Comments on two multi-server authentication protocols

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

1 Institute of information systems and applications, National Tsing Hua University
*: corresponding author
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
jschou@mail.nhu.edu.tw
Tel: 886+ (0)5+272-1001 ext.56226

Abstract

Recently, Tsai and Liao et al. each proposed a multi-server authentication protocol. They claimed their protocols are secure and can withstand various attacks. But we found some security loopholes in each protocol. We will show the attacks on their schemes.

**Keywords:** multi-server, password authentication protocol, server spoofing attack, parallel session attack

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 as networks getting larger and larger. Consequently, several multi-server protocols were proposed [1-13].

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 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 and Wang [2] proposed a secure dynamic ID scheme for multi-server environment. They claim that their protocol is secure. However, we found their scheme suffers from both the server spoofing attack and the parallel session attack. In this paper, we will show the attacks on [1] and [2], respectively.

The remainder of this paper is organized as follows: In Section 2, we review both Tsai’s and Liao-Wang’s protocols. In Section 3, we demonstrate the vulnerabilities in their schemes, respectively. Finally, a conclusion is given in Section 4.

2. Review of Tsai’s and Liao-Wang’s protocols

In this section, we review Tsai’s protocol in Section 2.1 and Liao-Wang’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
- **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**: RC’s two secret keys
- **p**: a large prime number
- **g**: the primitive element in a Galois field GF(p)
- **H( )**: a collision-resistant one-way hash function
- **(a, b)**: a string denotes that string a is concatenated with string b.
- **⊕**: a bitwise Xor operator
- **△T**: a tolerant time delay for messages transmission over network
- **⇒**: a secure channel
- **→**: a common channel

2.1 Review of Tsai’s protocol

Tsai’s protocol contains four phases. They are: (1) user registration phase, (2) login phase, (3) authentication of server and RC phase, and (4) authentication of server and user phase. We describe the protocol as follows and also depict phases (1), (2) in Figure 1, phase (3) in Figure 2, and phase (4) in Figure 3.

Assume that there are s servers in the system. At the beginning, RC computes and sends H(SID_j,y) to S_j, for j = 1 to s, with S_j keeping it secret, via a secure channel.
2.1.1 Registration phase

In this phase, \( U_u \) 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.

2.1.2 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 \( N_c \). He then computes \( C_1 = (B \oplus H(PW_u)) \oplus N_c = H(ID_u, x) \oplus N_c \).

2. \( U_u \) sends \( \{ID_u, C_1\} \) to \( S_j \).

2.1.3 Authentication of server and RC phase

In this phase, when receiving message \( \{ID_u, C_1\} \) from \( U_u \), \( S_j \) will run the following steps to let himself be authenticated by \( RC \), verify \( U_u \)’s legitimacy, and negotiate the session key with \( U_u \). Let the secret key shared between \( S_j \) and \( RC \) be \( H(H(SID_j, y), Ns+1, NRC +2) \), where \( Ns \) and \( NRC \) are \( S_j \)’s and \( RC \)’s randomly 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.
Authentication of server and RC phase

(A) the secret key is not generated

1. generates a nonce $N_s$
   computes $C_2 = H(SID_j, y) \oplus N_s$.

2. $ID_u, SID_j, C_1, C_2$

3. derives $N'_s = H(SID_j, y) \oplus C_2$
   generates a nonce $N_{RC}$
   computes $C_3 = N_{RC} \oplus H(SID_j, y)$

4. $C_3$

5. retrieves
   $N'_{RC} = C_3 \oplus H(SID_j, y)$
   calculates
   $C_4 = H(H(SID_j, y), N_s) \oplus N'_{RC}$

6. $C_4$

7. computes
   $C'_4 = H(H(SID_j, y), N'_s) \oplus N_{RC}$
   checks $C'_4 =? C_4$
   retrieves $N'_c = H(ID_u, x) \oplus C_1$
   computes
   $C_5 = H(H(SID_j, y), N'_s, N_{RC})$
   $C_6 = H(H(SID_j, y), N'_s + 1, N_{RC} + 2)$
   $\oplus H(H(ID_u, x), N'_c)$

