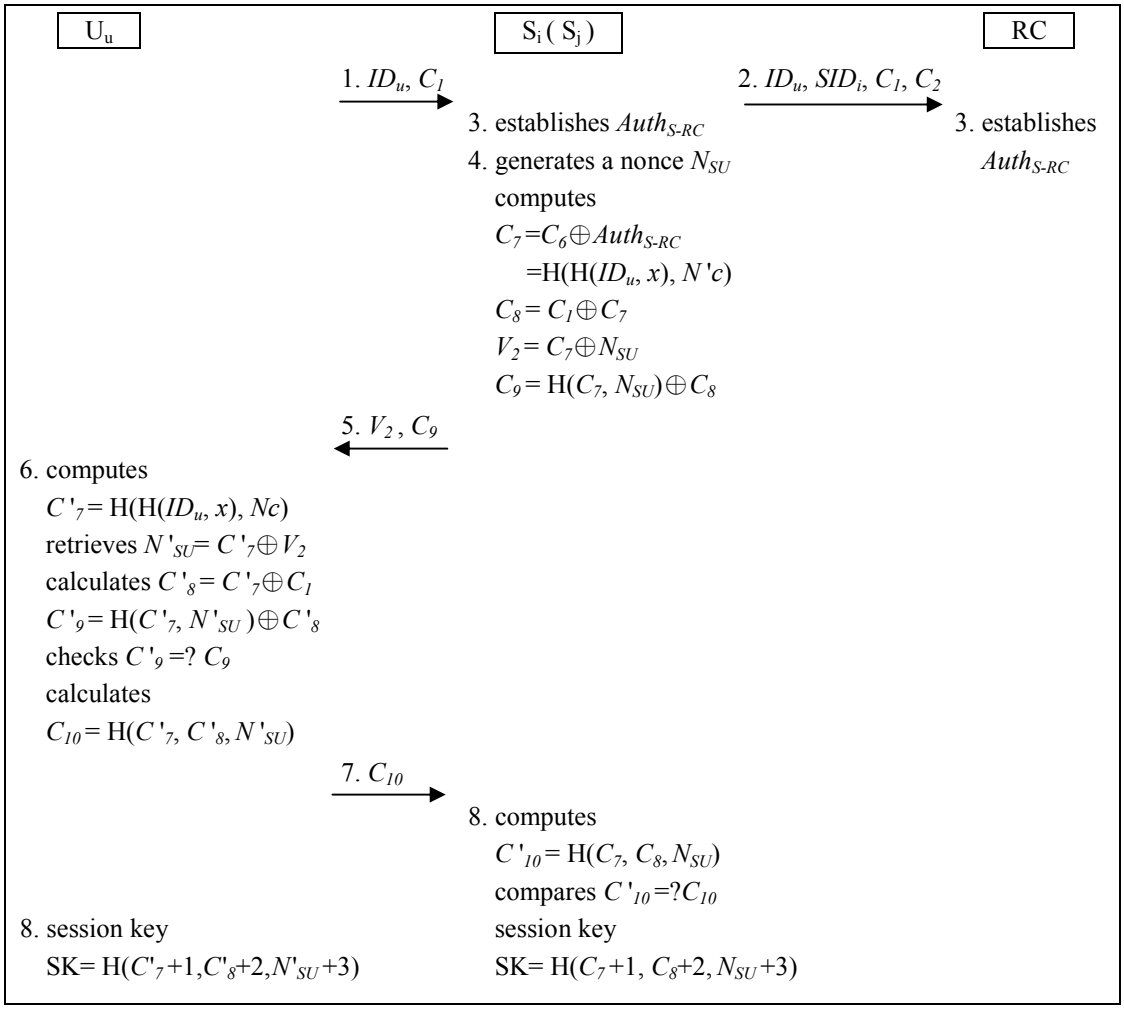


Fig.5. Server spoofing at tack by an insider server on Tsai's protocol for scenario (A) the secret key is not generated.

C_{10}. If so, U_u is authentic. They, U_u and S_i, compute the common session key as SK= H(C'_7+1, C'_8+2, N'_{SU}+3) and SK= H(C_7+1, C_8+2, N_{SU}+3), respectively.

From the above-mentioned steps, we can see that a server spoofing attack can besuccessfully launched by insider attacker S_i for this case.

(B) the secret key has been generated.

For this case, we describe the attack as follows and also illustrate it in Figure 6.

1.U_u starts the protocol and sends {ID_u, C_1} to S_i who masquerades as S_j.

2. When S_i runs the authentication of server and RC phase, he simply sends {ID_u, SID_i, C_1} to RC. RC deduces $N'c$=H(ID_u, x)$\oplus C_1$ and computes C_6 = H(H(SID_i, y), Ns'+1, N_{RC}+2)$\oplus$H(H(ID_u, x), $N'c$).

3. RC sends {C_6} to S_i. S_i then performs the authentication of server and user phase with U_u.

4. S_i generates a random nonce N_{SU} and uses the generated common secret key $Auth_{S\text{-}RC}$ to compute C_7 = $C_6\oplus Auth_{S\text{-}RC}$ =H(H(ID_u, x), $N'c$). He then calculates C_8 = $C_1\oplus C_7$, V_2 = $C_7\oplus N_{SU}$, and C_9 = H(C_7, N_{SU})$\oplus C_8$.

