Improvements on two password-based authentication protocols

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

[1] Institute of information systems and applications, National Tsing Hua University
d949702@oz.nthu.edu.tw
[2] Department of Information Management, Nanhua University, Taiwan, R.O.C
*: corresponding author
jschou@mail.nhu.edu.tw
Tel: 886+ (0)5+272-1001 ext.56536
[3] Department of Information Management, Nanhua University, Taiwan, R.O.C
g6451519@mail1.nhu.edu.tw

Abstract

Recently, Liao *et al.* and Hölbl *et al.* each proposed a user authentication protocol, respectively. Both claimed that their schemes can withstand various attacks. However, Xiang *et al.* pointed out Liao *et al.*'s protocol suffers from three kinds of attacks, the replay attack, the guessing attack, and the Denial-of-service (DoS) attack. Moreover, we and Munilla *et al.* also found Hölbl *et al.*'s protocol suffers from the password guessing attack. In this paper, we will propose the two protocols' improvements respectively. After analyses and comparisons, we conclude that our improvements are not only more secure but also more efficient in communication cost than all of the password based schemes that we know.

Keywords: *smart card, password authentication protocol, password change, man-in-the-middle attack, denial-of-service attack, smart-card-lost attack, off-line password guessing attack, mutual authentication*

1. Introduction

Password-based authentication protocols are widely adopted for logging into the remote servers. To a minimum, they can provide authentication between the client and the server to ensure the legality of the user over an open network. Many schemes in this area have been proposed such as, two-party password authenticated key exchange (PAKE) protocols for the client-server architecture [1-19, 30-36], three-party PAKE (3PAKE) protocols for the client-client-server architecture [20-26], and multi-server PAKE protocols for the client-servers architecture [27-29]. In a two-party PAKE (2PAKE) protocol, a client can access a server's resources; in a 3PAKE protocol, any two clients can communicate with each other after having been authenticated by the server; and in a multi-server PAKE protocol, a client can access many servers' resources. For that 2PAKE protocol is a fundamental concept in authentication, we therefore focus only on this type of protocols in this paper.

In a traditional two-party password-based authentication scheme without using smart card, the server has to store the verifier table for authenticating the client. It may therefore suffer from the stolen verifier table attack. Hence, a better solution is the use of a smart card. In 2006, Liao *et al.*[9] and Peyravian *et al.*[4] each proposed a password authentication scheme using smart card, respectively. Both claimed that their protocols are secure. However in 2008, Xiang *et al.*[14] found that [9] has the security loopholes: the replay attack, the guessing attack, and the denial-of-service (DoS) attack and Hölbl *et al.*[12] found that [4] suffers from the password guessing attack. In addition, Hölbl *et al.* further proposed an improvement on [4]. But we [17] and Munilla *et al.*[18] each found a password guessing attack on their improvement in 2008 and 2009, respectively. Also in 2008, Liu *et al.*[10] proposed a new mutual authentication scheme based on nonces and smart cards, Bindu *et al.*[7] proposed an improved remote user authentication scheme on Chien *et al.*'s protocol [1], and Juang *et al.*[8] proposed an efficient password authenticated key agreement scheme, respectively. All of them claimed that their schemes are secure. Nevertheless, in 2009, Sun *et al.*[16] demonstrated that [10] suffers the man-in-the-middle attack. In addition, we will demonstrate that [7] suffers from the smart-card-lost attack (see Appendix A.1), and [8] suffers from the password guessing attack if the smart card is lost (see Appendix A.2). In the same year, Rhee *et al.*[11] proposed a remote user authentication scheme without using smart card. But Tsai *et al.*[19] found that their protocol suffers the insider attack. Additionally, also in 2009, Goriparthi *et al.*[6], Kim *et al.*[30], Wang *et al.*[31], Hsiang *et al.*[32], Chung *et al.*[33], Xu *et al.*[34], and Hwang *et al.*[35] each proposed a 2PAKE protocol, respectively. However, after analysis, we found that both the protocols [6, 31], proposed by Goriparthi *et al.* and Wang *et al.* respectively, are vulnerable to the DoS attack on the password change phase (which makes the password invalid after the protocol run, see Appendix A.3), [30, 32] suffer from the smart-card-lost attack (see Appendix A.4), [34] is vulnerable to the insider impersonation attack (see Appendix A.5), and the others [33, 35] are not efficient enough for that they need four and three passes to establish the session keys, respectively. In the same year, Hölbl *et al.*[37] proposed two improved two-party key agreement protocols. But, we found that both of their protocols fail since the KGC's secret x_s can be found (see Appendix A.6). Also in 2009, Yang *et al.*[38] proposed a remote mutual authentication agreement protocol. However, their method isn't a password based scheme. Moreover, it is problematic in the registration phase, since they use the group, G_p, in the manner as if it were multiplicative, which contradicts their definition (in the system initializing phase) that G_p is a cyclic addition group. In 2010, Li *et al.*[36] propose a 2PAKE protocol, but we found it suffers from the smart-card-lost attack (see Appendix A.7)

From the above-mentioned, we know that there still lacks a secure protocol in this area. Accordingly in this paper, we will propose two improvements on schemes [9] and [12] respectively to make them safer and more efficient. We consider that the reason why a smart card based protocol suffers from the password guessing attack when it is lost is that an attacker can read out the parameters and guess the possible passwords to check whether his guessing is correct, and that for preventing DoS attack, a client should be able to change his password only after he has been authenticated by the server (in the password change phase). Based on these two considerations, we design our two improvements, hoping that they can get rid of the DoS attack and the password guessing attack when the smart card is lost, and that both can reduce the communicational passes and satisfy the requirement that the clients can choose and change their passwords at will. We will examine both of our improvements by checking whether they can resist against the replay attack, password guessing attack, insider attack, man-in-the-middle attack, DoS attack, and smart-card-lost attack, and examine whether the smart-card based improvement on Liao *et al.*'s protocol can achieve the ten requirements proposed by Liao *et al.* (for a password-based authentication protocol using smart card).

The remainder of this paper is organized as follows: In Section 2, we review both the protocols of Liao *et al.* and Hölbl *et al.*, respectively. In Section 3, we present our two improvements and then analyze the security in Section 4. After that, some discussions are made in Section 5. Finally, a conclusion is given in Section 6.

2. Review of Liao *et al.*'s and Hölbl *et al.*'s protocols

In this section, we review Liao *et al.*'s protocol [9] in Section 2.1 and Hölbl *et al.*'s protocol [12] in Section 2.2, respectively. Before that, the notations used are first described below.

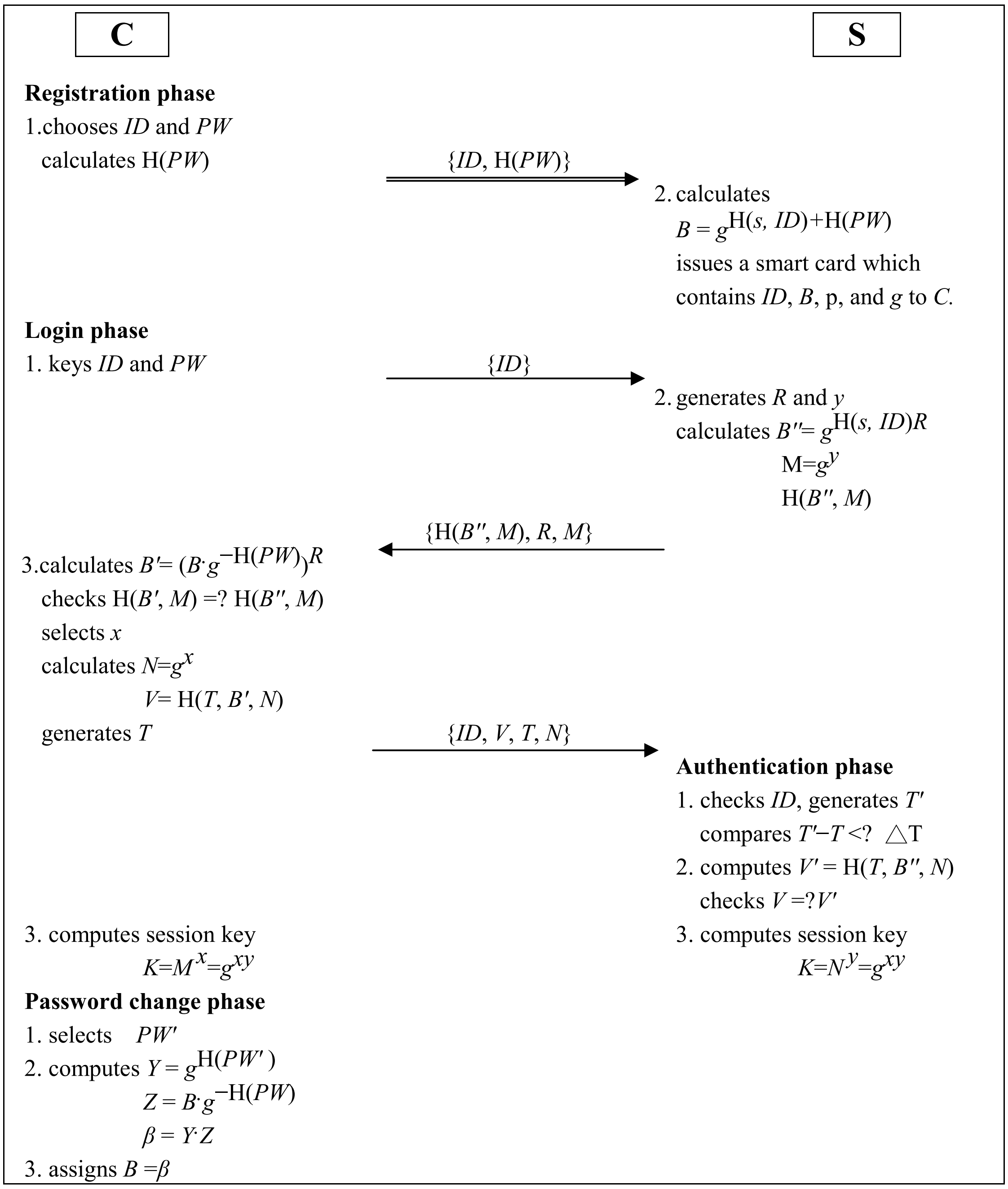
C, S	: a client and a server, respectively.
E	: an adversary/attacker.
ID	: the identity of C.
PW	: the password of C.
p	: a large prime number.
g	: a primitive element in a Galois field GF(p), where GF(p) is a set of integers $\{0,1,\cdots,p-1\}$ with arithmetic operations defined on modulo p.
H	: a collision-resistant one-way hash function.
(a,b)	: a string denotes that string a is concatenated with string b.
$\oplus$	: an exclusive-or operation.
$\triangle T$	: the tolerant time for transmission delay.
s	: S's secret key.

=> : a secure channel.

→ : a common channel.

2.1 Review of Liao *et al.*'s protocol

In this section, we first briefly review Liao *et al.*'s scheme in Section 2.1.1 through 2.1.4 then list their ten proposed requirements for a password-based authentication protocol using smart card in Section 2.1.5 (for more details see [9]). Their scheme consists of four phases, registration phase, login phase, authentication phase, and password change phase. We describe them as follows and also illustrate it in Fig. 1.



C | **S**

Registration phase

C: 1.chooses ID and PW
calculates $H(PW)$

C ⇒ S: $\{ID, H(PW)\}$

S: 2. calculates
$B = g^{H(s, ID)+H(PW)}$
issues a smart card which contains ID, B, p, and g to C.

Login phase

C: 1. keys ID and PW

C → S: $\{ID\}$

S: 2. generates R and y
calculates $B'' = g^{H(s, ID)R}$
$M = g^{y}$
$H(B'', M)$

S → C: $\{H(B'', M), R, M\}$

C: 3.calculates $B' = (B \cdot g^{-H(PW)})^{R}$
checks $H(B', M) =?\ H(B'', M)$
selects x
calculates $N = g^{x}$
$V = H(T, B', N)$
generates T

C → S: $\{ID, V, T, N\}$

Authentication phase

S: 1. checks ID, generates T'
compares $T' - T <?\ \triangle T$
2. computes $V' = H(T, B'', N)$
checks $V =?\ V'$
3. computes session key
$K = N^{y} = g^{xy}$

C: 3. computes session key
$K = M^{x} = g^{xy}$

Password change phase

1. selects PW'
2. computes $Y = g^{H(PW')}$
$Z = B \cdot g^{-H(PW)}$
$\beta = Y \cdot Z$
3. assigns $B = \beta$

Fig. 1. Liao *et al.*'s protocol

2.1.1 Registration phase

In this phase, C performs the following two steps to register at S for obtaining a smart card.

1. C freely chooses his *ID* and *PW*, and calculates H(*PW*). C then sends {*ID*, H(*PW*)} to S through a secure channel.
2. S calculates $B = g^{\mathrm{H}(s,\ ID)+\mathrm{H}(PW)}$ mod p and then issues C a smart card, which contains *ID*, *B*, p, and *g*, through a secure channel.

2.1.2 Login phase

When C wants to login to S, he inserts his smart card and cooperates with S to perform the following steps.

