南　華　大　學

資訊管理學系

碩士論文

運用智慧卡建立多伺服器認證的通訊協定

A novel multi-server authentication protocol

研 究 生：黃純慧

指導教授：周志賢 博士

中華民國九十八年六月

南　華　大　學

資訊管理學系

碩 士 學 位 論 文

運用智慧卡建立多伺服器認證的通訊協定
A novel multi-server authentication protocol

研究生：黃純慧

經考試合格特此證明

口試委員：

指導教授：

系主任(所長)：

口 試 日 期 ： 中 華 民 國 九 十 八 年 六 月 二 十 四 日

運用智慧卡建立多伺服器認證的通訊協定

學生：黃純慧　　指導教授：周志賢 博士

南　華　大　學　資訊管理學系碩士班

摘　　要

最近，Tsai 和 Hsiang 等人分別提出運用智慧卡建立多伺服器認證的通訊協定。他們宣稱，他們的通訊協定是安全並且可阻擋各種攻擊。然而，我們發現他們的通訊協定中存在著安全漏洞。在本文中，我們先說明這兩個機制中的安全漏洞，然後提出新的通訊協定。經過安全性分析後，這個新機制是運用智慧卡建立多伺服器認證的通訊協定中最安全且有效率的。

關鍵詞：　多伺服器，密碼認證協定，智慧卡，密碼變更，金鑰協議

A novel multi-server authentication protocol

Student：Chun-Hui Huang　　Advisors：Dr. Jue-Sam Chou

Department of Information Management
The M.I.M. Program
Nan-Hua University

ABSTRACT

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

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

目　錄

表　　目　　錄

圖　　目　　錄

Chapter 1 Introduction

For the possible extension of a communication network, a two-party password authentication protocol for a client-server architecture might not be sufficient for efficiently accommodating various users requirements. Due to this observation, several multi-server protocols were proposed [1-15] attempting to resolve this problem.

In 2003, Li *et al*. [4] proposed a multi-server protocol based on ElGamal digital signature and geometric transformations on an Euclidean plane. Unfortunately, their protocol had been broken by Cao and Zhong [12]. In 2004 and 2005, Tsaur *et al*. [9, 10] proposed two multi-server schemes. However, both of their schemes are based on Lagrange interpolating polynomial which is computationally intensive as indicated in [10]. In 2006 and 2007, Cao *et al*. [15] and Hu *et al*. [7] each proposed an authentication scheme for a multi-server environment, respectively. Both of their schemes assume that all servers are trustworthy. Nevertheless, this assumption is not always true, as stated in [6]. In 2008, Lee *et al*. [5] proposed an authenticated key agreement scheme for multi-server using mobile equipment. But their scheme can not add a server freely since when a server is added, all users who want to login to the new added server have to re-register at the registration center for getting a new smart card. This increases the registration center's card-issue overhead. Also, in 2008, Tsai [6] 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*. [14] proposed a secure dynamic ID scheme for multi-server environments. They claim that their protocol is secure. However, Hsiang and Shih [3] found their scheme suffers from both the server spoofing attack and the insider attack (We also illustrated the same weakness of Liao *et al*.'s scheme in [13]). Hence, they proposed an improvement on Liao *et al*.'s protocol to get rid of the found weakness. Yet, we found that Hsiang *et al*.'s improvement is still insecure. It is vulnerable to the insider attack. In this paper, we will first show the attacks on [6] and [3] respectively. After that, we show our scheme and then examine its security.

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

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

In this chapter, we will review Tsai's protocol in Section 2.1 and Hsiang-Shih's protocol in Section 2.2. 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*'.

H(·) : a collision-resistant one-way hash function

(*a*,*b*) : a string denotes that string *a* is concatenated with string *b*.

$\oplus$: a bitwise exclusive or operator

g : a primitive element in a Galois field GF(p), where p is a large prime number.

=> : a secure channel

→ : a common channel

2.1 Review of Tsai's protocol

Tsai's protocol consists of 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. It assumes that there are *s* servers existing in the system. At beginning, RC computes and sends H(SID_j,*y*) to each server S_j for j = 1 to *s*, with S_j keeping it secret, via a secure channel. We describe them as follows.

(1) Registration phase

In this phase (as shown in Fig. 1), U_u performs the following steps for obtaining

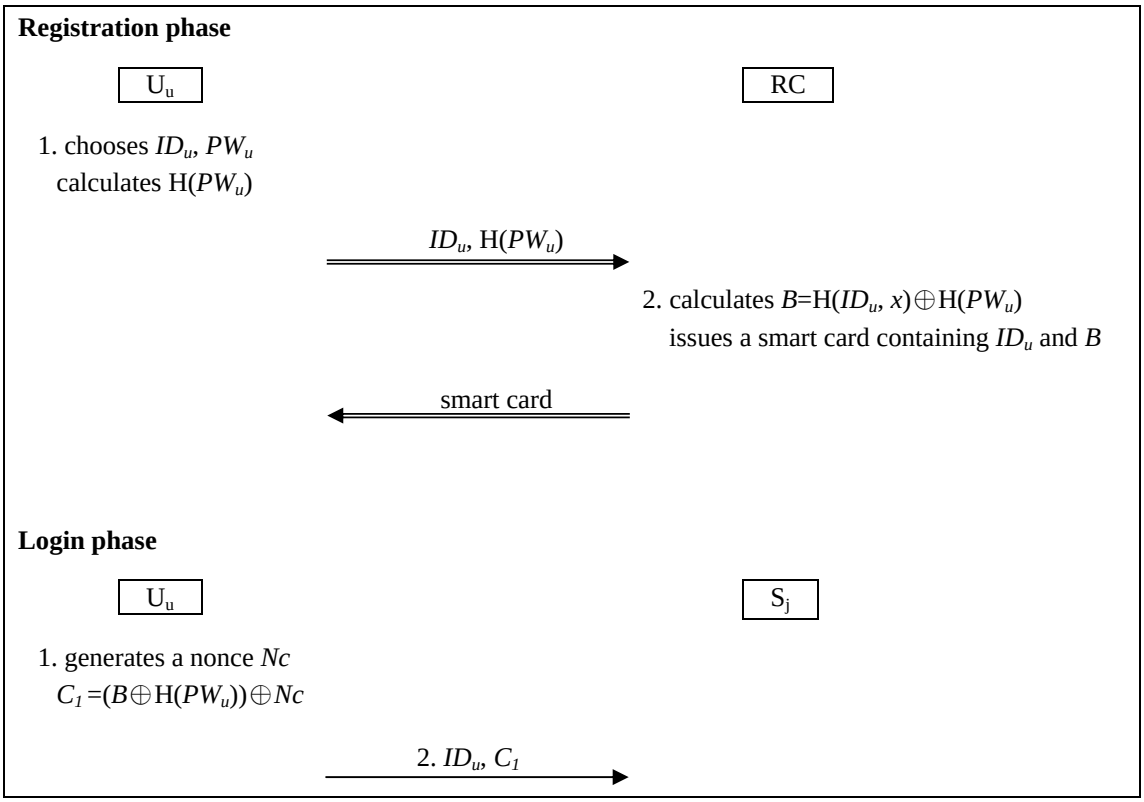

Fig. 1. Registration phase and login phase of Tsai's protocol

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

In this phase (also shown in Figure 1), 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.

(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 chosen nonces respectively. To increase the protocol's efficiency, 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.

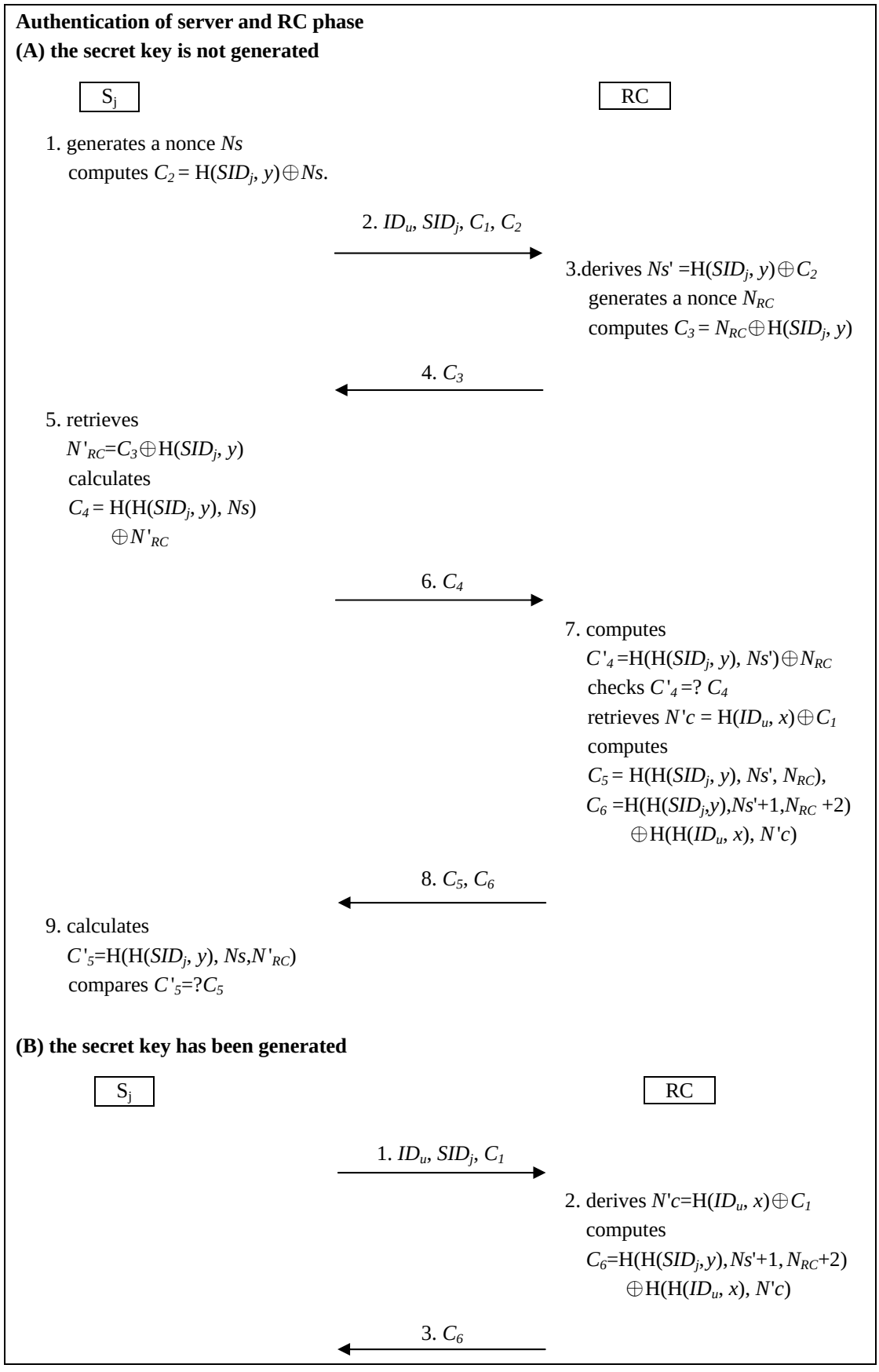

Fig. 2. Authentication of server and RC phase of Tsai's protocol

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 $Auth_{S\text{-}RC}=H(H(SID_j, y), Ns+1, N'_{RC}+2)$ in the verifier table for the next execution of this phase.