8. $C_5, C_6$

9. calculates
   $C'_5 = H(H(SID_j, y), N_s, N'_{RC})$
   compares $C'_5 =?= C_5$

(B) the secret key has been generated

1. $ID_u, SID_j, C_j$

2. derives $N'_c = H(ID_u, x) \oplus C_1$
   computes
   $C_6 = H(H(SID_j, y), N'_s + 1, N_{RC} + 2)$
   $\oplus H(H(ID_u, x), N'_c)$

3. $C_6$

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

(A) the secret key is not generated.

1. $S_j$ generates a random nonce $N_s$ and computes $C_2 = H(SID_j, y) \oplus N_s$.
2. $S_j$ sends \{ID$_u$, SID$_j$, C$_1$, C$_2$\} to RC.
3. RC derives $N'_s = H(SID_j, y) \oplus C_2$. He then generates a random nonce $N_{RC}$ and computes $C_3 = N_{RC} \oplus H(SID_j, y)$.
After receiving the message from RC, $S_j$ retrieves $N'_RC = C_5 \oplus H(SID_j, y)$ and calculates $C_4 = H(H(SID_j, y), Ns) \oplus N'_RC$.

5. $S_j$ sends $\{C_4\}$ to RC.

6. 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 $N'c = H(ID_u, x) \oplus C_I$ and computes $C_5 = H(H(SID_j, y), Ns', N_{RC})$, $C_6 = H(H(SID_j, y), Ns'+1, N'_{RC}+2) \oplus H(H(ID_u, x), N'c)$. $S_j$ sends $\{C_5, C_6\}$ to RC.

7. After receiving the message from RC, $S_j$ calculates $C'_5 = H(H(SID_j, y), Ns, N'_RC)$ and compares to see if $C'_5$ is equal to the received $C_5$. If so, RC is authentic. Both $S_j$ and RC will store the common secret key $Auth_{S, RC} = H(H(SID_j, y), Ns+1, N'_{RC}+2)$ for next execution of server and RC authentication to reduce the computational cost.

(B) the secret key has been generated.

1. $S_j$ sends $\{ID_u, SID_j, C_I\}$ to RC.

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

3. RC sends $\{C_5, C_6\}$ to $S_j$.

2.1.4 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.

1. $S_j$ generates a random nonce $N_{SU}$ and uses his $Auth_{S, RC}$ to compute $C_7 = C_6 \oplus H(H(SID_j, y), Ns+1, N'_{RC}+2) \oplus H(H(ID_u, x), N'c)$. He then calculates $C_8 = C_I \oplus C_7$, $V_2 = C_7 \oplus N_{SU}$, and $C_9 = H(C_7, N_{SU}) \oplus C_8$.

2. $U_u$ sends $\{V_2, C_9\}$ to $S_j$.

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 $C'_8 = C'_7 \oplus C_I$, $C'_9 = H(C'_7, N'_{SU}) \oplus C'_8$. He then checks to see if the newly computed $C'_9$ is equal to the received $C_9$. If so, $S_j$ is authentic. $U_u$ continues to 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 compares 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 $SK = H(C'_7+1, C'_8+2, N'_{SU}+3) = H(C_7+1, C_8+2, N_{SU}+3)$. 

