A Secure DoS-resistant User Authenticated Key Agreement Scheme with Perfect Secrecies

Jeng-Ping Lin¹, Jih-Ming Fu²

¹Department of Information Network Technology Chihlee Institute of Technology, Niaosong, Taiwan 833, R.O. China. Email: jplin@mail.chihlee.edu.tw
²Department of Computer Science & Information Engineering, Cheng Shiu University, Niaosong, Taiwan 833, R.O. China.

Received February 2, 2008

Abstract

The goal of a denial-of-service (DoS) attack is to deplete the resource of a targeted server in order that its intended clients cannot obtain the services. Recently, Hwang et al. proposed an ID-based password authentication scheme using smart cards against the DoS attack. In their scheme, the major merits include: (1) mutual authentication; (2) the password guessing attack; (3) the replay attack; (4) the impersonation attack; (5) session key establishment; and (6) the server resources exhaustion attack. However, two basic and the most important security properties of a session key establishment are not satisfied in their scheme. One is the perfect forward secrecy. If the long-term secret key is compromised, the previous session key should not be derived. The other is the perfect backward secrecy. If a used session key is compromised, subsequent communications should not be damaged. The intentions of this paper are to show that the above weaknesses exist in Hwang et al.’s scheme and to propose a security-enhanced user authentication scheme. The proposed scheme not only can achieve the above admired security requirements, but also can solve the smart card loss problem which is a troublesome security threat in our life and cannot be solved in most authentication and key agreement schemes. [Life Science Journal. 2010;7(1): 88 – 94] (ISSN: 1097 – 8135).

Keywords Authentication; Client puzzles; Perfect forward secrecy; Perfect backward secrecy

1 INTRODUCTION

In today, most on-line services over the Internet are based on the client/server architecture. In the architecture, there is a single server to serve a lot of clients. Authentication is basic and is the first step to identify whether a remote client is authorized or unauthorized. After the verification of the identity, the client can be held accountable and the system can decide to give a specific access privilege. Moreover, the system generates a session key to protect their future communications [1-5] [Bellar and Rogaway, 1993; Juang, Feb. 2004;Juang, March 2004;Juang, May 2004;Juang, 2006].

Password is widely adopted into authentication and session key generation schemes since a password-based scheme is easily implemented for many applications. Relatively, the entropy of a memorial password is low and is easily suffered from the guessing attack. Therefore, many password-based authentication schemes with the key agreement scheme were proposed to provide robust security requirements [6, 7] [Juang, 2008; Wen et al., 2005].

Owing to the openness of the Internet, a goal of malicious attackers is to make that the service from the remote server is unavailable. One of the tricks is that the attackers can launch a denial of service (DoS) attack or a distributed denial of service (DDoS) attack to deplete the resource of the remote server by sending a huge service requests[8,9] [Agah and Das, 2007;Peng et al., 2007]. Hence, the DoS and DDoS attacks should be taken into consideration in the design of a secure user authentication scheme.

By sending a large connection requests to a targeted victim, the attack will cause that the server exhausts the resource to reply the response due to the innateness of the TCP/IP protocol principle. As we know, the DoS or DDoS attacks are easily implemented, but the attacks are hard to be prevented for the server. In general, the defense mechanisms of the DoS/DDoS attacks can be divided into four types [9] [Peng et al., 2007]: attack prevention, attack detection, attack source identification, and attack reaction. Most previous schemes addressed the works in the network layer and tried to analyze the information of incoming and outgoing packets. The major ideas are to install firewall, intrusion detection system, and intrusion prevention system on the entrance of systems.