(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 together perform the following steps for mutual authentication and establishing a common session key (as shown in Fig. 3).

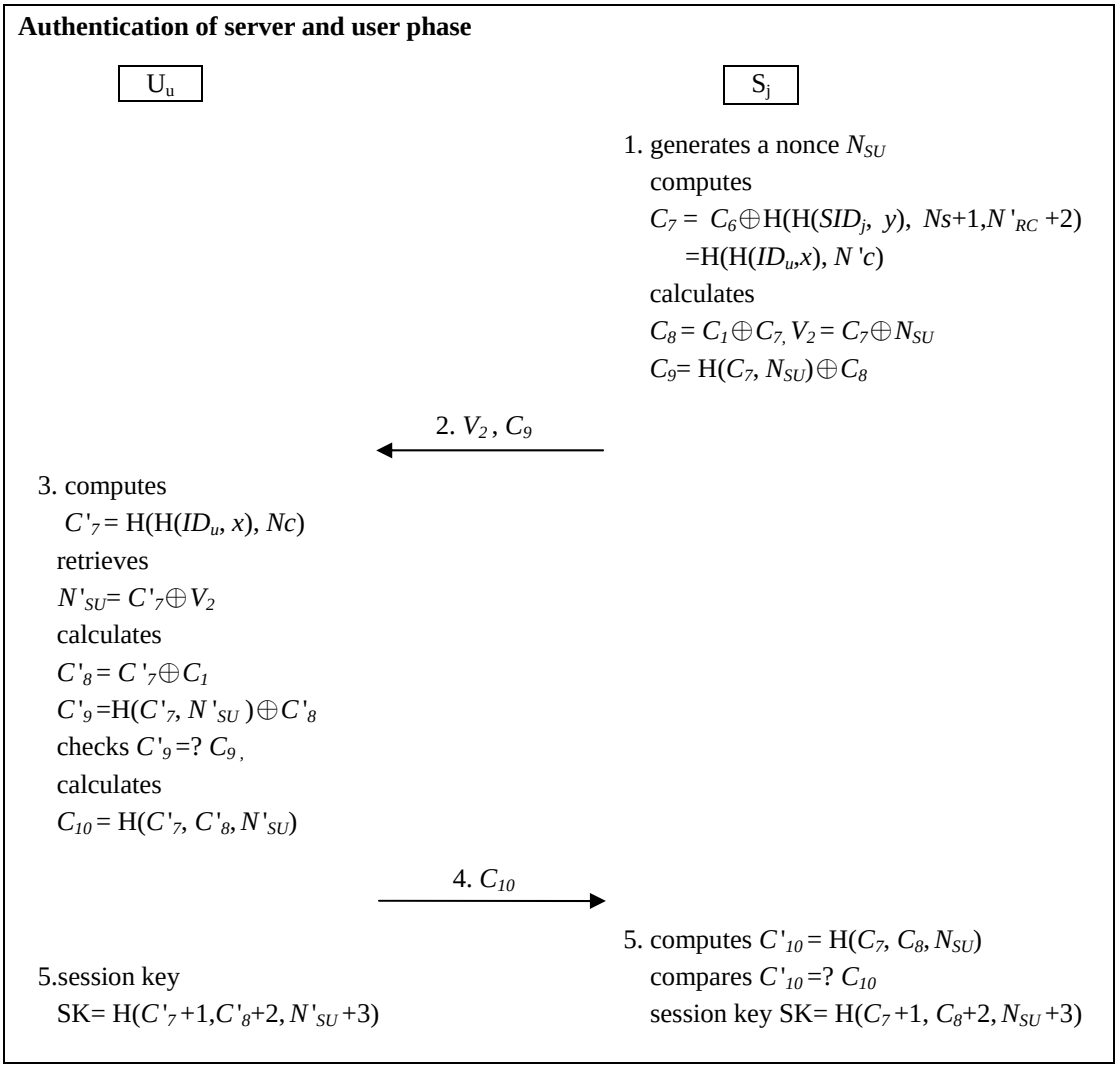

Fig. 3. Authentication of server and user phase of Tsai's protocol

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 $N'_{SU} = C'_7 \oplus V_2$, 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 calculates $C_{10} = H(C'_7, C'_8, N'_{SU})$.

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, U_u and S_j, 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.

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 has three secrets, x, y and r. He first computes and sends $H(SID_j, y)$ to the 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. U_u chooses PW_u and a random number b_u, and computes $H(b_u \oplus PW_u)$. He then

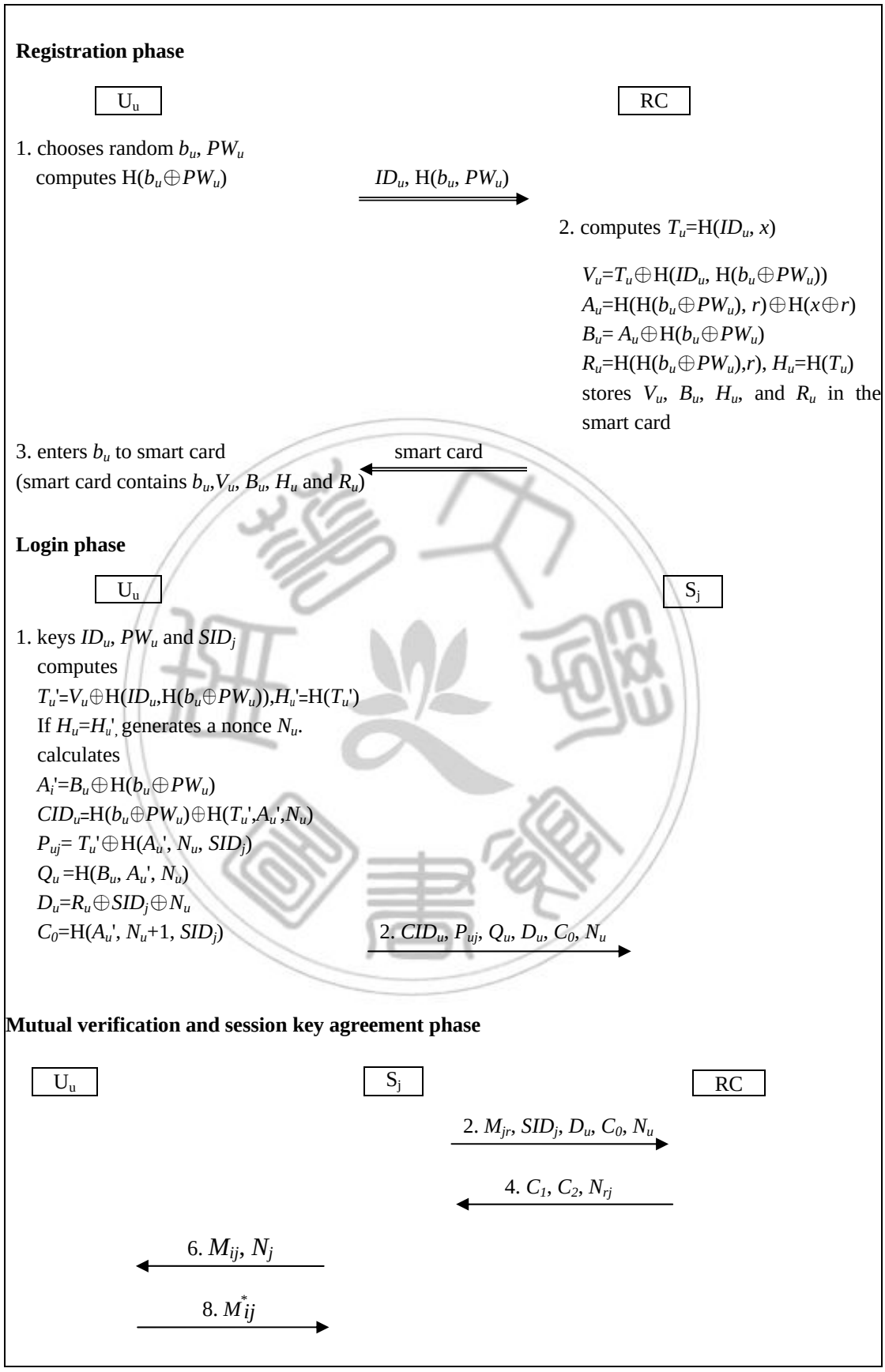

Fig. 4. Hsiang-Shih's protocol

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 $A_u'=B_u \oplus H(b_u \oplus PW_u)$, $CID_u= H(b_u \oplus PW_u) \oplus H(T_u', A_u', N_u)$, $P_{uj}= T_u' \oplus H(A_u', N_u, SID_j)$, $Q_u =H(B_u, A_u', N_u)$, $D_u=R_u \oplus SID_j \oplus N_u$, 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 for authenticating each other and computing 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 $N_{jr}^{*}=M_{jr}\oplus H(SID_j, y)$, $R_u^{*}=D_u\oplus SID_j\oplus N_u$, $A_u^{*}=R_u^{*}\oplus H(x\oplus r)$, and $C_o^{*}= H(A_u^{*}, N_u+1, SID_j)$. He checks to see if C_o^{*} is equal to the received C_0. If so, the message is accepted. 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 $B_u''= A_u''\oplus h_u$. After that, he computes $H(B_u'', A_u'', N_u)$ and compares it with the value Q_u 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 $M_{ij}=H(B_u'', N_i, A_u'', SID_j)$.

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}^{*}. If so, U_u is authentic.

10. After finishing mutual authentication, U_u and S_j can compute the common session key as $SK= H(B_u, A_u, N_u, N_j, SID_j)$ 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.

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

In this chapter, we will show that Tsai's protocol suffers from the server spoofing attacks on both scenarios and Hsiang-Shih's protocol suffers from the user impersonation attack, and the off-line password guessing attack if the smart card is lost. We demonstrate these 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 secret $H(SID_i, y)$. He then can masquerade as a legal server S_j to cheat a remote user. This is because in Tsai's protocol, in the authentication of server and user phase, a user doesn't examine if the message is sent from the correct server. In the following, we present the server spoofing attacks on scenarios (A) and (B), and illustrate them in Figure 5 and 6, respectively.

(A) the secret key is not generated

For this case, we show the attack as follows.