5
Authentication of server and user phase

1. generates a nonce $N_{SU}$
   computes
   \[ C_7 = C_6 \oplus H(SID_u, y, Ns+1, N'_{RC}+2) \]
   \[ = H(ID_{u,x}), N'c \]
   calculates
   \[ C_8 = C_7 \oplus C_7, V_2 = C_7 \oplus N_{SU} \]
   \[ C_9 = H(C_7, N_{SU}) \oplus C_8 \]

2. \[ V_2, C_0 \]

3. computes
   \[ C'_7 = H(ID_{u,x}, Nc) \]
   retrieves
   \[ N'_{SU} = C'_7 \oplus V_2 \]
   calculates
   \[ C'_8 = C'_7 \oplus C_7 \]
   \[ C'_9 = H(C'_7, N'_{SU}) \oplus C'_8 \]
   checks \[ C'_9 =? C_9 \]
   calculates
   \[ C'_{10} = H(C'_7, C'_8, N'_{SU}) \]

4. \[ C'_{10} \]

5. computes \[ C'_{10} = H(C_7, C_8, N_{SU}) \]
   compares \[ C'_{10} =? C_{10} \]
   session key \[ SK = H(C'_7+1, C'_8+2, N'_{SU}+3) \]

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

2.2 Review of Liao-Wang’s protocol

In this section, we review Liao-Wang’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, $y$ is a secret number shared among RC and all servers. We describe their protocol as follows and also depict it in Figure 4.

2.2.1 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 his $ID_u, PW_u$ and sends \{ID_u, PW_u\} to RC through a secure channel,
2. RC computes \[ B=H(ID_{u,x}), B_1=B \oplus H(ID_{u,x}), B_2=H(PW_u) \oplus H(x), \] and \[ B_3=H(B) \]. He then issues $U_u$ a smart card containing $B_1, B_2, B_3$, and $y$ through a secure channel.
**Registration phase**

1. chooses $ID_u$, $PW_u$

2. computes
   
   $B = H(ID_u, x)$,
   
   $B_1 = B \oplus H(ID_u, PW_u)$
   
   $B_2 = H(PW_u) \oplus H(x)$,
   
   $B_3 = H(B)$

   smart card contains $B_1$, $B_2$, $B_3$, $y$

**Login phase**

1. keys $ID_u$, $PW_u$ and $SID_j$

   computes
   
   $B' = B_1 \oplus H(ID_u, PW_u)$,
   
   $B'_2 = H(B')$

   If $B'_2 = B'_3$, generates a nonce $Nc$.

   calculates
   
   $CID_u = H(PW_u) \oplus H(B'_2, y, Nc)$
   
   $C_1 = B'_2 \oplus H(y, Nc, S)I_D_j)$
   
   $C_2 = H(B'_2, y, Nc)$

2. $CID_u$, $C_1$, $C_2$, $Nc$

**Mutual verification and session key agreement phase**

1. computes $B' = C_1 \oplus H(y, Nc, S)I_D_j)$,

   $H_{PW} = CID_u \oplus H(B', y, Nc)$,

   $B'_2 = H_{PW} \oplus H(x)$, $H(B'_2, y, Nc)$

   checks $H(B'_2, y, Nc) = C_2$, if so, generates a nonce $Ns$.

   calculates $C_3 = H(B'_2, Nc, y, S)I_D_j)$

2. $C_3$, $Ns$

3. computes $H(B'_2, Nc, y, S)I_D_j)$

   compares
   
   $H(B'_2, Nc, y, S)I_D_j) = C_2$, if so,

   calculates
   
   $C_4 = H(B'_2, Nc, y, S)I_D_j)$

4. $C_4$

5. computes $H(B'_2, Nc, y, S)I_D_j)$

   checks $H(B'_2, Nc, y, S)I_D_j) = C_2$

6. session key

   $SK = H(B'_2, Nc, Ns, y, S)I_D_j)$

   $SK = H(B'_2, Nc, Ns, y, S)I_D_j)$

---

Fig. 4. Liao-Wang’s protocol
2.2.2 Login phase

1. U_u keys his ID_u, PW_u and SID_j to the smart card. The smart card computes B'=B_j \oplus H(ID_u, \ PW_u), B_j'=H(B'), and compares to see if B_j stored is equal to the computed B_j'. If so, smart card knows U_u is the real card holder. It then generates a random nonce Nc and calculates CID_u=H(PW_u)\oplus H(B', y, Nc), C_j=B_j'\oplus H(y, Nc, SID_j), and C_2=H(B_2, y, Nc).