Recently, the idea of adopting a puzzle game is paid more attention for defeating the DoS/DDoS attacks [10, 11][Aura et al., 2001;Bocan, 2004]. The intention of the idea is to prevent the resources of the server from being exhausted and the sincerity of the client has been shown to the server by performing some expensive cryptographic operations. The goal is to design an acceptable solution of a puzzle for legal clients, but the computation cost is high for malicious outsiders. In general, a puzzle is designed that the challenge is to seek out the miss materials of a hashed value [12, 13] [Juels and Brainard, 1999; Laurens et al., 2006]. For instance, $z$ is a digest value of two variables $x$ and $y$. Given $z$ and $y$, the goal is to seek out $x'$ to satisfy $z = h(x', y)$. As we know, if $x$ and $y$ are known, it is computationally fast for the server from computing the digest value of $x$ and $y$. Without the knowledge of $x$ and $y$, the computation cost is heavy for the client. The computation cost is disequilibrium between the client and the server because the client could only perform the brute-force search to
seek out the solution of a puzzle.

In 2009, Hwang et al. proposed a password-based user authentication scheme with session key establishment against the server resource exhaustion attacks and some well-known attacks [14] [Hwang et al., 2010]. Unfortunately, two basic and important security properties of a secure key establishment scheme are not taken into their consideration and we introduce them as follows:

1) **Perfect Forward Secrecy.** A key establishment scheme is said to provide the perfect forward secrecy if the compromise of long-term keys for communicated parties cannot damage past session keys.

The idea of the perfect forward secrecy is that previous traffics can be locked securely in the past. A widely adopted method is to employ the concept of Diffie-Hellman key agreement to generate distinct session keys, wherein the exponentials are chosen randomly as short-term keys. If long-term secret keys are compromised, previous sessions are not affected by an active adversary [15] [Schneier, 1996]. An admired key agreement should provide this property.

2) **Perfect Backward Secrecy (Known-key Attack).** A key establishment scheme is said to be secure against a known-key attack if the compromise of past session keys cannot allow that either a passive adversary learns the future session keys, or an active adversary impersonates one of the communicated parties successfully in the future.

The perfect backward secrecy on a key establishment scheme is analogous to the known-plaintext attack [16, 17] [Minier et al., 2009; van Oorschot and Wiener, 1991] on an encryption algorithm. Firstly, from implementation and engineering decisions point of view, scholars consider that, the probability of the compromise of session keys which were established previously may be larger than that of long-term keys. Secondly, in terms of cryptographic techniques, if a key establishment scheme only took moderate strength into consideration, past session key may be recovered over time. Finally, for some reasons of applications, it is necessary that past session keys may be deliberately unsecured. A secure key agreement should be against this threat.

Another serious security threat is also not taken into consideration in most smart card-based authentication schemes. In a real life, we always worry about the damage of smart cards loss. In 1998 and 2002, Kocher et al., [18] [Kocher et al., 1999] and Messerges et al. [19][Messerges et al., 2002] stated that this security threat happened by monitoring the power consumption and analyzing the leaked information in the smart card. A secure and admired smart card-based authentication scheme should blockade this threat.

In this paper, we propose a user authentication with key agreement scheme where the perfect forward secrecy and the perfect backward secrecy can be satisfied at the same time and the merits in Hwang et al.’s scheme are also taken into our consideration. Apart from that, our proposed scheme also can be secure against the smart card loss threat. Most smart card-based schemes cannot solve this threat. It implies that if previous schemes want to withstand this threat, their schemes must rely on a tamper-resistant smart card [20] [Nordin, 2004]. As we know, in a tamper-resistant smart card-based scheme, the system cost is high.

In the next section, we first review Hwang et al.’s scheme and show their weakness. In Section 3, we present our method. In Section 4, we analyze the security of the proposed scheme and compare the satisfaction of some security criteria between our scheme and Hwang et al.’s scheme. Finally, we conclude this paper in Section 5.

I. Hwang et al.’s Scheme

In this section, we briefly review Hwang et al.’s scheme [14][Hwang et al., 2010]. Before we introduce the scheme, we first notify the used parameters as follows.

A. Notations

1) \( v_i \) is a solution of the puzzle which is decided by the server S.

2) \( N_i \) and \( N_i \) denote the nonces and are generated by the server and the smart card, respectively.

3) \( q_i \) is a session is chosen by the smart card.

4) \( h() \) is a 128bits one-way hash function.

5) \( SK \) is the secret key of the server.

6) \( sk_i \) is also a secret key of the server and is used for puzzle verification.