1. When U_u wants to communicate with S_j, he starts the protocol and sends $\{ID_u, C_1\}$ to S_i (who masquerades as S_j).
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 values C_3, C_4, C_5, and C_6 in the

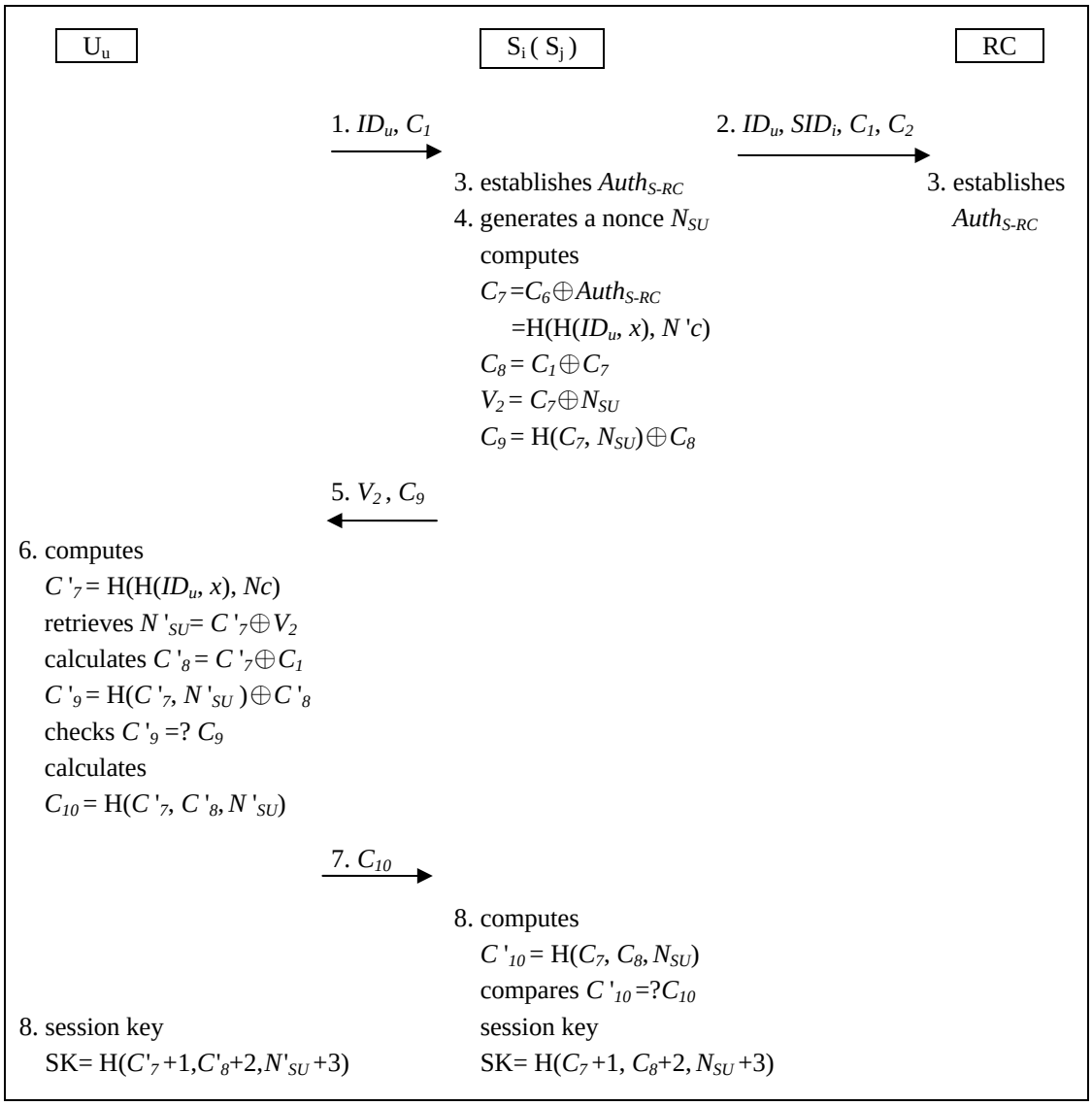

Fig. 5. Server spoofing attack by an insider server on Tsai's protocol for scenario (A) the secret key is not generated.

transmitted messages, except for C_6 which is not used for comparison, 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 can 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 C_{10}. If so, U_u is authentic. They, U_u and S_i, can then 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, we can see that a server spoofing attack can be successfully launched by the insider attacker S_i.

(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.

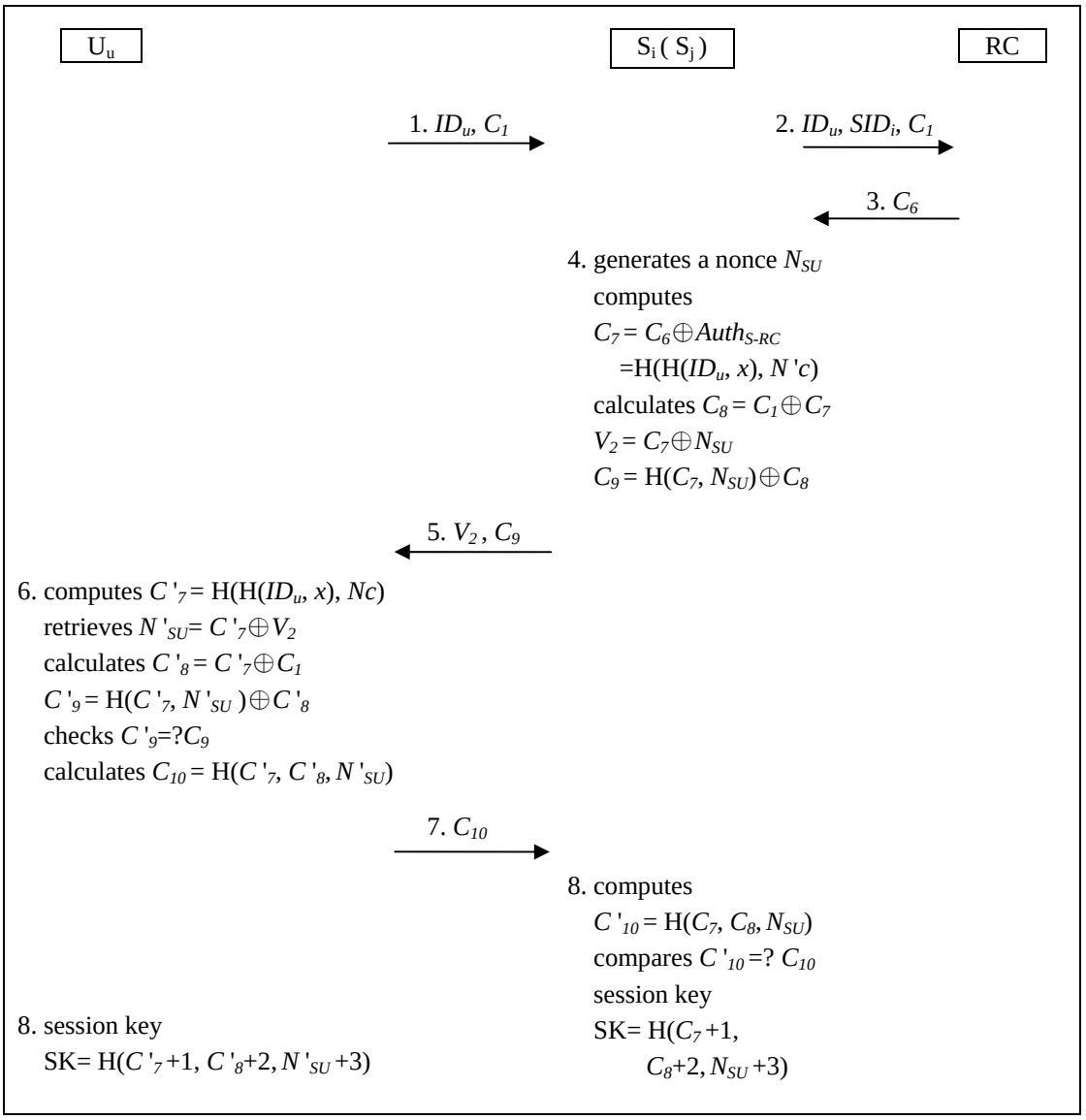

Fig. 6. Server spoofing attack by an insider server on Tsai's protocol for scenario (B) the secret key has been generated.

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 $C_6 = H(H(SID_i, y), Ns'+1, N_{RC}+2) \oplus H(H(ID_u, x), N'c) = Auth_{S\text{-}RC} \oplus H(H(ID_u, x), N'c)$.
3. RC sends $\{C_6\}$ to S_i. S_i then performs the authentication of server and user phase together 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})$\oplus C_8$.

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. Henceforth, S_i disguising himself as S_j is therefore regarded as being authentic by U_u. 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, we can see that the server spoofing attack launched by insider attacker S_i can be successfully accomplished.

3.2 Attack on Hsiang-Shih's protocol

In the following, we demonstrate two attacks, (1) the impersonation attack and (2) the off-line password guessing attack if the smart card is lost, on Hsiang-Shih's protocol.

(1) The impersonation attack

For this attack, we further divide it into two cases: (a) outsider impersonation

attack, and (b) insider impersonation attack.

(a) 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 impersonate U_u to access all servers's resources successfully.

(b) Insider impersonation attack

Assume that attacker E is a malevolent user registered at RC. He can use his secret b_e, PW_e, B_e, and 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 successfully in the same manner as the just described (a).

(2) 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. Hence, 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 exists in Hsiang-Shih's protocol.

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.

Chapter 4 Our protocol

After presenting the attacks on protocols [6] and [3], in this chapter, we present our novel method. 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. In the following, we describe the first two phases and also depict them in Figure 7. Then, for more clarity, we combine the last two phases into the phase "login for authentication and session key agreement or for authentication and password change phase" and divided it into three scenarios: (A) the first time execution, (B) not the first time execution, and (C) login for authentication and password change, with each scenario containing two phases. In the following, we discuss these three scenarios in turn.

(1) Preparation phase

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

1. RC computes $RSj=H(SID_j, y)$.
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, U_u can use the card to login to any eligible server for accessing resources.

