Design of improved password authentication and update scheme based on elliptic curve cryptography

SK Hafizul Islam *, G.P. Biswas
Department of Computer Science and Engineering, Indian School of Mines, Dhanbad-826004, India

A R T I C L E   I N F O

Article history:
Received 29 December 2010
Received in revised form 29 June 2011
Accepted 1 July 2011

Keywords:
Password authentication
Elliptic curve cryptography
Cryptographic hash function
Impersonation attack
Insider attack

A B S T R A C T

Secured password authentication and update of passwords are two essential requirements for remote login over unreliable networks. In this paper, an elliptic curve cryptography (ECC) based technique has been proposed that not only satisfies the above two requirements, but also provides additional security requirements that are not available in some schemes proposed so far. For instances, the Peyravian and Zunic's scheme does not provide the protection against the password guessing attack, server spoofing attack and data eavesdropping attack. Although some modifications to remove these attacks have been proposed by Hwang and Yeh, Lee et al., it has been found that some attacks like replay attack, server spoofing attack, data eavesdropping attack, etc. are still possible. Subsequently, Hwang and Yeh’s scheme is further improved by Lin and Hwang, which has been analyzed in this paper and certain security flaws have been identified. We have attempted to remove these security flaws and proposed an ECC-based scheme that in addition to the secured password authentication and password update, it protects several related attacks efficiently. As a proof of our claim, the detailed security analysis of the proposed scheme against the attacks has been given. One advantage of the proposed scheme is that it generates an ECC-based common secret key that can be used for symmetric encryption, which requires lesser processing time than the time required in the public key encryption-based techniques.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Client authentication needs security for remote login while the client's program tries to communicate with the server's program over insecure networks like Internet. The identity and a secret password of a client are used for mutual authentication and access control; password can be compromised during transmission, if an efficient scheme is not followed. Also some systems in a hostile network needs change of the client's password periodically for the protection of the valuable resources from adversary, and until a secured password change protocol that allows the client to change the old password to a new password safely, the systems are not well protected. Basically there are four basic approaches based on public key encryption, private key encryption, hash function and their combinations, to design password authentication and password change scheme.

Lamport [1] suggested a hash-based password authentication scheme that mutually authenticates the client and the server successfully, although it is immune from server's data eavesdropping and impersonation attacks, but vulnerable to reply attack, also its high hash computation and password resetting problem decreases its applicability for practical use. The password authentication and password change protocols using only collision-resistant one-way hash function without...
2.1. Theory of elliptic curve

The equation of a non-singular elliptic curve \( E_q(a, b) \) over a finite field \( Z_q \) \((q > 3) \) and is a large prime number) can be written as

\[
y^2 \mod q \equiv x^3 + ax + b \mod q
\]

where \( a \) and \( b \) are two constant such that \( 4a^3 + 27b^3 \neq 0 \mod q \) must be satisfied for its non-singularity. Any point \( P(x, y) \in E_q(a, b) \), \( x, y \in Z_q \) together with \( O \), called 'point at infinity' forms an additive cyclic group \( E = \{(x, y) \in E_q(a, b) \} \cup \{O\} \).
where $O$ serves as additive identity element of the group. The point multiplication is computed by repeated addition as,

$$k \cdot P = \overbrace{P + P + \cdots + P}^{k \text{ times}}.$$ 

A more details of elliptic curve group properties can be found in [30].

2.2. Computational problems

**Definition 1 (Elliptic Curve Discrete Logarithm Problem (ECDLP)).** Given $Q, R \in E$, find an integer $k \in \mathbb{Z}_q^*$ such that $R = kQ$.

**Definition 2 (Computational Diffie–Hellman Problem (CDHP)).** Given $(P, aP, bP)$ for any $a, b \in \mathbb{Z}_q^*$, computation of $abP$ is hard to the group $E$.

**Definition 3 (Decisional Diffie–Hellman Problem (DDHP)).** Given $(P, aP, bP, cP)$ for any $a, b, c \in \mathbb{Z}_q^*$, decide whether or not $cP = abP$, i.e. decide $c = ab \mod q$ or not.

2.3. Bilinear pairings

Let $G_1$ denotes an additive group of prime order $q$. $G_2$ is a multiplicative group of the same order and $P$ is a generator of $G_1$. Also let $\hat{e} : G_1 \times G_2 \rightarrow G$ is an admissible mapping, which satisfies the following properties.