7) \( \text{puzzle}(p, x_1, x_2, ..., x_n) \) denotes that given \( (p, x_1, x_2, ..., x_n) \) to find \( v \) such that \( h(x_1, x_2, ..., x_n, v) = p. \)

B. Registration Phase

Client \( U_i \) sends the identity \( ID_i \) and the chosen password \( PW_i \) for registration. Upon receiving the request, the server generates a smart card’s identifier \( C1ID_i \) and calculates \( S_i = ID_i^{SK} \mod n \), \( h_i = g^{PW_i^{SK}} \mod n \), and \( W_i = h(ID_i, SK) \) where \( n \) is a large prime number and \( g \) is a generator of \( Z_n^* \). The server stores \( (n, g, ID_i, C1ID_i, S_i, h_i, W_i) \) into a smart card and issues it back to the client. The phase is finished through a secure channel and the smart card adopted a fingerprint technology to verify the fingerprint of the client.

C. Login Phase

Client \( U_i \) enters the password \( PW_i \) and imprints the personalized fingerprint through a fingerprint input device. If it succeeds, the card performs the following steps:

1) The card extracts the content \( (ID_i, C1ID_i) \), generates a random nonce \( N_i \) and forwards them to the server as its login request.

2) Upon receiving the request \( (ID_i, C1ID_i, N_i) \), the server determines a puzzle solution \( v_i \) and calculates \( p = h(ID_i, N_i, v_i) \) and \( token_i = h(p, ID_i, N_i, N_i, v_i, sk_i) \). The server sends \( (p, N_i, token_i) \) back to the card.

http://www.sciencepub.net
3) The cards tries to seek out the solution \( v_i \) to satisfy 
\[ h(D_i, N_i, N_{\text{v}}) = p. \]
It should apply a brute-force method to find of the solution without 
the knowledge of the server. After the solution is 
found, the card calculates 
\[ X_i = g^{\ast + pW_i} \mod n, \]
\[ Y_i = S_i \ast h_i \ast \text{token}_i \mod n, Z_i = W_i \oplus q_i, \text{and } T_i = h(X_i, \]
\[ Y_i, \text{token}_i, q_i), \] where \( q_i \) is a chosen session key for 
future communications. The card sends \((D_i, X_i, Y_i, \]
\[ Z_i, T_i, v_i, N_i, N_{\text{v}}) \) to the server.
4) The server checks the validity of \((D_i, N_i, N_{\text{v}})\) and 
verifies whether \text{token}_i is equal to \( h(p, D_i, N_i, N_{\text{v}}, \)
\[ s_{\text{new}}) \) for proving the solution of the puzzle. If the 
above verification holds, the server extracts \( q_i = Z_i \oplus \]
h(D_i, SK) and verifies whether \( T_i \) is the same as \( h(X_i, \)
\[ Y_i, \text{token}_i, q_i). \] If it is also true, the server checks 
whether \( Y_i^{SK^{-1}} \) is equal to \( D_i \ast X_i \ast \text{token}_i \mod n \). If the 
verification is also correct, the server sets \( q_i \) as the 
session key and sends \( h(q_i) \) back to the card for the 
mutual authentication.
5) Similarly, the card verifies the correctness of \( h(q_i). \) 
If it is true, the card sets \( q_i \) as the session key.

D. Perfect Forward Secrecy

Since the communications \((D_i, X_i, Y_i, Z_i, T_i, v_i, N_i, N_{\text{v}})\) 
are always eavesdropped from outsiders, if the 
attacker compromises the long-term secret key \( SK \), all
the session keys can be derived.
1) The attack can construct all the secret keys of clients, 
\[ W_i = h(D_i, SK) \]
2) Then the attacker can derive all the session keys \( q_i = \]
\[ Z_i \oplus W_i. \]

