Cryptanalysis and improvement of ‘an improved authentication with key agreement scheme on elliptic curve cryptosystem for global mobility networks’

Mohammad Sabzinejad Farash* †

Faculty of Mathematical Sciences and Computer, Kharazmi University, Tehran, Iran

SUMMARY

Authentication schemes assure that authorised user can fraudulently obtain his/her required services from home domains. Recently, Li et al. (International Journal of Network Management, 2013; 23(5):311–324) proposed a remote user authentication scheme. They claimed that their protocol is secure against known security attacks. However, in this paper, we indicate that Li et al.’s scheme is insecure against user impersonation attack. We show that an active adversary can easily masquerade as a legitimate user without knowing the user’s secret information. As a remedy, we also proposed an improved authentication scheme to overcome the security weaknesses of Li et al.’s scheme. To show the security of our scheme, we prove its security the random oracle model. The implementation results show that our improved scheme offers a reduction of 58% in computational cost and a communication cost reduction of 48% with respect to Li et al.’s scheme. Copyright © 2014 John Wiley & Sons, Ltd

Received 3 May 2014; Revised 30 August 2014; Accepted 5 November 2014

1. INTRODUCTION

Mobile communication recently has become more pervasive and gained popularity. People can roam freely and use mobile services almost everywhere. However, people are worried about two major security issues in the mobile communication: privacy and authentication. Privacy assures that the communication messages are not intercepted by an eavesdropper. On the other hand, authentication assures that any unauthorised user cannot fraudulently obtain his/her required services from home domains [1]. Thus, efficient and secure mechanisms for mobile clients are required in order to provide the authenticity and privacy in online transactions. Authenticated key exchange schemes can be used to achieve these security goals.

An authenticated key exchange scheme allows two or more parties to share a secret key to encrypt/decrypt the messages transmitted between the communication parties (e.g., [2–8]). Password-based authenticated key exchange (PAKE) protocol is a type of authenticated key exchange protocols that enables two or more communication entities, who only share a weak, low-entropy and easily memorable passwords, to authenticate each other and establish a high-entropy secret session key. Various PAKE schemes have been proposed to provide security in mobile communications for a decade [9–21]. However, most of them suffer from crucial security weaknesses and cannot provide the expected security goals.
One of the most important threats to the authentication scheme is masquerading or impersonation attack. In such attach, an active adversary tries to masquerade as a legitimate entity. Thus, an authentication scheme that is vulnerable to the impersonation attack will not be suitable for using in any security environment. For this reason, the aim of this paper is to analyse an authentication scheme to show its vulnerability to the impersonation attack and improve it.

1.1. Related works

In 2005, Yang et al. [22] proposed a PAKE scheme based on Diffie–Hellman key exchange protocol. However, Huang et al. [23] pointed out that the Yang et al.’s scheme may not be suitable for users with limited computational power and further proposed a new scheme. Jo et al. [24] demonstrated that the schemes by Yang et al. and Huang et al. are both vulnerable to off-line password guessing attack. Based on Yang et al.’s scheme, Durlanik and Sogukpinar [25] introduced an efficient authentication scheme by using Elliptic Curve Cryptography (ECC). Because of the adoption of elliptic curves, Durlanik and Sogukpinar’s scheme reduced the total execution time and the requirements for memory in comparison with Yang et al.’s scheme. However, Yoon and Yoo [26] indicated that Durlanik and Sogukpinar’s scheme still suffered from off-line password guessing and Denning–Sacco attacks, and projected an improved scheme to overcome the weaknesses. However, Liu and Koenig [27] demonstrated that Yoon and Yoo’s scheme still suffers from off-line password guessing and insider attacks.

In 2009, Tsai [28] proposed an efficient authentication protocol based on random nonce, in which one-way hash functions and exclusive-or operations were only utilised for computing all the communication messages. As a result, the computation cost was very low, and it was suitable for low computation equipments. However, it was still defenseless to off-line password guessing, Denning–Sacco and stolen-verifier attacks; furthermore, it did not provide any key agreement, known-key secrecy and perfect forward secrecy [29–31]. To deal with the problems, Arshad and Ikram [31] proposed an ECC-based authentication scheme. But, Tang and Liu [32] demonstrated the vulnerability of Arshad and Ikram’s scheme to off-line password guessing attack and introduced an improved scheme to overcome the weakness.

In 2010, Yoon et al. [33] also proposed an authentication scheme based on ECC to deal with the problems in Tsai et al.’s scheme. In 2012, Xie [34] pointed out that Yoo et al.’s scheme still suffers from stolen-verifier and off-line password guessing attacks and proposed an improved scheme. However, Farash and Attari [35] show that Xi’s scheme is also insecure and proposed an enhanced scheme. In 2013, Zhang et al. [36] proposed a new password-based authenticated protocol, but Tu et al. [37] found out that it is insecure against impersonation attacks. Tu et al. proposed an improved authentication protocol for session initiation protocol using smart card to overcome the security flaws of Zhang et al.’s protocol.

Recently, Li et al. [38] analysed and pointed out that user impersonation attack was effective in Rhee et al.’s scheme [39] because of the mathematical homomorphism of the registration information. Then, they proposed an improved remote user authentication scheme and claimed that their scheme is more secure, practical and robust. However, in this paper, we analyse the weaknesses in Li et al.’s scheme [38] and propose an improved scheme.

1.2. Contribution

The main contribution of this paper is to analyse and improve Li et al.’s remote user authentication scheme [38]. Li et al. proposed their scheme to enable a remote server to authenticate claimant users. However, we show that Li et al.’s scheme fails to achieve the secure authentication goal. We found out that an active attacker can easily masquerade as a legitimate user in Li et al.’s scheme. We prove that this vulnerability is due to wrong design of the login and the authentication phases. Thus, we improve these phases of Li et al.’s scheme to overcome the security weakness and achieve better functionality.
1.3. Outline

The rest of this paper is organised as follows. We review Li et al.’s scheme in Section 2. In Section 3, we propose the security weaknesses of Li et al.’s scheme. Our improved scheme and its security proof are proposed in Sections 4 and 5, respectively. A comparison between our improved scheme and Li et al.’s scheme is proposed in Section 6. Finally, we conclude our paper in Section 7.