1. C keys his *ID* and *PW* to the smart card and sends {*ID*} to S.
2. S selects two random numbers *R* and *y*, and calculates $B''= g^{\mathrm{H}(s,\ ID)R}$ mod p and $\mathrm{M}=g^{y}$. He then computes H(*B''*, *M*) and sends {H(*B''*, *M*), *R*, *M*} to C.
3. After receiving the message from S, C calculates $B'= (B\cdot g^{-\mathrm{H}(PW)})^{R}$ mod p and checks to see if H(*B'*, *M*) is equal to the received H(*B''*, *M*). If so, S is authentic. C then selects a random number *x*, calculates $N=g^{x}$ mod p, and computes *V*= H(*T*, *B'*, *N*), where *T* is the timestamp of the system. He then sends {*ID*, *V*, *T*, *N*} to S.

2.1.3 Authentication phase

In this phase, S executes the following steps to determine whether C is allowed to login or not.

1. S checks *ID*, generates the timestamp *T'*, and compares if $T'-T$ is less than △T. If ID is invalid or $T'-T >$△T, the login request is rejected.
2. S computes *V'* = H(*T*, *B''*, *N*), and then checks if *V* is equal to *V'*. If it is, C is authentic. Otherwise, S stops the protocol.
3. After authenticating C, S computes the session key as $K=N^{y}=g^{xy}$. And C also computes the session key as $K=M^{x}=g^{xy}$.

2.1.4 Password change phase

When C wants to change his password from *PW* to *PW'*, he performs the following steps.

1. selects a new password PW'.
2. computes $Y = g^{H(PW')} \bmod p$, $Z = B \cdot g^{-H(PW)} \bmod p$, and $\beta = Y \cdot Z$, where PW is the old password and B is the variable stored in the smart card.
3. assigns $B = \beta$ in the smart card.

2.1.5 The ten requirements for a password-based authentication protocol using smart card

In 2006, Liao *et al.*[9] proposed ten requirements for evaluating the goodness of a password-based authentication protocol using smart card. We list them as follows.

R1. It needs no password or verifier table.
R2. The clients can choose and change their passwords freely.
R3. The clients need not to reveal their passwords to the server.
R4. The passwords are not transmitted in plaintext over the Internet.
R5. It can resist the insider (a legal user) attack.
R6. It can resist replay attack, password guessing attack, modification-verifier-table attack, and stolen-verifier attack.
R7. The length of a password is appropriate for memorization.
R8. It is efficient and practical.
R9. It can achieve mutual authentication.
R10. It can resist password guessing attack even if the smart card is lost.

2.2 Review of Hölbl *et al.*'s protocol

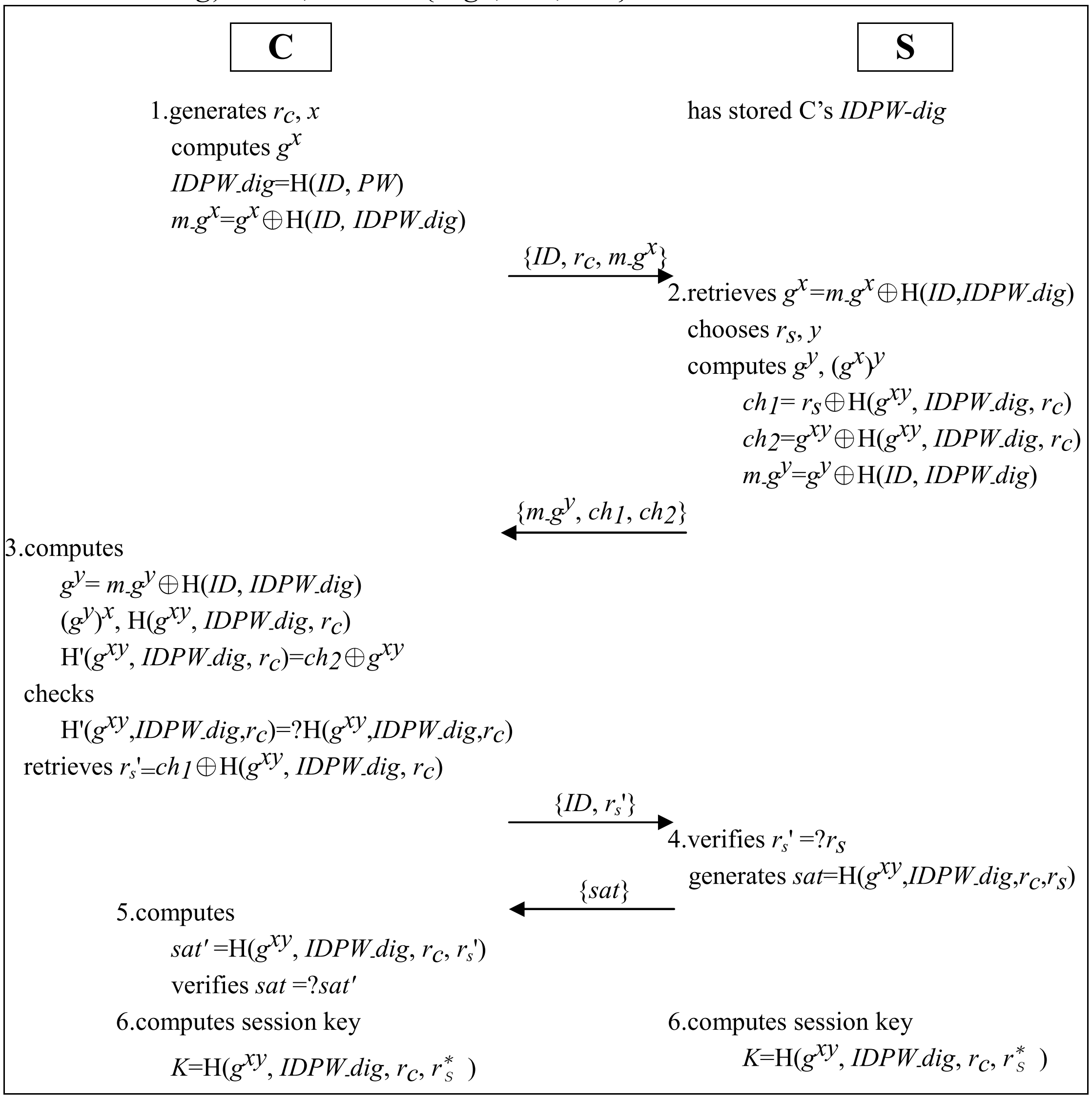
In this section, we first review Hölbl *et al.*'s user authentication protocol in Section 2.2.1 and then review the password change protocol in Section 2.2.2.

2.2.1 User authentication protocol

In this protocol, user C has to register at server S to become a legal client and S stores C's $IDPW\_dig$(=H(ID, PW)), instead of PW. They both perform the following steps. We also depict them in Fig. 2.

1. C chooses two random values r_C, x and computes g^x. Then, C masks g^x as m_g^x by computing $m\_g^x = g^x \oplus H(ID, IDPW\_dig)$, and sends message $\{ID, r_C, m\_g^x\}$ to S.
2. After receiving the message, S retrieves g^x by computing $g^x = m\_g^x \oplus H(ID, IDPW\_dig)$. Then, S chooses two random values r_S, y and computes g^y. He calculates $(g^x)^y$, generates $ch1 = r_S \oplus H(g^{xy}, IDPW\_dig, r_C)$ and $ch2 = g^{xy} \oplus H(g^{xy}, IDPW\_dig, r_C)$, and masks g^y as m_g^y by computing $m\_g^y = g^y \oplus H(ID,$

IDPW_dig). Then, S sends $\{m\_g^y, ch_1, ch_2\}$ to C.



C
S
1.generates r_C, x
computes g^x
$IDPW\_dig$=H(ID, PW)
m_g^x=$g^x \oplus$H(ID, $IDPW\_dig$)
has stored C's $IDPW$-dig
$\{ID, r_C, m\_g^x\}$
2.retrieves g^x=$m_g^x \oplus$H(ID,$IDPW\_dig$)
chooses r_S, y
computes g^y, $(g^x)^y$
ch_1= $r_S \oplus$H(g^{xy}, $IDPW\_dig$, r_C)
ch_2=$g^{xy} \oplus$H(g^{xy}, $IDPW\_dig$, r_C)
m_g^y=$g^y \oplus$H(ID, $IDPW\_dig$)
$\{m\_g^y, ch_1, ch_2\}$
3.computes
g^y= $m_g^y \oplus$H(ID, $IDPW\_dig$)
$(g^y)^x$, H(g^{xy}, $IDPW\_dig$, r_C)
H'(g^{xy}, $IDPW\_dig$, r_C)=$ch_2 \oplus g^{xy}$
checks
H'(g^{xy},$IDPW\_dig$,r_C)=?H(g^{xy},$IDPW\_dig$,r_C)
retrieves r_s'=$ch_1 \oplus$H(g^{xy}, $IDPW\_dig$, r_C)
$\{ID, r_s'\}$
4.verifies r_s' =?r_S
generates sat=H(g^{xy},$IDPW\_dig$,r_C,r_S)
$\{sat\}$
5.computes
sat' =H(g^{xy}, $IDPW\_dig$, r_C, r_s')
verifies sat =?sat'
6.computes session key
K=H(g^{xy}, $IDPW\_dig$, r_C, r_S^*)
6.computes session key
K=H(g^{xy}, $IDPW\_dig$, r_C, r_S^*)

Fig. 2. Hölbl *et al.*'s user authentication protocol

3. After receiving the message, C derives g^y= $m_g^y \oplus$H(ID, $IDPW\_dig$). Then, he computes $(g^y)^x$ and H(g^{xy}, $IDPW\_dig$, r_C), and derives H'(g^{xy}, $IDPW\_dig$, r_C) by computing $ch_2 \oplus g^{xy}$. C checks to see if the derived H' (g^{xy}, $IDPW\_dig$, r_C) is equal to the computed H(g^{xy}, $IDPW\_dig$, r_C). If it is, C then retrieves r_s' by computing $ch_1 \oplus$H(g^{xy}, $IDPW\_dig$, r_C) and sends $\{ID, r_s'\}$ to S. Otherwise, S is not genuine and C terminates the protocol.
4. After receiving $\{ID, r_s'\}$, S verifies if the received r_s' is the same as his generated r_S (in step 2). If they are the same, C is authentic. Then, S generates an authentication token sat =H(g^{xy}, $IDPW\_dig$, r_C, r_S) and sends $\{sat\}$ to C.
5. After receiving $\{sat\}$, C computes sat'=H(g^{xy}, $IDPW\_dig$, r_C, r_s') and verifies if the received sat is equal to sat'. If the verification succeeds, S is authentic.

6. After the successful mutual authentication, they can compute the common session key as K=H(g^{xy}, $IDPW\_dig$, r_C, r_S^*), where r_S^* is the result of r_S plus some

fixed value in order for *K* to be different from *sat*.

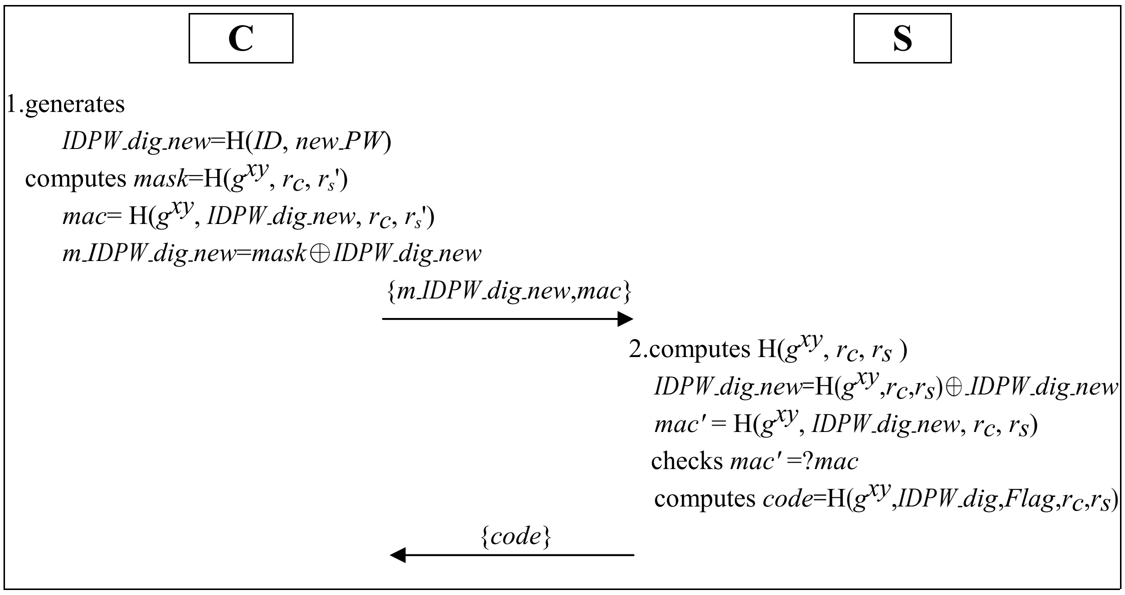

C
S
1.generates
$IDPW\_dig\_new$=H(ID, new_PW)
computes $mask$=H(g^{xy}, r_C, r_s')
mac= H(g^{xy}, $IDPW\_dig\_new$, r_C, r_s')
$m\_IDPW\_dig\_new$=$mask \oplus IDPW\_dig\_new$
{$m\_IDPW\_dig\_new$,mac}
2.computes H(g^{xy}, r_C, r_S)
$IDPW\_dig\_new$=H(g^{xy},r_C,r_S)$\oplus$ $\_IDPW\_dig\_new$
mac' = H(g^{xy}, $IDPW\_dig\_new$, r_C, r_S)
checks mac' =?mac
computes $code$=H(g^{xy},$IDPW\_dig$,$Flag$,r_C,r_S)
{$code$}

Fig. 3. Password update protocol of Hölbl *et al.*'s password change protocol

2.2.2 Password change protocol

In this protocol, when C wants to update his password *PW* to *new_PW*, he performs the password update protocol as follows. We also show it in Fig. 3.