E. Perfect Backward Secrecy

Similarly, the communications \((D_i, X_i, Y_i, Z_i, T_i, v_i, N_i, N_{\text{v}})\) 
are always eavesdropped from outsiders, if the 
attacker compromises a used session key \( q_i \), we can show 
that the attacker can impersonate the client \( U_i \) to 
communicate with the server.
1) Firstly, the attacker employs \( q_i \) to extract the long-
term secret key \( W_i = h(D_i, SK) = Z_i \oplus q_i. \)
2) Now, the attacker sends a login request \((D_i, C_iD_i, \)
\[ N_i) \) to the server. Without loss of generality, the 
server will return \((p_{\text{new}}, s_{\text{new}}, \text{token}_{\text{new}}) \) back to the 
attacker.
3) The goal of the attacker is to forge \((D_i, X_{\text{new}}, Y_{\text{new}}, \)
\[ Z_{\text{new}}, T_{\text{new}}, v_{\text{new}}, N_{\text{new}}, N_{\text{v}}) \) for passing the 
verifications of the servers.
I. For simplicity, we assume that the attacker has 
unlocked the solution \( v_{\text{v}} \) of the puzzle \( p_{\text{new}}. \)
II. Find an integer \( a \) to satisfy \( a \ast \text{token}_{\text{new}} = \text{token}_i \mod n, \) 
where \( a = \text{token}_i \ast \text{token}_{\text{new}}^{-1} \mod n. \)
III. Select a new session key \( q_{\text{new}}. \)
IV. Calculate \( X_{\text{new}} = X_i \ast a \mod n, Z_{\text{new}} = W_i \oplus q_{\text{new}}, \text{and } T_{\text{new}} = h(X_{\text{new}}, Y_{\text{new}}, \text{token}_{\text{new}}, \text{q}_{\text{new}}) \)
V. Send \((D_i, X_{\text{new}}, Y_{\text{new}}, Z_{\text{new}}, T_{\text{new}}, v_i, N_i, N_{\text{new}}) \)
4) Without loss of generality, the server will verify:
I. Check the validity of \((D_i, N_i, N_{\text{v}})\);
II. Verifies whether \text{token}_{\text{new}} is equal to \( h(p, D_i, N_i, N_{\text{v}}, \)
\[ s_{\text{new}}, v_{\text{new}}, s_{\text{v}}) \) for proving the solution of the 
puzzle.

III. Extract \( q_{\text{new}} = Z_{\text{new}} \oplus h(D_i, SK) \)
IV. Verify whether \( T_{\text{new}} = h(X_{\text{new}}, Z_{\text{new}} \oplus \text{token}_{\text{new}}, q_{\text{new}}). \) If it is also true, the server checks 
whether \( Y_{\text{new}} = x_{\text{new}}^{SK^{-1}} \) is equal to \( D_i \ast X_i \ast \text{token}_{\text{new}} \mod n. \)
If the verification is also correct, the server sets \( q_{\text{new}} \) as the 
session key and sends \( h(q_{\text{new}}) \) back to the card for the mutual 
authentication.

V. The forged request will pass the verification of 
the server and the server will believe the session key is \( q_{\text{new}}. \)

5) Correctness.
(1) \[ Y_{\text{new}}^{SK^{-1}} = Y_i^{SK^{-1}} = (S_i \ast h_i \ast \text{token}_i)^{-1} \mod n \]
\[ = (D_i \ast g^{pW_i \ast \text{token}_i}) \mod n \]
(2) \[ (D_i \ast X_{\text{token}_{\text{new}}}) = (D_i \ast X_i \ast \text{token}_{\text{new}}) \mod n \]
\[ = (D_i \ast g^{pW_i \ast \text{token}_i}) \mod n \]

II. Our Scheme

In this section, we propose a novel user 
authentication scheme with key agreement. The 
proposed scheme not only can keep the same merits of 
Hwang et al.’s scheme, but also can add more admired 
security properties. The used parameters are the same 
Hwang et al.’s scheme.

A. Registration Phase

Client \( U_i \) sends the identity \( ID_i \) and the chosen 
password \( PW_i \) for registration. Upon receiving the 
request, the server generates a smart card’s identifier 
\( C_iD_i \) and calculates \( S_i’ = (ID_i^{SK} \mod n) \oplus h(PW_i), h_i’ \]
\[ = (g^{PW_i \ast SK} \mod n) \oplus h(PW_i), \) and \( W_i’ = h(D_i, SK) \oplus \]
h(PW_i). The server stores \( n, g, ID_i, C_iD_i, S_i’, h_i’, W_i’ \) into 
a smart card and issues it back to the user. The phase is 
finished through a secure channel.