2. REVIEW OF LI ET AL.’S PROTOCOL

In this section, we review Li et al.’s password-based authenticated key agreement protocol [38] using the notations shown in Table 1. This protocol has six phases: initialisation, registration, login, authentication with key agreement, mutual authentication and key confirmation, and secret update phases.

2.1. Initialisation phase

The server $S$ generates system parameters as follows:

1. A large prime number $p$;
2. An elliptic curve $E_p(a, b)$ over the finite field $\mathbb{F}_p$;
3. A base point $P$ over $E_p(a, b)$ of order $n$;
4. $S$’s private key $x_S \in \mathbb{Z}_n^*$; and
5. Secure one-way hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$.

Finally, $S$ keeps $x_S$ secret and publishes $\{E_p(a, b), P, n, H\}$.

2.2. Registration phase

In this phase, the client $U_i$ who wants to become a legal user of a remote server performs the following steps as shown in Figure 1:

1. A client $U_i$ chooses his/her valid identifier $ID_i$ with password $pw_i$ and then sends $(ID_i, pw_i)$ to $S$ over a secure channel.
2. Upon receiving the registration request message $(ID_i, pw_i)$ from $U_i$, $S$ computes $U_i$’s authentication information

$$Y_i = (Y_{i,1}, Y_{i,2}) = (ID_i r_i n_i x_S P + pw_i P, r_i P)$$

(1)

where $r_i$ is a random number only used once in this phase and $n_i$ is a large unique number generated randomly by $S$ for every user.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>A client</td>
</tr>
<tr>
<td>$S$</td>
<td>The server</td>
</tr>
<tr>
<td>ID</td>
<td>Client’s identity</td>
</tr>
<tr>
<td>pw</td>
<td>Client’s password</td>
</tr>
<tr>
<td>$p, n$</td>
<td>Two prime numbers</td>
</tr>
<tr>
<td>$F_p$</td>
<td>A finite field</td>
</tr>
<tr>
<td>$E_p(a, b)$</td>
<td>An elliptic curve over $F_p$</td>
</tr>
<tr>
<td>$P$</td>
<td>An element of $E$ with the prime order $n$</td>
</tr>
<tr>
<td>$\mathbb{Z}_n^*$</td>
<td>The non-zero integers modulus $n$</td>
</tr>
<tr>
<td>$H$</td>
<td>The hash function $h : {0, 1}^* \rightarrow {0, 1}^k$</td>
</tr>
<tr>
<td>$x_S$</td>
<td>The secret key of the server</td>
</tr>
</tbody>
</table>
3. $S$ sends $\{H(\cdot), n, E_p(a, b), P, Y_1\}$ to $U_i$ over a secure (or public) channel and stores $(ID_i, n_i)$ in its database privately.
4. Upon receiving the authentication information, $U_i$ stores it in his/her storage device and remembers his/her ID$_i$ with pw$_i$.

2.3. Login phase

$U_i$ can perform the following steps to log in to the authentication server as shown in Figure 1:

1. $U_i$ inputs his/her ID$_i$ with pw$_i$ into his/her device.
2. The device chooses temporary secret random numbers $a, b, c, d, k_1 \in \mathbb{Z}_n^*$. 

Figure 1. Li et al.’s scheme [38]
3. Compute

\begin{align*}
Y'_{i,1} &= Y_{i,1} - pw_i P = ID_i r_i n_i x_S P \\
C_1 &= a Y'_{i,1} = a(ID_i r_i n_i x_S P) \\
C_2 &= a Y_{i,2} = a r_i P \\
C_3 &= b Y_{i,2} = b r_i P \\
C_4 &= c Y_{i,2} = c r_i P \\
C_5 &= c Y'_{i,1} + k_1 P = c ID_i r_i n_i x_S P + k_1 P \\
C_6 &= d Y_{i,2} = d r_i P
\end{align*}

4. \( U_i \) sends to \( S \) the login request message \( M_1 = \{ID_i, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\} \).

2.4. Authentication with key agreement phase

Upon receiving the login request message \( M_1 = \{ID_i, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\} \), \( S \) performs the following steps as shown in Figure 1:

1. \( S \) firstly checks whether \( ID_i \) is valid in the registration table and extracts \( n_i \) corresponding to \( ID_i \) in its database and then verifies if the equation

\[ ID_i n_i x_S C_2 = ?C_1 \]  

holds. If it does not hold, \( S \) rejects \( U_i \); otherwise, \( S \) accepts \( U_i \)'s login request and performs the following step.

2. \( S \) computes \( k_1 P = C_5 - ID_i n_i x_S C_4 \). Then \( S \) randomly chooses \( k_2 \in Z_n^* \) and computes \( K = k_1 k_2 P \) and the session key \( sk = H(K_x) \), where \( K_x \) is the \( x \)-coordinate of the point \( K \).

3. \( S \) computes

\begin{align*}
C_7 &= ID_i n_i x_S C_3 \\
C_8 &= ID_i n_i x_S C_6 + k_2 P \\
C_9 &= E_{sk}(ID_i || m || S)
\end{align*}

where \( m \) is a session identifier.

4. Finally, \( S \) sends to \( U_i \) the message \( M_2 = \{C_7, C_8, C_9\} \) for mutual authentication and key confirmation.
2.5. Mutual authentication and key confirmation phase

Upon receiving the message $M_2$, $U_i$ performs the following steps as shown in Figure 1:

1. $U_i$ verifies whether the equation
   \[ bY_{i,1}' = C_7 \]  
   holds. If so, $U_i$ believes that the response of the message is correct from the responding server; otherwise, it rejects.
2. After the mutual authentication process, $U_i$ computes $k_2 P = C_8 - dY_{i,1}'$ and $K = k_1 k_2 P$, and obtains the session key $sk = H(K_x)$, where $K_x$ is the $x$-coordinate of the point $K$. $U_i$ can decrypt the message $C_9$ with $sk$ and confirm the session key if $S$ and ID$_i$ are correct.
3. $U_i$ computes $C_{10} = E_{sk} (S||m||ID_i)$ and sends the message $M_3 = \{C_{10}\}$ to $S$.
4. At the end of the session, $S$ verifies $C_{10}$ by checking the equation $C_{10} = E_{sk} (S||m||ID_i)$. If it holds, the scheme is finished successfully; otherwise, it terminates in failure.