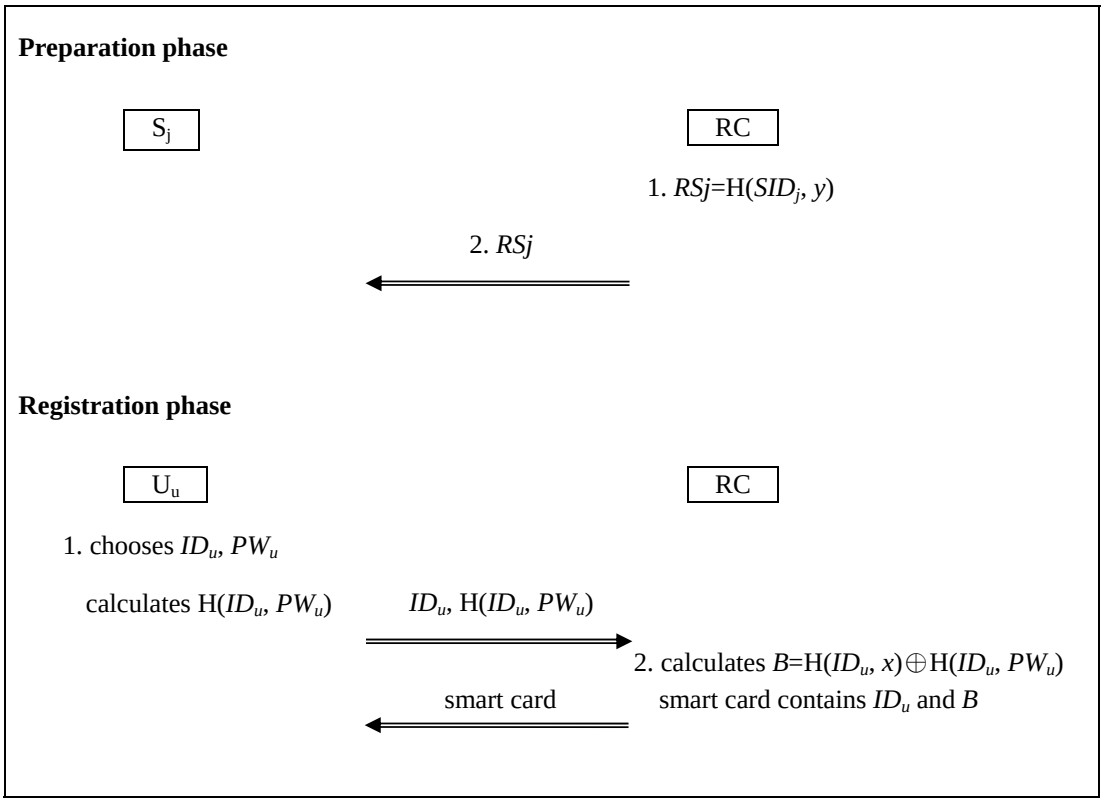

Fig. 7. Preparation phase and registration phase

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)⊕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 authentication and password change phase

In our scheme, when U_u wants to login S_j, he may want to execute either authentication and session key agreement or authentication and password change. We first describe the 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 authentication and password change phase. We describe them as follows.

(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 8.

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*'.

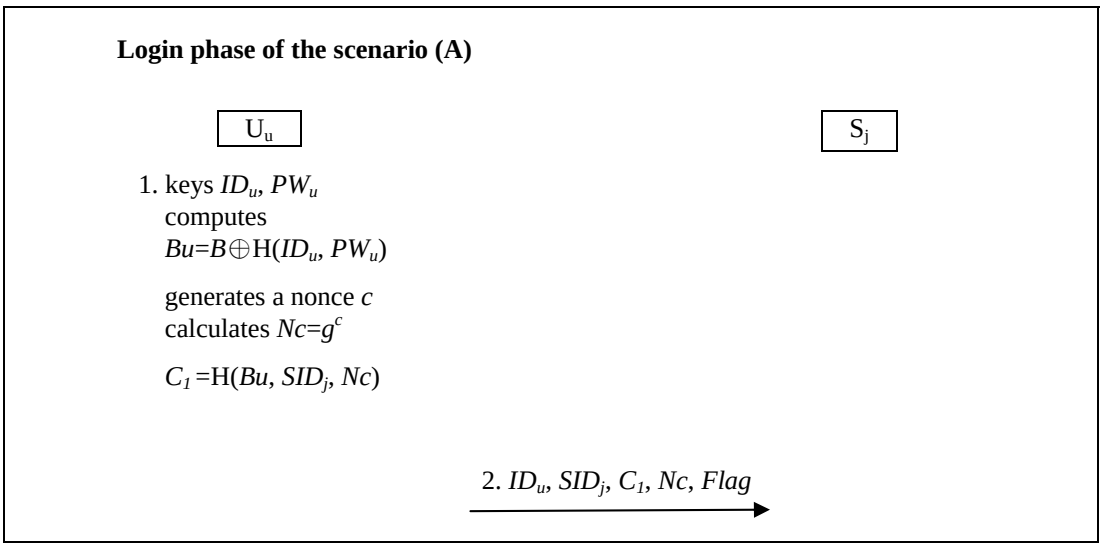

Fig. 8. Scenario (A): the first time execution (of login for authentication and session key agreement phase

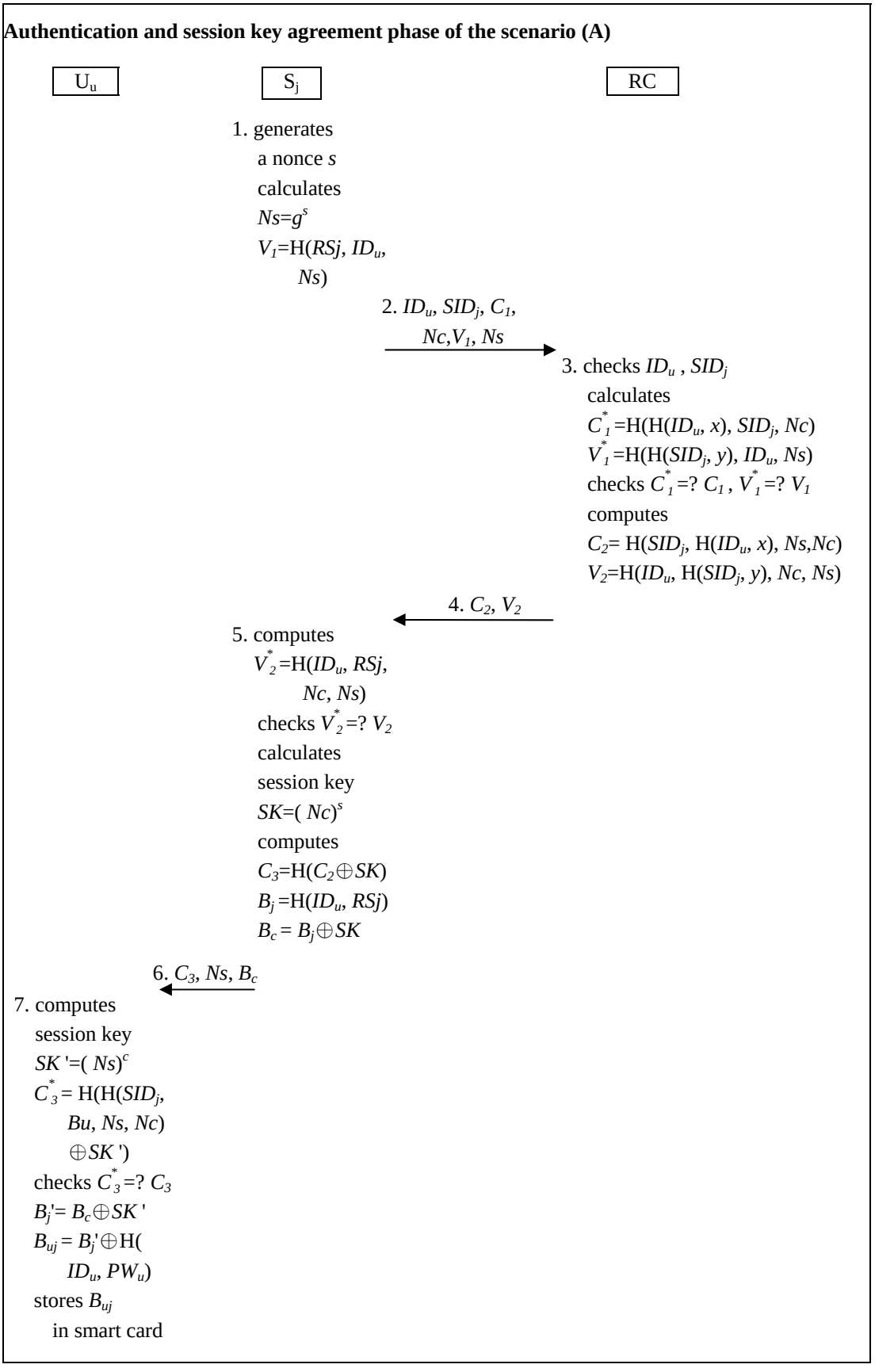

Fig. 8-continued. Scenario (A): the first time execution (of login for authentication and session key agreement phase)

(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. The steps are also illustrated in Figure 8-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 $B_j' = B_c \oplus SK'$, $B_{uj} = B_j' \oplus H(ID_u, PW_u)$ and stores the common secret key B_{uj} in his smart card for the use of next time login without the help of RC's authentication. 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 two phases. The steps are also illustrated in Figure 9.

(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 doesn't login 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 to the received C. If so, S_j confirms that U_u is authentic. He then calculates the session key $K=(Nu)^j$ and

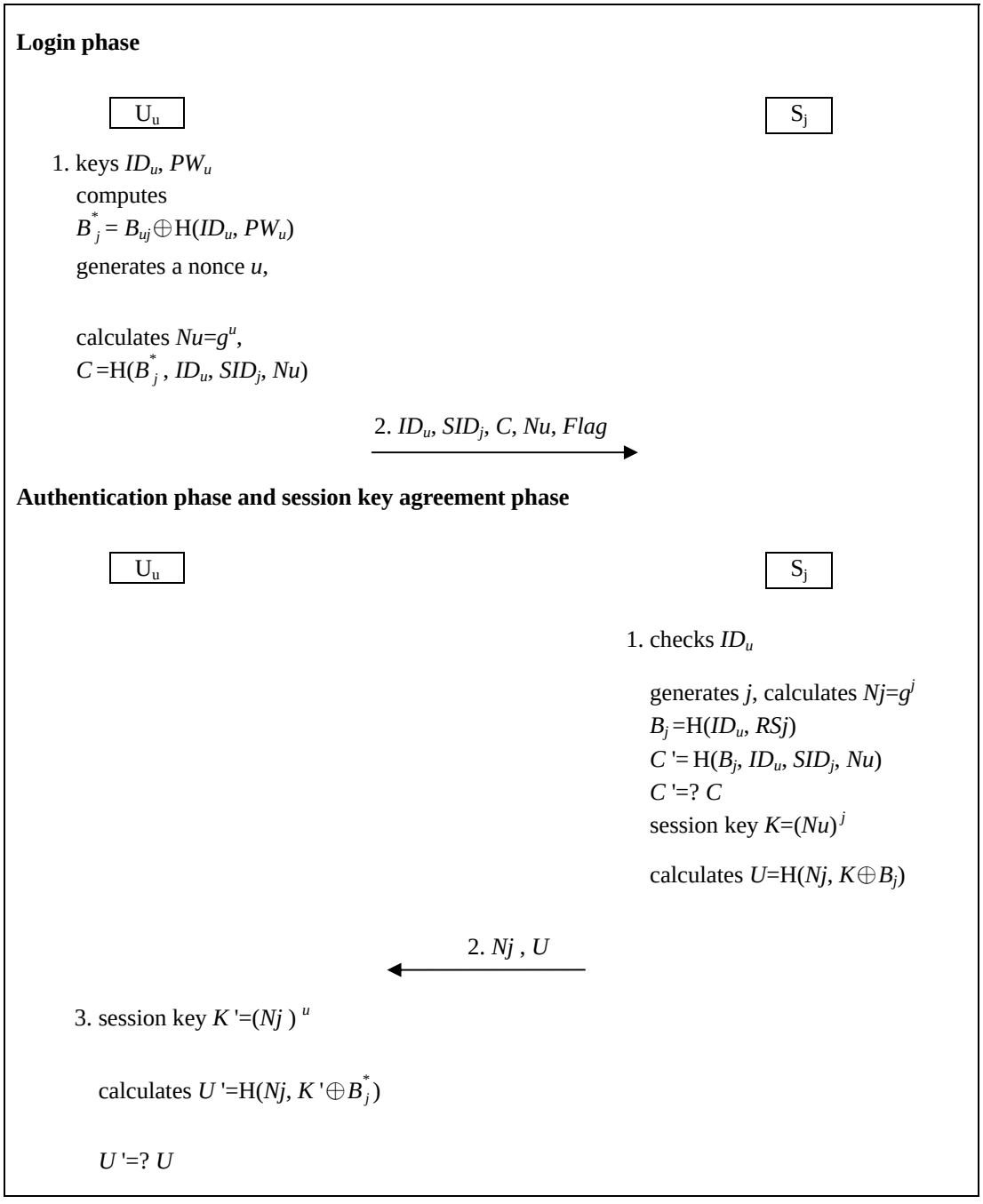

Fig. 9. Scenario (B): not the first time execution (of login for authentication and session key agreement phase)

$U=H(Nj, K \oplus B_j)$.