B. Login Phase

User \( U_i \) enters the password \( PW_i \) into a card reader. 
Then the smart card performs the following steps to 
achieve the mutual authentication with the server.
1) The card extracts the content \((D_i, C_iD_i, S_i = S_i’ \oplus \)
h(PW_i), \( h_i = h_i’ \oplus h(PW_i), W_i = W_i’ \oplus h(PW_i)), \)
generates a random number \( N_i \) and 
calculates \( g^{N_i \ast PW_i} \mod n. \) Then the card forwards 
\((D_i, C_iD_i, g^{N_i \ast PW_i} \mod n) \) to the server as its login 
request.
2) Upon receiving the request, the server determines a 
puzzle solution \( v_i \) and calculates \( g^{N_i} \mod n \) and \( p = \]
h(D_i, \( g^{N_i \ast PW_i} \mod n, g^{N_i} \mod n, v_i.) \) The server 
also calculates \( p = h(D_i, g^{N_i \ast PW_i} \mod n, \)
\( g^{N_i} \mod n) \oplus h(D_i, SK), v_i) \) and \text{token}_i = h(p, D_i, \)
\( g^{N_i \ast PW_i} \mod n, g^{N_i} \mod n) \oplus h(D_i, SK), v_i, s_{\text{v}}) \) 
and sends \( p, (g^{N_i} \mod n) \oplus h(D_i, SK), \text{token}_i) \)
back to the card.

3) The cards employs $W_i$ to extract $g^{N_i} \pmod n$ and tries to seek out the solution $v_i$ to satisfy $h(D_i, g^{N_i+PW_i} \pmod n, (g^{N_i} \pmod n) \oplus h(D_i, SK), v_i) = p_i$. It should apply a brute-force method to find of the solution without the knowledge of the solution. After the solution is found, the card calculates $Y_i = S_i + h_{N_i, token_i} \pmod n$, $Sess = g^{N_i+PW_i} \pmod n$, and $T_i = h(Y_i, token_i, Sess, v_i)$. The card sends $(ID_i, token_i, Y_i, T_i)$ to the server.

4) The server checks the validity of $ID_i$ and verifies whether $token_i$ is equal to $h(p, ID_i, g^{N_i+PW_i} \pmod n, g^{N_i} \pmod n \ll 0, SK_i)$ for proving the solution of the puzzle. If the above verification holds, the server verifies whether $S_i^{SK_i^{-1}}$ is equal to $ID_i \ast g^{N_i+PW_i \ast token_i} \pmod n$. If it holds, the server calculates $Sess = g^{N_i+PW_i \ast N_i} \pmod n$ and verifies whether $T_i$ is the same as $h(Y_i, token_i, Sess, v_i)$. If all of the conditions are held, the server authenticate the identity of the user and sets the session key $SK_{US} = h(Sess)$ as their session key. The server sends $h(Y_i, token_i, Sess + 1, v_i)$ back to the card.

5) Similarly, the card verifies the correctness of $h(Y_i, token_i, Sess + 1, v_i)$. If it true, the card also sets the session key $SK_{US} = h(Sess)$ as their session key. We use Figure 1 to introduce our scheme.

### Figure 1. The Proposed Scheme

#### U_i (Smart Card)

1. Enter the password $PW_i$

   Extract $(ID_i, CID_i, S_i) = S_i \oplus h(PW_i)$,

   $h_i = h_i \oplus h(PW_i)$, $W_i = W_i \oplus h(PW_i)$

   Generate $N_i$

   Calculate $g^{N_i \ast PW_i} \pmod n$ ($ID_i, CID_i, g^{N_i+PW_i} \pmod n$)

2. Determine a puzzle solution $v_i$

   Generate $N_i$

   Calculate $h(ID_i, SK), (g^{N_i} \pmod n) \oplus h(ID_i, SK), v_i$.