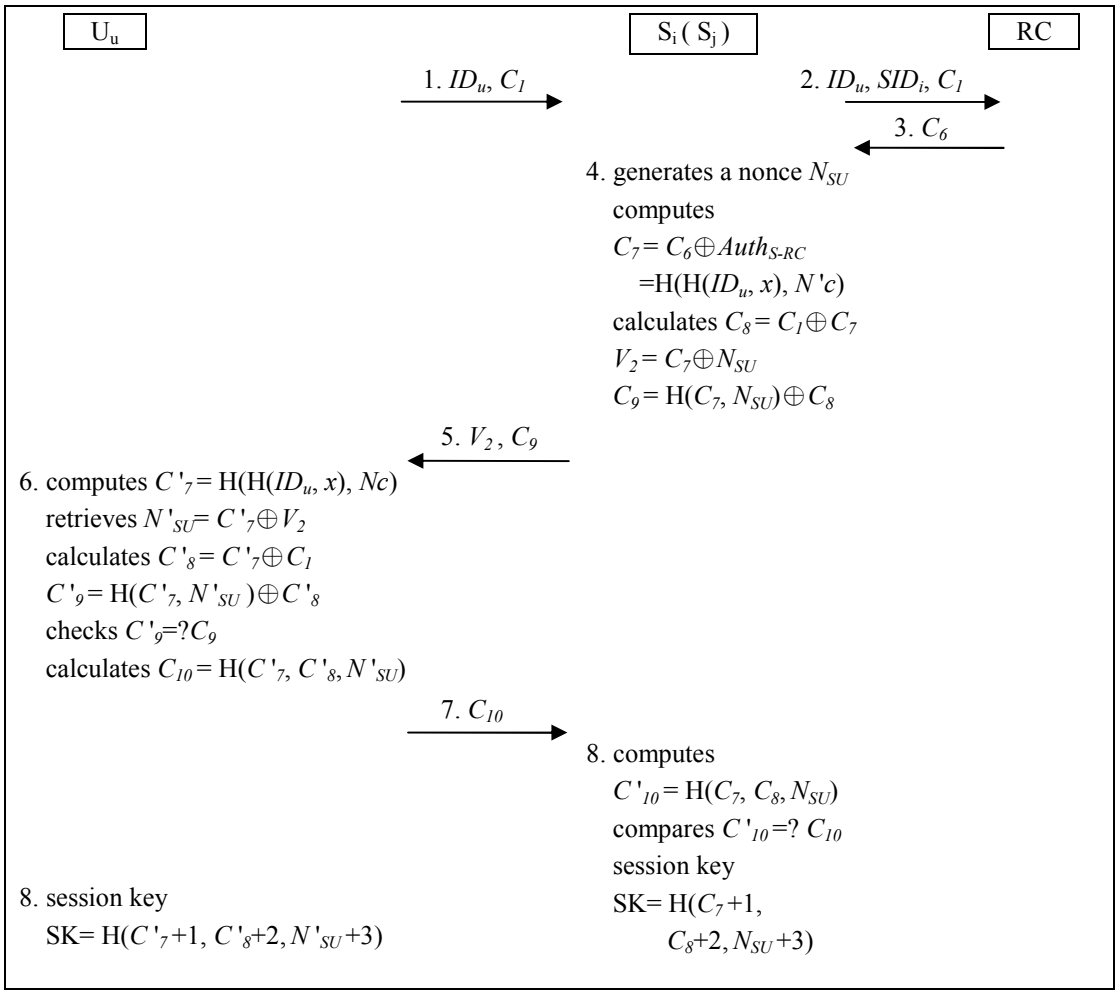


Fig. 6. Server spoofing attack by an insider server on Tsai's protocol for scenario (B) the secret key has been generated.

5. S_i sends $\{V_2, C_9\}$ to U_u.
6. After receiving the message, U_u computes $C'_7 = H(H(ID_u, x), Nc)$, retrieves $N'_{SU} = C'_7 \oplus V_2$, and calculates $C'_8 = C'_7 \oplus C_1$, $C'_9 = H(C'_7, N'_{SU}) \oplus C'_8$. He then checks to see if C'_9 is equal to the received C_9. If so, U_u confirms that the message is sent from the sender who has received his C_1 in the login phase. S_i disguising himself as S_j is therefore regarded as being authentic. U_u then proceeds to calculate $C_{10} = H(C'_7, C'_8, N'_{SU})$.
7. U_u sends $\{C_{10}\}$ to S_i.
8. After obtaining the message from U_u, S_i computes $C'_{10} = H(C_7, C_8, N_{SU})$ and checks to see if C'_{10} is equal to the received C_{10}. If so, U_u is authentic. They then compute the common session key $SK = H(C'_7+1, C'_8+2, N'_{SU}+3)$ and $SK = H(C_7+1, C_8+2, N_{SU}+3)$, respectively.

From the above-mentioned steps, we can see that a server spoofing attack launched by insider attacker S_i has been successfully accomplished in case (B).

3.2 Attack on Hsiang-Shih's protocol

In the following, we demonstrate two attacks, (a) the impersonation attack and (b) the off-line password guessing attack if the smart card is lost, on Hsiang-Shih's protocol.

(a) The impersonation attack

We further divide this kind of attack into following two cases: (1) outsider impersonation attack, and (2) insider impersonation attack.

(1) Outsider impersonation attack

In Hsiang-Shih's protocol, it can easily be seen that any passive attacker can deduce all of a user U_u's secrets stored in the smart card from the messages, $\{M_{jr}, SID_j, D_u, C_0, N_u\}$, $\{CID_u, P_{uj}, Q_u, D_u, C_0, N_u\}$, and $\{C_1, C_2, N_{rj}\}$, transmitted among U_u, S_j and RC. For he can deduce $A_u = C_2 \oplus H(M_{jr})$, and then obtain R_u, T_u, $H(b_u \oplus PW_u)$ by computing $R_u = D_u \oplus SID_j \oplus N_u$, $T_u = P_{uj} \oplus H(A_u, N_u, SID_j)$, and $H(b_u \oplus W_u) = CID_u \oplus H(T_u, A_u, N_u)$. Hence, he can impersonate U_u to login to S_j by sending a login request. For example, he sends the login request $\{CID_u, P_{uj}', Q_u', D_u', C_0', N_u'\}$ to S_j by selecting a new random nonce N_u' and computing $CID_u = H(b_u \oplus PW_u) \oplus H(T_u, A_u, N_u')$, $P_{uj} = T_u \oplus H(A_u, N_u', SID_j)$, $Q_u = H(B_u, A_u, N_u')$, $D_u = R_u \oplus SID_j \oplus N_u'$, and $C_0 = H(A_u, N_u'+1, SID_j)$. Obeying their protocol, it is obvious that he can pretend U_u and access all servers's resources successfully.

(2) Insider impersonation attack

An insider attacker E is a malevolent user registered at RC. He can use his secret b_e, PW_e, B_e, R_e to deduce $H(x \oplus r)$ by computing $H(x \oplus r)=B_e \oplus R_e \oplus H(b_e \oplus PW_e)$. Then, he can use his computed $H(x \oplus r)$, the eavesdropped message $\{CID_u, P_{uj}, Q_u, D_u, C_0, N_u\}$ transmitted between U_u and S_j in the login phase, and the public parameter SID_j to deduce R_u, A_u, T_u, $H(b_u \oplus PW_u)$ by computing $R_u=D_u \oplus SID_j \oplus N_u$, $A_u=R_u \oplus H(x \oplus r)$, $T_u=P_{uj} \oplus H(A_u, N_u, SID_j)$, and $H(b_u \oplus W_u)=CID_u \oplus H(T_u, A_u, N_u)$. He then can calculate all of U_u's secrets $\{V_u, B_u\}$ stored in the smart card by computing $V_u= T_u \oplus H(ID_u, H(b_u \oplus PW_u))$ and $B_u= A_u \oplus H(b_u \oplus PW_u)$. For E has all the secret data of U_u, he can therefore impersonate U_u in the same manner successfully as described in Section 3.2.(a).(1).

(b) Off-line password guessing attack if the smart card is lost

In the password change phase, when a user wants to change his password, the smart card has to verify the correctness of the card holder's password. If the smart is lost or stolen, the attacker can read the secret data $\{b_u, V_u, H_u\}$ stored in the smart card. He then can compute $T' =V_u \oplus H(ID_u, H(b_u \oplus PW'))$, where PW' is his guessing password, and check to see if his computed $H(T')$ is equal to the stored value of H_u without the help of any other entities. If the two values equal, he successfully launches the attack. Else, he can repeat the above password guessing attack until he obtains the correct one. Therefore, a smart-card-lost off-line password guessing attack can be launched.