2.6. Secret update phase

There are two secrets in Li et al.’s scheme: the password $pw_i$ of the client $U_i$ and the secret key $x_S$ of the authentication server. The secrets may need to be updated in the future for security.

2.6.1. Password update phase

The client $U_i$ could change his/her password off-line anytime and anywhere by computing

\[ Y_i^* = (Y_{i,1}', Y_{i,2}) = (Y_{i,1} - pw_i P + pw_i^* P, Y_{i,2}) \]

and replacing $Y_i$ by $Y_i^*$ with a new password $pw_i^*$.

2.6.2. Secret number update phase

The server $S$ could change its secret number $x_S$ online by interacting with its clients. Suppose that this phase is executed after the authentication with key agreement procedures and a secure channel based on the session key $sk$. Thus, they can communicate with each other securely using symmetric cryptography algorithm; that is, all of the following information is encrypted by $sk$ using the symmetric cryptography algorithm. $U_i$ sends the update request. Then $S$ computes the new values of $Y_{i,1}' = ID_r r_i^* n_x^* P + pw_i^* P$ and $Y_{i,2}' = r_i^* P$, and sends these new values to $U_i$. Finally, $U_i$ computes $Y_{i,1}' = Y_{i,1}' + pw_i P$ and replaces the original authentication information $Y_i = (Y_{i,1}, Y_{i,2})$ by $Y_i^* = (Y_{i,1}', Y_{i,2}')$.

3. CRYPTANALYSIS OF LI ET AL.’S SCHEME

In this section, we show that an active adversary can mount an impersonation attack on Li et al.’s protocol [38]. The key vulnerability of Li et al.’s scheme is due to this fact that the secret parameter $Y_{i,1}'$ is easy to be recovered. The proposed impersonation attack contains three rounds: eavesdropping round, learning round and masquerading round. In the eavesdropping and learning rounds, the secret parameter $Y_{i,1}'$ is obtained as an input of third round.

3.1. Eavesdropping round

In this round, the adversary $A$ can passively eavesdrop a session executed between the legitimate user $U$ and the remote server and obtain the message $M_1 = \{ID_1, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\}$ sent by $U_i$ in the login phase. In respect to equations (1)–(8), we recall that
CRYPTANALYSIS AND IMPROVEMENT OF AN IMPROVED AUTHENTICATION WITH KEY

3.2. Learning round

In this round, the attacker $A$ who captured the message $M_1 = \{ID_1, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\}$ wants to obtain the secret parameter $Y_{i,1}$ of $U$. The details of this round are described in the following phases as shown in Figure 2:

\[
Y_{i,1} = ID_1r_in_iXS P + pw_i P
\]  
(14)

\[
Y_{i,2} = r_i P
\]  
(15)

\[
Y'_{i,1} = Y_{i,1} - pw_i P = ID_1r_in_iXS P
\]  
(16)

\[
C_1 = aY'_{i,1} = a(ID_1r_in_iXS P)
\]  
(17)

\[
C_2 = ar_i P
\]  
(18)

\[
C_3 = br_i P
\]  
(19)

\[
C_4 = cr_i P
\]  
(20)

\[
C_5 = cY'_{i,1} + k_1 P = c(ID_1r_in_iXS P) + k_1 P
\]  
(21)

\[
C_6 = dr_i P
\]  
(22)
Step 1: On behalf of $U_i$, $A$ performs the following steps to log in to the authentication server:

1. $A$ sets the parameters

   \[ C'_1 = C_1 = aY'_{i,1} \quad (23) \]
   \[ C'_2 = C_2 = ar_i P \quad (24) \]
   \[ C'_3 = Y_{i,2} = r_i P \quad (25) \]
   \[ C'_4 = C_4 = cr_i P \quad (26) \]
   \[ C'_5 = C_5 = cY'_{i,1} + k_1 P \quad (27) \]
   \[ C'_6 = C_6 = dr_i P \quad (28) \]

2. $A$ sends the login request message $M_1 = \{ID_i, Y_{i,2}, C'_1, C'_2, C'_3, C'_4, C'_5, C'_6\}$ to $S$.

As can be clearly seen, the parameters $C'_1, C'_2, C'_4, C'_5, C'_6$ are the same as the original protocol, but $C'_3$ only is changed to $Y_{i,2}$.

Step 2: Upon receiving the login request message $M_1 = \{ID_i, Y_{i,2}, C'_1, C'_2, C'_3, C'_4, C'_5, C'_6\}$, $S$ performs as follows:

1. $S$ firstly checks whether $ID_i$ is valid in the registration table and extracts $n_i$ corresponding to $ID_i$ in its database and then verifies if

   \[ ID_in_i \times S \cdot C'_2 = C'_1 \quad (29) \]

   It is clear that the equation holds because $C'_1$ and $C'_2$ are respectively the same as $C_1$ and $C_2$, which were computed by the legal user $U_i$. Therefore, $S$ accepts $A$ as the legitimate user $U_i$ and performs the following step.

2. $S$ computes

   \[ k_1 P = C'_5 - ID_in_i \times S \cdot C'_4 \quad (30) \]

   Then $S$ randomly chooses $k_2 \in \mathbb{Z}_n^*$, and computes

   \[ K = k_1k_2 P \quad (31) \]

   and the session key

   \[ sk = H(K_x) \quad (32) \]

   where $K_x$ is the $x$-coordinate of the point $K$.