   $p = h(ID_i, g^{N_i+PW_i} \pmod n, (g^{N_i} \pmod n) \oplus h(ID_i, SK), v_i)$

   $v_i = h(p, g^{N_i+PW_i} \pmod n, (g^{N_i} \pmod n) \oplus h(ID_i, SK), v_i, sk_i)$

3. Extract $g^{N_i} \pmod n$

   Seek out the solution $v_i$

   Use $p$ to verify $v_i$.

   Calculate $Y_i = S_i + h_{N_i, token_i} \pmod n$.$Sess = g^{N_i+PW_i} \pmod n$.

   and $T_i = h(Y_i, token_i, Sess, v_i)$

4. Verify the validity of $ID_i$

   Verify $token_i \ast h(p, g^{N_i+PW_i} \pmod n, (g^{N_i} \pmod n) \oplus h(ID_i, SK), v_i, sk_i)$

   $Y_i = h(ID_i, g^{N_i+PW_i} \pmod n, (g^{N_i} \pmod n) \oplus h(ID_i, SK), v_i, sk_i)$

   Calculate $Sess = g^{N_i+PW_i \ast N_i} \pmod n$

   Verify $T_i \oplus h(Y_i, token_i, Sess, v_i)$

#### Server

4. $U_i$ will verify whether the received hashed value is correct or not. If it holds, $U_i$ will believe that $(g^{N_i} \pmod n) \oplus h(ID, SK)$ is generated by $S$ and believe $U_i \leftarrow SK_{US} \rightarrow S$.

#### III. DISCUSSIONS

##### A. Security Analysis

We analyze that the proposed scheme is secure against some well-known security threats.

1. **Mutual Authentication.** The goal of the mutual authentication is to establish an agreed session key $SK_{US}$ between $U_i$ and the server. Let $U_i \leftarrow SK_{US} \rightarrow S$ denote that $U_i$ shares a secret key $SK_{US}$ with the server $S$. The mutual authentication is complete between $U_i$ and $S$ if there is a session key $SK_{US}$ such that $U_i$ believes $U_i \leftarrow SK_{US} \rightarrow S$, and $S$ also believes $U_i \leftarrow SK_{US} \rightarrow S$. A strong mutual authentication may lead to the following statement [21][Burrows et al., 1990]:

   I. $U_i$ believes that $S$ believes $U_i \leftarrow SK_{US} \rightarrow S$, and

   II. $S$ believes that $U_i$ belives $U_i \leftarrow SK_{US} \rightarrow S$.

   $U_i$ and $S$ can do mutual authentication in the login phase.

   I. Upon receiving $h(Y_i, token_i, Sess + 1, v_i)$ in Step

   4, $U_i$ will verify whether the received hashed value is correct or not. If it holds, $U_i$ will believe that $(g^{N_i} \pmod n) \oplus h(ID, SK)$ is generated by $S$ and believe $U_i \leftarrow SK_{US} \rightarrow S$.

   II. Since $N_i$ is generated by $U_i$, $U_i$ believes $N_i$ is fresh and believes that $S$ believes $U_i \leftarrow SK_{US} \rightarrow S$.

   III. Using the same way, upon receiving $(ID_i, token_i, Y_i, T_i)$ in Step 3, $S$ will verify the validity of $Y_i$ and $T_i$. If all the conditions hold, $S$ believes that $g^{N_i+PW_i} \pmod n$ is generated by $U_i$ and believe $U_i \leftarrow SK_{US} \rightarrow S$.

   IV. Since $N_i$ is generated by $S$, $S$ believes $N_i$ is fresh and believes that $U_i$ believes $U_i \leftarrow SK_{US} \rightarrow S$.