2. U_u sends \{CID_u, C_1, C_2, Nc\} to S_j.

2.2.3 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 computes B^* = C_j' \oplus H(y, Nc, SID_j), H_{PW} = CID_u \oplus H(B^*, y, Nc), and B_j' = H_{PW} \oplus H(x). He then computes H(B_j', y, Nc) and checks to see if it is equal to the received C_2. If so, S_j then generates a random nonce Ns and calculates C_j = H(B_j', Nc, y, SID_j).

2. S_j sends \{C_3, Ns\} to U_u.

3. U_u computes H(B_2, Nc, y, SID_j) and compares to see if it is equal to the received C_3. If it is, S_j is authentic. U_u then calculates C_d = H(B_2, Ns, y, SID_j).

4. U_u sends \{C_d\} to S_j.

5. After receiving the message from U_u, S_j computes H(B_j', Ns, y, SID_j) and checks to see if it is equal to the received C_d. If so, U_u is authentic.

6. After finishing mutual authentication, U_u and S_j can compute the common session key SK = H(B_2, Nc, Ns, y, SID_j) = H(B_j', Nc, Ns, y, SID_j), respectively.

2.2.4 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 B'=B_j \oplus H(ID_u, \ PW_u), B_j'=H(B') and compares to see if B_j in the smart card is equal to the computed B_j'. If so, U_u is the real card holder.

3. The smart card allows U_u to submit a new password PW_u^{new}.

4. The smart card computes B_1^{new} = B_1' \oplus H(ID_u, \ PW_u^{new}), B_2^{new} = B_2 \oplus H(PW_u) \oplus H(PW_u^{new}) and replaces B_1, B_2 with B_1^{new}, B_2^{new}, respectively.

3. Security loopholes in Tsai’s and Liao-Wang’s protocols

After analysis, we found Tsai’s protocol suffers server spoofing attacks in both scenarios and Liao-Wang’s protocol suffers server spoofing attack and parallel session
attack. In this section, we will show the security loopholes 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 on Tsai’s protocol. It 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 mentioned scenarios, (A) and (B), and also illustrate them in Figure 5 and 6, respectively.

(A) the secret key is not generated.

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 N_s, computes C_2 = H(SID_i, y)⊕N_s, and sends {ID_u, SID_i, C_1, C_2} to RC. For the subsequent messages C_3, C_4, C_5 and C_6, except C_6, sent between RC and S_i to authenticate each other are independent on U_u’s secrecy H(H(ID_u, x), N_e) as depicted in scenario (A) of Figure 2. RC and S_i will thus achieve mutual authentication successfully.
3. RC and S_i then negotiate to establish the common secret key Auth_{S_RC}=H(H(SID_i, y), N_s+1, N_RC'+2)=H(H(SID_i, y), N_s'+1, N_RC +2) in the phase of server and RC authentication. After that, S_i and U_u will perform the following steps for the authentication of server and user phase.
4. S_i generates a random nonce N_SU and uses his Auth_{S_RC} to compute C_7 = C_e ⊕ Auth_{S_RC} =H(H(ID_u, x), N'e). He then calculates C_8 = C_1 ⊕ C_7, V_2 = C_7 ⊕ N_SU, and C_9 = H(C_7, N_SU) ⊕ 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), N'e), retrieves N'_SU = C'_7 ⊕ V_2, and calculates C'_8 = C'_7 ⊕ C_1, C'_9 = H(C'_7, N'_SU) ⊕ 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 from the sender who had received his C_7 in the login phase. S_i disguising as S_j is thus regarded as authentic. U_u continues to calculate 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 compares to see if C'_{10} is equal to his 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) = H(C_7+1, C_8+2, N_SU+3).