3. $S$ computes

   \[ C'_7 = ID_in_i \times S \cdot C'_3 \quad (33) \]
   \[ C'_8 = ID_in_i \times S \cdot C'_6 + k_2 P \quad (34) \]
   \[ C_9 = E_{sk}(ID_i \parallel m \parallel S) \quad (35) \]

   where $m$ is a session identifier.
3.3. Masquerading round

In this round, the attacker $A$ who obtained the secret parameter $Y_{i,1}'$ from the previous round starts a new session to introduce himself/herself to the server as the legal user $U_i$. The details of this round, outlined in Figure 3, are as follows:

- **Step 3**: $A$ intercepts and stores the message $M_2 = \{C_7, C_8, C_9\}$ and terminates the session. At this point, the attacker $A$ has obtained the secret parameter $Y_{i,1}'$, which is equal to the received parameter $C_7$, because

$$C_7 = \text{ID}_i n_i x S C_3'$$

by equation (33)

$$= \text{ID}_i n_i x S Y_{i,2}$$

by equation (25)

$$= \text{ID}_i n_i x S t_i P$$

by equation (15)

$$= Y_{i,1}'$$

by equation (16)

Figure 3. Masquerading round of the proposed attack on Li et al.’s scheme
Login phase: $A$ can perform the following steps to log in to the authentication server:

1. $A$ chooses random numbers $a, b, c, d, k_1, \in \mathbb{Z}_n^*$.
2. Compute

\[
\begin{align*}
    C_1 &= aY_{i,1} = a(\text{ID}_i r_i n_i x P) \quad (36) \\
    C_2 &= aY_{i,2} = a r_i P \quad (37) \\
    C_3 &= bY_{i,2} = b r_i P \quad (38) \\
    C_4 &= cY_{i,2} = c r_i P \quad (39) \\
    C_5 &= cY'_{i,1} + k_1 P = c \text{ID}_i r_i n_i x P + k_1 P \quad (40) \\
    C_6 &= dY_{i,2} = d r_i P \quad (41)
\end{align*}
\]

3. $A$ sends to $S$ the login request message $M_1 = \{\text{ID}_i, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\}$.

Authentication with key agreement phase: Upon receiving the login request message $M_1 = \{\text{ID}_i, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\}$, $S$ performs the following steps:

1. $S$ firstly checks whether $\text{ID}_i$ is valid in the registration table and extracts $n_i$ corresponding to $\text{ID}_i$ in its database and then verifies if the equation

\[
\text{ID}_i n_i x S C_2 = C_1 \quad (42)
\]

holds. If it does not hold, $S$ rejects $U_i$; otherwise, $S$ accepts $U_i$’s login request and performs the following step.

2. $S$ computes

\[
    k_1 P = C_5 - \text{ID}_i n_i x S C_4 \quad (43)
\]

Then $S$ randomly chooses $k_2, \in \mathbb{Z}_n^*$, and computes

\[
    K = k_1 k_2 P \quad (44)
\]

and the session key $sk = H(K_x)$, where $K_x$ is the $x$-coordinate of the point $K$.

3. $S$ computes

\[
\begin{align*}
    C_7 &= \text{ID}_i n_i x S C_3 \\
    C_8 &= \text{ID}_i n_i x S C_6 + k_2 P \quad (45) \\
    C_9 &= E_{sk}(\text{ID}_i || m || S)
\end{align*}
\]

where $m$ is a session identifier.

4. Finally, $S$ sends to $U_i$ the message $M_2 = \{C_7, C_8, C_9\}$ for mutual authentication and key confirmation.

Mutual authentication and key confirmation phase: $A$ intercepts the message $M_2$ and performs the following steps:
1. \( \mathcal{A} \) computes
\[
k_2 P = C_8 - dY_{i,1}
\]
and obtains the session key \( sk = H(K_x) \), where \( K_x \) is the x-coordinate of the point \( K \).

2. \( \mathcal{A} \) computes
\[
C_{10} = E_{sk}(S||m||ID_i)
\]
and sends the message \( M_3 = \{C_{10}\} \) to \( S \).

3. At the end of the session, \( S \) verifies \( C_{10} \) by checking the equation
\[
C_{10} = E_{sk}(S||m||ID_i)
\]
If it holds, the scheme is finished successfully; otherwise, it terminates in failure.

**Proposition 1.** At the end of the proposed impersonation attack, the attacker \( \mathcal{A} \) has been accepted as the legal user \( U_i \) by the server \( S \).

**Proof.** As mentioned in the authentication with key agreement phase, \( S \) ensures that the received message was generated by the user \( U \) if equation (42) holds. According to equations (36) and (37), we show that equation (42) holds as follows:

\[
\begin{align*}
\text{ID}_i n_i xS C_2 &= \text{ID}_i n_i xS aY_{i,2} \quad \text{By equation (37)} \\
&= \text{ID}_i n_i xS a(r_i P) \quad \text{By equation (15)} \\
&= a(\text{ID}_i n_i xS r_i P) \\
&= a(Y'_{i,1}) \quad \text{By equation (16)} \\
&= C_1 \quad \text{By equation (36)}
\end{align*}
\]

Moreover, at the end of the mutual authentication and key confirmation phase, \( S \) successfully finished the session if equation (48) holds. This equation holds if the encryption session key \( sk = H(K_x) \) computed by the server \( S \) and the adversary \( \mathcal{A} \) is the same. The session key is the same for \( S \) and \( \mathcal{A} \) if the parameter \( K \) computed by \( \mathcal{A} \) is equal to \( K \), which was computed by \( S \). We show that the parameter \( K \) is the same for \( S \) and \( \mathcal{A} \) and is equal to \( k_1k_2 P \). The parameter \( K \) computed by \( S \) is as follows:

On the other hand, the parameter \( K \) computed by \( \mathcal{A} \) is as follows:

Therefore, \( \mathcal{A} \) succeeded to masquerade as the legitimate user \( U \) and shared the session key \( sk = H(K_x) \) with \( S \).

\[
K = k_2(C_5 - \text{ID}_i n_i xS C_4) \quad \text{By equations (43) and (44)}
\]
\[
= k_2 ((cY'_{i,1} + k_1 P) - \text{ID}_i n_i xS C_4) \quad \text{By equation (40)}
\]
\[
= k_2 ((cY'_{i,1} + k_1 P) - \text{ID}_i n_i xS (cY_{i,2})) \quad \text{By equation (39)}
\]
\[
= k_2 ((cY'_{i,1} + k_1 P) - c\text{ID}_i n_i xS (r_i P)) \quad \text{By equation (15)}
\]
\[
= k_2 ((cY'_{i,1} + k_1 P) - cY'_{i,1}) \quad \text{By equation (16)}
\]
\[
= k_1k_2 P
\]
4. OUR IMPROVED SCHEME

As can be seen in Section 3, the security flaws of Li et al.’s scheme [38] were due to the design of the phases: login phase, authentication with key agreement phase and mutual authentication and key confirmation phase. Thus, in this section, we only improve these phases to overcome the security flaws of Li et al.’s scheme. The outline of our improvement is shown in Figure 4.