2. S_j sends $\{Nj, U\}$ to U_u.

3. After receiving the message from S_j, U_u computes the session key $K'=(Nj)^u$, $U'=$

($Nj, K' \oplus B_j^*$) 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

To get rid of the weakness caused by losing the smart card as described in Section 3.2.(2) of Chapter 3, when executing the password change phase, the client had better go through the supervision of RC. Under such a limitation, it not only can let the client choose and change his password freely but also can prevent the password guessing attack if the smart card is lost. Due to this observation, the password change phase of our design goes through the intervention of RC. It 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 two phases. The steps are also illustrated in Figure 10.

(a) Login phase

1. U_u keys his ID_u, PW_u, and a 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*'.

(b) Authentication and password change phase

1. From *Flag*, RC knows this login request from U_u is for password change. He

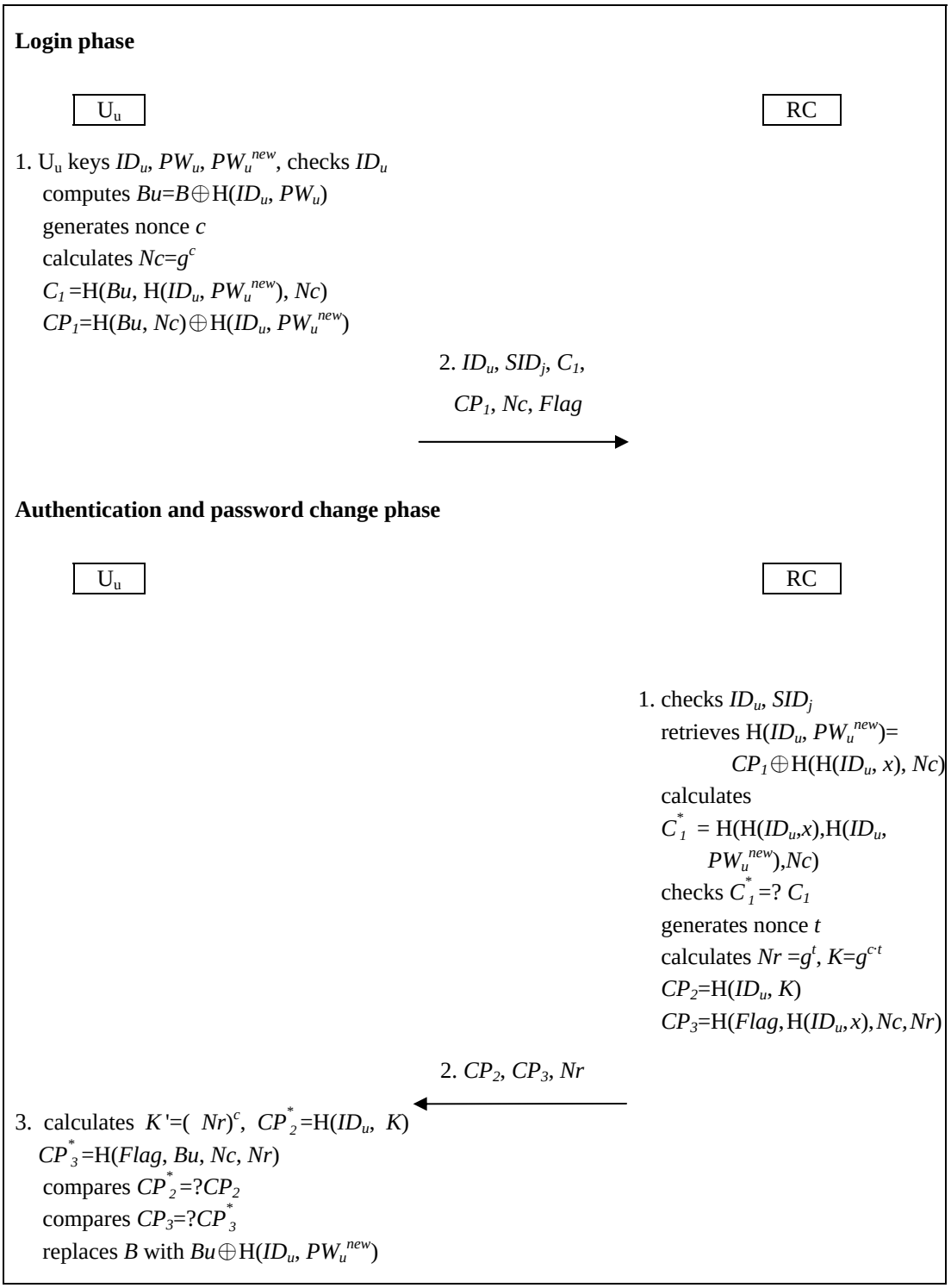

Fig. 10. Scenario (C): Login for authentication and password change phase

checks to see if ID_u and SID_j are valid. If they are valid, RC then 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 then generates a random nonce t and calculates $Nr = g^t$, $K = g^{c \cdot t}$, 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 by RC. The smart card then replaces the stored B with $Bu \oplus$ H(ID_u, PW_u^{new}).

Chapter 5 Security analysis of 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), smart-card-lost attack, and so on. For abbreviation, in the following, we use notations D(A), D(B), and D(C), to denote the discussion for the three scenarios, (A), (B), and (C), respectively (as described in Section 4.(3) of Chapter 4). In each discussion, we demonstrate how our protocol can achieve each of the above-mentioned security property, if the analysis for that scenario is needed.

5.1 Mutual authentication

D(A): Mutual authentication between each pair among the three parties, user U_u, server S_j and RC.

As shown in Figure 8-continued, 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, if we use A $\rightarrow$ B to represent A authenticates B, or equivalently, B is regarded as being authentic by A, we can demonstrate these

relations using the two solid arrows, ① and ②, as indicated in Figure 11. RC 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 also depicted in Figure 11 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. It is obvious that authenticity relationship has transitive property by way of the intermediate node when the identities of both communicating parties and their common secret are committed, *e.g.*, to a hash function. For example, if A $\longrightarrow$ B and B $\longrightarrow$ C, then A $\longrightarrow$ C by way of B. According to this rule, we can obtain the dashed arrow ⑤ from ③ and ④ (as shown in Figure 11). The remaining dashed arrow ⑥ can be obtained by using the following proof of contradiction. For we already know the facts that RC $\longrightarrow$ S_j and RC $\longrightarrow$ U_u by way of S_j, if S_j $\longrightarrow$ U_u doesn't hold, then from the transitive property, RC $\longrightarrow$ U_u by way of S_j can't hold as well. This contradicts the fact that RC $\longrightarrow$ U_u by way of S_j. Hence, the dashed arrow ⑥ must exist. This completes the proof and Figure 11 thus exists. From the figure, we can see that the mutual authentications between each pair of the three parties are satisfied.

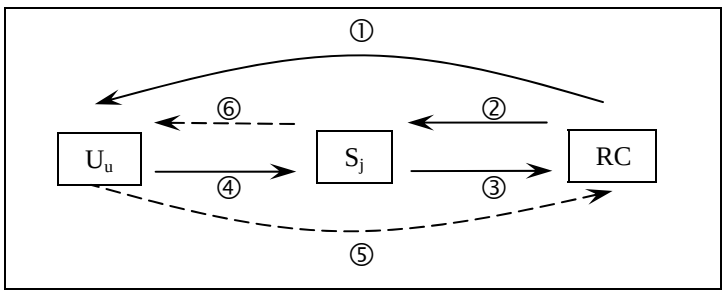

Fig. 11. Authenticity relationship

D(B): Mutual authentication between user U_u and server S_j.

For authenticating U_u after receiving his login request {ID_u, SID_j, C, Nu, $Flag$}, S_j verifies the validity of C. If it is valid, he confirms that U_u is authentic and then sends Nj and U to U_u. U_u verifies the validity of U. If it is valid, he confirms that S_j is authentic.

D(C): Mutual authentication between user U_u and RC.

For authenticating U_u after receiving his password change request {ID_u, SID_j, C_1, CP_1, Nc, $Flag$}, RC verifies the validity of C_1. If it is valid, he confirms that U_u is authentic and then sends CP_2, CP_3 and Nr to U_u. U_u verifies the validities of CP_2 and CP_3. If they are valid, he confirms that RC is authentic.

5.2 Session key agreement

D(A): 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) of Chapter 4.

D(B): It can also be easily seen in step 1 and step 3 of the authentication and key agreement phase executed by S_j and U_u in Section 4.(3).(B).(b) of Chapter 4.

5.3 Perfect forward and backward 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 and thereafter in each session between the user and server are independent. Accordingly, our scheme provides perfect forward and backward secrecy in both scenarios, (A) and (B).

5.4 Changing password freely and securely

In our protocol, user U_u can change his password freely and securely. Even an attacker E can temporarily obtain U_u's smart card, E can't change U_u's password from PW_1 to PW_2 without the knowledge of U_u's password, where PW_1 and PW_2 are two passwords guessed and selected by E respectively. He can't replace B with $B \oplus H(ID_u, PW_1) \oplus H(ID_u, PW_2)$, where B is the value stored in U_u's smart card. 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) of Chapter 4.

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 the stolen-verifier attack.

5.6 Preventing the insider-server spoofing attack

Our scheme needn't to assume that all servers are trustworthy as required in [7, 9,

12]. For in our protocol, if S_i wants to masquerade as S_j, he will be rejected. Since without the knowledge of S_j's secret $RSj = H(SID_j, y)$, he cannot be regarded as S_j successfully by both RC and U_u. In the following, we will discuss this for the two scenarios, (A) and (B), in D(A) and D(B) respectively.

D(A): If S_i sends the message, as in step 2 of Section 4.(3).(A) of Chapter 4, to RC by impersonating S_j, then without the knowledge of S_j's *RSj* and RC's secret key *y*, S_i can't obtain the value $V_1 = H(H(SID_j, y), ID_u, Ns)$ to be regarded as authentic by RC.