1. After authenticating the server, C computes *mask*=H(g^{xy}, r_C, r_s'), *mac*=H(g^{xy}, *IDPW_dig_new*, r_C, r_s'), and *m_IDPW_dig_new*=*mask*$\oplus$*IDPW_dig_new*, where *IDPW_dig_new*=H(*ID*, *new_PW*). Then, he sends {*m_IDPW_dig_new*, *mac*} to S.
2. After receiving the message, for verifing the validity of *mac*, S retrieves *IDPW_dig_new* by computing H(g^{xy}, r_C, r_S)$\oplus$*m_IDPW_dig_new*, computes *mac'*= H(g^{xy}, *IDPW_dig_new*, r_C, r_S), and compares *mac'* with the received *mac*. If they are equal, S accepts the password change and replaces *IDPW_dig* with *IDPW_dig_new* (Otherwise, he rejects the password change.). S then sends a message *code*=H(g^{xy}, *IDPW_dig*, *Flag*, r_C, r_S) to C, where *Flag* is set to be either '*accept*' or '*reject*', depending upon whether the password change is accepted or rejected.

3. Our improvements

In this section, we present our two improvements on Liao *et al.*'s protocol and Hölbl *et al.*'s protocol in Section 3.1 and Section 3.2, respectively.

3.1 Improvement on Liao *et al.*'s protocol

Our improvement, for correcting the security flaw found by Xiang *et al.*, consists of three phases, registration phase, login phase, and authentication phase. We describe them as follows.

3.1.1 Registration phase

Our registration phase is the same as in the original scheme. That is, C gets a smart card containing *ID*, p, and $B(= g^{\mathrm{H}(s,\, ID)+\mathrm{H}(PW)})$.

3.1.2 Login and authentication phases

Since the purpose of logining to a server is usually for authentication only, seldom for password change, for efficiency consideration, we divide our improvement into two scenarios: (A) authentication only (as shown in Fig. 4), and (B) authentication and password change (as shown in Fig. 5).

(A) Authentication only

(1) Login phase

If C wants to communicate with S without changing his password, C will run the following steps.

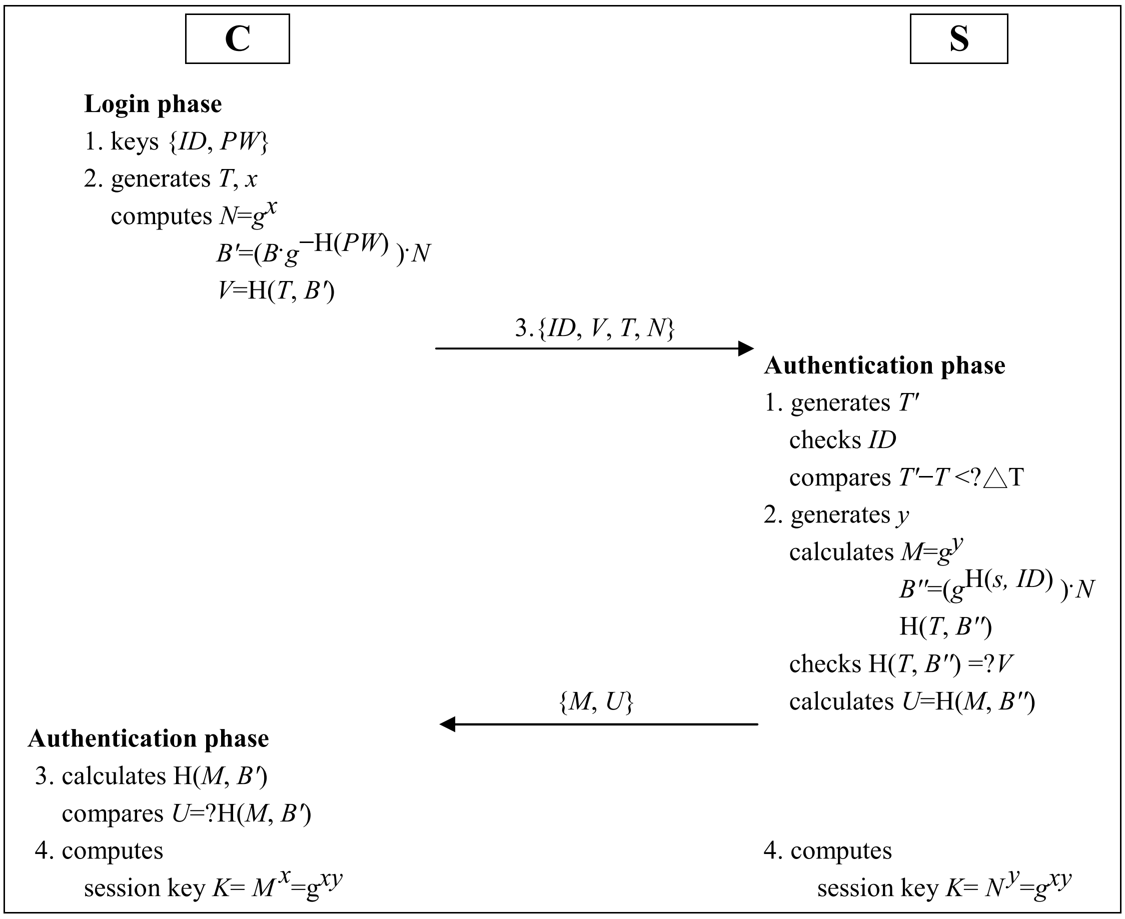

C
S
Login phase
1. keys $\{ID, PW\}$
2. generates T, x
computes $N=g^x$
$B'=(B\cdot g^{-\mathrm{H}(PW)})\cdot N$
$V=\mathrm{H}(T, B')$
3.$\{ID, V, T, N\}$
Authentication phase
1. generates T'
checks ID
compares $T'-T <? \triangle \mathrm{T}$
2. generates y
calculates $M=g^y$
$B''=(g^{\mathrm{H}(s,\, ID)})\cdot N$
$\mathrm{H}(T, B'')$
checks $\mathrm{H}(T, B'') =? V$
calculates $U=\mathrm{H}(M, B'')$
$\{M, U\}$
Authentication phase
3. calculates $\mathrm{H}(M, B')$
compares $U=?\mathrm{H}(M, B')$
4. computes
session key $K= M^x=g^{xy}$
4. computes
session key $K= N^y=g^{xy}$

Fig. 4. Our improvement for authentication only (on Liao *et al.*'s protocol)

1. inserts his smart card and keys $\{ID, PW\}$.
2. generates a timestamp T and a random nonce x, and computes $N=g^x$ mod p, $B'=(B\cdot g^{-H(PW)}$ mod p)$\cdot N$, and V=H(T, B').
3. sends $\{ID, V, T, N\}$ to S.

(2) Authentication phase

When receiving the message $\{ID, V, T, N\}$ from C, S runs the following steps to verify the legitimacy of C and negotiates their common session key.

1. S generates a timestamp T', checks ID, and compares to see if $T'-T$ is less than $\triangle$T. If ID is valid and $T'-T < \triangle$T, S accepts the login request.
2. S generates a random nonce y and calculates $M=g^y$ mod p, $B''=(g^{H(s,\ ID)}$ mod p)$\cdot N$ mod p, and H(T, B''). He checks to see if H(T, B'') is equal to V. If so, C is authentic. S then computes U=H(M, B'') and sends $\{M, U\}$ to C.
3. After receiving the message, C calculates H(M, B') and compares if U is equal to H(M, B'). If it is, S is authentic.
4. After above successful mutual authentication, S can compute the common session key as $K=N^y=g^{xy}$ and C also can compute the common session key as $K = M^x = g^{xy}$.

(B) Authentication and password change

(1) Login phase

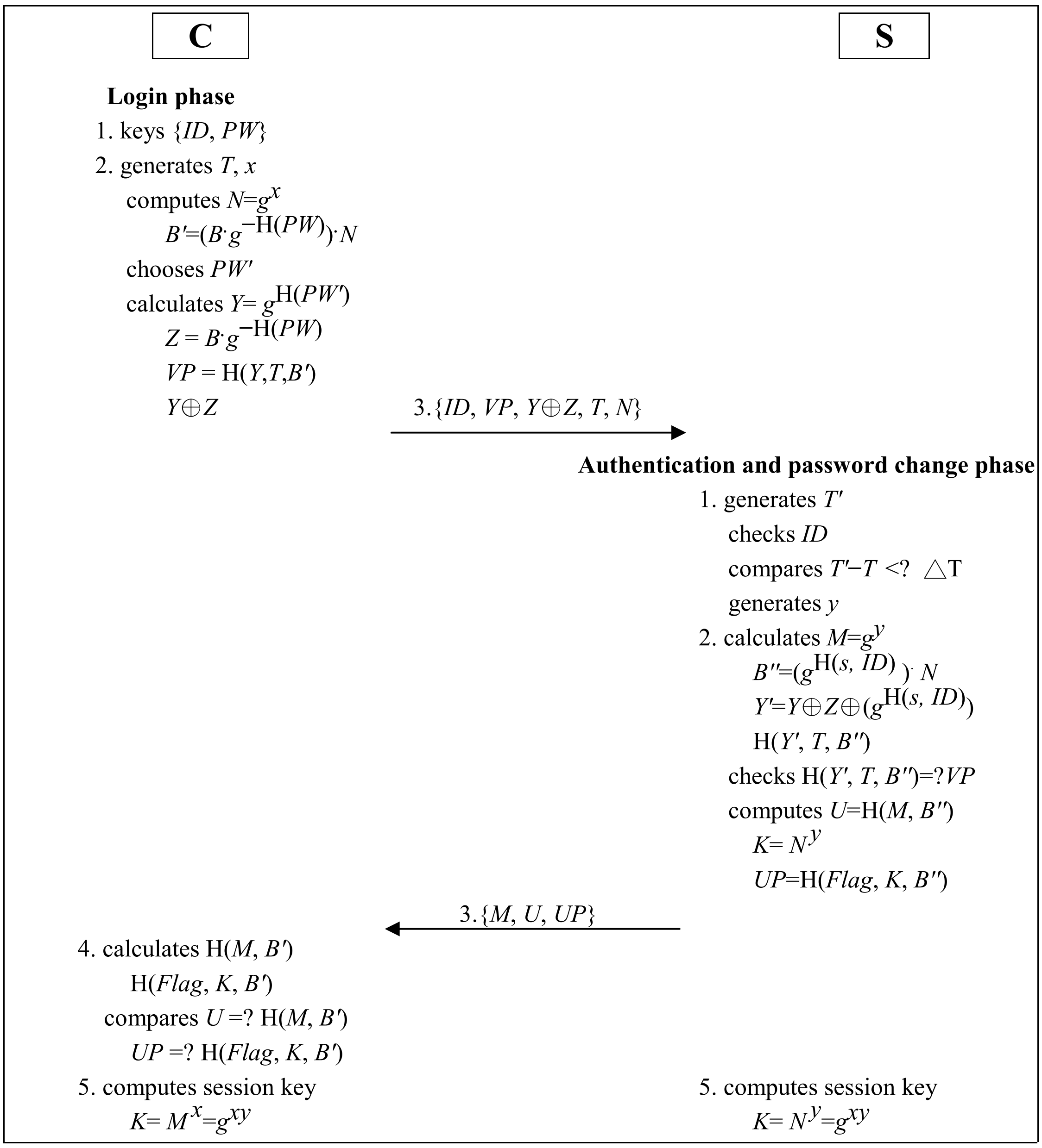
Although in the password change phase of Liao *et al.*'s protocol, C can change his password without communicating with S. However, Xiang *et al.* found that it suffers from the DoS attack. Hence, in the following, we propose our improvement to resist such an attack.

Assume that C wants to change his password from PW to PW', he performs the following steps.

1. inserts his smart card and keys $\{ID, PW\}$.
2. generates a timestamp T and a random nonce x, and computes $N=g^x$ mod p, $B'=(B\cdot g^{-H(PW)}$ mod p)$\cdot N$. He chooses a new password PW' and calculates $Y= g^{H(PW')}$ mod p, $Z = B\cdot g^{-H(PW)}$ mod p ($=g^{H(s,\ ID)}$ mod p), VP = H(Y, T, B'), and $Y\oplus Z$ ($= g^{H(PW')}\oplus g^{H(s,\ ID)}$ mod p).
3. sends $\{ID, VP, Y\oplus Z, T, N\}$ to S.