Furthermore, after guessing the correct password, the attacker can enforce the password change phase. Subsequently, from then on, the real card holder cannot use his password to login to the remote server anymore. That is, their scheme suffers from Denial-of-service attack as well.

4. Our protocol

After presenting the attacks on protocols [1] and [14], in this section, we present our scheme. Our protocol contains four phases. They are: (1)preparation phase, (2)registration phase, (3)login phase, and (4)authentication and session key agreement phase or authentication and password change phase. In our protocol, RC is trustworthy and has two secret keys, x and y. All identities of users and servers are public, *e.g.* ID_u and SID_j. We describe the first two phases in Section 4.1 and Section 4.2 respectively and also depict them in Figure 10. The last two phases are combined and divided into three scenarios. We discuss them in Section 4.3 and also depict them in Figure 11, 12, and 13, respectively.

(1) Preparation phase

In this phase, for each server S_j with identity SID_j, RC performs the following steps.

1. RC computes RSj=H(SID_j, y).

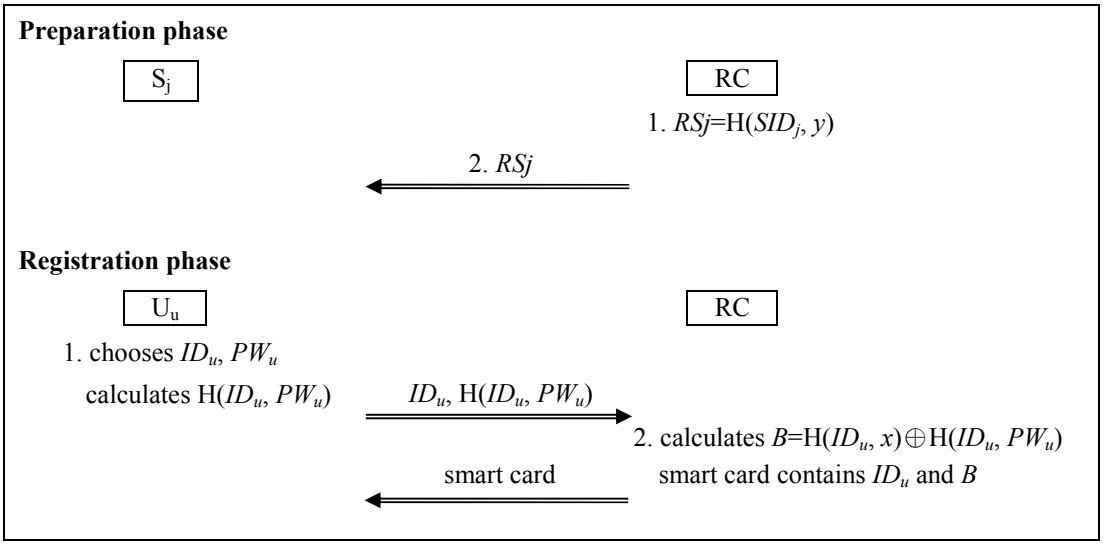


Fig. 10. Preparation phase and registration phase

2. RC sends RSj to S_j via a secure channel.

(2) Registration phase

In this phase, U_u performs the following steps to register at RC for obtaining a smart card. Once having registered at RC, he can use the card to login to any eligible server for accessing resources.

1. U_u randomly chooses his ID_u, PW_u and calculates H(ID_u, PW_u). He then sends {ID_u, H(ID_u, PW_u)} to RC through a secure channel.
2. RC calculates B=H(ID_u, x)$\oplus$H(ID_u, PW_u) and issues U_u a smart card containing ID_u and B through a secure channel.

(3) Login for authentication and session key agreement or for password change

In our scheme when U_u wants to login S_j, he may want to execute either authentication and session key agreement or password change. We first describe former case using two scenarios: (A) the first time execution (of login for authentication and session key agreement phase), and (B) not the first time execution (of login for authentication and session key agreement phase). Then, we describe the latter case using scenario (C) (login for password change phase). We describe them as follows and also depict them in Figure 11, 12, and 13, respectively.

(A) the first time execution (of login for authentication and session key agreement phase)

(a) Login phase

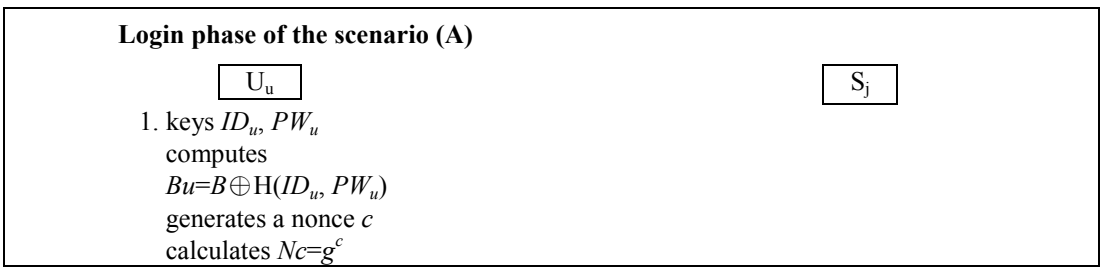
When U_u wants to access S_j's resources, he inserts his smart card and performs the following steps. The steps are also illustrated in Figure 11.

1. U_u keys his ID_u and PW_u to the smart card. The smart card computes $Bu=B\oplus H(ID_u, PW_u)$, generates a random nonce c and calculates $Nc=g^c$, $C_1=H(Bu, SID_j, Nc)$.
2. U_u sends $\{ID_u, SID_j, C_1, Nc, Flag\}$ to S_j, where *Flag* is set to '*the first time login*'.

(b) Authentication and session key agreement phase

When receiving the login message from U_u, S_j executes the following steps to determine if U_u is valid. If so, he negotiates the session key with U_u. We describe them using the following steps. The steps are also illustrated in Figure 11-continued.

1. After receiving $\{ID_u, SID_j, C_1, Nc, Flag\}$ from U_u, S_j reads *Flag* and knows that U_u is the first time login. He then generates a random nonce s and calculates $Ns=g^s$, $V_1=H(RSj, ID_u, Ns)$.
2. S_j sends $\{ID_u, SID_j, C_1, Nc, Tu, V_1, Ns\}$ to RC.
3. After receiving the authentication request, RC first checks to see if ID_u and SID_j are valid. If so, RC calculates $C_1^*=H(H(ID_u, x), SID_j, Nc)$, $V_1^*=H(H(SID_j, y), ID_u, Ns)$ and checks to see if they are equal to the received C_1 and V_1, respectively. If so, RC confirms that both U_u and S_j are authentic and knows that U_u attempts to login to S_j. He then computes $C_2=H(SID_j, H(ID_u, x), Ns, Nc)$ and $V_2=H(ID_u, H(SID_j, y), Nc, Ns)$.
4. RC sends $\{C_2, V_2\}$ to S_j.
5. After receiving the message from RC, S_j computes $V_2^*=H(ID_u, RSj, Nc, Ns)$ and checks to see if it is equal to the received V_2. If it is, S_j confirms that RC is authentic. He then calculates the session key $SK=(Nc)^s$ to be shared with U_u and computes $C_3=H(C_2\oplus SK)$, $B_j=H(ID_u, RSj)$, and $B_c=B_j\oplus SK$.
6. S_j sends $\{C_3, Ns, B_c\}$ to U_u.
7. After receiving the message from S_j, U_u computes $SK'=(Ns)^c$, $C_3^*=H(H(SID_j, Bu, Ns, Nc)\oplus SK')$ and checks to see if this computed C_3^* is equal to the received C_3. If so, U_u confirms that S_j is authentic. He then calculates $B_j'=B_c\oplus SK'$, $B_{uj}=B_j'\oplus$