Assume that S_i, intercepting the messages $\{Nc\}$ in step 2 and $\{C_2\}$ in step 4 as shown in Figure 8-continued, wants to impersonate S_j to U_u. He sends the message $\{C_3, g^{s'}, B_c\}$ to U_u for step 6, where $C_2 = H(SID_j, H(ID_u, x), Ns, Nc)$, $Ns = g^s$, *s* is a noce selected by S_j, s' is a noce selected by S_i, $C_3 = H(C_2 \oplus (Nc)^{s'})$, and $B_c = H(ID_u, RSi) \oplus SK$. U_u computes $C_3^* = H(H(SID_j, Bu, g^{s'}, Nc) \oplus (g^{s'})^c)$ and compares it with C_3, where $Bu = H(ID_u, x)$, $Nc = g^c$. U_u will find that they are not equal and terminate the session. Hence, S_i can't impersonate S_j to be authenticated by U_u successfully.

D(B): Assume that S_i, intercepting the login messages $\{ID_u, SID_j, C, Nu, Flag\}$ from U_u as depicted in Figure 9, wants to impersonate S_j to U_u. He chooses a random number *i* and sends the message $\{g^i, U\}$ to U_u, where $U = H(g^i, (Nu)^i \oplus H(ID_u, RSi))$. U_u computes $U' = H(g^i, (g^i)^c \oplus B_j^*)$ and compares it with the received *U*, where $B_j^* = B_{uj} \oplus H(ID_u, PW_u) = H(ID_u, RSj)$. He will find they are not equal and terminate the session. Hence, S_i can't impersonate S_j to U_u successfully in this

case.

5.7 Preventing insider-user impersonating attack

In our scheme, if a legal malicious U_n wants to impersonate U_u to login to S_j. Without the knowledge of U_u's B and PW_u, he cannot be regarded as U_u by both RC and S_j successfully. In the following, we demonstrate that how our protocol can prevent this kind of attack in scenarios, (A) and (B).

D(A): In this scenario, assume that U_n impersonates U_u and sends the loging request $\{ID_u, SID_j, C_1, g^{c'}, Flag\}$ to S_j, where C_1 =H(Bn, SID_j, $g^{c'}$), Bn=$B \oplus$H(ID_n, PW_n) =H(ID_n, x), and c' is a nonce selected by U_n. The value of C_1 which U_n computes would be different from the value of RC's computation C_1^*=H(H(ID_u, x), SID_j, $g^{c'}$) as shown in Figure 8-continued. Hence, he can't be authenticated by RC.Therefore, the insider impersonating attack fails.

In another case of this scenario, assume U_n intercepting U_u's loging request $\{ID_u, SID_j, C_1, g^c, Flag\}$ wants to impersonate U_u and resends it to S_j without modification, where c is a nonce selected by U_u. After the completion of mutual authentication between S_j and RC, S_j sends $\{C_3, g^s, B_c\}$ to U_n, where $B_c = B_j \oplus (g^c)^s$ =H(ID_u, RSj)$\oplus(g^c)^s$ and s is a nonce selected by S_j. U_n can deduce neither B_j nor SK by computing $B_j = B_c \oplus (g^s)^c$ and SK= $(g^s)^c$. Since U_n doesn't have the knowledge of c. Hence, the insider impersonating attack fails.

D(B): Similarly, in this scenario, if U_n wants to impersonate U_u and sends a login request to S_j, he can not be authenticated by S_j successfully. Since he doesn't

have the correct B_j= H(ID_u, RSj) for computing C to be authenticated by S_j as indicated in Figure 9. Even if U_n is lucky enough to be authenticated successfully by S_j by resending him the login message from U_u and receives the message {Nj, U} from S_j, he can't obtain the correct session key $K'=(Nj)^u$ for he doesn't have the knowledge of the random nonce u which is selected by U_u. Hence, the insider impersonating attack fails.

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

D(A): In this scenario, 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, B_c}, to and from through the Internet. Assume that an attacker E who hasn't got U_u's smart card wants to guess U_u's password PW_u. (If E has got U_u's smart card, this case is termed as smart-card-lost attack. We will discuss this attack in section 5.11.) We argue that E will not succeed. For among all the transmitted messages (M_1 through M_4), 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, E can hardly succeed in guessing PW_u due to the one-way property of a hash function and the unknown value of B, E therefore can't implement an 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. The reasons for scenarios (B) and (C) are similar to the above-mentioned. We omit them here.

5.9 Preventing replay attack

D(A): Scenario (A) 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, C_3 and B_c. Henceforth, it is obvious that our protocol in this scenario can prevent any replay attack.

D(B): Similarly, in scenario (B), it also uses two different random nonces, *u* and *j*, each time in computing *Nu* and *Nj* which are needed in computing the related values, such as *C* and *U*. Hence, the replay attack in this scenario can also be avoided.

D(C): Again, this scenario uses different random nonces, *c* and *t*, each time in computing *Nc* and *Nr* which are needed in computing the related values, such as C_1, CP_1, CP_2 and CP_3. Therefore, this scenario also can prevent the replay attack.

From the above analysis, we can see that in our protocol, an attacker cannot be authenticated successfully by resending any previous transmitted values to the intended party.

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

Assume that an attacker E wants to launch a parallel session attack by

masquerading as both S_j to U_u and U_u to S_j. We demonstrate how our protocol can withstand this kind of attack in the following.

D(A): After receiving the login message $M=\{ID_u, SID_j, C_1, Nc, Flag\}$ from U_u, E masquerading as U_u starts another protocol by resending M to S_j. For more clarity, we roughly show this case in Figure 12. Although, using value C_1, E can pass RC's authentication after S_j sends M 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. This is because E, without the knowledge of c, can't compute the correct session key SK', $(Ns)^c$, shared with S_j. Similarly, E 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, E can't obtain C_2, V_2 from RC to compute a valid $C_3=H(C_2 \oplus SK)$ which will be transmitted to U_u for U_u to authenticate S_j. In other words, E can not succeed in impersonating S_j to U_u. Therefore, the parallel session attack fails in this scenario.

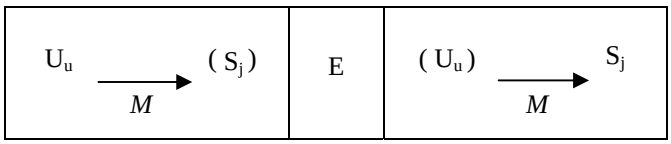

Fig. 12. Parallel session attack

D(B): After receiving the login message $M=\{ID_u, SID_j, C, Nu, Flag\}$ from U_u to S_j, E masquerading as U_u starts another protocol by resending M to S_j. Although, using value C, E can pass S_j's authentication. However, without the knowledge of U_u's nonce u, he can't finish the authentication and session key

agreement phase to impersonate U_u to S_j. This is because E, without the knowledge of *u*, can't compute the correct session key *K'*, $(Nj)^u$, shared with S_j. Hence, the parallel session attack fails.

D(C): Since, E doesn't obtain U_u's smart card, he can't replace the stored value *B* with any one as in step 3 of Figure 10 to finish the password change phase. Hence, the attack on this scenario is meaningless and doesn't exist.

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*. We will show that even under such a situation, our protocol still can prevent various attacks including: (1) insider impersonating attack, (2) outsider impersonation attack, (3) off-line password guessing attack, (4) on-line password guessing attack, (5) replay attack. We analyze each type of attack as follows.

(1) Insider impersonating attack: Although, U_n obtains the smart card and knows the value *B*. However, he can't use his guessing password *PW'* to be authenticated by RC without the knowledge of U_u's PW_u. For in scenario (A), the value C_1 =H($B \oplus$H(ID_u, *PW'*), SID_j, *Nc*) which U_n computes would be different from the value of RC's computation C_1^*=H(H(ID_u, *x*), SID_j, *Nc*), where *B*=H(ID_u, *x*)$\oplus$ H(ID_u, PW_u). And in scenario (B), the value *C* =H(B_j^*, ID_u, SID_j, *Nu*) U_n computes would be different from the value of S_j's computation *C'* =H(B_j, ID_u, SID_j, *Nu*), where $B_j^* = B_{uj} \oplus$H(ID_u, PW_u)= H(ID_u, *RSj*)$\oplus$H(ID_u, *PW'*)$\oplus$H(ID_u, PW_u), B_j= H(ID_u, *RSj*). As for scenario (C), the value C_1 =H(*Bu*, H(ID_u, PW_u^{new}),

Nc) which U_n computes would be different from the value of RC's computation $C_1^* = H(H(ID_u, x), H(ID_u, PW_u^{new}), Nc)$, where $Bu=B \oplus H(ID_u, PW') = H(ID_u, x) \oplus H(ID_u, PW_u) \oplus H(ID_u, PW') \neq H(ID_u, x)$. Hence, the insider U_n's login request for each of the three scenarios, (A), (B), and (C), can be detected. Therefore, the insider attack fails.

(2) Outsider impersonation attack: The reason for this attack prevention is similar to the analysis just mentioned above. Assume that an un-registered user getting U_u's smart card wants to impersonate U_u by starting the protocol with S_j for scenario (A) or (B), or with RC for scenario (C). However, without the knowledge of U_u's password PW_u, he can not deduce the correct value of $Bu=B \oplus H(ID_u, PW_u)=H(ID_u, x)$ to compute C_1 to be verified by RC for scenarios (A) and (C), or the correct value of $B_j^* = B_{uj} \oplus H(ID_u, PW_u) = H(ID_u, RSj)$ to compute C to be verified by S_j for scenario (B). Since each password guessing for each impersonation needs passing the verification of RC or S_j. Hence, the attacker could hardly be authenticated successfully. This means the outsider impersonation attack fails in our protocol.

(3) Off-line password guessing attack: Assume U_n gets the smart card, he wants to launch an off-line password guessing attack. For we know that the smart card contains ID_u, $B=H(ID_u, x) \oplus H(ID_u, PW_u)$ for scenarios (A) and (C), and ID_u and $B_{uj} = B_j' \oplus H(ID_u, PW_u) = H(ID_u, H(SID_j, y)) \oplus H(ID_u, PW_u)$ for scenario (B). We assume that U_n has the knowledge of $H(ID_u, x)$ or $H(ID_u, H(SID_j, y))$, he can check to see if his guessing password PW' is correct by comparing his computed

$B \oplus H(ID_u, PW')$ with $H(ID_u, x)$ or his computed $B_{uj} \oplus H(ID_u, PW')$ with $H(ID_u, H(SID_j, y))$. 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. However, any legal user or un-registered user can't obtain $H(ID_u, x)$ or $H(ID_u, H(SID_j, y))$ without the knowledge of RC's secrets x and y. Though U_n can get $H(ID_n, x)$ by computing $H(ID_n, x)=B \oplus H(ID_n, PW_n)$, where B is stored in U_n's smart card, he can't deduce x from $H(ID_n, x)$ to calculate $H(ID_u, x)$. Since $H(\cdot)$ is a collision-resistant one-way hash function. Hence, no one can implement the off-line password guessing attack.