(2) Authentication and password change phase

After receiving the message $\{ID, VP, Y \oplus Z, T, N\}$ from C, S executes the



C **S**

Login phase

1. keys $\{ID, PW\}$
2. generates T, x
   computes $N=g^{x}$
   $B'=(B \cdot g^{-\mathrm{H}(PW)}) \cdot N$
   chooses PW'
   calculates $Y= g^{\mathrm{H}(PW')}$
   $Z = B \cdot g^{-\mathrm{H}(PW)}$
   $VP = \mathrm{H}(Y,T,B')$
   $Y \oplus Z$

$\xrightarrow{3.\{ID,\ VP,\ Y \oplus Z,\ T,\ N\}}$

Authentication and password change phase

1. generates T'
   checks ID
   compares $T'-T <? \triangle\mathrm{T}$
   generates y
2. calculates $M=g^{y}$
   $B''=(g^{\mathrm{H}(s,\ ID)}) \cdot N$
   $Y'=Y \oplus Z \oplus (g^{\mathrm{H}(s,\ ID)})$
   $\mathrm{H}(Y',\ T,\ B'')$
   checks $\mathrm{H}(Y',\ T,\ B'')=?VP$
   computes $U=\mathrm{H}(M,\ B'')$
   $K= N^{y}$
   $UP=\mathrm{H}(Flag,\ K,\ B'')$

$\xleftarrow{3.\{M,\ U,\ UP\}}$

4. calculates $\mathrm{H}(M,\ B')$
   $\mathrm{H}(Flag,\ K,\ B')$
   compares $U =? \mathrm{H}(M,\ B')$
   $UP =? \mathrm{H}(Flag,\ K,\ B')$
5. computes session key
   $K= M^{x}=g^{xy}$

5. computes session key
   $K= N^{y}=g^{xy}$

Fig. 5. Our improvement for authentication and password change (on Liao *et al.*'s protocol)

following steps to identify C. If C is legal, S accepts the login request for password change and then computes the session key.

1. After receiving the message, S generates a timestamp T', checks ID, and compares if $T'-T$ is less than $\triangle$T. If ID is valid and $T'-T < \triangle$T, the login request continues.
2. S generates a random nonce y and calculates $M=g^{y}$ mod p, $B''= g^{\mathrm{H}(s,ID)} \cdot N$ mod p, $Y'=Y \oplus Z \oplus g^{\mathrm{H}(s,\ ID)}$ mod p ($=g^{\mathrm{H}(PW')}$ mod p), and $\mathrm{H}(Y',\ T,\ B'')$. He checks to see if $\mathrm{H}(Y',\ T,\ B'')$ is equal to VP. If it isn't, S

refuses the request and terminates the protocol. Otherwise, C is authentic. S accepts the password change request and computes U=H(M, B'') , K= N^y mod p, and UP=H($Flag$, K, B''), where $Flag$ is set to '*accept*'.

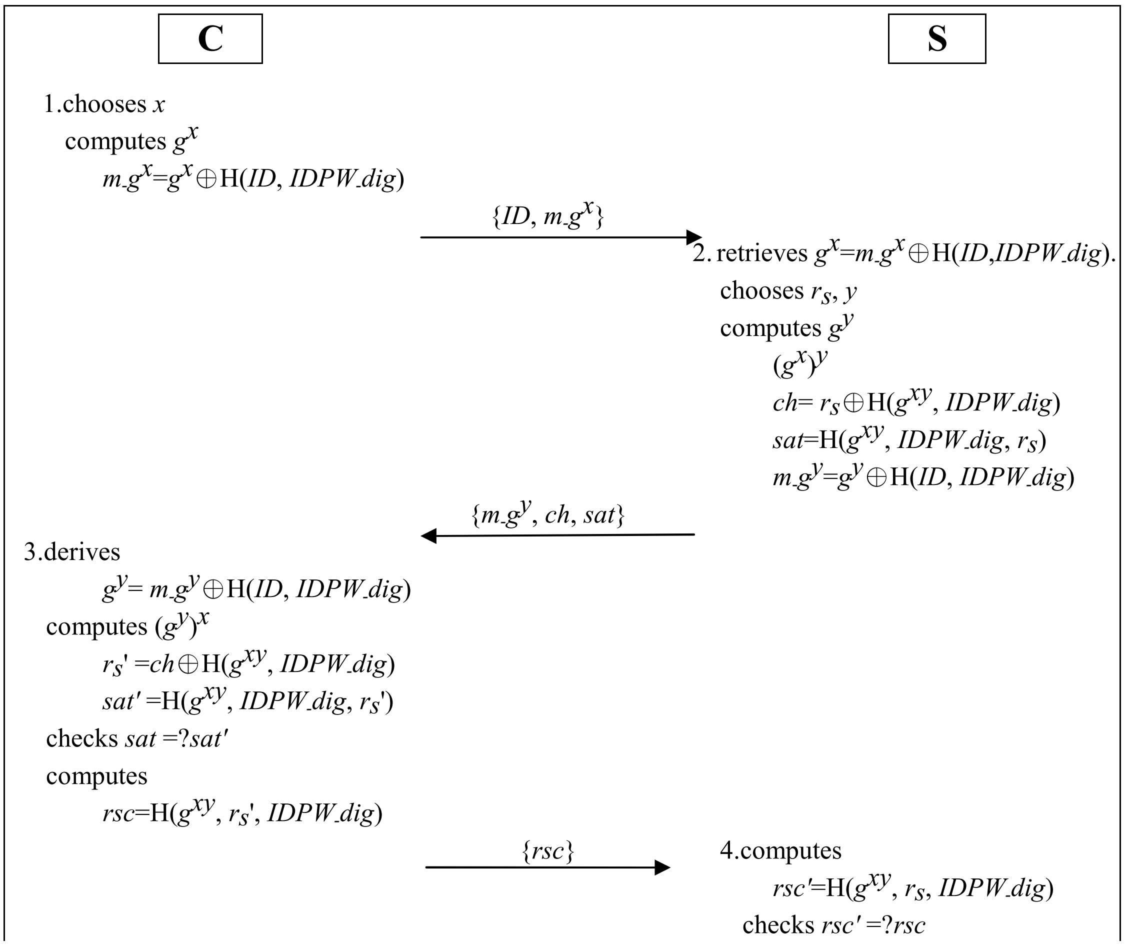
3. S sends {M, U, UP} to C.
4. After receiving the message, C calculates H(M, B') and H($Flag$, K, B'). He compares if U is equal to H(M, B'). If it is, S is authentic. C then compares UP with the value H($Flag$, K, B'). If they are equal, C confirms that his password change request is accepted.
5. After successful mutual authentication, S can compute the common session key as K= N^y= g^{xy} and C also can compute the common session key as K= M^x= g^{xy}.

3.2 Improvement on Hölbl *et al.*'s protocol

In this section, we describe our improvement on Hölbl *et al.*'s authentication protocol as follows and also show it in Fig. 6.

Our improvement performs the following steps.

1. C generates a random nonce x, computes g^x mod p, and masks it as m_g^x by


C
S
1.chooses x
computes g^x
$m\_g^x = g^x \oplus$ H(ID, $IDPW\_dig$)
{ID, m_g^x}
2. retrieves $g^x = m\_g^x \oplus$ H(ID,$IDPW\_dig$).
chooses r_S, y
computes g^y
$(g^x)^y$
$ch = r_S \oplus$ H(g^{xy}, $IDPW\_dig$)
sat=H(g^{xy}, $IDPW\_dig$, r_S)
$m\_g^y = g^y \oplus$ H(ID, $IDPW\_dig$)
{m_g^y, ch, sat}
3.derives
$g^y = m\_g^y \oplus$ H(ID, $IDPW\_dig$)
computes $(g^y)^x$
$r_S' = ch \oplus$ H(g^{xy}, $IDPW\_dig$)
sat' =H(g^{xy}, $IDPW\_dig$, r_S')
checks sat =?sat'
computes
rsc=H(g^{xy}, r_S', $IDPW\_dig$)
{rsc}
4.computes
rsc'=H(g^{xy}, r_S, $IDPW\_dig$)
checks rsc' =?rsc

5.computes session key K=H(g^{xy}, r_S')	5.computes session key K=H(g^{xy}, r_S)

Fig. 6. Improvement on Hölbl *et al.*'s authentication protocol

computing $m_g^x=g^x \oplus$H(ID, $IDPW\_dig$). Then C sends message {ID, m_g^x} to S.

2. After receiving the message, S retrieves g^x by computing $g^x=m_g^x \oplus$H(ID, $IDPW\_dig$). Then, he chooses two random nonces r_S, y and computes g^y mod p. S calculates $(g^x)^y$ mod p, generates $ch= r_S \oplus$H(g^{xy}, $IDPW\_dig$), sat=H(g^{xy}, $IDPW\_dig$, r_S), and masks g^y as m_g^y by computing $m\_g^y=g^y \oplus$ H(ID, $IDPW\_dig$). Then, S sends {m_g^y, ch, sat} to C.
3. After receiving the message from S, C derives $g^y= m_g^y \oplus$H(ID, $IDPW\_dig$), computes $(g^y)^x$ mod p, and retrieves r_S' by computing $ch \oplus$H(g^{xy}, $IDPW\_dig$). He computes sat'=H(g^{xy}, $IDPW\_dig$, r_S') and checks to see if the received sat is equal to the computed sat'. If it is, S is authentic. C then computes rsc=H(g^{xy}, r_S', $IDPW\_dig$) and sends {rsc} to S.
4. After receiving {rsc}, S computes rsc'=H(g^{xy}, r_S, $IDPW\_dig$) and verifies if rsc' is the same as the received rsc. If they are, C is authentic.
5. After successful mutual authentication, C and S each can compute the common session key K=H(g^{xy}, r_S')=H(g^{xy}, r_S).

4. Security analysis for the two improvements

In this section, we show that both of our improvements not only can provide mutual authentication, perfect forward and backward secrecy and key freshness but also can resist the following attacks: replay attack, off-line password guessing attack, insider attack, man-in-the-middle attack, on-line password guessing attack, and DoS attack. We show them in turn below. For abbreviation, we make the analysis behind notation (a) denotes that it is for the improvement on Liao *et al.*'s protocol. The notations ① and ② following notation (a) stand for that they are the analyses for authentication only (as shown in Fig. 4) and for authentication and password change (as shown in Fig. 5), respectively. In addition, we also make the analysis behind notation (b) stands for that it is for the improvement on Hölbl *et al.*'s protocol (as shown in Fig. 6).

4.1 Mutual authentication

Mutual authentication means that both the server and the user can authenticate each other before generating the common session key. In the following, we demonstrate how our protocol can achieve this goal.

(a)① In the phase of authentication only, to authenticate C, S has to verify the validity of V=H(T, B') and to authenticate S, C must check the validity of U=H(M, B''), where B'=B''. Since, these two evidences, V and U, are computed with the common secret B'/B'' and only C and S know the common secret, no one else can forge V and U. In other words, when both the validities of V and U are confirmed by S and C respectively, the mutual authentication between them is achieved.

② In the phase of authentication and password change, to authenticate C, S has to verify the validity of VP = H(Y, T, B') and to authenticate S, C must check the validities of both U=H(M, B'') and UP=H($Flag$, K, B''). Since, VP, U and UP are computed with the common secret B'/B'' and only C and S know the common secret, no one else can forge these values. In other words, when the validities of VP, U and UP are confirmed by S and C respectively, the mutual authentication between them is achieved.

(b) For authenticating S, C has to verify the validity of sat=H(g^{xy}, $IDPW\_dig$, r_S'). Conversely, for authenticating C, S must check the validity of rsc=H(g^{xy}, r_S, $IDPW\_dig$). For only S and C can know or deduce the common secrets, g^{xy}, $IDPW\_dig$, and r_S, no one else can forge the value of sat or rsc. In other words, after the validities of sat and rsc are confirmed by C and S respectively, the mutual authentication in our protocol is achieved.

4.2 Perfect forward and backward secrecy

Perfect forward and backward secrecy means that if an intruder gets the session key, he can’t reconstruct any previous or subsequent session keys. In both of our improvements, a compromised password PW can’t be used to reconstruct any previous or subsequent session keys for that we use the Diffie-Hellman key agreement protocols whose session keys are based on large random nonces. If an intruder gets PW in our two protocols, he can’t deduce K=g^{xy} in (a), or K=H(g^{xy}, r_S) in (b), without the knowledge of the two random numbers, x and y. Therefore, both of our protocols can provide perfect forward and backward secrecy.

4.3 Key freshness

Key freshness means that the key used in each session is different from the ones used in other sessions. Since each party picks his random nonce secretly when computing the session key in our protocols, it can be easily seen that the freshness of the used session keys in our protocols is guaranteed.

4.4 Preventing the replay attack

Replay attack means that a legal peer's transmission message is intercepted and replayed by an adversary for fooling another legal peer to regard him as authentic. However, the fresh nonces chosen at each protocol run are used to avoid such replay attacks in our improvements. More clearly, we list the reason for each case below.

(a) An adversary cannot be authenticated by resending previous messages transmitted by a legal client for that we use the random nonces x, y, and the timestamp T to withstand such kind of attack. Hence, the replay attack can be avoided in this improvement.