2. **The Replay Attack.** The attack could be classified into two categories. Firstly, if the attacker re-submits a used message $(ID_i, CID_i, g^{N_i+PW_i} \pmod n)$ to the server as a new login request. Without loss of generality, the server responses $p_{new}$,
The attacker cannot retrieve \((g^{N_{sk}} \mod n)\) without \(W_s\). It implies that the attacker has no ability to send the response to the server in Step 3. Secondly, if the attacker re-submits a used message \((p, (g^{N_i} \mod n) \oplus h(ID_i, SK), \text{token}_i)\) back to the card. The card believes that the received message is fresh. Based on the difficulty of the computational Diffie-Hellman problem, without the knowledge of \(N_i\), the attacker still has no ability to send the response \(h(Y_i, \text{token}_i, \text{Sess}+1, v_i)\) back in Step 4.

3) The Impersonation Attack without the Smart Card. Consider that the attacker can send a forged request \((ID_i, CID_i, g^{N_i} \mod n)\) and will get an honest response \((p, (g^{N_i} \mod n) \oplus h(ID_i, SK), \text{token}_i)\) back. Without the secret key \(W_s\), the attacker cannot retrieve \(g^{N_i} \mod n\). It implies that the attacker has no ability to forge a valid \(T_i = h(Y_i, \text{token}_i, \text{Sess}, v_i)\) to the server based on the difficulty of the computational Diffie-Hellman problem.

4) The Guessing Attack / Impersonation Attack with the Smart Card. Consider that the attacker has the ability to extract the content of the smart card, \((S_i = (ID_i, SK, g^{N_i} \mod n) \oplus h(PW_i), h_Y = (g^{PW_i \cdot SK} \mod n) \oplus h(PW_i), \) and \(W_s = h(ID_i, SK) \oplus h(PW_i)).\) Owing to the openness of the Internet, the attacker also can eavesdrop the communicated messages \((ID_i, CID_i, g^{N_i \cdot PW_i} \mod n, p), \text{token}_i = h(p, ID_i, g^{N_i \cdot PW_i} \mod n, (g^{N_i} \mod n) \oplus h(ID_i, SK), v_i, s_k, Y_i = S_i \cdot h^{N_i \cdot \text{token}_i} \mod n, T_i = h(Y_i, \text{token}_i, \text{Sess}, v_i), h(Y_i, \text{token}_i, \text{Sess}+1, v_i)).\)

I. Case 1. The attacker guesses the password \(PW_i\), and tries to verify whether the guessed value is correct on the eavesdropped messages. By the difficulty of the discrete logarithm and the computationally Diffie-Hellman problems and without the knowledge of \(SK\), it is hard for the attacker to launch the off-line guessing attack on the messages \((g^{N_i \cdot PW_i} \mod n, \text{token}_i, Y_i = S_i \cdot h^{N_i \cdot \text{token}_i} \mod n)).\)

II. Case 2. The attacker guesses the password \(PW_i\), and tries to verify whether the guessed value is correct on the response of the server.

1) The attacker calculates a forged request \((ID_i, CID_i, g^{N_i \cdot PW_i} \mod n)\) and sends it to the server.

1) Without loss of generality, the server sends \((p = h(ID_i, g^{N_i \cdot PW_i} \mod n, (g^{N_i} \mod n) \oplus h(ID_i, SK), v_i)\) and \(\text{token}_i = h(p, ID_i, g^{N_i \cdot PW_i} \mod n, (g^{N_i} \mod n) \oplus h(ID_i, SK), v_i, s_k, Y_i = S_i \cdot h^{N_i \cdot \text{token}_i} \mod n, T_i = h(Y_i, \text{token}_i, \text{Sess}, v_i))\) back to the card.

The Perfect Forward Secrecy. If the long-term secret key \(SK\) of the system is compromised, the session key is still secure in the proposed scheme. Following the scheme, the client sends \((g^{N_i \cdot PW_i} \mod n) \oplus h(ID_i, SK)\) back in Step 3. Then the client and the server can establish the same session key \(SK_{US} = h(Sess = g^{N_i \cdot PW_i} \mod n)\) based on the difficulty of the computational Diffie-Hellman problem and without the knowledge of the ephemeral keys \(N_i\) and \(N_i\), the attacker cannot derive the session key \(SK_{US}\).