4.1. Login phase

$U_i$ can perform the following steps to log in to the authentication server as shown in Figure 4:

1. $U_i$ inputs his/her $ID_i$ with $pw_i$ into his/her device.
2. The device chooses a temporary secret random number $a \in \mathbb{Z}_n^*$.
3. Compute

$$K = k_1 \left( C_8 - dY'_{i,1} \right)$$

By equations (46) and (47)

$$= k_1 \left( (ID_i n_i xS C_6 - k_2 P) - dY'_{i,1} \right)$$

By equation (45)

$$= k_1 \left( (ID_i n_i xS dY_{i,2} - k_2 P - dY'_{i,1}) \right)$$

By equation (41)

$$= k_1 \left( (d(ID_i n_i xS r_i P - k_2 P) - dY'_{i,1}) \right)$$

By equation (15)

$$= k_1 \left( (d(Y'_{i,2}) - k_2 P - dY'_{i,1}) \right)$$

By equation (16)

$$= k_1 k_2 P$$
\[ Y'_{i,1} = Y_{i,1} - pw_i P = ID_i r_i n_i x_S P \]  
(50)

\[ C_1 = aY'_{i,1} = a (ID_i r_i n_i x_S P) \]  
(51)

\[ C_2 = aY_{i,2} = ar_i P \]  
(52)

\[ C_3 = H (ID_i, C_1, C_2) \]  
(53)

4. \( U_i \) sends to \( S \) the login request message \( M_1 = \{ID_i, Y_{i,2}, C_2, C_3\} \).

4.2. Authentication with key agreement phase

Upon receiving the login request message \( M_1 = \{ID_i, Y_{i,2}, C_2, C_3\} \), \( S \) performs the following steps as shown in Figure 4:

1. \( S \) firstly checks whether \( ID_i \) is valid in the registration table and extracts \( n_i \) corresponding to \( ID_i \) in its database and then computes

\[ C'_1 = ID_i n_i x_SC_2 \]  
(54)

and verifies if the equation

\[ H (ID_i, C'_1, C_2) = C_3 \]  
(55)

holds. If it does not hold, \( S \) rejects \( U_i \); otherwise, \( S \) accepts \( U_i \)'s login request and performs the following step.

2. \( S \) randomly chooses a number \( b \in Z_n^* \) and computes

\[ K = bC'_1 \]  
(56)

\[ C_4 = bID_i n_i x_i Y_{i,2} \]  
(57)

\[ C_5 = H (ID_i, C'_1, C_2, C_3, C_4, K) \]  
(58)

3. Finally, \( S \) sends to \( U_i \) the message \( M_2 = \{C_4, C_5\} \) for mutual authentication and key confirmation.

4.3. Mutual authentication and key confirmation phase

Upon receiving the message \( M_2 \), \( U_i \) performs the following steps:

1. \( U_i \) computes

\[ K' = aC_4 \]  
(59)

and verifies whether the equation

\[ H (ID_i, C_1, C_2, C_3, C_4, K') = C_5 \]  
(60)

holds. If it does not hold, it rejects; otherwise, \( U_i \) believes that the response of the message is correct from the responding server.
2. \( U_i \) computes
\[
C_6 = H \left( \text{ID}_i, C_1, C_2, C_3, C_4, C_5, K' \right)
\] (61)
and sends the message \( M_5 = \{C_6\} \) to \( S \). Finally, \( U_i \) computes the session key
\[
sk = H \left( \text{ID}_i, C_1, C_2, C_3, C_4, C_5, C_6, K' \right)
\] (62)

3. At the end of the session, \( S \) verifies \( C_6 \) by checking the equation
\[
H \left( \text{ID}_i, C_1, C_2, C_3, C_4, C_5, K' \right) = \{C_6\}
\] (63)
If it does not hold, \( S \) terminates in failure; otherwise, the scheme is finished successfully with the session key
\[
sk = H \left( \text{ID}_i, C_1', C_2, C_3, C_4, C_5, C_6, K' \right)
\] (64)

5. SECURITY ANALYSIS OF THE IMPROVED PROTOCOL

In this section, we show that our protocol is secure in the random oracle model. We start with the formal security model and the algorithm assumption that will be used in our proof.

5.1. Security model

In order to make our scheme resist the known attacks to the authentication protocols, we use the method of provable security. The security proof is based on the model proposed by Abdalla and Pointcheval [40]. The model that we use is as follows:

5.1.1. Participants

An authentication protocol \( \Pi \) runs in a network of a number of interconnected participants where each participant is either a client \( U \in \mathcal{U} \) or a trusted server \( S \in \mathcal{S} \). The set \( S \) is assumed to involve only a single server for simplicity. Each of the participants may have several instances called oracles involved in distinct executions of the protocol \( \Pi \). We refer to \( i \)-th instance of \( U \) (resp. \( S \)) in a session as \( \Pi^i_U \) (resp. \( \Pi^i_S \)). Every instance \( \Pi^i_U \) (resp. \( \Pi^i_S \)) has a partner ID \( \text{id}^i_U \) (resp. \( \text{id}^i_S \)), a session ID \( \text{id}^i_U \) (resp. \( \text{id}^i_S \)) and a session key \( \text{sk}^i_U \) (resp. \( \text{sk}^i_S \)). \( \text{id}^i_U \) (resp. \( \text{id}^i_S \)) denotes the set of the identities that are involved in this instance. \( \text{id}^i_U \) (resp. \( \text{id}^i_S \)) denotes the flows that are sent and received by the instance \( \Pi^i_U \) (resp. \( \Pi^i_S \)). An instance \( \Pi^i_U \) (resp. \( \Pi^i_S \)) is said to be accepted if it holds a session key \( \text{sk}^i_U \) (resp. \( \text{sk}^i_S \)), a session identifier \( \text{id}^i_U \) (resp. \( \text{id}^i_S \)) and a partner identifier \( \text{id}^i_U \) (resp. \( \text{id}^i_S \)).