(b) Similarly, we use the random nonces r_S, x, y to prevent replay attack in this protocol. An attacker thus cannot be authenticated by resending previous messages transmitted by a legal client. Hence, our protocol can prevent the replay attack.

4.5 Preventing the off-line password guessing attack

Off-line password guessing attack means that a passive attacker slinkingly intercepts the communication line between a legal client and the server, and tries to guess the client's password off line. In the following, we prove why our protocols can resist against such an off-line password guessing attack.

(a) ① In the authentication only scenario, although an adversary E may intercept the message $\{V, T, N\}$, where $N = g^x$, $V = \mathrm{H}(T, B')$, and $B' = (B \cdot g^{-\mathrm{H}(PW)}) \cdot N$. However, he doesn't know both the values of B (stored in C's smart card) and C's password. Therefore, he may guess PW as PW_1, but without the knowledge of B, he can't compute the value $\mathrm{H}(T, (B \cdot g^{-\mathrm{H}(PW_1)}) \cdot N)$ to compare with the intercepted value V. Thus, the off-line password guessing attack can't work.

② In the authentication and password change scenario, although E might intercept the message $\{VP, Y \oplus Z, T, N\}$, where $N = g^x$, $Z = B \cdot g^{-\mathrm{H}(PW)}$, $Y = g^{\mathrm{H}(PW')}$, $VP = \mathrm{H}(Y, T, B')$, and $B' = (B \cdot g^{-\mathrm{H}(PW)})$. However, he doesn't know the values of B (stored in C's smart card), PW (kept secret by C), and PW' (chosen by C). Therefore, E may guess PW and PW' as PW_1 and PW_2, respectively but without the knowledge of B, he can't compute the value $\mathrm{H}(g^{\mathrm{H}(PW_2)}, T, B \cdot g^{-\mathrm{H}(PW_1)})$ to compare with the intercepted value VP. In other words, the off-line password guessing attack fails.

(b) Assume that E has intercepted the transmitted messages, $\{ID, m\text{-}g^x\}$, $\{m\_g^y, ch, sat\}$, and $\{rsc\}$, between C and S. However, since $m\_g^x = g^x \oplus \mathrm{H}(ID, IDPW\_dig)$,

m_g^y=$g^y\oplus$H(*ID*, *IDPW_dig*), *ch*=$r_S\oplus$H(g^{xy}, *IDPW_dig*), *sat*=H(g^{xy}, *IDPW_dig*, r_S), and *rsc*=H(g^{xy}, r_S', *IDPW_dig*), even E has the knowledge of g^x or g^y, due to the one-way property of the hash function, E can hardly figure out *IDPW_dig* from both m_g^x and m_g^y. Similarly, even E has the knowledge of r_S, r_S' and g^{xy}, he can by no means figure out *IDPW_dig* from *ch*, *sat* or *rsc*. Not to mention, all the above parameters are kept secret. Therefore, E can't implement the off-line password guessing attack.

4.6 Preventing the insider attack

Insider attack means that a legal client D can impersonate another legal client C to gain the service of server S.

(a) ① Assume that D wants to impersonate C to login to S. However, without the knowledge of C's password *pw* and *B*, he can not deduce *V* and consequently be authenticated by S. Therefore, the insider attack fails.

② Similarly, if D wants to impersonate C to login to S. Without the knowledge of *B* and C's password, he can not deduce *VP* and be successfully authenticated by S. Therefore, the insider attack fails.

(b) If D wants to impersonate C to login to S. Without the knowledge of C's *IDPW_dig*, the value g^x he uses would be different from the value S retrieves. Hence, the value of *rsc* which D will produce in step 3 would be different from the value of *rsc'* computed by S in step 4 (of Fig. 6). That is, D can't be authenticated by S successfully. Therefore, the insider attack fails.

4.7 Preventing man-in-the-middle attack (MIMA)

Man-in-the-middle attack means that an active attacker intercepts the communication line between a legal user and the server and uses some means to successfully masquerade as both the server to the user and the user to the server. Then, the user will believe that he is talking to the intended server and vice versa.

(a) ① We now launch such a MIMA on our protocol for authentication only (as described in Section 3.1.2.(A)) and illustrate it in Fig. 7. In the figure, after E intercepts the communication line between S and C, he impersonates C by sending {*ID, V', T', N'*} to S and masquerades as S by sending {*M', U'*} to C. If both C and S can successfully verify *U'* and *V'* respectively, E is regarded as being authentic by both of the two communicating parties and will have the two same session keys shared with C and S, respectively.

However, since that for verifying *U'*, C should compute H(*M*, *B'*) and for verifying *V'*, S should calculate H(*T*, *B''*), where $B' = (B \cdot g^{-\mathrm{H}(PW)}) \cdot N = B'' = (g^{\mathrm{H}(s,\ ID)}) \cdot N$, without the knowledge of *PW* or *s*, E can't send valid *U'* and *V'*. Hence, the MIMA fails.

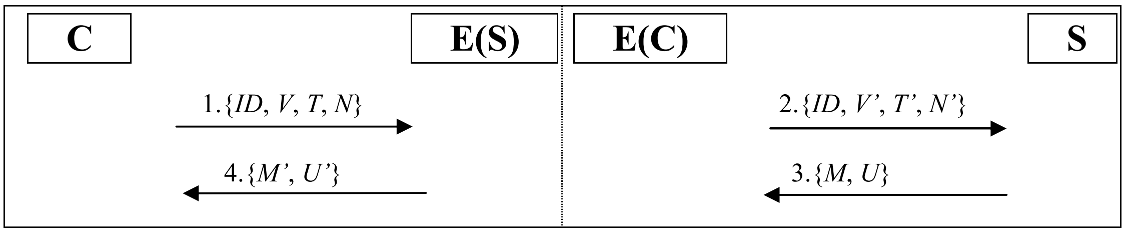


Fig. 7. The MIMA on authentication only in section 3.1.2.(A)

② In Fig. 8, we depict a MIMA on the authentication and password change (as described in Section 3.1.2.(B)). In the figure, after E intercepts the message transmitted between S and C, he impersonates C by sending {*ID*, *VP'*, *Y'*⊕ *Z'*, *T'*, *N'*} to S and masquerades as S by sending {*M'*, *U'*, *UP'*} to C. If C can successfully verify *U'* and *UP'* and S can successfully verify *VP'*, E is regarded as being authentic by both of them. That is, E would have the two same session keys shared with C and S, respectively. Therefore, C can make them believe that they each are talking to the intended party. However, for verifying *U'* and *UP'*, C should compute H(*M*, *B'*) and H(*Flag*, *K*, *B'*). And for verifying *VP'*, S should calculate H(*Y'*, *T*, *B''*), where $B' = (B \cdot g^{-\mathrm{H}(PW)}) \cdot N = B'' = (g^{\mathrm{H}(s,\ ID)}) \cdot N$. Without the knowledge of *PW* or *s*, E can't compute valid *U'*, *UP'*, and *VP'*. Hence, the MIMA fails.

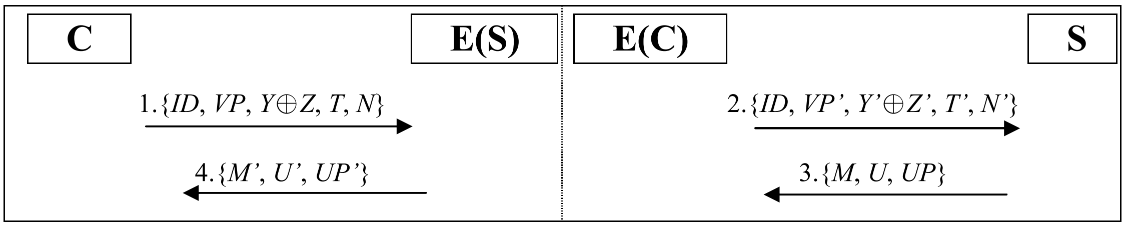


Fig. 8. The MIMA on authentication and password change in section 3.1.2.(B)

(b) Similarly, we launch a MIMA on the proposed improvement in section 3.2 and also illustrate it in Fig. 9. In the figure, after C sends {*ID*, m_g^{x}} to S, for impersonating S, E forges m_g^{x} to be $m\_g^{x'}$, then he forwards {*ID*, $m\_g^{x'}$} to S. Likewise, after S sends out {m_g^{y}, *ch*, *sat*}, E replaces it and sends his forged message {$m\_g^{y'}$, *ch'*, *sat'*} to C. Subsequently, after C sends out {*rsc*}, E replaces it and sends {*rsc'*} to S. If C and S can successfully verify *sat'* and *rsc'* respectively, E is authenticated by the two communicating parties. Then, E will have the two right session keys shared with C and S, respectively. However, for verifying *sat'*, C should compute H(g^{xy}, *IDPW_dig*, r_S'), and for verifying *rsc'*, S should calculate H(g^{xy}, r_S, *IDPW_dig*), where x is a random number chosen by C, and r_S, y are chosen by S (as shown in Fig. 6). Obviously, without the knowledge of *IDPW_dig*, E can't successfully be authenticated by C and S, respectively. Not to mention, E doesn't know the values of r_S and g^{xy}. Therefore, the MIMA on this improvement fails.

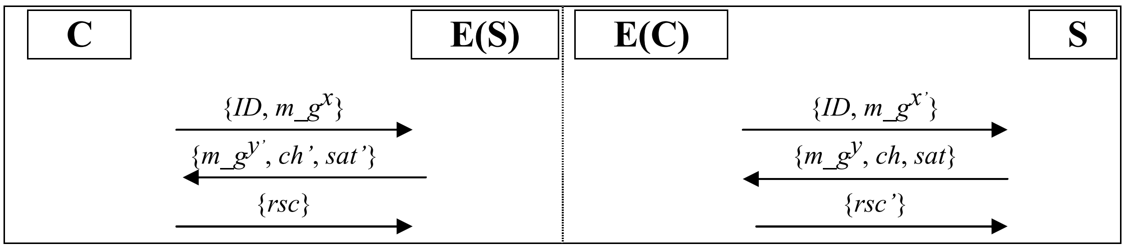


Fig. 9. The MIMA on (b) in section 3.2

4.8 Preventing the on-line password guessing attack

Suffering on-line password guessing attack means that an attacker can successfully guess a legal user's password on line. Since the two improvements we proposed have the mutual authentication function. Only the user with the right password can pass the authentication of the server. Therefore, any attempt to launch a password guessing attack will be detected by the server. Moreover, we can set both improvements to tolerate some times of wrong password logins, e.g., three time. If the number of wrong login times is reached, the system would reject the login request. Under such a setting, both of our schemes can resist the on-line password guessing attack.

4.9 Preventing smart-card-lost attack

Smart-card-lost attack means an attacker can launch various attacks when he gets a legal user's smart card. In the following, we discuss two of the most common attacks launched under such a situation, impersonation attack and off-line password guessing attack.

(a) Suppose C's smart card is lost and obtained by E. Through, E can read the value of B. However, without the knowledge of s, E still can't get the value of $g^{\mathrm{H}(s,\ ID)}$, where s is the secret of S. Hence, E can't launch the off-line password guessing attack, for example, he may guess password PW as PW' and verify whether $B{\cdot}g^{-\mathrm{H}(PW')}$ is equal to $g^{\mathrm{H}(s,\ ID)}$. Therefore, the off-line password guessing attack can't work. As for the impersonation attack prevention, we analyze them respectively below for the two scenarios.

① In the scenario of authentication only, if E has got C's smart card and knows the value of B. He starts the authentication protocol for being authenticated by S. However, he doesn't know C's password *PW*. He can not deduce C's $B'(=(B\cdot g^{-\mathrm{H}(PW)})\cdot N)$ and henceforth $V(=\mathrm{H}(T, B'))$ which will be verified by S (as shown in Fig. 4). Therefore, he can't be authenticated by S successfully. In other words, the impersonation attack fails.

② Similarly, in the scenario of authentication and password change, assume that E has got C's smart card and knows *B*. He starts the authentication protocol for being authenticated by S (as shown in Fig. 5). However, he faces the same problem as stated above in ① that he doesn't know C's password *PW*. He can not deduce C's B' and hence *VP*. Since for being authenticated by S, E generates a random *x*, computes $N=g^x$, and randomly selects two passwords PW'' and PW''' as the old and new one respectively. He then computes $Y=g^{-\mathrm{H}(PW''')}$, $B'=B\cdot g^{-\mathrm{H}(PW'')}\cdot N$, and $Z=B\cdot g^{-\mathrm{H}(PW'')}$, and let $VP=\mathrm{H}(Y, T, B')$. However, due to $B=g^{\mathrm{H}(s,\ ID)+\mathrm{H}(PW)}$ and *PW* is not equal to PW'' with nonnegligible probability, without the knowledge of *s* and *PW*, *VP* can hardly be authenticated by S since B' (computed by E) is not equal to B'' (deduced by S). That is, he can not be authenticated by S. Therefore, E can't launch the impersonation attack.

(b) For the protocol using no smart cards, we needn't examine it.