- **Bilinearity:** For any $P, Q, R \in G_1$ then we have $\hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R)$ and $\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R)$. Therefore, for any $a, b \in \mathbb{Z}_q^*$: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab} = \hat{e}(abP, Q) = \hat{e}(P, abQ)$ holds.
- **Non-degenerate:** $\hat{e}(P, P) \neq 1_{G_2}$, where $1_{G_2}$ is the identity element of the group $G_2$.
- **Computability:** There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

In general, $G_1$ is group of points on an elliptic curve, and $G_2$ is a multiplicative subgroup of a finite field. The map $\hat{e}$ will be derived either from the modified Weil pairing or Tate pairing over a finite field. For more comprehensive description about bilinear pairings, selection of suitable parameters, elliptic curves and these groups can be found in [33–35] for efficiency and security.

3. Review of Lin and Hwang’s scheme

In this section, a brief description of the Lin and Hwang [9] scheme that contains three parts: password authentication, password change and key distribution, are given below, where the following notations have been used (Table 1).

Now the Lin and Hwang’s scheme for password authentication, password change and distribution of secure session key are given below.

3.1. Password authentication protocol

It consists of the following steps:

**Step 1.** Client → Server: $id, \{r_c, pw\}K_s$.

**Step 2.** Server → Client: $r_s \oplus r_c, H(r_c)$.

**Step 3.** Client → Server: $id, H(r_c, r_s)$.

**Step 4.** Server → Client: Access Granted/Denied.

In brief, the server stores $H(pw)$ instead of $pw$, to protect the password. During the password authentication, a client selects a random number $r_c$ and encrypts $r_c$ and $pw$ with server’s public key $K_S$ and sends the same with client’s id to the server as shown in step 1. The server decrypts $\{r_c, pw\}K_s$ using its own private key and retrieves $r_c$ and $pw$, then compares hashed result of extracted $pw$ with $H(pw)$, which stored in the server’s database. If the result is matched then the server selects a random number $r_s$ computes $r_c \oplus r_s$ and $H(r_c)$ and sends back $r_c \oplus r_s$ and $H(r_c)$ to the client. After receiving $r_c \oplus r_s, H(r_c)$ from the server, the client XORs $r_s$ with $r_c \oplus r_s$ and retrieves $r_s$. The client compares if the hashed value of retrieved $r_s$ and received $H(r_c)$, depending on this condition client computes the authentication token $H(r_c, r_s)$ and sends back id, $H(r_c, r_s)$ to the server. Therefore, the server computes $H(r_c, r_s)$ using its own copies of $r_c$ and $r_s$ and compares with received $H(r_c, r_s)$. If it is matched then the server sends a message ‘Access granted’ otherwise send an error message ‘Access Denied’ to the client.

3.2. Password change protocol

The four steps of this protocol are given below:

**Step 1.** Client → Server: $id, \{r_c, pw\}K_s$.

**Step 2.** Server → Client: $r_c \oplus r_s, H(r_c)$.

**Step 3.** Client → Server: $id, H(r_c, r_s), H(new\_pw) \oplus H(r_c + 1, r_s), H(H(new\_pw), r_s)$.

**Step 4.** Server → Client: Access Granted/Denied.

Step 2. Server → Client: $g^x + g^y$, $H(g^x)$.
Step 3. Client → Server: id, $H(g^x, g^y)$.

The common secret session key is then computed by the client and the server as $(g^x)^y$ and $(g^y)^x$ respectively.

### 4. Weaknesses of Lin and Hwang's scheme

The cryptanalysis of the Lin and Hwang’s scheme [9] has been made in this section, and some of the common weaknesses are given below.

#### 4.1. Stolen-verifier attack

The stolen-verifier attack, which is described in [6, 13], means that an outsider theft the password-verifier from the server’s database and applies an off-line guessing attack on to get the client's exact password and hence, he can impersonate as a legitimate client. In Lin–Hwang’s scheme [9], the client $C$ registers to the remote server $S$ with $id, H(pw)$ and $S$ then stores the pair $(id, H(pw))$ to the database. The outsider $A$ can successfully find out the $C$’s password $pw$ by performing the following procedure.