Two instances \( \Pi^i_U \) and \( \Pi^j_S \) are considered partnered if and only if (a) both of them have accepted, (b) \( \text{id}^i_U = \text{id}^j_S \), (c) \( \text{id}^j_U = \text{id}^i_S \) and (d) \( \text{id}^i_U = \text{id}^j_S \).

5.1.2. Long-lived keys

Each client \( U \in \mathcal{U} \) holds a password \( \text{pw}_U \). Each server \( S \in \mathcal{S} \) holds a vector \( \text{pw}_S = \langle \text{pw}_U \rangle_{U \in \mathcal{U}} \) with an entry for each client.
5.1.3. Adversary model
The communication network is assumed to be fully controlled by an adversary $A$, which schedules and mediates the sessions among all the parties. The adversary $A$ is allowed to issue the following queries in any order:

**Execute($\Pi_U^i, \Pi_S^j$):** This query models passive attacks in which the attacker eavesdrops on honest executions among the client instance $\Pi_U^i$ and trusted server instance $\Pi_S^j$. The output of this query consists of the messages that were exchanged during the honest execution of the protocol $\Pi$.

**SendClient($\Pi_U^i, m$):** The adversary makes this query to intercept a message and then modify it, create a new one or simply forward it to the client instance $\Pi_U^i$. The output of this query is the message that the client instance $\Pi_U^i$ would generate upon receipt of message $m$. Additionally, the adversary is allowed to initiate the protocol by invoking $\text{SendClient}(\Pi_U^i, \text{Start})$.

**SendServer($\Pi_S^j, m$):** This query models an active attack against a server. The adversary makes this query to obtain the message that the server instance $\Pi_S^j$ would generate on receipt of the message $m$.

**Reveal($\Pi_U^i$):** This query models the known session key attack. The adversary makes this query to obtain the session key of the instance $\Pi_U^i$.

**Corrupt($U$):** This query returns to the adversary the long-lived key $pw_U$ for participant $U$.

**Test($\Pi_U^i$):** Only one query of this form is allowed to be made by the adversary to a fresh oracle. To respond to this query, a random bit $b \in \{0, 1\}$ is selected. If $b = 1$, then the session key held by $\Pi_U^i$ is returned. Otherwise, a uniformly chosen random value is returned.

5.1.4. Fresh oracle
An oracle $\Pi_U^i$ is called fresh if and only if the following conditions hold: (i) $\Pi_U^i$ has accepted, and (ii) $\Pi_U^i$ or its partner (if exists) has not been asked a Reveal query after their acceptance.

5.1.5. Protocol security
The security of an authentication protocol $\Pi$ is modelled by the game $\text{Game}(\Pi, A)$. When playing this game, the adversary $A$ can make many queries mentioned earlier to $\Pi_U^i$ and $\Pi_S^j$. If $A$ asks a single query, $\text{Test}(\Pi_U^i)$, where $\Pi_U^i$ has accepted and is fresh, then $A$ outputs a single bit $b'$. The aim of $A$ is correctly guessing the bit $b$ in the test session. More precisely, we define the advantage of $A$ as follows:

$$\text{Adv}_{\Pi, D}(A) = |2 \Pr[b' = b] - 1|$$

The protocol $\Pi$ is said to be secure if $\text{Adv}_{\Pi, D}(A)$ is negligible.

5.2. Computational assumption
We define the decisional Diffie–Hellman (DDH) assumption, which we use in the security proof of our scheme.

**Definition 1:** The DDH assumption can be precisely defined by two experiments, $\text{Exp}^{\text{DDH} \text{- real}}_{P,n}(W)$ and $\text{Exp}^{\text{DDH} \text{- rand}}_{P,n}(W)$. An adversary $W$ is provided with $uP$, $vP$ and $uvP$ in the experiment $\text{Exp}^{\text{DDH} \text{- real}}_{P,n}(W)$, and $uP$, $vP$ and $wP$ in the experiment $\text{Exp}^{\text{DDH} \text{- rand}}_{P,n}(W)$, where $u$, $v$ and $w$ are drawn at random from $\mathbb{Z}_n^*$. Define the advantage of $W$ in violating the DDH assumption, $\text{Adv}^{\text{DDH}}_{P,n}(W)$, as follows:

$$\text{Adv}^{\text{DDH}}_{P,n}(W) = \max \{|\Pr[\text{Exp}^{\text{DDH} \text{- real}}_{P,n}(W) = 1] - \Pr[\text{Exp}^{\text{DDH} \text{- rand}}_{P,n}(W) = 1]|\}$$
5.3. Security proof

Theorem 1: Let $D$ be a uniformly distributed dictionary of possible passwords with size $|D|$. Let $\Pi$ describe the improved authentication protocol defined in Figure 4. Suppose that DDH assumption holds. Then,

$$\text{Adv}_{\Pi,D}(A) \leq \frac{q_H^2}{2^k} + \frac{(q_s + q_e)^2}{n^2} + 2q_e \cdot \text{Adv}_{P,n}^{\text{DDH}}(W) + 2 \max \left\{ \frac{q_H}{2^k}, \frac{q_s}{|D|} \right\}$$

where $q_e$ denotes the number of Send queries, $q_e$ denotes the number of Execute queries and $q_H$ denotes the number of hash queries to $H$.

Proof. This proof consists of a sequence of hybrid games, starting at the real attack $G_0$ and ending up at game $G_4$ where the adversary has no advantage. For each game $G_i (0 \leq i \leq 4)$, we define $\text{Succ}_i$ as the event that $A$ correctly guesses the bit $b$ in the test session.