4.10 Preventing DoS attack after password changing

Suffering DoS attack means that if an attacker temporarily gets access to the client's smart card and successfully guesses the password. Then, he can perform the password change phase to replace the old password with his new one. This would result in making the legal user's password invalid and thereafter the server will deny providing any service to the legal user. However, we have shown above that both of our improvements can prevent smart-card-lost attack, on-line, and off-line password guessing attack (as shown in section 4.5, 4.8, and 4.9 respectively). That is, even an attacker E can temporarily get access to the client's smart card, he can't successfully launch smart-card-lost attack, on-line, or off-line password guessing attack. Consequently, our two improvements can resist against the DoS attack.

5. Discussion and comparison

In this section, we first demonstrate that both of our two improvements, (1) (on [9]) and (2) (on [12]), are more efficient and secure than [9], [12], and all other existing 2PAKE schemes. Then, we show why the improvement on [9] can meet the ten requirements as indicated in Section 2.1.5

5.1 Low communication cost

From in Fig. 1 and Fig. 2, we know that Liao *et al.*'s protocol [9] needs three passes and Hölbl *et al.*'s protocol [12] needs four passes. However, our improvement on Liao *et al.*'s protocol needs only two passes in both of the two scenarios (as shown in Fig. 4 and 5 respectively), and the improvement on Hölbl *et al.*'s protocol needs only three passes. Due to the reason that when we estimate the efficiency of a protocol, the number of passes is always the dominant factor when compared with its computational overhead. Therefore, our improvements are more efficient than [9] and [12], respectively.

In the following, we make comparisons, in the aspects of needed number of passes and whether it can satisfy the ten security features (STSF) listed in Section 4, among our two schemes and other existing 2PAKE protocols [1-12, 30-36] as mentioned in Section 1 (Here, we ignore schemes [13-19] since they didn't propose improvements or new methods). We show the results for the smart-card password based schemes in Table 1, and for the non-smart-card password based schemes in Table 2, respectively. For convenience, in the tables, we use notations i(1) and i(2) to denote the improvements on [9] and [12], respectively.

Table 1. Comparisons in passes and STSF for smart-card password based schemes

schemes	i(1)	[1]	[2]	[3]	[5]	[6]	[7]	[8]	[9]	[10]	[30]	[31]	[32]	[33]	[34]	[35]	[36]
passes	2	2	3	3	3	1	3	3	3	4	2	2	2	3	2	4	3
STSF	○	×	○	○	○	×	×	×	×	×	×	×	×	○	×	○	×

Table 2. Comparison in passes and STSF for the non-smart-card password based schemes

schemes	i(2)	[11]	[12]	[4]
passes	3	2	4	4
STSF	○	×	×	×

From Table 1 and 2, we can see that both of our improvements, i(1) (smart-card password based) and i(2) (non-smart-card password based), are the most efficient and secure than all of the proposed schemes.

5.2 Ten requirements satisfaction of our improvement on Liao *et al.*'s scheme

For the improvement on Hölbl *et al.*'s protocol using no smart card, in this section, we only show that the improvement on Liao *et al.*'s scheme can meet Liao *et al.*'s ten requirements for a smart-card password based authentication protocol.

R1. It needs no password or verifier tables.

Our improvement (as described in Section 3.1) needs no password or verifier tables stored in the server's memory or in the clients' smart cards. Hence, it meets this requirement.

R2. The clients can choose and change their passwords at will.

Since in our schemes, the password change request can only be accepted after successful mutual authentication. This guarantees that only the real card holder can securely and freely change his password. We have shown this in Section 3.1.2.(B). Put it another way, the improvement we proposed on the password change protocol can let the client choose and change his password freely and securely.

R3. The clients need not reveal their password to the administrator of the server.

Our register phase is the same as the one in Liao *et al.*'s protocol. Hence, the password is not revealed to the administrator of the server.

R4. The passwords are not transmitted in plaintext over the Internet.

In the two scenarios, (A) and (B), of our improvement in Section 3.1, the password is not transmitted in a clear form. Hence, the improvement can satisfy this requirement.

R5. It can resist the insider (a legal user) attacks.

We have demonstrated this in Section 4.6.(a).

R6. It can resist against the replay attack, password guessing attack, modification-verifier-table attack, and stolen-verifier attack.

Since our improvement needs no verifier table (as does in Liao *et al.*'s protocol), it can resist the modification-verifier-table attack and stolen-verifier attack. In addition, we have demonstrated that our protocol can resist the replay attack in Section 4.4, and password guessing attack in Section 4.5 and 4.8, respectively.

R7. The length of a password is appropriate for memorization.

Although Liao *et al.*'s protocol [9] suffers from the password guessing attack, our improvement can withstand this kind of attack. Since, the client's password is first hashed then computed with several secret parameters which are unknown to any attacker. Although, an attacker may eavesdrop on the communication line and get the value of V (in Fig. 4) or VP (in Fig. 5). However, the value of B in both of them is first randomized by $g^{-\mathrm{H}(PW)} \cdot N$ and then hashed by a hash function H. Due to the one-way property of a secure hash function, the attacker has no idea about the value of B in both cases. Therefore, he hasn't any information available to guess the password. Not to mention, our scheme can prevent on-line and off-line password guessing attack. That is, our improvement is really secure even if the client uses a weak easily-memorized password. In other words, our protocol can let the client use an easy-remember password.

R8. It is efficient and practical.

Other than its low communication cost as described in Section 5.1, our improvement has another advantage that it needs no complex computation (such as bilinear pairing [1, 6, 7]), it only uses the computations of hash functions and Diffie-Hellman key agreement as performed in the original one [9]. Hence, our improvement is efficient and practical.

R9. It can achieve mutual authentication.

We have demonstrated this in Section 4.1.(a).

R10. It can resist password guessing attacks even if the smart card is lost.

We have demonstrated this in Section 4.9.(a).

6. Conclusion

In this article, we have analyzed and improved the two password-based authentication protocols, [9] and [12], and shown that both of our improvements are not only more secure but also more efficient than the two original schemes (reviewed in Section 2). Especially, we also have shown that our improvement on Liao *et al.*'s protocol not only can satisfy the ten requirements (proposed by Liao *et al.* themselves) but also can withstand both DoS attack on the password change phase and the password guessing attack if the smart card is lost. Moreover, we have made comparisons among our two improvements and the other proposed schemes on both the communicational cost (needed number of passes) and the ten security features (listed in Section 4) in Table 1 and Table 2, respectively. From the tables, we conclude that both of our improvements can satisfy the ten security features and have lower communication cost while compared with all of the other proposed password based schemes nowadays.

Appendix A

In the following (A.1 to A.7), we first briefly review the scheme then show the attack.

A.1 Why does Bindu et al.'s scheme [7] suffer from the smart-card-lost attack launched by an insider?

Review

There are three phases in Bindu *et al.*'s scheme: the registration phase, the login phase and the authentication phase.

In the registration phase, the server S issues to legal user i a smart card which contains m_i and I_i, where m_i=H($ID_i \oplus s$)$\oplus$H(s)$\oplus$H(PW_i), I_i=H($ID_i \oplus s$)$\oplus s$, and s is S's secret key.

When i wants to login to S, he starts the login phase by computing $r_i=g^x$ (x is a random number chosen by i), $M=m_i \oplus$H(PW_i), $U=M\oplus r_i$, $R=I_i\oplus r_i=$ H($ID_i\oplus s$)$\oplus s \oplus r_i$, and E$_R$[r_i, ID_i, T] (T is a timestamp, and E$_R$[r_i, ID_i, T] is a ciphertext encrypted by the secret R). He then sends {U, T, E$_R$[r_i, ID_i, T]} to S.

In the authentication phase, after receiving {U, T, E$_R$[r_i, ID_i, T]} at timestamp T_s, S computes $R=$ $U\oplus$H(s)$\oplus s$ $=M\oplus r_i\oplus$H(s)$\oplus s$ $=m_i\oplus$H(PW_i)$\oplus r_i\oplus$H(s)$\oplus s$ = H($ID_i\oplus s$)$\oplus$H(s)$\oplus$H(PW_i)$\oplus$H(PW_i)$\oplus r_i\oplus$H(s)$\oplus s$ = H($ID_i\oplus s$)$\oplus r_i\oplus s$, decrypts E$_R$[r_i, ID_i, T], checks to see if T_s-T is less than $\triangle$T, and compares R with H($ID_i \oplus s$)$\oplus s\oplus r_i$ to see if they are equal. If they are, he sends { T_s, E$_R$[r_s, r_i+1, T_s]} to i, where $r_s=g^y$ and y is a random number chosen by S. After that, i verifies the validity of the time interval, decrypts E$_R$[r_s, r_i+1, T_s], and checks to see if r_i+1 is correct or not. If it is, S is authentic. Then, i sends {E$_{Kus}$[r_s+1]} to S, where $Kus=r_s^x=g^{xy}$. S decrypts the message and checks to see if r_s+1 is correct or not. If it is, i is authentic.

Attack

If C lost his smart card and the card is got by an insider E, E can impersonate C to log into S. We show the attack in the following.

For that C's smart card stores m_c=H($ID_c\oplus s$)$\oplus$H(s)$\oplus$H(PW_c) and I_c=H($ID_c\oplus s$)$\oplus s$, and E's smart card stores m_e=H($ID_e\oplus s$)$\oplus$H(s)$\oplus$H(PW_e) and I_e=H($ID_e\oplus s$)$\oplus s$, suppose E gets C's smart card but doesn't have the knowledge of PW_c, E can choose a random number x and computes $r_c=g^x$, $V=$ $m_e\oplus I_e\oplus$H(PW_e)=H(s)$\oplus s$, $M=I_c\oplus$ $V=$ H($ID_c\oplus s$)$\oplus s\oplus$H(s)$\oplus s$ =H($ID_c\oplus s$)$\oplus$H(s) which equals $m_c\oplus$H(PW_c), $U=M\oplus r_c$, and $R=$ $I_c\oplus r_c$. Then, E masquerades as C by sending {U, T, E$_R$[r_c, ID_c, T]} to S. After receiving the message, S computes $R=U\oplus$H(s)$\oplus s$ and compares R with H($ID_c\oplus s$)$\oplus s\oplus r_c$. If they are equal, S sends C the message { T_s, E$_R$[r_s,

r_c+1, T_s]}. E intercepts the message, decrypts $E_R[r_s, r_c+1, T_s]$, and uses r_s to compute $Kus=r_s^x=g^{xy}$. E then can send a correct message $\{E_{Kus}[r_s+1]\}$ to S, to let S authenticate him as C. In other words, insider E can successfully launch a smart-card-lost attack.

More clarity, we demonstrate why $R=U\oplus H(s)\oplus s$ is equal to $H(ID_c\oplus s)\oplus s\oplus r_c$ by the following equations.

$$
\begin{aligned}
R &= U\oplus H(s)\oplus s \\
&= M\oplus r_c\oplus H(s)\oplus s \quad\cdots\cdots\quad \because\ U=M\oplus r_c \\
&= I_c\oplus V\oplus r_c\oplus H(s)\oplus s \quad\cdots\cdots\quad \because\ M=I_c\oplus V \\
&= H(ID_c\oplus s)\oplus s\oplus V\oplus r_c\oplus H(s)\oplus s \quad\cdots\cdots\quad \because\ I_c=H(ID_c\oplus s)\oplus s \\
&= H(ID_c\oplus s)\oplus s\oplus H(s)\oplus s\oplus r_c\oplus H(s)\oplus s \quad\cdots\cdots\quad \because\ V=H(s)\oplus s \\
&= H(ID_c\oplus s)\oplus s\oplus r_c
\end{aligned}
$$

A.2 Why is Juang et al.'s [8] protocol vulnerable to the password guessing attack if the smart card is lost?

Review

In [8], if an attacker gets C's smart card, he can successfully launch an off-line password-guessing attack for impersonating C to log into the server S. In the following, we first review Juang *et al.*'s protocol and then show the attack.

Their protocol consists of four phases: the setup phase, the registration phase, the login and authentication phase, and the password changing phase.

In the setup phase, S chooses two secrets s, x and publishes $P_s=sP$, where P is a generator of an additive cyclic group G_1 with a prime order q.

In the registration phase, the server S issues to legal user i a smart card which contains b_i ($b_i=E_x[H(PW_i, b), ID_i, H(H(PW_i, b), ID_i)]$ and $E_x[M]$ is a ciphertext of M encrypted by S's secret key x) and b (a random number chosen by i).

When i wants to log into S, i starts the login and authentication phase and sends $\{aP, \alpha\}$ to S, where a is a random number chosen by i, $\alpha=E_{Ka}[b_i]$, $Ka=H(aP, P_s, Q, \hat{e}(P_s, aQ))$, $\hat{e}: G_1\times G_1\rightarrow G_2$ is a bilinear mapping, $Q=h(ID_s)$, $h(\cdot)$ is a map-to-point hash function $h:\{0,1\}^*\rightarrow G_1$, and ID_s is S's identification. Subsequently, S chooses a random number r, computes the session key $sk=H(H(aP, P_s, Q, \hat{e}(aP, sQ)), r, ID_i, ID_s)$ $=H(Ka, r, ID_i, ID_s)$ since $\hat{e}(P_s, aQ)=\hat{e}(aP, sQ)$, and sends $\{Auth_s, r\}$ to user i, where ID_i is i's identification, $Auth_s=H(Ka, H(PW_i, b), r, sk)$, and $H(PW_i, b)$ is obtained from decrypting α and b_i. Then, i computes the session key sk. For authenticating S, he verifies $Auth_s$ to see if it is equal to $H(Ka, H(PW_i, b), r, sk)$. If it is, i computes and sends $\{Auth_i\}$

to S, where $Auth_i$=H(Ka, H(PW_i, b), r+1, sk) and H(PW_i, b) is the hash result of b stored in the smart card with PW_i inputted by i. Finally, for authenticating i, S checks to see if $Auth_i$ is equal to H(Ka, H(PW_i, b), r+1, sk).