From the above-mentioned steps, we can see that a server spoofing attack can be successfully launched by insider attacker S_i.
Fig. 5. Server spoofing attack by an insider server on Tsai’s protocol: (A) the secret key is not generated.

(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 $C_6=H(H(SID_i, y), Ns^i+1, N_{RC}+2)\oplus H(H(ID_u, x), N’c)$.
3. RC sends $\{C_6\}$ to $S_i$ as depicted in scenario (B) of Figure 2. $S_i$ then proceeds the following steps with $U_u$ for the authentication of server and user phase.
4. $S_i$ generates a random nonce $N_{SU}$ and uses the generated common secret key $Auth_{S,RC}$ to compute $C_7=C_6\oplus Auth_{S,RC}=H(H(ID_u, x), N’c)$. 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
Fig. 6. Server spoofing attack by an insider server on Tsai’s protocol: (B) the secret key has been generated.

<table>
<thead>
<tr>
<th>U_u</th>
<th>S_i (S_j)</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $ID_u, C_i$</td>
<td>2. $ID_u, SID_{10}, C_i$</td>
<td>3. $C_6$</td>
</tr>
<tr>
<td>4. generates a nonce $N_{SU}$ computes $C_7 = C_6 \oplus \text{Auth}<em>{S, R_C}$ $= H(H(ID_u, x), N_c)$ calculates $C_9 = C_I \oplus C_7$ $V_2 = C_7 \oplus N</em>{SU}$ $C_9 = H(C_7, N_{SU}) \oplus C_8$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. $V_2, C_9$</td>
<td>7. $C_{10}$</td>
<td></td>
</tr>
<tr>
<td>6. computes $C' = H(H(ID_u, x), N_c)$ retrieves $N'_{SU} = C'_7 \oplus V_2$ calculates $C'_8 = C'_7 \oplus C_I$ $C'_9 = H(C'<em>7, N'</em>{SU}) \oplus C'_8$ checks $C'<em>9 =? C_9$ calculates $C</em>{10} = H(C'_7, C'<em>8, N'</em>{SU})$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. session key $SK = H(C'_8 + 1, C'<em>8 + 2, N'</em>{SU} + 3)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

3.2 Attack on Liao-Wang’s protocol

In Liao-Wang’s protocol, it can easily be seen that an insider peer (either a server or a user) can launch an off-line password guessing attack by eavesdropping on the transmitted message \{C$ID_u$, $C_i$, $C_2$, Nc \} and comparing $C_2$ with his computation $H(H(PW') \oplus H(x), y, N_c)$, where $y$ is the value stored in his smart card and shared
with RC, $PW'$ is his guessing password, and $H(x)$ is shared by all legal servers in their protocol and also can be derived by all legal users by computing $H(x) = B_2 \oplus H(PW)$, where $B_2$ is the value stored in the smart card and $PW$ is the user’s password.

In addition, it also can be seen that anyone who has got $U_u$’s smart card can launch a password guessing attack by comparing $B_3$ with his computation $B_1 \oplus H(ID_u, PW')$, where $B_3$, $B_1$ are the values stored in $U_u$’s smart card and $PW'$ is his guessing password.

Besides, in this section, we will show two server spoofing attacks on Liao-Wang’s protocol in section 3.2.1 and section 3.2.2, respectively. Then, we also show a parallel session attack on their scheme in section 3.2.3.

### 3.2.1 Server spoofing attack by an insider server

Assume that $S_i$ is a legal server who has registered at RC. He also has his secrets $H(x)$, $y$ to authenticate users. We will show that $S_i$ can masquerade as any server (Here, we assume $S_j$) to cheat a remote user. It is because each server has the same secret data, $H(x)$ and $y$, for faking messages to cheat users. We describe the server spoofing attack below and also depict it in Figure 7.

1. $U_u$ starts the protocol and sends $\{CID_u, C_1, C_2, Nc\}$ to $S_i$, where $C_1 = B' \oplus H(y, Nc, SID_j)$, as in the login phase of Figure 4.