Game $G_0$. This game is the real protocol, in the random oracle model. In this game, all the instances of $U$ and the trusted server $S$ are modelled as the real execution in the random oracle. By definition of event $\text{Succ}_i$, which means that the adversary correctly guesses the bit $b$ involved in the Test-query, we have

$$\text{Adv}_{\Pi,D}(A) = 2\left| \Pr[\text{Succ}_0] - \frac{1}{2} \right|$$

(65)

Game $G_1$. This game is as the same as the game $G_0$, except that we simulate the hash oracle $H$ as usual by maintaining hash list $H_{\text{List}}$ with entries of the form $(\text{Inp}, \text{Outp})$. On hash query for which there exists a record $(\text{Inp}, \text{Outp})$ in the hash list, return $\text{Outp}$. Otherwise, randomly choose $\text{Outp} \in \{0,1\}^k$, send it to $A$, and store the new tuple $(\text{Inp}, \text{Outp})$ into the hash list. We also simulate all the instances, as the real players would do, for the Send-query and for the Execute, SendClient, SendServer, Reveal, Corrupt and Test queries. From the viewpoint of the adversary, we easily see that the game is perfectly indistinguishable from the real attack. Hence,

$$\Pr[\text{Succ}_1] = \Pr[\text{Succ}_0]$$

(66)

Game $G_2$. In this game, we simulate all the oracles in game $G_1$, except that we cancel the game in which some collisions appear on the partial transcripts $M_1$ and $M_2$ and on hash values. According to the birthday paradox, the probability of collisions in output of hash oracle is at most $q_H^2/2^k + 1$, where $q_H$ denotes the maximum number of hash queries. Similarly, the probability of collisions in the transcripts is at most $(q_s + q_e)^2/(2n^2)$, where $q_s$ represents the number of queries to the SendClient and SendServer oracles and $q_e$ represents the number of queries to the Execute oracle. So we have

$$|\Pr[\text{Succ}_2] - \Pr[\text{Succ}_1]| \leq \frac{q_H^2}{2^k + 1} + \frac{(q_s + q_e)^2}{2n^2}$$

(67)

Game $G_3$. In this game, we change the simulation of queries to the SendClient oracle. First, we randomly select a session executed by partner instances $\Pi^U_i$ and $\Pi^S_i$.

- When SendClient $(\Pi^U_i, \text{Start})$ is asked, we choose random values $u \in \mathbb{Z}_n^*$ and compute $C_1 = uY_{i,1}$, $C_2 = uY_{i,2}$ and $C_3 = H(\text{ID}_i, C_1, C_2)$ and return $M_1 = \{\text{ID}_i, Y_{i,2}, C_2, C_3\}$ to $A$. 

Copyright © 2014 John Wiley & Sons, Ltd

Int. J. Network Mgmt (2014)
DOI: 10.1002/nem
Combining equations (65)–(70), one obtains the announced result as follows:

\[ \Pr[\text{Succ}_3] = \Pr[\text{Succ}_2] \]  

\[ \text{Game } G_4. \text{ In this game, we once again change the simulation of queries to the } \text{SendClient} \text{ oracle for the selected session in game } G_3. \text{ This time, we change the way we compute } K \text{ so that it becomes independent of password and ephemeral keys. When } \text{SendServer} \left( \Pi^i_5, M_1 = \{ID_1, Y_{i,2}, C_2, C_3\} \right) \text{ and } \text{SendClient} \left( \Pi^i_4, M_2 = \{C_4, C_5\} \right) \text{ are asked, we set } K = wP, \text{ where } w \text{ is selected from } Z^n_\ast \text{ at random. The difference between the game } G_4 \text{ and the game } G_3 \text{ is as follows:} \]

\[ |\Pr[\text{Succ}_4] - \Pr[\text{Succ}_3]| \leq q_e \cdot \text{Adv}^{\text{DDH}}(W) \]

By assuming a successful adversary } A \text{ to distinguish } G_3 \text{ and } G_4, \text{ we construct a DDH solver } W.\]

In game } G_4, \text{ the Diffie–Hellman key } K \text{ is random and independent with the user’s password and ephemeral keys. So, there are three possible cases where the adversary distinguishes the real session key } SK \text{ and the random key as follows:}

\begin{itemize}
  \item Case 1. The adversary queries } (ID_1, C_1, C_2, C_3, C_4, C_5, K) \text{ to } H. \text{ The probability that this event occurs is } q_H/2^k.
  \item Case 2. The adversary asks } \text{SendClient} \text{ query except } \text{SendClient}(\Pi^i_5, m) \text{ and successfully impersonates } U \text{ to } S. \text{ The adversary is not allowed to reveal static key } pw_i \text{ of } U. \text{ Thus, in order to impersonate } U, \text{ the adversary has to obtain some information of the password } pw_i \text{ of } U. \text{ The probability is } 1/D. \text{ Because there are at most } q_s \text{ sessions of this kind, the probability that this event occurs is lower than } q_s/|D|.
\end{itemize}

As a conclusion,

\[ \Pr[\text{Succ}_4] = \frac{1}{2} + \max \left\{ \frac{q_H}{2^k}, \frac{q_s}{|D|} \right\} \]

Combining equations (65)–(70), one obtains the announced result as follows:

\[ \text{Adv}_{\Pi, D}(A) = 2|\Pr[\text{Succ}_0] - \frac{1}{2}| \]

\[ = 2 \left| \Pr[\text{Succ}_0] - \Pr[\text{Succ}_4] + \max \left\{ \frac{q_H}{2^k}, \frac{q_s}{|D|} \right\} \right| \]

\[ \leq 2 \left( |\Pr[\text{Succ}_0] - \Pr[\text{Succ}_4]| + \max \left\{ \frac{q_H}{2^k}, \frac{q_s}{|D|} \right\} \right) \]

\[ \leq 2 \left( |\Pr[\text{Succ}_1] - \Pr[\text{Succ}_2]| + |\Pr[\text{Succ}_3] - \Pr[\text{Succ}_4]| + \max \left\{ \frac{q_H}{2^k}, \frac{q_s}{|D|} \right\} \right) \]

\[ \leq \frac{q_H^2}{2^k} + \frac{(q_s + q_e)^2}{n^2} + 2q_e \cdot \text{Adv}^{\text{DDH}}(W) + 2 \max \left\{ \frac{q_H}{2^k}, \frac{q_s}{|D|} \right\} \]

\[ \square \]
6. SECURITY AND PERFORMANCE COMPARISONS

The security comparison between our proposed scheme and some related schemes are summarised in Table 2. As can be clearly seen, the proposed scheme not only prevents the attacks that are applicable to other schemes but also provides some new security properties. As a result, the proposed scheme is more secure and has many functionality compared with other schemes.