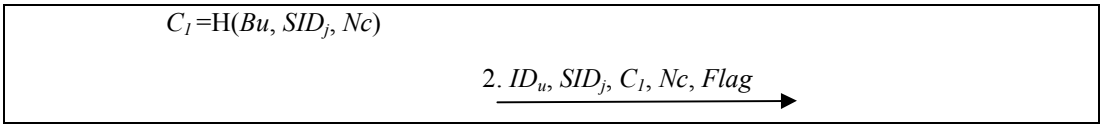


Fig. 11. Scenario (A): the first time execution (of login for authentication and session key agreement phase

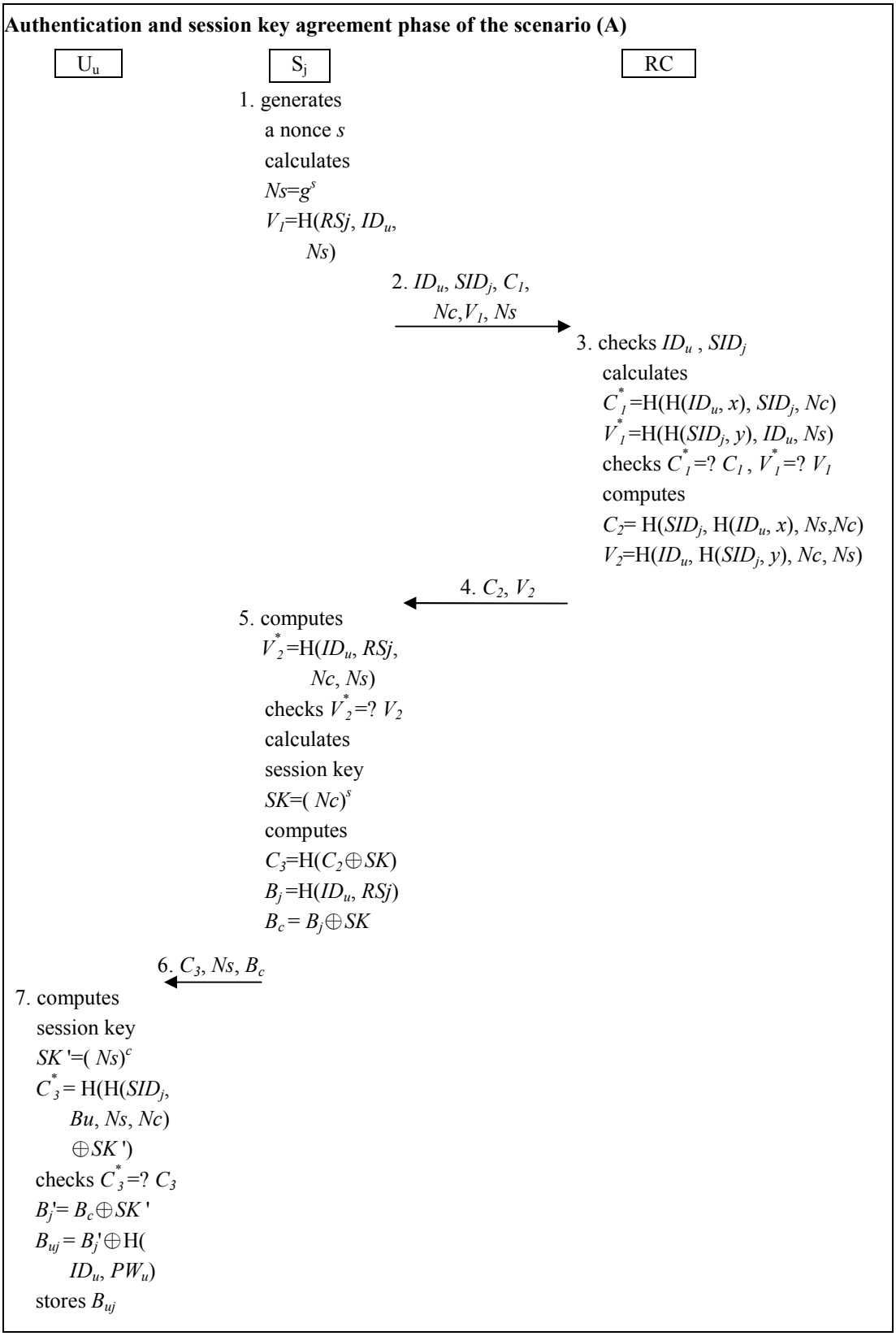


Fig. 11-continued. Scenario (A): the first time execution (of login phase for authentication and session key agreement phase)

H(ID_u, PW_u) and stores the common secret key B_{uj} in his smart card for login S_j without the help of RC's authentication next time. U_u and S_j then have the common session key $SK' = SK = g^{c \cdot s}$.

(B) not the first time execution (of login for authentication and session key agreement phase)

When U_u wants to access S_j's resources again, he inserts his smart card and performs the following steps. The steps are also illustrated in Figure 12.

(a) Login phase

1. U_u keys his ID_u and PW_u to the smart card. The smart card computes $B^*_j = B_{uj} \oplus$ H(ID_u, PW_u), generates a random nonce u and calculates $Nu = g^u$, C =H(B^*_j, ID_u, SID_j, Nu).
2. U_u sends {ID_u, SID_j, C, Nu**,** $Flag$} to S_j**,** where $Flag$ is set to '*not the first time login*'.

(b) Authentication and session key agreement phase

When receiving the login message from U_u, S_j executes the following steps to determine if U_u is valid. If so, he negotiates the session key with U_u.