(4) On-line password guessing attack: For an invalid login request or password chang request can be detected with the wrong password as described in (1), and each password guessing for each impersonation attack needs the help of RC's or S_j's verification as described in (2), the attacker could hardly be authenticated successfully. In other words, without a correct password PW_u to compute the specified $Bu=B \oplus H(ID_u, PW_u)$ or $B_j^* = B_{uj} \oplus H(ID_u, PW_u)$, U_n can't pass RC's or S_j's verification (see Figure 8, 9). Moreover, if we set the protocol to tolerate three times of wrong password logins, the impossibility of on-line password guessing attack can be easily seen. Hence, the on-line password guessing attack doesn't exist in our scheme.

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

Chapter 6 Discussion

In this chapter, we first show that our protocol meets the requirement of single registration property in Section 6.1. Then, to show the advantages of our scheme, in Section 6.2, we compare our protocol with both Tsai's [6] and Hsiang-Shih's [3] in communication cost. In Section 6.3, we compare our protocol with [1, 5, 8, 9, 10] in the aspects of card-issue cost. Section 6.4 discusses why our scheme doesn't adopt the dynamic ID concept as do in [14, 3] and Section 6.5 shows the importance of RC-off-line authentication. Finally, we make a comparison table for our protocol and the other proposed schemes so far in Section 6.6.

6.1 Single registration

In our protocol, it is not necessary for a user to register at each server. He is required to register at RC only once. Thereafter, he can access all of the legal servers' resources by only using one password.

6.2 Low communication cost

As indicated in Figure 1 through Figure 3, Tsai's protocol [6] needs seven passes for scenario (A) (one pass for login phae, four passes for authentication of server and RC phase, and two phases for authentication of server and user phase) and five passes for scenario (B) (one pass for login phase, two passes for authentication of server and

RC phase, and two passes for authentication of server and user phase). Hsiang-Shih's protocol [3] needs five passes whenever a user wants to login to a server (one pass for login phase, and four passes for mutual verification and session key agreement phase) as indicated in Figure 4. However, ours needs only four passes for scenario (A) (one pass for login phae and three passes for authentication and session key agreement phase) and only two passes for scenario (B) (one pass for login phae and one pass for authentication and session key agreement phase). Therefore, our protocol is significantly more efficient than [6] and [3]. Since when we estimate the efficiency of a protocol, the number of passes needed is always the dominant factor when compared with its computation overhead. That is, our scheme is the most efficient one when compared with theirs.

6.3 Increasing servers freely/ Low card-issue cost

Several schemes, [1, 5, 8, 9, 10, 15], can't add a server freely. For in them, when a server is added, all of the users who want to login to the newly added server need to re-register at RC to get a new smart card. It increases the system's card-issue cost. However, in our protocol, when a server is added, all users don't need to re-register. He can use his card to access all of this newly added server's resources. That is, our protocol can let the number of servers increase freely and thus has low card-issue cost.

6.4 Why we don't adopt dynamic ID

In [3, 14], both schemes use the concept of dynamic ID which can let the server

authenticate the user without the knowledge of user ID. But Hsiang and Shih [3] has pointed out that Liao-Wang's dynamic ID protocol [14] is vulnerable to the insider attack. In addition, also based on this dynamic ID concept, they proposed an improvement attempting to get rid of the weakness. However, in this paper, we further present an insider attack on Hsiang-Shih's protocol [3]. Note that the two insider attacks, pointed by [3] and us respectively, only eavesdrop on the transmitted messages, without any help of the smart card. This means that a dynamic-ID based protocol is fragile by using only the computation of a hash function even the function is embedded with some modular exponential operation. For example, U_u's dynamic ID, CID_u, is computed as $CID_u = H(b_u \oplus PW_u) \oplus H(T_u', A_u', N_u)$ as described in Section 2.2 of Chapter 2. U_u then can send $\{CID_u, P_{uj}, Q_u, D_u, C_0, N_u\}$ to S_j, where $Q_u = H(B_u, A_u', N_u)$ is originally designed to be checked by S_j to see if the login request is sent from the legal user U_u (see Figure 4). But indeed Q_u can only be used to decide if the login request is sent from a legal user (insider) due to the fact that the identity of the real intended party, ID_u, is not embedded in $Q_u (= H(B_u, A_u', N_u) = H(B_u, B_u \oplus H(b_u \oplus PW_u), N_u))$.

From another point of view, the insider U_n can randomly select four numbers, N_u, b, t, and v, and set $B_u = b$, $T_u = t$, $R_u = v$, and $A_u' = v \oplus H(x \oplus r)$, where $H(x \oplus r)$ can be obtained as described in 3.2.(1).(b). After these setting, he can impersonate U_u and send the login request $\{CID_u, P_{uj}, Q_u, D_u, C_0, N_u\}$ to S_j by computing $CID_u = v \oplus H(x \oplus r) \oplus b \oplus H(t, v \oplus H(x \oplus r), N_u)$, $P_{uj} = t \oplus H(v \oplus H(x \oplus r), N_u, SID_j)$, $Q_u = H(b, v \oplus H(x \oplus r), N_u) = H(b, A_u', N_u)$, $D_u = v \oplus SID_j \oplus N_u$, and $C_0 = H(v \oplus H(x \oplus r), N_u + 1, SID_j)$. After that,

S_j sends the message to RC as indicated in the mutual verification and session key agreement phase in Figure 4. In RC's turn, RC can deduce the same A_u^* as A_u' which U_n generates and S_j can deduce the same A_u'' and B_u'' as A_u' and b, which are also generated by U_n. Therefore, U_n can be authenticated successfully by both RC and S_j. Since, RC authenticates U_n by checking if his computed $C_o^* = H(A_u^*, N_u+1, SID_j)$ equals the received $C_0 (=H(A_u', N_u+1, SID_j))$ from U_n, and S_j authenticates U_n by checking if his computed $H(B_u'', N_j, A_u'', SID_j)$ equals the received $M_{ij}^* (=H(B_u, N_j, A_u, SID_j))$ (also refer to steps 3 and 5 of Section 2.2.(3) of Chapter 2). This is why the insider U_n can successfully launch an impersonation attack. Henceforth, we don't adopt the dynamic ID concept in our scheme. In the following, we show the deductions by equations (1), (2), and (3) respectively.

$$
\begin{aligned}
A_u^* &= R_u^* \oplus H(x \oplus r) \\
&= D_u \oplus SID_j \oplus N_u \oplus H(x \oplus r) \quad \text{by } R_u^* = D_u \oplus SID_j \oplus N_u \\
&= v \oplus SID_j \oplus N_u \oplus SID_j \oplus N_u \oplus H(x \oplus r) \\
&= v \oplus H(x \oplus r) \quad \text{by } D_u = v \oplus SID_j \oplus N_u \\
&= A_u' \qquad \cdots\cdots (1)
\end{aligned}
$$

$$
\begin{aligned}
A_u'' &= C_2 \oplus H(H(SID_j, y), N_{rj}) \\
&= A_u^* \oplus H(H(SID_j, y), N_{rj}) \oplus H(H(SID_j, y), N_{rj}) \\
&= A_u^* \quad \text{by } C_2 = A_u^* \oplus H(H(SID_j, y), N_{rj}) \\
&= A_u' \quad \text{by (1)} \qquad \cdots\cdots (2)
\end{aligned}
$$

$$B_u'' = A_u'' \oplus h_u$$

$= v \oplus \mathrm{H}(x \oplus r) \oplus h_u$ by (2) and $A_u' = v \oplus \mathrm{H}(x \oplus r)$

$= v \oplus \mathrm{H}(x \oplus r) \oplus CID_u \oplus \mathrm{H}(T_u'', A_u'', N_u)$

by $h_u = CID_u \oplus \mathrm{H}(T_u'', A_u'', N_u)$

$= v \oplus \mathrm{H}(x \oplus r) \oplus v \oplus \mathrm{H}(x \oplus r) \oplus b \oplus \mathrm{H}(t, v \oplus \mathrm{H}(x \oplus r), N_u) \oplus \mathrm{H}(T_u'', A_u'', N_u)$

$= b$ by $CID_u = v \oplus \mathrm{H}(x \oplus r) \oplus b \oplus \mathrm{H}(t, v \oplus \mathrm{H}(x \oplus r), N_u)$ ..... (3)

6.5 RC-off-line authentication

In protocol [6] and [3] (see Figure 2 for [6] and Figure 4 for [3]), when a user wants to log into a server, the server has to send the authentication request to RC for authenticating the user. Even in scenario (B) of [6], after RC and the server had generated the secret key, the server still needs to send the user's authentication request to RC. This means RC needs to be always on-line for authenticating a user. However, our protocol only requires the server sending the user's authentication request to RC when the user first logs into the server (as described in Section 4.(3).(A) of Chapter 4). If the user isn't the first time execution of login for authentication and session key agreement phase, the login request sent to the server can be verified by the server itself, without the intervention of RC (as described in Section 4.(3).(B) of Chapter 4). This can significantly reduce both the computational and the communicational overhead when the system is busy. This is the reason why our protocol is designed to be a RC-off-line authentication.

6.6 Comparisons

In this section, we compare the security features mentioned in Chapter 5 among our

scheme and the other proposed schemes listed in the reference, except [12] and [13] which only point out the weakness of [4] and [14] respectively without proposing a new method. After comparing with those schemes, we can see that our scheme not only can provide the secure RC-off-line authentication and resist against all kinds of attacks but also can add a server freely. We summarize the comparison result of each property in Table 1 and list the reasons for why the corresponding scheme in the table can't attain the security features in Appendix A. From Table 1, we can see that our protocol is the most favorite protocol for a multi-server environment.

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

	Ours	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[14]	[15]
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	○	○	○	○	○	○	×	○	○	○	○	×	○	○
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	○	○	○	×	×	×	○	×	×	×	×	○	×	○

Chapter 7 Conclusion

In this paper, we have analyzed the security of Tsai's and Hsiang-Shih's protocols and found several attacks existing in their schemes. In addition, we also propose a novel secure protocol for multi-server environments. After the security analysis, we conclude that our protocol is not only the most secure but also the most efficient scheme using smart card in a multi-server environment up to date. We have shown this in Table 1.

References

[1] C.C. Chang, and 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.

[2] C.C. Chang, and 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.

[3] H.C. Hsiang, and 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.

[4] I.C. Lin, M.S. Hwang, and 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.

[5] J. H. Lee, and 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.

[6] 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.

[7] L. Hu, X. Niu, and 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] R.J. Hwang, and 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.

[9] W.J. Tsaur, C.C. Wu, and 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.

[10] W.J. Tsaur, C.C. Wu, and 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.

[11] 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.

[12] X. Cao, and S. Zhong, “Breaking a remote user authentication scheme for multi-server architecture”, *IEEE Communications Letters*, Vol. 10, No. 8, pp. 580-581, August 2006.

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

[14] Y.P. Liao, and 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.

[15] Z.F. Cao, and 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.