We evaluate the performance of the proposed scheme in terms of the computation and the communication cost and compare it with the related scheme. To estimate the computation cost, we define the following notations: $T_{PM}$ is the time complexity of elliptic curve scalar point multiplication, $T_{PA}$ is the time complexity of elliptic curve point addition, and $T_X$ is the time complexity of one-way hash function or symmetric encryption/decryption. The execution time computed by Farash et al. [5] for the cryptographic operations on different hardware platform is shown in Table 3.

To estimate the communication cost, same as [41], we assume that the size of $n$ used in the ECC of the scheme is 160 bits, the block size of cipher text of the symmetric encryption/decryption is 128 bits, the digest message size of hash function (e.g., SHA-1) is 160 bits and the identity size is 80 bits. To compute the communication cost, we consider all exchanged messages during a session. In our scheme, the exchanged messages are $M_1 = \{\text{ID}_i, Y_{i,2}, C_2, C_3\}$, $M_2 = \{C_4, C_5\}$ and $M_3 = \{C_6\}$. These messages are including one identity $\text{ID}_i$ with 80-bit length, three points $\{Y_{i,2}, C_2, C_4\}$ on elliptic curve with 160-bit length, and three hash values $\{C_3, C_5, C_6\}$ with 160-bit length. Thus, the communication cost of our scheme can be computed as $1(80) + 3(160) + 3(160) = 1040$ bits.

In Li et al.’s scheme, the exchanged messages are $M_1 = \{\text{ID}_i, Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6\}$, $M_2 = \{C_7, C_8, C_9\}$ and $M_3 = \{C_{10}\}$. These messages are including one identity $\text{ID}_i$ with 80-bit length, nine points $\{Y_{i,2}, C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$ on elliptic curve with 160-bit length, and two cipher texts $\{C_9, C_{10}\}$ with 256-bit length. Thus, the communication cost of our scheme can be computed as $1(80) + 9(160) + 2(256) = 2032$ bits.

The performance of three phases, that is, login phase, authentication with key agreement phase and mutual authentication phase, of the proposed improved scheme and a comparison with the related schemes are summarised in Table 4. As can be seen, the execution time of the improved scheme is about 42% of the execution time of Li et al.’s scheme. Moreover, the communication cost of the improved scheme is less than 52% of the communication cost of Li et al.’s scheme. Therefore, the improved scheme not only conquers the security problems of Li et al.’s scheme but also provides a better performance.

Table 2. Security comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Zhang et al.’s scheme [36]</th>
<th>Tu et al.’s scheme [37]</th>
<th>Li et al.’s scheme [38]</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>No verification table</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of guessing attacks</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of replay attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of stolen-verifier attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of stolen smart card attack</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of impersonation attack</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevention of modification attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Known-key security</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Providing of perfect forward secrecy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Provable security</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3. The execution time of cryptographic operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>$T_{PM}$</th>
<th>$T_{PA}$</th>
<th>$T_X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>0.86 ms</td>
<td>0.001 ms</td>
<td>&lt; 0.001 ms</td>
</tr>
</tbody>
</table>
Table 4. Performance comparison

<table>
<thead>
<tr>
<th></th>
<th>Zhang <em>et al.</em>'s protocol [36]</th>
<th>Tu <em>et al.</em>'s protocol [37]</th>
<th>Li <em>et al.</em>'s protocol [38]</th>
<th>Improved protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User's computation</strong></td>
<td>$4T_{PM} + 1T_{PA} + 6T_{X}$</td>
<td>$4T_{PM} + 1T_{PA} + 5T_{X}$</td>
<td>$11T_{PM} + 3T_{PA} + 3T_{X}$</td>
<td>$4T_{PM} + 1T_{PA} + 3T_{X}$</td>
</tr>
<tr>
<td><strong>Server's computation</strong></td>
<td>$4T_{PM} + 1T_{PA} + 5T_{X}$</td>
<td>$3T_{PM} + 5T_{X}$</td>
<td>$6T_{PM} + 2T_{PA} + 3T_{X}$</td>
<td>$3T_{PM} + 3T_{X}$</td>
</tr>
<tr>
<td><strong>Total computation</strong></td>
<td>$8T_{PM} + 2T_{PA} + 11T_{X}$</td>
<td>$7T_{PM} + 1T_{PA} + 10T_{X}$</td>
<td>$17T_{PM} + 5T_{PA} + 6T_{X}$</td>
<td>$7T_{PM} + 1T_{PA} + 6T_{X}$</td>
</tr>
<tr>
<td><strong>Execution time</strong></td>
<td>$\approx 8.89$ ms</td>
<td>$\approx 6.03$ ms</td>
<td>$\approx 14.625$ ms</td>
<td>$\approx 6.026$ ms</td>
</tr>
<tr>
<td><strong>Communication cost</strong></td>
<td>1200 bits</td>
<td>1200 bits</td>
<td>2032 bits</td>
<td>1040 bits</td>
</tr>
</tbody>
</table>
7. CONCLUSIONS

In this paper, we analysed Li et al.’s remote user authentication scheme. We pointed out that Li et al.’s scheme suffers from impersonation attack by which an attacker can masquerade as a legal user to share a common session key with the server. Moreover, we proposed an improvement of Li et al.’s scheme to overcome the security problems. The security of the improved protocol was proven in the random oracle model. The implementation results indicated that the improved scheme requires 58% computational cost and 48% communication cost less than Li et al.’s scheme.

REFERENCES


**AUTHORS’ BIOGRAPHY**

Mohammad Sabzinejad Farash received his BSc degree in Electronic Engineering from Shahid Chamran College of Kerman in 2006 and MSc degree in Communication Engineering from I. Hossein University in 2009. He also received his PhD degree in Cryptographic Mathematics from the Department of Mathematics and Computer Sciences of Kharazmi University in Iran in 2013. His research interests are security protocols and provable security models.