1. After receiving {ID_u, SID_j, C, Nu**,** $Flag$ } from U_u, S_j first checks to see if ID_u is valid. If ID_u is legal, from the flag, S_j knows that U_u logins not the first time. He generates a random nonce j, calculates $Nj = g^j$, B_j =H(ID_u, RSj), C' = H(B_j, ID_u, SID_j, Nu), and checks to see if C' is equal o the received C. If so, S_j confirms

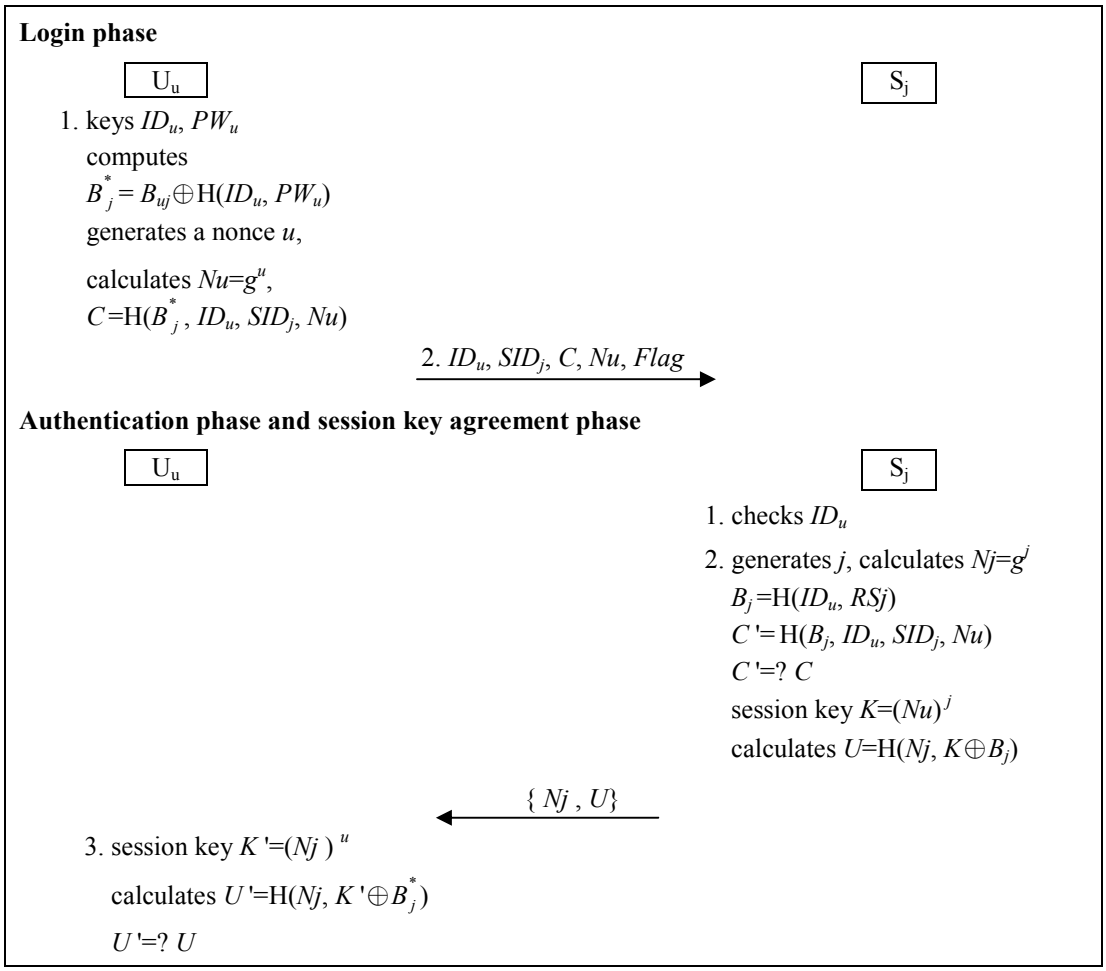


Fig. 12. Scenario (B): not the first time execution (of login for authentication and session key agreement phase)

that U_u is authentic. He then calculates the session key $K=(Nu)^j$ and $U=H(Nj, K \oplus B_j)$.

2. S_j sends $\{Nj, U\}$ to U_u.
3. After receiving the message from S_j, U_u computes the session key $K'=(Nj)^u$, $U'=H(Nj, K' \oplus B^*_j)$ and checks to see if U' is equal to the received U. If it is, U_u confirms that S_j is authentic. U_u and S_j then have the common session key $K'=K=g^{uj}$.

(C) Login for authentication and password change phase

To get rid of the weakness as described in Section 3.2.(b), the clients must change his password under the intervention of RC when executing the password change phase. Under such limitation, it not only can prevent the password guessing attack if the smart card is lost but also meet the requirement that the client can choose and change his password at will. To attain this purpose, the password change phase of our protocol contains two phases, (a) Login phase and (b) Authentication and password change phase.

When U_u wants to change his password, he performs the following steps. The steps

are also illustrated in Figure 13.

(a) Login phase

1. U_u keys his ID_u, PW_u, and new password PW_u^{new} to the smart card. The smart card checks PW_u to see if ID_u is the real cardholder. If so, the card computes $Bu=B \oplus H(ID_u, PW_u)$, generates a random nonce c, and calculates $Nc=g^c$, $C_1=H(Bu, H(ID_u, PW_u^{new}), Nc)$, and $CP_1=H(Bu, Nc) \oplus H(ID_u, PW_u^{new})$.
2. U_u sends $\{ID_u, SID_j, C_1, CP_1, Nc, Flag\}$ to RC, where *Flag* is set to '*for password change*'.