Step 1. $A$ steals $H(pw)$ from the $S$’s database and tries to find out $C$’s password $pw$ by using an off-line password guessing attack on stolen $H(pw)$.
Step 2. $A$ guess a password $pw''$, computes $H(pw'')$ and then compares the result with stolen $H(pw)$. 

---


---

### Table 1
Notations are used in Lin and Hwang’s scheme.

<table>
<thead>
<tr>
<th>Notations used</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$id$</td>
<td>Identity of the client, publicly known to all</td>
</tr>
<tr>
<td>$pw$</td>
<td>Secret password of a client</td>
</tr>
<tr>
<td>$K_s$</td>
<td>Public key of the server</td>
</tr>
<tr>
<td>$(M)K_s$</td>
<td>Encryption of the message $M$ with the public key $K_s$</td>
</tr>
<tr>
<td>$r_c, r_s$</td>
<td>Random numbers chosen by the client and the server respectively</td>
</tr>
<tr>
<td>$q, g$</td>
<td>A large prime $q$, order of the group $Z_q$ and $g$ is the generator of the cyclic group $Z_q^*$</td>
</tr>
<tr>
<td>$x, y$</td>
<td>Random exponents chosen by the client and the server respectively</td>
</tr>
<tr>
<td>$H(\cdot)$</td>
<td>Collision-resistant one-way secure hash function</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>Bitwise XOR operator</td>
</tr>
</tbody>
</table>

### Table 2
Notations used in the proposed scheme.

<table>
<thead>
<tr>
<th>Notations used</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_A$</td>
<td>Identity of the client $A$</td>
</tr>
<tr>
<td>$pw_A$</td>
<td>Secret password of the client $A$</td>
</tr>
<tr>
<td>$d_s$</td>
<td>Secret key of the server $S$</td>
</tr>
<tr>
<td>$U_s$</td>
<td>Public key of the server $S$, where $U_s = d_s \cdot G$</td>
</tr>
<tr>
<td>$U_A$</td>
<td>Password-verifier of the client $A$, where $U_A = pw_A \cdot G$</td>
</tr>
<tr>
<td>$K_s$</td>
<td>Secret key computed either using $K = pw_A \cdot U_s = (K_s, K_p)$ or $K = d_s \cdot U_A = (K_s, K_p)$</td>
</tr>
<tr>
<td>$E_{EA}(\cdot)$</td>
<td>Symmetric encryption (AES) with $K_s$</td>
</tr>
<tr>
<td>$G$</td>
<td>Bases point of the elliptic curve group of order $n$ such that $n \cdot G = 0$, where $n$ is a large prime number</td>
</tr>
<tr>
<td>$H(\cdot)$</td>
<td>A collision-resistant one-way secure hash function</td>
</tr>
<tr>
<td>$rA/rS$</td>
<td>Random numbers chosen by the client/server from $[1, n - 1]$ respectively</td>
</tr>
<tr>
<td>$+/-$</td>
<td>Elliptic curve point addition/subtraction</td>
</tr>
</tbody>
</table>
Step 3. If $H(pw') = H(pw)$, then $pw' = pw$ i.e., $A$ correctly guesses $C$’s password. Otherwise, $A$ can repeat the process until to have the correct password $pw$. The correctness of the password can be checked by testing all possible passwords from the search space $|PW|$, where $PW$ is the set of all possible passwords and $|·|$ represents the cardinality of the set. It is known that a client generally chooses the weak password (low intensity) for easy memorization, so the space $|PW|$ is not large enough.

Therefore, the stolen-verifier attack is possible in Lin–Hwang’s scheme. The stolen-verifier attack in Lin–Hwang’s scheme is illustrated in Fig. 1.

4.2. Insider attack

In insider attack as stated in [36,37], a client $C$ may register with a number of servers $S_1, S_2, \ldots, S_n$ using a common password $pw$ and the identity $id$ for his convenience, and if the privileged-insider $U_1$ of $S_1$ has the knowledge of $C$’s $pw$ and $id$, then $U_1$ may try to access other servers $S_2, S_3, \ldots, S_n$ by using the same $pw$ and $id$. In Lin–Hwang’s scheme [9], initially the remote server stores the pair $(id, H(pw))$ of the client $C$ to the database. Thus the insider attack in Lin–Hwang’s scheme may be done by using the following three steps:

Step 1. $U_1$ steals the password-verifier $H(pw)$ from the $S_1$’s database.
Step 2. $C$ chooses an easy-memorable password and therefore, it is not difficult for $U_1$ to figure out $C$’s password $pw$ from $H(pw)$ by executing an off-line password guessing attack.
Step 3. $U_1$ tries to use $C$’s identity-password pair $(id, pw)$, follows the password authentication protocol of Lin–Hwang’s scheme and can easily login to the other remote servers $S_2, S_3, \ldots, S_n$.