6) The Perfect Backward Secrecy. Assume that a used session key \(SK_{US} = h(Sess = g^{N_i \cdot PW_i} \mod n)\) with the communicated messages \((ID_i, CID_i, g^{N_i \cdot PW_i} \mod n, p), \text{token}_i = h(p, ID_i, g^{N_i \cdot PW_i} \mod n, (g^{N_i} \mod n) \oplus h(ID_i, SK), v_i, s_k, Y_i = S_i \cdot h^{N_i \cdot \text{token}_i} \mod n, T_i = h(Y_i, \text{token}_i, \text{Sess}, v_i))\) are compromised by the attacker. The goal of the attacker is to do the following cases successfully.

I. Case 1. The attacker eavesdrops the communications and tries to compromise the future session keys. Since the ephemeral keys \(N_i\) and \(N_i\) are chosen randomly, based on the difficulty of the computationally Diffie-Hellman problem, it is infeasible to derive the session key \(SK_{US} = h(Sess = g^{N_i \cdot PW_i} \mod n)\).

II. Case 2. The attacker sends a forged login request \((ID_i, CID_i, g^{N_i} \mod n)\) to the server and gets a response \((p, (g^{N_i} \mod n) \oplus h(ID_i), \text{token}_i))\) back to the server.
function operation; multiplication; used to solve the puzzle. 

*: cost in client and server sides respectively.

Table II. To provide the perfect forward secrecy and the security criteria with the related schemes and show the

\[ B. \text{ Comparisons} \]

**1) Satisfaction of the Criteria**

In this subsection, we compare some admired security criteria with the related schemes and show the result in Table I.

**2) Computation Cost**

We denote that \( T_H \) is the time of one hash function operation; \( T_{MUL} \) is the time for one modular multiplication; \( T_{MOD} \) is the time for one exclusive OR operation; and \( T_{EXP} \) is the time for one modular exponentiation operation.

We show the comparison of the computation cost in Table II. To provide the perfect forward secrecy and the perfect backward secrecy properties and to solve the smart card loss problem, we add a slight computation cost in client and server sides respectively.

**Table I. Satisfaction of the security criteria between our scheme and the related schemes**

<table>
<thead>
<tr>
<th></th>
<th>Kim et al. [22][Kim et al., 2003]</th>
<th>Hwang et al. [14][Hwang et al., 2010]</th>
<th>Our Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual authentication</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>On-line password guessing attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Off-line password guessing attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message replay attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Impersonation attack</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Server resource exhaustion attack</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Session key establishment</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Perfect forward secrecy</td>
<td>Not supported</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Perfect backward secrecy</td>
<td>Not supported</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Security against the smart card loss problem</td>
<td>Not supported*</td>
<td>Not supported*</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*: even if the schemes follow our idea, the schemes still cannot achieve the same requirement.

**Table II. Comparison of the Computation Cost**

<table>
<thead>
<tr>
<th></th>
<th>Computation Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client Side</td>
</tr>
<tr>
<td><strong>Our Scheme</strong></td>
<td>((n+7)T_R + 4T_B + 5T_{MUL} + 3T_{EXP})</td>
</tr>
<tr>
<td><strong>Kim et al.'s scheme</strong> [22][Kim et al., 2003]**</td>
<td>(3T_{MUL} + 2T_{EXP})</td>
</tr>
<tr>
<td><strong>Hwang et al.'s scheme</strong> [14][Hwang et al., 2010]**</td>
<td>((n+3)T_R + 1T_B + 3T_{MUL} + 2T_{EXP})</td>
</tr>
</tbody>
</table>

*: \( n \) is the number of hash function operations that is used to solve the puzzle.

**Conclusions**

We have proposed a security enhanced password-based user authentication scheme with key agreement. By Tables I and II, the proposed scheme not only satisfies the same security criteria with Hwang et al.'s scheme, but also uses a slight computation cost to provide more admired security requirement such as the perfect forward secrecy, the perfect backward secrecy and the smart card loss problem.

**References**

[13] Laurens, V., Saddik, A.E., Nayak, A., Requirements for client puzzles to defeat the denial of service and the distributed denial of