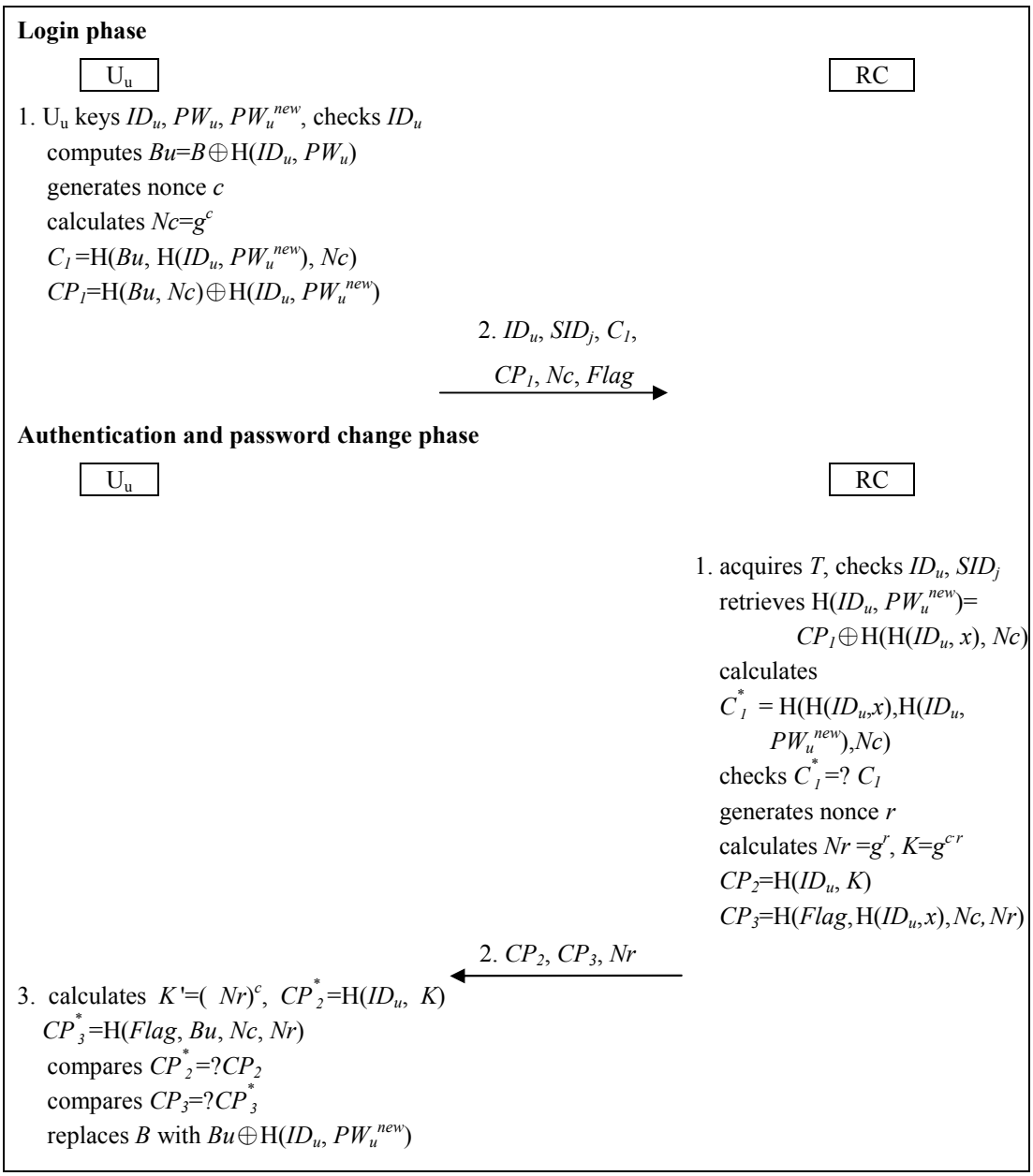


Fig. 13. Scenario (C): Login for authentication and password change phase

(2) Authentication and password change phase

1. After receiving the password change request, RC checks to see if ID_u and SID_j are valid. If they are valid, RC retrieves $H(ID_u, PW_u^{new})$ by computing $CP_1 \oplus H(H(ID_u, x), Nc)$, calculates $C_1^* = H(H(ID_u, x), H(ID_u, PW_u^{new}), Nc)$, and checks to see if C_1^* is equal to the received C_1. If so, U_u is authentic. RC generates a random nonce r and calculates $Nr = g^r$, $K = g^{c \cdot r}$, $CP_2 = H(ID_u, K)$, and $CP_3 = H(Flag, H(ID_u, x), Nc, Nr)$, where $Flag$ is set to '*accept*'.
2. RC sends $\{CP_2, CP_3, Nr\}$ to U_u.
3. After receiving the message from RC, U_u calculates $K' = (Nr)^c$, $CP_2^* = H(ID_u, K)$, and $CP_3^* = H(Flag, Bu, Nc, Nr)$ and checks to see if CP_2^* is equal to the received CP_2. If so, U_u confirms that RC is authentic. He then compares CP_3 with CP_3^*. If they are equal, U_u knows that his password change request has been accepted. The smart card then replaces the stored B with $Bu \oplus H(ID_u, PW_u^{new})$.

5. Security analysis for our protocol

We will show that our protocol not only can provide mutual authentication, perfect forward secrecy, changing password freely and securely, and session key agreement but also can resist various attacks such as, stolen-verifier attack, insider-server spoofing attack, insider-user impersonating attack, off-line password guessing attack, on-line password guessing attack, replay attack, parallel session attack (Man-in-the -Middle attack), and smart-card-lost attack. In the following, for each security attribute analysis, we mainly concern the most complex part, scenario (A) as indicated in Figure 11, in our protocol. The other two scenarios, (B) and (C), can be reasoned in a similar manner. We omit them unless stated otherwise.

5.1 Mutual authentication

In the following, we demonstrate that scenario (A) depicted in Figure 11 can provide mutual authentication between each pair among the three parties, user U_u, server S_j and RC. We describe it below.

As in the figure, for authenticating U_u after receiving his login request, S_j first sends $\{ID_u, SID_j, C_1, Nc, V_1, Ns\}$ to RC. RC verifies the validities of both C_1 and V_1. If they are valid, RC confirms that both U_u and S_j are authentic. Here, we demonstrate these relations using the two solid arrows, ① and ②, as involved in Figure 14. He then sends C_2 and V_2 to S_j. S_j verifies the validity of V_2. If it is valid, S_j confirms that RC is authentic. This is depicted in Figure 14 by the solid arrow ③. He then sends $\{C_3, Ns\}$ to U_u. Also, U_u has to verify $\{C_3, Ns\}$ to authenticate S_j. If C_3 is valid, U_u confirms that S_j is authentic. This is depicted using the solid arrow ④ in the figure. Obviously, authenticity relationship has transitive property when the identities of both

communicating parties and their common secret are committed. Otherwise, the PKI infrastructure will not work. From this observation, if there is an authenticity from A to B and from B to C, then there is an authenticity from A to C. According to this rule, we can obtain the dashed arrow ⑤ from ③ and ④ (as shown in Figure 14). The remaining dashed arrow ⑥ can be obtained by using the following proof of contradiction. If we use A $\longrightarrow$ B to represent A authenticates B, or equivalently, B is regarded as authentic by A, for we already know the facts that RC $\longrightarrow$ S_j and RC $\longrightarrow$ U_u, if S_j $\longrightarrow$ U_u doesn't hold, then from the transitive property, RC $\longrightarrow$ U_u cann't hold as well. This contradicts the fact that RC $\longrightarrow$ U_u. Hence, the dashed arrow ⑥ exists. This completes the figure. From the figure, we can see that the mutual authentications between each pair of the three parties can be satisfied.

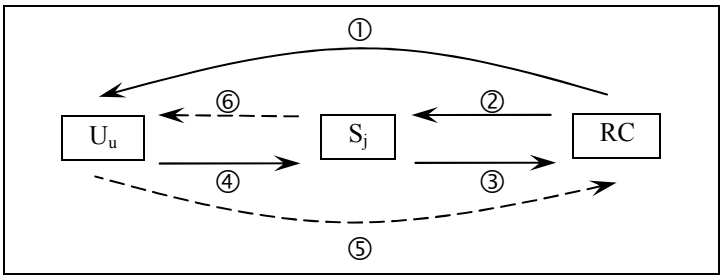


Fig. 14. Authenticity relationship

5.2 Session key agreement

In our protocol, after a legal user U_u has logged into an eligible server and finished the authentication and session key agreement phase, they have the same session key. This can easily be seen in the two steps, step 5 and step 7, of the authentication and key agreement phase executed by S_j and U_u respectively in Section 4.(3).(A).(b).

5.3 Perfect forward secrecy

In our scheme, a compromised password can't be used to construct previous session keys for that we use the Diffie-Hellman key agreement protocol which are based on random nonces. In other words, the session keys generated before in each session between the user and server are independent. Accordingly, our scheme provides perfect forward secrecy.