2. After receiving the message $\{CID_u, C_1, C_2, Nc\}$ from $U_u$, $S_i$ runs the mutual verification and session key agreement phase with $U_u$. He uses his secret data, $H(x)$ and $y$, and the public parameter $SID_j$ to compute $B^* = C_1 \oplus H(y, Nc, SID_j)$, $H_{pw} = CID_u \oplus H(B'^*, y, Nc)$, and $B'^* = H_{pw} \oplus H(x)$. He then generates a random nonce $Ns$ and calculates $C_3 = H(B'^*, Nc, y, SID_j)$.

3. $S_i$ sends $\{C_3, Ns\}$ to $U_u$.

4. $U_u$ computes $H(B_2, Nc, y, SID_j)$ and compares to see if it is equal to the received $C_3$. If so, $U_u$ confirms that $S_i$ is authentic. $U_u$ then calculates $C_4 = H(B_2, Ns, y, SID_j)$.

5. $U_u$ sends $\{C_4\}$ to $S_i$.

6. After obtaining the message, $S_i$ computes $H(B'^*, Ns, y, SID_j)$ and checks to see if it is equal to the received $C_4$. If so, $U_u$ is authentic.

7. After finishing the mutual authentication, $U_u$ and $S_i$ can compute the common session key $SK = H(B_2, Nc, Ns, y, SID_j) = H(B'^*, Nc, Ns, y, SID_j)$.

From the above-mentioned steps, we can see that the server spoofing attack has been successfully launched by $S_i$ who masquerades as $S_j$.

### 3.2.2 Server spoofing attack by an insider user

Assume that $U_n$ is a legal user who has registered at RC. He also has a smart card
to access servers’ resources. We will show that he can use his $B_2'$ and $y$ both stored in the smart card to masquerade as any server to cheat a remote user. It is because $U_u$ can first uses $B_2'$ and his password $PW_u$ to compute $B_2' \oplus H(PW_u)$, obtaining $H(x)$. Then he uses $H(x)$ and $y$ to fake desired messages to cheat the remote user. We describe this attack by using the following steps and also depict it in Figure 8.

1. $U_u$ starts the protocol and sends $\{CID_u, C_1, C_2, Nc\}$ to $U_n$ who impersonates $S_i$.
2. $U_n$ uses his $PW_n$ and $B_2'$ in his smart card to derive the value of $H(x)$ by computing $B_2' \oplus H(PW_n)$. He then uses $\{CID_u, C_1, C_2, Nc\}$, $H(x)$, $y$, and the public parameter $SID_i$ to compute $B^* = C_1 \oplus H(y, Nc, SID_i)$, $H_{PW_u} = CID_u \oplus H(B^*, y, Nc)$ and $B'_{2} = H_{PW_u} \oplus H(x)$. In addition, he also generates a random nonce $Ns$ and calculates $C_3 = H(B'_{2}, Nc, y, SID_i)$.
3. $U_n$ sends $\{C_3, Ns\}$ to $U_u$.
4. After receiving the message, $U_u$ uses his stored $B_2$ to compute $H(B_2, Nc, y, SID_i)$ and compares to see if it is equal to the received $C_3$. If it is, $U_u$ authenticates $U_n$ as $S_i$ unconsciously. He then calculates $C_4 = H(B_2, Ns, y, SID_i)$.
5. $U_u$ sends $\{C_4\}$ to $U_u$.
6. After obtaining the message, $U_u$ computes $H(B'_{2}, Ns, y, SID_i)$ and checks to see if it is equal to the received $C_4$. If so, $U_u$ is authentic.
7. After finishing the mutual authentication, $U_u$ and $U_n$ can compute the common session key $SK = H(B_2, Nc, Ns, y, SID_i) = H(B'_{2}, Nc, Ns, y, SID_i)$.

From the above-mentioned steps, we can see that the insider spoofing attack, launched by $U_u$ masquerading as $S_i$, has been successfully accomplished.