Attack

In this protocol, it can be easily seen that if user C lost his smart card and the card is got by an insider E, E can impersonate C to log into S. We show the attack in the following.

E reads out b and b_c (which equals E_x[H(PW_c, b), ID_i, H(H(PW_i, b), ID_i)]) stored in C's smart card but he doesn't have the knowledge of PW_c. He can choose a random number c, computes cP, Kc=H(cP, P_s, Q, ê(P_s, cQ)), α=E_{Kc}[b_c], starts the protocol, and masquerades as C to send { cP, α} to S. After receiving the message, S chooses a random number r, computes session key sk=H(Kc, r, ID_c, ID_s), $Auth_s$=H(Kc, H(PW_c, b), r, sk), and sends {$Auth_s$, r} to C. E intercepts the message and launches an off-line password guessing attack. He chooses a possible password PW', computes Kc=H(cP, P_s, Q, ê(P_s, cQ)), sk=H(Kc, r, ID_c, ID_s), H(Kc, H(PW', b), r, sk) and checks to see if it is equal to the received $Auth_s$. If it is, the attacker successfully gets C's password PW_c which is equal to PW'. Subsequently, E can masquerade as C by using PW' and C's smart card to log into S.

A.3 Why are the protocols of Goriparthi et al.[6] and Wang et al.[31] vulnerable to the DoS attack on the password change phase which can make the password invalid after their protocol run?

Review of [6]

In the password change phase of Goriparthi *et al.*'s protocol [6], when client C wants to change his password PW, he keys his ID and PW to his smart card. The smart card verifies ID (without verifying his password PW) to see if it is correct. If it is, the smart card will subsequently receive a new password PW* submitted by C and compute Reg^*_{ID}= Reg_{ID} –h(PW)+h(PW*)= s·h(ID)+h(PW*), where Reg_{ID}= s·h(ID) + h(PW) is stored in C's smart card, h(·) is a map-to-point hash function h:{0,1}*→G_1, and G_1 is a group on an elliptical curve. Finally, the smart card will replace Reg_{ID} with Reg^*_{ID}.

Attack on [6]

In [6], assume that there is an attacker temporarily gets access to C's smart card. He randomly selects two passwords PW' and PW'' as the old and the new ones, respectively. The smart card will then compute Reg'_{ID}=Reg_{ID}–h(PW') +h(PW'')= s·h(ID)+h(PW)–h(PW') +h(PW'') and replace Reg_{ID} with Reg'_{ID}. This would make

C’s password PW invalid.

Review of [31]

In Wang *et al.*’s protocol [31], C inserts his smart card, keys PW, and requests to change the password PW to a new one PW^*. Then, the smart card computes $N_i^* = N_i \oplus \mathrm{H}(PW) \oplus \mathrm{H}(PW^*)$ and replaces N_i with N_i^*, where $N_i=\mathrm{H}(PW_i) \oplus \mathrm{H}(x)$ is stored in C’s smart card, PW_i is chosen by the user when he registers at the remote server S, and x is S’s secret key.

Attack on [31]

Obviously, protocol [31] also exits the same security loophole as does in [6]. Since if an attacker temporarily gets access to C’s smart card and reads the value of N_i, he can use two random values PW' and PW'' to compute $N_i' = N_i \oplus \mathrm{H}(PW') \oplus \mathrm{H}(PW'')$ and replace N_i with N_i'. From then on, client C can never pass the authentication and the attack succeeds.

A.4 Why do [30, 32] suffer from the smart-card-lost attack?

Review of [30]

In [30], when user C wants to change his password, he inserts his card and types his ID and PW. The smart card computes $K^*_1=R \oplus \mathrm{H}(PW)$ and compares K^*_1 with K_1 to see if they are equal, where $R(=K_1 \oplus \mathrm{H}(PW_c))$ and $K_1(=\mathrm{H}(ID \oplus x) \oplus N)$ are stored in C’s smart card, PW_c is chosen by the user when he registers at the remote server S, and N is a random number. If they are, the card accepts the password change request and C inputs a new password PW^*. Then, the card computes $R^*= K^*_1 \oplus \mathrm{H}(PW^*)$ and $K^*_2= K_2 \oplus \mathrm{H}(PW \oplus \mathrm{H}(PW)) \oplus \mathrm{H}(PW^* \oplus \mathrm{H}(PW^*))$, where $K_2=\mathrm{H}(ID \oplus x \oplus N) \oplus \mathrm{H}(PW_c \oplus \mathrm{H}(PW_c))$ is also stored in C’s smart card. Finally, the smart card will replace R and K_2 with R^* and K^*_2, respectively.

Attack on [30]

An attacker who gets C’s smart card and reads the values of R, K_1, and K_2 can launch a password-guessing attack. He chooses a possible password PW', computes $K'_1=R \oplus \mathrm{H}(PW')$, and checks to see if K'_1 and K_1 are equal. If they are, PW' is the correct password. Then, for changing the password from PW' to PW^*, the attacker logins to the server and computes $R^*= K'_1 \oplus \mathrm{H}(PW^*)$ and $K_2^*= K_2 \oplus \mathrm{H}(PW' \oplus \mathrm{H}(PW')) \oplus \mathrm{H}(PW^* \oplus \mathrm{H}(PW^*))$. He then replaces R and K_2 with R^* and K_2^*, respectively. Eventually, he can masquerade as C to log into the server. That is, he can successfully implement a smart-card-lost attack.

Review of [32]

In [32], when user C wants to change his password, he inserts his card and types his ID and PW. The smart card computes $P^*=R \oplus \mathrm{H}(b \oplus PW)$, and $V^*=\mathrm{H}(P^*$

$\oplus$H(PW)), and compares V^* with V, where PW is C's password inputted for being changed, and R, b, and V are stored in C's smart card. If they are equal, the card accepts the password change request and then computes $R_{new}=P^*\oplus$H($b\oplus PW^*$) and V_{new}=H($P^*\oplus$H(PW^*)), where PW^* is a new password submitted by C. Finally, the smart card replaces V with V_{new}.

Attack on [32]

Assume that an attacker who can get C's smart card reads the values of R, b, and V and implements a password-guessing attack. He chooses a possible password PW', computes $P'=R\oplus$H($b\oplus PW'$) and V'=H($P'\oplus$H(PW')), and checks to see if V' and V are equal. If they are, PW' is the correct password. Then, for changing the password from PW' to PW'', the attacker logins to the server and computes $R''=P'\oplus$H($b\oplus PW''$) and V''=H($P'\oplus$H(PW'')), where PW'' is a new password submited by E. Finally, the smart card replaces R and V with R'' and V'', respectively. The attacker can therefore masquerade as C to log into the server. That is, the attacker successfully implements a smart-card-lost attack.

A.5 Why does Xu *et al.*'s protocol [34] is vulnerable to the insider impersonation attack?

Review

We first briefly review the protocol [34] and then present our attack.

Xu *et al.*'s protocol consists of three phases: the registration phase, the login phase and the authentication phase.

In the registration phase, user C submits his ID_c and PW_c to the server S. S issues C a smart card which stores C's identity ID_c, and B=H$(ID_c)^x$+H(PW_c), where x is S's secret key and PW_c is C's password.

In the login phase, user C inputs ID_c and PW_c to his smart card. The card obtains system's timestamp T, chooses a random number v, computes B_c=(B-H(PW_c))^v =H$(ID_c)^{x\,v}$, W=H$(ID_c)^{\,v}$, and C_1=H(T, B_c, W, ID_c), and sends { ID_c, C_1, W, T } to S.

In the authentication phase, after receiving { ID_c, C_1, W, T } at time T^*, S computes B_s= W^x, and checks to see if ID_c is valid, $T^*-T <$ △T, and C_1 is equal to H(T, B_s, W, ID_c). If they are, S selects a random number m, sets T_s to be the current time, computes M=H$(ID_c)^m$, C_s=H(M, B_s, T_s, ID_c), and sends { ID_c, C_s, M, T_s } to C. After receiving the message, C validates ID_c and T_s, computes H(M, B_c, T_s, ID_c), and compares it with the received C_s. If they are equal, S is authentic. Then, C and S can compute the common session key as sk=H(ID_c, M, W, M^v) and

sk=H(ID_c, M, W, W^m), respectively.

Attack

Assume that a malicious insider U wants to masquerade as C to access S's resource. He reads B from his smart card, obtains system's timestamp T_u, chooses a random number r, computes B_u=(B-H(PW_u))^r = H(ID_u)^{xr}, W=H(ID_c)^r, C_1=H(T_u, B_u, W, ID_c), and sends { ID_c, C_1, W, T_u } to S.

After receiving the message, S validates ID_c and T_u, computes B_s= W^x =H(ID_c)$^{r x}$, and checks to see if the received C_1 is equal to the computed H(T_u, B_s, W, ID_c). In this case, we can see that C_1 is doomed to be equal to H(T_u, B_s, W, ID_c). So, U (who masquerades as C) is authentic. Finally, S obtains the system's timestamp T_s and sends { ID_c, C_s, M, T_s } to U, where M=H(ID_c)^m and m is a random number chosen by S. U also can compute the session key as sk=H(ID_c, M, W, M^r) shared with S. Therefore, user U's impersonation attack succeeds.

A.6 Why do both Hölbl *et al.*'s protocols [37] fail ?

Hölbl *et al.* proposed two improvements of two-party key agreement protocols. In the following, we first briefly review then present our attack on both of their protocols, respectively.

Review of the first protocol

Hölbl *et al.*'s first protocol consists of three phases: the system setup phase, the private key extraction phase, and the key agreement phase.

In the system setup phase, key generation center (KGC) chooses a random number x_s and keeps it secret. He computes y_s=g^{x_s} and publishes it.

In the private key extraction phase, with each user having his identity ID, KGC selects a random number k_i, and calculates i's private key v_i=$I_i k_i$+$x_s u_i$ (mod p-1) and public key u_i=g^{k_i} (mod p), where I_i=H(ID_i).

In the key agreement phase, user A chooses a random number r_a, computes t_a= g^{r_a} , and then sends { u_a, t_a, ID_a } to user B. After receiving { u_a, t_a, ID_a }, B chooses a random number r_b , calculates t_b= g^{r_b}, and then sends { u_b, t_b, ID_b } back to A. Finally, A and B can compute their common session key, K=($u_b^{I_b}$·$y_s^{u_b}$·t_b)$^{(v_a+r_a)}$= $g^{(v_b+r_b)\cdot(v_a+r_a)}$ and K=($u_a^{I_a}$·$y_s^{u_a}$·t_a)$^{(v_b+r_b)}$=$g^{(v_a+r_a)\cdot(v_b+r_b)}$, respectively, where I_a=H(ID_a) and I_b=H(ID_b).

Attack on the first protocol

Assume that an insider C calculates I_c=H(ID_c) and q=gcd(I_c, u_c), and computes w=I_c/q, z=u_c/q, and j=v_c/q, where v_c is C's private key. Hence, gcd(w, z)=1. Then, he can use the extended Euclid's algorithm to find α and β both satisfying that

$\alpha \cdot w+\beta \cdot z=1$. As a result, he can obtain both x_s and k_c, since $v_c=1 \cdot j_c \cdot q_c=(\alpha \cdot w+\beta \cdot z) \cdot j_c \cdot q_c =(\alpha \cdot I_c/q+\beta \cdot u_c/q) \cdot j \cdot q=(\alpha \cdot I_c+\beta \cdot u_c) \cdot j=I_c \cdot (\alpha \cdot j)+(\beta \cdot j) \cdot u_c$ and $v_c=I_c \cdot k_c+x_s \cdot u_c$, where x_s is KGC's secret key and k_c is a random number selected by KGC satisfying $u_c=g^{k_c}$. More clearly, the value x_s he obtains is equal to $\beta \cdot j$.

After obtaining x_s, C can deduce any user's private key in the same manner. As an example, in the following, we demonstrate how C can deduces i's private key, k_i. C calculates I_i=H(ID_i) and q_i=gcd(I_i, u_i), computes $w_i=I_i/q_i$ and $z_i=u_i/q_i$, and then uses the extended Euclid's algorithm to compute γ and ε satisfying that $\gamma \cdot w_i+\varepsilon \cdot z_i=1$. Finally, since $v_i=1 \cdot j_i \cdot q_i =(\gamma \cdot w_i+\varepsilon \cdot z_i) \cdot j_i \cdot q_i =(\gamma \cdot I_i/q_i+\varepsilon \cdot u_i/q_i) \cdot j_i \cdot q_i =(\gamma \cdot I_i+\varepsilon \cdot u_i) \cdot j_i =I_i \cdot (\gamma \cdot j_i)+(\varepsilon \cdot j_i) \cdot u_i$ and $v_i =I_i \cdot k_i+x_s \cdot u_i$, he can calculate $j_i=x_s/\varepsilon$ and thus obtains i's private key by computing $v_i=j_i \cdot q_i$. With the knowledge of i's private key, insider C can impersonate user i to communicate with any other legal user. That is, to a minimum, an insider attack exists.