5.4 Changing password freely and securely

In our protocol, user U_u can change his password securely. Even if an attacker E can temporarily obtain U_u's smart card; however, without the knowledge of U_u's password, he can't change U_u's password by choosing two different passwords, PW_1 guessed by E as the old and PW_2 as the new one, and replacing B with $B \oplus \mathrm{H}(ID_u, PW_1)$

$\oplus$H(ID_u, PW_2), where B is the value stored in U_u's smart card. This is because the password change request can only be accepted after successful mutual authentication between U_u and RC as stated in Section 4.(3).(C).(b).

5.5 Preventing the stolen-verifier attack

The protocol we proposed doesn't hold any verifier table. RC holds only two secret keys, x and y. Each user has only one secret H(his identity, x) and each server has only one secret H(his identity, y). Therefore, our scheme gets rid of using verifier tables and thus can prevent stolen-verifier attack.

5.6 Preventing the insider-server spoofing attack

It is unnecessary to assume that all servers are trustworthy as needed in [7, 9, 12], because in our protocol, an illegal or spoofing server's request will be rejected. More precisely, if a legal server S_i having his own *RSi* sends message 2 in Figure 11-continued to RC by impersonating S_j, without the knowledge of S_j's *RSj* and RC's secret key y, the value of V_1 he computes would be different from the value of RC's computation V_1^*. Hence, he can't be authenticated by RC. Therefore, the server spoofing attack fails.

5.7 Preventing insider-user impersonating attack

If a legal client U_n wants to impersonate client U_u to login to S_j. Without the knowledge of U_u's B and PW_u, the value of C_1 he computes would be different from the value of RC's computation C_1^* as shown in Figure 11-continued. Hence, he can't be authenticated by RC. Therefore, the insider-user impersonating attack fails.

5.8 Preventing off-line and on-line password guessing attack

In scenario (A) of our protocol, there are four messages: M_1={ID_u, SID_j, C_1, Nc, $Flag$}, M_2={ID_u, SID_j, C_1, Nc, V_1, Ns}, M_3={ C_2, V_2} and M_4={ C_3, Ns}, to and from through the Internet. Assume that an attacker U_n who hasn't got U_u's smart card wants to guess U_u's password PW_u. (If U_n has got U_u's smart card, this case is termed as smart-card-lost attack. We will discuss it in section 5.11.) We argue that U_n will not succeed. For among the transmitted messages, except for C_1 computed by U_u in M_1 and M_2, S_j and RC needn't use U_u's password PW_u to compute any values. Moreover, PW_u in C_1 is protected by a hash function iterating two times, *i.e.*, C_1=H($B\oplus$H(ID_u, PW_u), SID_j, Nc). Not to mention, its inner hash result is Xor-ed by an unknown value B. Hence, U_n can hardly succeed in guessing PW_u due to the one-way property of a hash function. That is, due to the unknown value B, U_n definitely can't implement the off-line password guessing attack. For the same reason, we can easily see that an

attacker can not launch an on-line password guessing attack if we set our protocol to tolerate three times of wrong password logins.

5.9 Preventing replay attack

For our protocol uses different random nonces, *c* and *s*, each time in computing *Nc* and *Ns* which are needed in computing the related values, such as C_1, V_1, C_2, V_2 and C_3, to prevent any replay attack. Therefore, an attacker cannot be authenticated successfully by resending any previous transmitted values.

5.10 Preventing parallel session (Man-in-the-Middle) attack

In scenario (A), assume that U_n wants to launch a parallel session attack by masquerading as both S_j to U_u and U_u to S_j. After receiving the message $M_1=\{ID_u, SID_j, C_1, Nc, Flag\}$ from U_u in the login phase, U_n masquerading as U_u starts another protocol by resending M_1 to S_j. For more clarity, we briefly show this scenario in Fig. 15. Although, using value C_1, U_n can pass RC's authentication after S_j sending M_1 to RC. However, without the knowledge of U_u's nonce *c*, he can't finish the authentication and session key agreement phase to impersonate U_u to S_j. Because U_n, without the knowledge of *c*, can't compute the correct session key *SK*', $(Ns)^c$, shared with S_j. Similarly, U_n can't impersonate S_j to U_u for he hasn't S_j's secret *RSj* to compute V_1 for being verified by RC. Hence, U_n can't obtain C_2, V_2 from RC to compute a valid C_3 which will be transmitted to U_u for U_u to authenticate S_j. In other words, U_n can not succeed in impersonating S_j to U_u. Therefore, the parallel session attack fails.

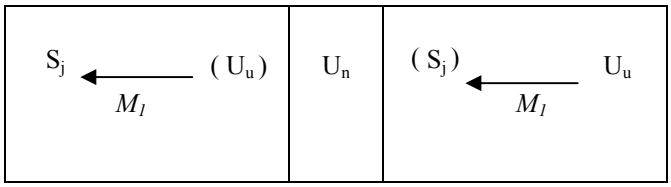


Fig. 15. Parallel session attack

5.11 Preventing smart-card-lost attack

Assume that an attacker U_n has got U_u's smart card and knows the stored value *B*. Even under such a situation, our protocol still can prevent various attacks including: (1) insider impersonating attack, (2) off-line/on-line password guessing attack, (3) outsider impersonation attack, (4) replay attack. We analyze each type of attack as follows.

(1) Insider impersonating attack: Although, U_u obtains the smart card and knows the value *B*. However, without the knowledge of U_u's PW_u, the value $C_1 = H(B \oplus H(ID_u, PW_u), SID_j, Nc)$ which U_n computes would be different from the

value of RC's computation C_1^*. Hence, he can't be authenticated by RC. Therefore, the insider attack fails.

(2) Off-line/On-line password guessing attack: When U_n gets the smart card, he may want to launch an off-line password guessing attack. However, for the smart card only stores ID_u and B, without the help of on-line RC which will responds with C_2, V_2 to S_j only under the situation that C_1 equals his computation C_1^*. That can happen only U_n can guess the correct PW_u. Hence, he can't launch such an off-line password guessing attack. In other words, without a correct password PW to compute the specified $Bu=B\oplus H(ID_u, PW)$, U_n can't compute the right value of $C_1=H(Bu, SID_j, Nc)$ to pass RC's verification. More precisely, since the computed C_1 for each guessing needed to be verified by RC to confirm its correctness, U_n can't implement the off-line password guessing attack without the help of RC. In the same fashion, the impossibility of on-line password guessing attack can be obviously seen if we set the protocol to tolerate three times of wrong password logins.

(3) Outsider impersonation attack: Assume that an un-registered user getting U_u's smart card wants to impersonate U_u by starting the protocol with S_j. Similarly, without the knowledge of U_u's password PW_u, he can not deduce the correct value of $Bu=B\oplus H(ID_u, PW_u)=H(ID_u, x)$ and henceforth C_1, which will be verified by RC. In other words, since each password guessing for each impersonation needs the help of RC's verification, the attacker could hardly be authenticated successfully. This means the outsider impersonation attack fails.

Except for the resistance of the above mentioned attacks, our protocol also can prevent replay attack under the situation of smart card loss. The reasons are the same as the ones previously described in Section 5.9. We omit them here.