---

**Fig. 7. Server spoofing attack by an insider server on Liao-Wang’s protocol**
3.2.3 Parallel session attack by an insider user

Assume that $U_n$ is a legal user. He also has his smart card containing $B_2'$. We will show that he can masquerade as any other user to cheat a remote server on Liao-Wang’s protocol. It is because the remote server doesn’t examine if the message is indeed sent from the correct user. $U_n$ can thus use his $B_2'$ and $y$ to masquerade as any valid user. We demonstrate this attack by using the following steps and also depict it in Figure 9.

1. $U_u$ starts the protocol and sends $\{CID_u, C_1, C_2, Nc\}$ to $U_n$ who masquerades as $S_j$.
   After receiving the message, $U_n$ now masquerades as $U_u$ to start another protocol with real $S_j$ by sending him $\{CID_u, C_1, C_2, Nc\}$.
2. $S_j$ runs the mutual verification and session key agreement phase with $U_n$ and computes $B^*=C_1 \oplus H(y, Nc, SID_j)$, $B'_2=H_{pw} \oplus H(x)$. He then computes $H(B'_2, y, Nc)$ and checks to see if it is equal to the received $C_2$. If so, $S_j$ generates a random nonce $Ns$ and calculates $C_3=H(B'_2, Nc, y, SID_j)$.
3. $S_j$ sends $\{C_3, Ns\}$ to $U_n$.
4. $U_n$ computes $B_2'=H(PW_u) \oplus H(x)$, $B''=C_1 \oplus H(y, Nc, SID_j)$, $H''_{pw}=CID_u \oplus H(B'', y, Nc)$, $B'_2=H''_{pw} \oplus B_2'$, and $C_4=H(B'_2, Ns, y, SID_j)$.
5. $U_n$ sends $\{C_4\}$ to $S_j$.
6. After receiving the message, $S_j$ computes $H(B'_2, Ns, y, SID_j)$ and checks to see if it is equal to the received $C_4$. If so, $S_j$ confirms that $U_n$ is authentic and therefore regards $U_n$ as $U_u$ unconsciously.
1. \( C_{ID_u}, C_f, \)  
\( C_2, N_c \)

2. computes  
\( B'^* = C_f \oplus H(y, N_c, S-ID_j) \)  
\( B'' = H_{PW} \oplus H(x) \)  
computes \( H(B'_2, y, N_c) \)  
checks \( H(B'_2, y, N_c) = C_2 \)  
generates a nonce \( N_s \)  
calculates  
\( C_f = H(B'_2, N_c, y, S-ID_j) \)

3. \( C_s, N_s \)

4. computes  
\( B_2'' = H(PW_u) \oplus H(x) \)  
\( B'' = C_f \oplus H(y, N_c, S-ID_j) \)  
\( H_{PW} = C_{ID_u} \oplus H(B'_2, y, N_c) \)  
\( B_2'' = H_{PW} \oplus B'_2 \)  
\( C_4 = H(B_2'', N_s, y, S-ID_j) \)

5. \( C_4 \)

6. computes  
\( H(B_2'', N_s, y, S-ID_j) \)  
checks  
\( H(B_2'', N_s, y, S-ID_j) = C_4 \)

7. session key  
\( SK = H(B_2', N_c, N_s, y, S-ID_j) \)

Fig. 9. Parallel session attack by an insider user on Liao-Wang’s protocol

7. After finishing mutual authentication, \( U_n \) and \( S_j \) have the common session key  
\( SK = H(B_2', N_c, N_s, y, S-ID_j) = H(B'_2, N_c, N_s, y, S-ID_j) \).

From the above-mentioned steps, we can see that the insider user \( U_n \) has successfully launched a parallel session attack.

4. Conclusion

We have analyzed the security of Tsai’s and Liao-Wang’s protocols. Although, they claim their protocols can resist against various attacks, we have showed that their protocols are indeed insecure against some attack that we have described in this article.

References