Appendix A

(1) Why are they not “RC-off-line authentication” in [3, 6] ?

We have demonstrated this in Section 6.5 of Chapter 6.

(2) Why are they not “Increasing servers freely” in [1, 5, 8, 9, 10, 15] ?

In [1], RC stores $C_u \oplus PW_u$ in U_u’s smart card, where C_u=CRT(K_{u1}, K_{u2},…, K_{uk}), K_{uj}=H($ID_u \oplus RK_j$), RK_j =H(x, SID_j), $j=1,\ldots,k$, and k is the number of servers. If one server is added, all of the users have to get a new card for a new C_u.

In [5], RC computes $lcm_u=lcm(r_{u1},.., r_{uj},.., r_{un})$, where $1 \leqq j \leqq n$, r_{uj}=H(ID_u, H(SID_j,x)), and $lcm(\cdot)$ means the lowest common multiple of two or more numbers. He stores them in U_u’s smart card as described in the registration phase of their scheme. If U_u wants to access the newly added ($n+1$)th server, he has to re-register at RC for getting a new lcm_u. Therefore, their scheme increases the system’s card-issue cost.

In [8], TM (the trusted management server is similar to RC in our scheme) stores C_{uj} in U_u’s smart card, where C_{uj}=H($ID_u \oplus d_{sj}$)$\oplus$H(PW_u) and d_{sj} denotes the secret key of S_j. If the number of servers is w, the number of C_{uj} stored in the card is equal w. Therefore, when U_u wants to access a new added ($w+1$)th server, he has to go back to RC for updating his smart card to contain $C_{u(w+1)}$. This increases the system overhead.

In [9, 10], the CA (Central Authority is similar to RC in our protocol) uses all servers’ secret keys to construct a Lagrange interpolating polynomial $f_u(X)$ for U_u and stores $f_u(X)$ in U_u’s smart card as described in the user registration stage. Hence, if the number of servers increases, CA has to issue a new card to U_u for him to obtain a new

polynomial $f_u(X)$ to log into the newly added server.

In [15], CIC (Card Initialization Center is similar to RC) stores Ser_u in U_u's smart card. Each bit of Ser_u represents a U_u's access right parameter in each server. For example, if Ser_u is 1000100, U_u is permited to access S_3 and S_7. Therefore, when a new server is added into the system, each user's Ser_u has to be changed. Hence, all users have to get a new smart card for the new Ser_u.

(3) Why are they not "No assuming that all servers are trustworthy" in [2, 7, 15] ?

Chang *et al*. [2] assumed that RC and all servers in the system are trustworthy.

Hu *et al*.'s [7] is an improvement on Chang-Lee's scheme. In Chang-Lee's scheme, it is assumed that RC and all servers are trustworthy. Hu *et al.* didn't change this assumption in their proposed scheme.

Cao *et al*. [15] pointed out that all servers should be trusted just like CIC, because CIC shares the secret x with all servers. Hence, it means all servers in their protocol are trustworthy.

(4) Why are they not "Low computationally intensive" in [9, 10] ?

For both of the schemes [9, 10] are based on Lagrange interpolating polynomial which is computationally intensive as stated in [6].

(5) Why are they not "No verifier table in RC or any server" in [6, 11] ?

Since Tsai's scheme [6] uses the verifier table, and in [11] states that "While U_u registers at RC, RC has to send an authentication code $Ew_j(H(H(x, ID_u), SID_j), ID_u)$ to each server S_j, where $Ew_j(m)$ denotes the ciphertext of m encrypted using the secret key $w_j=H(x, SID_j)$. Then, each S_j has to store it in the verifier table for using $H(H(x, ID_u)$,

SID_j) to authenticate U_u.”.

(6) Why are they not “Mutual authentication” in [4, 9, 10, 15] ?

The proposed schemes in [4, 9, 10, 15] are not mutual authentication. Since the server doesn’t respond to the user for the user to check if the server is the intended one.

The protocols [4, 9, 10] are only one pass schemes, a message from the user to the server. It lacks the authentic message from the server to the user. Hence, the user can’t authenticate the server.

The protocol [15] is three-pass. The message of the first pass is only UID_u, the second a nonce N, and the third $\{Ser_u, C\}$. The former two messages in the first two passes can’t be used to authenticate the user or the server. The last pass is used for the server to authenticate the user. Hence, this scheme can’t let a user to authenticate the server.

(7) Why are they not “Preventing insider-server spoofing attack” in [2, 6, 7, 14, 15] ?

Chang *et al*. [2], Hu *et al*. [7], and Cao *et al*. [15] each assume that all servers are trustworthy. However, in the real word, this is usually not true. It will be vulnerable to the insider-server spoofing attack if a legal server is malevolent and masquerades as any other one.

Tsai’s scheme [6] is vulnerable to the insider-server spoofing attack as described in Section 3.1 of Chapter 3.

Liao *et al.*’s scheme [14] suffers from this attack which Hsiang-Shih had pointed out in [3]. We also pointed out it in [13].

(8) Why are they not “ Preventing off-line password guessing attack” in [4, 14] ?

In the proposed protocol [4], CM delivers the public parameters {SP_j, (r_j, s_j), K_j} and LS to the registered user U_u, where $SP_j=d_j*r_j+k_j*s_j$, d_j selected by CM is S_j's secret key, k_j selected by CM is a random number, and LS is a line passing through the point (K_j, Q_j) which is also in the line L_j:$f_j(X)=Y$ passing through two points ($X_j=ID_u^{ej}$, $Y_j=ID_u^{dj}$) and ($D_j=e_j^{IDu}$, $W_j=e_j^{PWu}$). When U_u wants to login to S_j, U_u sends the login message {ID_u, (K_j, Q_j), Z_j, A_j, T, SP_j, (r_j, s_j)} to S_j, where $Z_j= f_j(B_j)$, $A_j=g^{Ran}$, $B_j=e_j^{Ran*T}$ and T is a timestamp. Apparently, by eavesdropping on the message, attacker E can use the extended Euclid's algorithm to obtain d_j and k_j if r_j is relatively prime to s_j. Then he can compute the point ($X_j=ID_u^{ej}$, $Y_j=ID_u^{dj}$). After obtaining the two points (X_j, Y_j) and (K_j, Q_j), he can reconstruct the line L_j:$f_j(X)=Y$. Hence, E can deduce the value e_j^{PWu} by computing $e_j^{PWu} =W_j = f_j(e_j^{IDu})$ without guessing the user's correct password PW_u. This weakness is more serious than the off-line password guessing attack.

It can be easily seen that Liao *et al.*'s scheme [14] suffers from the insider off-line password guessing attack launched by either an insider server or insider user. We had described this in [13].

(9) Why are they not " Preventing parallel session attack" in [4, 9, 10, 14] ?

The protocols [4, 9, 10] each have only one pass, so they can't withstand the parallel session attack.

Liao *et al.*'s scheme [14] suffers from this attack which we had described in [13].

(10) Why are they not " Preventing smart-card-lost attack" in [3, 4, 5, 7, 8, 9, 10, 14] ?

In these protocols, assume that an attacker E obtains the smart card and reads all the secrets stored in the card. E can enforce a smart-card-lost off-line password guessing

attack on each scheme. We show them as follows.

In scheme [3], the smart card stores b, V_u=H(ID_u, x)⊕H(ID_u, H(b, PW_u)), and H_u=H(H(ID_u, x)). E can repeat with using his guessing PW' to compute H(V_u⊕H(ID_u, H(b, PW'))) and check to see if it is equal to H_u until he obtains the correct one. Therefore, a smart-card-lost off-line password guessing attack can be launched.

In scheme [4], (K_j, Q_j) is the intersection point of two line, LS and L_j. L_j is constructed by two points (X_j=ID_u^{ej}, Y_j=ID_u^{dj}) and (D_j=e_j^{IDu}, W_j=e_j^{PWu}). Therefore, an attacker E can eavesdrop U_u's two different points, (K_i, Q_i) and (K_j, Q_j), on LS from two different login messages to reconstruct LS. Hence, he can have the U_u's secret LS stored in the smart card without stealing the smart card. The weakness is more serious than the smart-card-lost attack.

In scheme [5], the smart card stores lcm_u, L_u=h(lcm_u,ID_u)⊕ H(PW_u), where h(·) is a hash function. E can repeat with using his guessing PW' to compute L_u⊕H(PW') and check to see if it is equal to h(lcm_u, ID_u) until he obtains the correct one. Therefore, their scheme suffers from the smart-card-lost off-line password guessing attack.

In scheme [7], the smart card stores the values of G=H(x, ID_u) and V=H(PW_u) ⊕G. E can repeat with using his guessing PW' to compute H(PW')⊕G and check to see if it is equal to V until he obtains the correct one. Therefore, their scheme also suffers from the smart-card-lost off-line password guessing attack.

In scheme [8], the smart card stores C_{uj}= d_{uj}⊕H(PW_u) and D_{uj}= H(v_{uj})⊕H(PW_u). E can send the login request { ID_u, C_{uj}⊕D_{uj}} to the server S_j. S_j will respond {C_1, C_2, R_1} to E, where C_1=a⊕d_{uj}, C_2=b⊕H(v_{uj}), R_1=H(SID_j, f(H(v_{uj})), a), and f(x)=a∗x+b. Then, E

can repeat with using his guessing *PW'* to compute H(SID_j, $a*x+b$)= H(SID_j, $(C_1 \oplus C_{uj} \oplus H(PW')) * (H(v_{uj})) + (C_2 \oplus D_{uj} \oplus H(PW_u))$, $C_1 \oplus C_{uj} \oplus H(PW')$) and check to see if it is equal to R_1 until he obtains the correct one. Therefore, E can successfully launch a smart-card-lost off-line password guessing attack.

In scheme [9], the smart card stores the values of $U\_S_u = r^{1/(PWu)}$ and $f_u(X)$, where $f_u(U\_ID_u) = g^r$. E can repeatly use his guessing *PW'* to compute $(U\_S_u)^{PW'}$ and check to see if it is equal to $f_u(U\_ID_u)$ until he obtains the correct one. Therefore, their scheme can't resist a smart-card-lost off-line password guessing attack.

In scheme [10], the smart card stores $U\_S_u = g^{r*d}$ and $f_u(X)$, where $f_u(U\_ID_u) = g^{PWu* r}$, $e*d=1 \bmod \phi(n)$ and e is CA's public key. E can repeat with using his guessing *PW'* to compute $(U\_S_u)^{e* PW'}$ and check to see if it is equal to $f_u(U\_ID_u)$ until he obtains the correct one. Therefore, a smart-card-lost off-line password guessing attack exists in their scheme.

In scheme [14], the smart card stores $V_u = H(H(ID_u, x)) \oplus H(ID_u, PW_u)$ and $H_u = H(H(ID_u, x))$. E can compute $T' = V_u \oplus H(ID_u, PW')$, where *PW'* is his guessing password, and check to see if his computed H(*T'*) is equal to the stored H_u without the help of any other entities. If the two values equal, the attack he launches succeeds. 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.