6. Discussion

In this section, we first show that our protocol meets the requirement of **s**ingle registration property. Then, to show the advantage of our scheme, we compare our protocol with both Tsai's [1] and Hsiang-Shih's [14] in communication cost and with [3,4,6,10,11] in the aspect of card-issue cost.

6.1 Single registration

In our protocol, a user needs not to register at each server. Instead, he only needs to register at RC. Consequently, he needs to remember only one password and can access all of the legal servers' resources.

6.2 Low communication cost

Tsai's protocol needs seven passes from login to complete mutual authentication in the first scenario and five passes in the second scenario as indicated in Figure 1 through Figure 3. Hsiang-Shih's protocol needs five passes as indicated in Figure 4. However, ours needs four passes in scenario (A) and only two passes in scenario (B). Therefore, our protocol is significantly more efficient than [1] and [14]. That is because when estimatly the efficiency of a protocol, the number of passes is the dominant factor when compared with the computation overhead the protocol requires. That is, our scheme has the lowest communication cost when compared with theirs.

6.3 Increasing servers freely/ Low card-issue cost

Several schemes [3,4,6,10,11] can't add a server freely. For in them, when a server is added, all users who want to login to the newly added server need to re-register to get a new smart card. It increases the system's card-issue cost. In our protocol, when a server is added, all users don't need re-register. He can use his own card to access the new server's resources. That is, our protocol can let the number of servers increase freely and thus has low card-issue cost.

6.4 Comparison

In this section, we compare the security features and some other properties among our scheme and the other proposed schemes listed in the reference, except [8] and [15] which only pointed out the weakness of [5] and [2] respectively without proposing a new method. Comparing with those schemes, our scheme not only can provide the secure RC-off-line authentication and resist against all attacks but also can add a server freely. We summarize the comparison of each property in Table 1. From Table 1, we can see our protocol is the most favorite one in multi-server environments.

Tab. 1. The comparison of our scheme and other proposed schemes

	Ours	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[9]	[10]	[11]	[12]	[13]	[14]
1. RC-off-line authentication	○	×	○	○	○	○	○	○	○	○	○	○	×	×
2. Increasing servers freely	○	○	○	×	×	○	×	○	×	×	×	○	○	○
3. No assuming that all servers are trustworthy	○	○	○	○	○	○	○	×	×	○	○	×	○	○
4. Low computationally intensive	○	○	○	×	×	×	○	○	○	○	○	○	○	○
5. No verifier table in RC or any server	○	×[a.]	○	○	○	○	○	○	○	○	○	○	×	○
6. Mutual authentication	○	○	○	×	×	×	○	○	×	○	○	○	○	○
7. Preventing insider-server spoofing attack	○	×	×	○	○	○	○	×	×	○	○	×	○	○
8. Preventing off-line password guessing attack	○	○	×	○	○	×	○	○	○	○	○	○	○	○
9. Preventing parallel session attack	○	○	×	×	×	×	○	○	○	○	○	○	○	○
10. Preventing smart-card-lost attack	○	○	×	×	○	×	×	×	○	×	×	×	×	×

a. It is described in Section 2.1.(3).(A).

7. Conclusion

We have analyzed the security of Tsai's and Hsiang-Shih's protocols. Although they claimed their protocols are secure, we have showed several attacks on their schemes. In addition, we propose a novel secure efficient protocol. After the security analysis, we conclude that our protocol has the following merits including: 1. single registration, 2. no verifier table, 3. low communicational cost, 4. increasing servers freely, 5. mutual authentication, 6. session key agreement, 7. perfect forward secrecy, 8. changing password freely and securely, and 9. preventing various attacks such as, stolen-verifier attack, insider-server spoofing attack, insider-user impersonating attack, off-line password guessing attack, on-line password guessing attack, replay attack, parallel session attack, and smart-card-lost attack. That is, up to date, our protocol is not only the most secure but also the most efficient scheme using smart card in a multi-server environment.

References

[1] J.L. Tsai, "Efficient multi-server authentication scheme based on one-way hash function without verification table", *Computers & Security*, Vol. 27, No. 3-4, pp. 115-121, May-June 2008.

[2] Y.P. Liao, S.S. Wang, "A secure dynamic ID based remote user authentication scheme for multi-server environment", *Computer Standards & Interfaces*, Vol. 31, No. 1, pp. 24-29, January 2009.

[3] W.J. Tsaur, C.C. Wu, W.B. Lee, "An enhanced user authentication scheme for multi-server Internet services", *Applied Mathematics and Computation*, Vol. 170, No. 1-1, pp. 258-266, November 2005.

[4] W.J. Tsaur, C.C. Wu, W.B. Lee, "A smart card-based remote scheme for password authentication in multi-server Internet services", *Computer Standards & Interfaces*, Vol. 27, No. 1, pp. 39-51, November 2004.

[5] I.C. Lin, M.S. Hwang, L.H. Li, "A new remote user authentication scheme for multi-server architecture", *Future Generation Computer Systems*, Vol. 19, No. 1, pp. 13-22, January 2003.

[6] J. H. Lee, D. H. Lee, "Efficient and Secure Remote Authenticated Key Agreement Scheme for Multi-server Using Mobile Equipment", *Proceedings of International Conference on Consumer Electronics*, pp. 1-2, January 2008.

[7] L. Hu, X. Niu, Y. Yang, "An Efficient Multi-server Password Authenticated Key Agreement Scheme Using Smart Cards", *Proceedings of International Conference on Multimedia and Ubiquitous Engineering*, pp. 903-907, April 2007.

[8] X. Cao, S. Zhong, "Breaking a remote user authentication scheme for multi-

server architecture", *IEEE Communications Letters*, Vol. 10, No. 8, pp. 580-581, August 2006.

[9] Z.F. Cao, D.Z. Sun, "Cryptanalysis and Improvement of User Authentication Scheme using Smart Cards for Multi-Server Environments", *Proceedings of International Conference on Machine Learning and Cybernetics*, pp. 2818-2822, August 2006.

[10] C.C. Chang, J.Y. Kuo, "An efficient multi-server password authenticated key agreement scheme using smart cards with access control", *Proceedings of International Conference on Advanced Information Networking and Applications*, Vol. 2, No. 28-30, pp. 257-260, March 2005.

[11] R.J. Hwang, S.H. Shiau, "Password authenticated key agreement protocol for multi-servers architecture", *Proceedings of International Conference on Wireless Networks*, Vol. 1, No. 13-16, pp. 279-284, June 2005.

[12] C.C. Chang, J.S. Lee, "An efficient and secure multi-server password authentication scheme using smart cards", *Proceedings of International Conference on Cyberworlds*, No. 18-20, pp. 417-422, November 2004.

[13] W.S. Juang, "Efficient multi-server password authenticated key agreement using smart cards", *IEEE Transactions on Consumer Electronics*, Vol. 50, No. 1, pp. 251-255, February 2004.

[14] H.C. Hsiang, W.K. Shih, "Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment", *Computer Standards & Interfaces*, In Press, Available online December 2008.

[15] Y. Chen, C.H. Huang, J.S. Chou, "Comments on two multi-server authentication protocols", http://eprint.iacr.org/2008/544, December 2008.