Review of the second protocol

Hölbl *et al.*'s second protocol consists of three phases: the system setup phase, the private key extraction phase, and the key agreement phase.

The system setup phase of this protocol is the same as the one in the first protocol.

In the private key extraction phase, with each user having his identity *ID*, KGC selects a random number k_i, and calculates i's private key $v_i=k_i+x_s \cdot$H(ID_i, u_i) and public key $u_i=g^{k_i}$.

In the key agreement phase, user A chooses a random number r_a, computes $t_a= g^{r_a}$, and then sends { u_a, t_a, ID_a } to user B. After receiving { u_a, t_a, ID_a }, B chooses a random number r_b , calculates $t_b= g^{r_b}$, and then sends { u_b, t_b, ID_b } to A. Finally, A and B can compute their common session key, $K= (u_b \cdot y_s^{\mathrm{H}(ID_b,u_b)} \cdot t_b)^{(v_a+r_a)} = g^{(v_b+r_b) \cdot (v_a+r_a)}$ and $K= (u_a \cdot y_s^{\mathrm{H}(ID_a,u_a)} \cdot t_a)^{(v_b+r_b)} = g^{(v_a+r_a) \cdot (v_b+r_b)}$, respectively.

Attack on the second protocol

Likewise, we can launch the same attack, as do in the first one, on this scheme. Since gcd(1, H(ID_c, u_c))=1, an insider C can use the extended Euclid's algorithm to find α and β both satisfying that $\alpha \cdot 1+\beta \cdot$H($ID_c$, u_c) =1. And since $v_c=k_c+x_s \cdot$H(ID_c, u_c) and 1=(k_c/v_c)$\cdot$1+(x_s/v_c)$\cdot$H(ID_c, u_c), he can obtain both x_s and k_c by letting $x_s=\beta \cdot v_c$ and $k_c=\alpha \cdot v_c$, where v_c is C's private key, x_s is KGC's secret key and k_c is a random number selected by KGC satisfying $u_c=g^{k_c}$. Consequently, similar to the result as shown in the attack of the first protocol, insider C can impersonate user i to communicate with any other legal user. That is, to the minimum, there exists an insider attack in their second scheme. Therefore, the protocol fails.

A.7 Why does Li *et al.*'s protocol [36] suffer from the smart-card-lost attack?

Review

We first briefly review the registration phase, the login phase and the authentication phase of protocol [36] and then present our attack.

In the registration phase, user C submits his ID_c, PW_c, and his personal biometric B_c to the server S. S issues C a smart card which stores the values of ID_c, f_c=H(B_c), and e_c=H(ID_c, x)⊕H(PW_c, f_c), where x is S's secret key.

In the login phase, user C inputs ID_c and PW_c to his smart card and inputs his personal biometric B_c on the specific device to check if H(B_c) is equal to f_c stored in the smart card. If it is, the card selects a random number R_c, computes $M_1= e_c \oplus H(PW_c, f_c)=H(ID_c, x)$, $M_2 = M_1 \oplus R_c$, and sends { ID_c, M_2 } to S.

In the authentication phase, after receiving { ID_c, M_2 }, S checks to see if ID_c is valid. If it is, S chooses a random number R_S, computes M_3=H(ID_c, x), $M_4= M_2 \oplus M_3= R_c$, $M_5 =M_3 \oplus R_S$, M_6=H(M_2, M_4), and sends { M_5, M_6} to C. After receiving S's message, C verifies whether M_6 is equal to H(M_2, R_c). If it is, S is authentic. C then computes $M_7=M_5 \oplus M_1=M_3 \oplus R_S \oplus M_1=H(ID_c, x) \oplus R_S \oplus H(ID_c, x)=R_S$, M_8=H(M_5, M_7), and sends {M_8} to S. After receiving C's message, S verifies whether M_8 is equal to H(M_5, R_s). If it is, C is authentic. S then accepts C's login request.

Attack

Assume that an attacker E gets C's smart card and reads the values of ID_c, f_c and e_c. He can successfully launch a password-guessing attack as shown below. E chooses a random number M_e and sends {ID_c, M_e} to S. After receiving the message, S checks to see if ID_c is valid. If it is, S chooses a random number R_S, computes M_3=H(ID_c, x), $M_4= M_e \oplus M_3$, $M_5= M_3 \oplus R_S$, M_6=H(M_e, M_4), and sends { M_5, M_6} to E. After receiving S's message, E terminates the communication, chooses a possible password PW', computes M'=H(M_e, $M_e \oplus e_c \oplus$H(PW', f_c)), and compares to see if M' is equal to M_6. If they are, PW' is the correct password, since $M_e \oplus e_c \oplus H(PW', f_c)=M_e \oplus H(ID_c, x) \oplus H(PW_c, f_c) \oplus H(PW', f_c)$. If $PW' =PW_c$, then the equation equals to $M_e \oplus$H(ID_c, x) which equals to $M_e \oplus M_3= M_4$. That is, M'=H(M_e, M_4)=M_6. E can therefore masquerade as C to log into the server. In other words, the attacker can successfully implement a smart-card-lost attack.

References

[1] H. Y. Chien, C. H. Chen, “A Remote Password Authentication Preserving User Anonymity,” *Proceedings of the 19th International Conference on Advanced Information Networking and Applications* (AINA ’05), Vol.2, pp. 245-248, March 2005.

[2] T. H. Chen, W. B. Lee, “A new method for using hash functions to solve remote user authentication”, Computers *& Electrical Engineering*, Vol. 34, No. 1, pp. 53-62, January 2008.

[3] G. Yang, D. S. Wong, H. Wang, X. Deng, “Two-factor mutual authentication based on smart cards and passwords”, *Journal of Computer and System Sciences*, Vol. 74, No. 7, pp.1160-1172, November 2008.

[4] M. Peyravian, C. Jeffries, “Secure remote user access over insecure networks”, *Computer Communications*, Vol. 29, No. 5, pp. 660-667, March 2006.

[5] W. S. Juang, S. T. Chen, H. T. Liaw, “Robust and efficient password-authenticated key agreement using smart cards”, *IEEE Transactions on Industrial Electronics*, Vol. 55, No. 6, pp. 2551-2556, June2008.

[6] T. Goriparthi, M. L. Das, A. Saxena, “An improved bilinear pairing based remote user authentication scheme”, *Computer Standards & Interfaces*, Vol. 31, No. 1, pp. 181-185, January 2009.

[7] C. S. Bindu, P. C. S. Reddy, B. Satyanarayana, “Improved remote user authentication scheme preserving user anonymity”, *International Journal of Computer Science and Network Security*, Vol. 8, No. 3, pp. 62-65, March 2008.

[8] W. S. Juang, W. K. Nien, “Efficient password authenticated key agreement using bilinear pairings”, *Mathematical and Computer Modelling*, Vol. 47, No. 11-12, pp. 1238-1245, June 2008.

[9] I. E. Liao, C. C. Lee, M. S. Hwang, “A password authentication scheme over insecure networks”, *Journal of Computer and System Sciences*, Vol. 72, No. 4, pp. 727-740, June 2006.

[10] J. Y. Liu, A. M. Zhou, M. X. Gao, “A new mutual authentication scheme based on nonce and smart cards”, *Computer Communications*, Vol. 31, No. 10, pp. 2205-2209, June 2008.

[11] H. S. Rhee, J. O. Kwon, D. H. Lee, “A remote user authentication scheme without using smart cards”, *Computer Standards & Interfaces*, Vol. 31, No. 1, pp. 6-13, January 2009.

[12] M. Hölbl, T. Welzer, B. Brumen, “Improvement of the Peyravian-Jeffries’s user authentication protocol and password change protocol”, *Computer Communications*, Vol. 31, No. 10, pp. 1945-1951, June 2008.

[13] J. S. Chou, M. D. Yang, G. C. Lee, “Cryptanalysis and improvement of Yang-Wang password authentication schemes”, http://eprint.iacr.org/2005/466,

December 2005.

[14] T. Xiang, K. Wong, X. Liao, “Cryptanalysis of a password authentication scheme over insecure networks”, *Computer and System Sciences*, Vol. 74, No. 5, pp. 657-661, August 2008.

[15] Y. Lee, J. Nam, D. Won, “Vulnerabilities in a remote agent authentication scheme using smart cards”, *LNCS: AMSTA*, Vol. 4953, pp. 850-857, April 2008.

[16] D. Z. Sun, J. P. Huai, J. Z. Sun, J. X. Li, “Cryptanalysis of a mutual authentication scheme based on nonce and smart cards”, *Computer Communications*, Vol. 32, No. 6, pp. 1015-1017, April 2009.

[17] Y. Chen, H. M. Sun, C. H. Huang, J. S. Chou, “Comments on two password based protocols”, *http://eprint.iacr.org/2008/400*, September 2008.

[18] J. Munilla, A. Peinado, “Security flaw of Hölbl *et al.*’s protocol”, *Computer Communications*, Vol. 32, No. 4, pp.736-739, March 2009.

[19] J. L. Tsai, “Impersonation attacks on Rhee et al.’s authentication scheme”, *http://dtim.mis.hfu.edu.tw/2008/paper/C044.pdf*, June 2008.

[20] H. Guo, Z. Li, Y. Mu, X. Zhang, “Cryptanalysis of simple three-party key exchange protocol”, *Computers & Security*, Vol. 27, No. 1-2, pp. 16-21, March 2008.

[21] H. B. Chen, T. H. Chen, W. B. Lee, C. C. Chang, “Security enhancement for a three-party encrypted key exchange protocol against undetectable on-line password guessing attacks”, *Computer Standards & Interfaces*, Vol. 30, No. 1-2, pp. 95-99, January 2008.

[22] E. J. Yoon, K. Y. Yoo, “Improving the novel three-party encrypted key exchange protocol”, *Computer Standards & Interfaces*, Vol. 30, No. 5, pp. 309-314, July 2008.

[23] J. Nam, Y. Lee, S. Kim, D. Won, “Security weakness in a three-party pairing-based protocol for password authenticated key exchange”, *Information Sciences*, Vol. 177, No. 6, pp. 1364-1375, March 2007.

[24] H. R. Chung, W. C. Ku, “Three weaknesses in a simple three-party key exchange protocol”, *Information Sciences*, Vol. 178, No. 1-2, pp. 220-229, January 2008.

[25] R. C. Phan, W. C. Yau, B. M. Goi, “Cryptanalysis of simple three-party key exchange protocol (S-3PAKE)”, *Information Sciences*, Vol. 178, No. 13, pp. 2849-2856, July 2008.

[26] C. C. Chang, J. S. Lee, T. F. Cheng, “Security design for three-party encrypted key exchange protocol using smart cards”, *ACM ISBN: 978-1-59593-993-7*, pp. 329-333, 2008.

[27] 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.

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

[29] H.C. Hsiang, W.K. Shih, “Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment”, *Computer Standards & Interfaces*, Vol. 31, No. 6, pp. 1118-1123, November 2009.

[30] S. K. Kim , M. G. Chung, “More secure remote user authentication scheme”, *Computer Communications*, Vol. 32, No. 6, pp. 1018-1021, April 2009.

[31] Y. Y. Wang, J. Y. Liu, F. X. Xiao, J. Dan, “A more efficient and secure dynamic ID-based remote user authentication scheme”, *Computer Communications*, Vol. 32, No. 4, pp. 583-585, March 2009.

[32] H. C. Hsiang, W. K. Shih, “Weaknesses and improvements of the Yoon–Ryu–Yoo remote user authentication scheme using smart cards”, *Computer Communications*, Vol. 32, No. 4, pp. 649-652, March 2009.

[33] H. R. Chung, W. C. Ku, M. J. Tsaur, “Weaknesses and improvement of Wang et al.'s remote user password authentication scheme for resource-limited environments”, *Computer Standards & Interfaces*, Vol. 31, No. 4, pp. 863-868, June 2009.

[34] J. Xu, W. T. Zhu, D. G. Feng, “An improved smart card based password authentication scheme with provable security”, *Computer Standards & Interfaces*, Vol. 31, No. 4, pp. 723-728, June 2009.

[35] M. S. Hwang, S. K. Chong, T. Y. Chen, “DoS-resistant ID-based password authentication scheme using smart cards”, *Journal of Systems and Software*, In Press, Available online 12 August 2009.

[36] C. T. Li, M. S. Hwang, “An efficient biometrics-based remote user authentication scheme using smart cards”, *Journal of Network and Computer Applications*, Vol. 33, No. 1, pp. 1-5, January 2010.

[37] M. Hölbl, T. Welzer, “Two improved two-party identity-based authenticated key agreement protocols”, *Computer Standards & Interfaces*, Vol. 31, No. 6, pp. 1056-1060, November 2009.

[38] J. H. Yang, C. C. Chang, “An ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem”, *Computers & Security*, Vol. 28, No. 3-4, pp. 138-143, May-June 2009.