The detailed description of this attack is given in the Fig. 2.

4.3. Impersonation attack

According to the impersonation attack in [7,13], it is found that Lin and Hwang’s scheme [9] is not free from this kind of impersonation attack, a brief description of which is given now.

Step 1. Client $C$ sends the authentication message $(id, [r_c, pw]K_s)$ to $S$.
Step 2. The server $S$, on receiving the client $C$’s authentication message $(id, [r_c, pw]K_s)$, $S$ decrypts $[r_c, pw]K_s$ with its own private key and gets $C$’s original password $pw$.
Step 3. If $S$ is not trusted, then $C$’s password $pw$ may be compromised with an attacker $A$ who may try to impersonate $C$ to login with $S$ as described below:
3.1. A selects a random number $r_A$, generates $\{r_A, pw\}K_s$ and sends the authentication message $(id, \{r_A, pw\}K_s)$ to $S$.

3.2. Then $S$ decrypts $\{r_A, pw\}K_s$ with his own private key, computes $H(pw)$ and compares it $H(pw)$ stored on the database. Since computed $H(pw)$ equal to the stored $H(pw)$, so $S$ selects a random number $r_S$ and replies with the message $(r_A \oplus r_S, H(r_S))$ to $A$.

3.3. Then $A$ retrieves $r_S$ by XORing $r_A$ with $r_A \oplus r_S$, computes $H(r_S)$ and compares it with received $H(r_S)$. Now $A$ computes the message $(id, H(r_A, r_S))$ and sends it back to $S$.

3.4. $S$ computes $H(r_A, r_S)$ from its own $r_S$ and received $r_A$ and compares it with received $H(r_A, r_S)$. Since computed $H(r_A, r_S)$ equal to received $H(r_A, r_S)$, so the server $S$ allows the attacker $A$ to access $C$’s account into the server $S$.

In addition, if the private key of $S$ is leaked accidentally to an adversary $A$, he can impersonate the client $C$ after revealing the $C$’s password $pw$ from the eavesdropped message $(id, \{r_A, pw\}K_s)$ sent by $C$ to $S$ during password authentication phase. Thus the Lin–Hwang’s scheme fails to protect this kind of impersonation attack. For clarity, the details of this attack are given in Fig. 3.

4.4. Many logged-in users’ attack

The many logged-in users’ attack is defined as the simultaneous access of a legitimate client’s account in a remote server by multiple adversaries using the same identity and password of the client. In Lin–Hwang’s scheme [9], the remote server $S$ stores identity, password-verifier pair $(id, H(pw))$ of the client $C$ to the database. Assume $C$’s legitimate id and $pw$ is accidentally exposed to the many adversaries $A_1, A_2, \ldots , A_m$, then all who knows $id$ and $pw$, can login to the remote server $S$, at the same time by executing the following steps:

Step 1. $A_1, A_2, \ldots , A_m$ choose random numbers $r_{A1}, r_{A2}, \ldots , r_{Am}$ and send the login requests $(id, \{r_{A1}, pw\}K_s), (id, \{r_{A2}, pw\}K_s), \ldots , (id, \{r_{Am}, pw\}K_s)$ to $S$ concurrently.

Step 2. $S$ decrypts all the messages $(id, \{r_{A1}, pw\}K_s), (id, \{r_{A2}, pw\}K_s), \ldots , (id, \{r_{Am}, pw\}K_s)$ and gets the same identity-password pair $(id, pw)$. Thus $S$ allows all of $A_1, A_2, \ldots , A_m$ to login and access $C$’s account concurrently.

This attack in Lin–Hwang’s scheme is further illustrated using a flow chart as given in Fig. 4.

4.5. Known session-specific temporary information attack

The detailed explanation about the known session-specific temporary information attack is given in [38–40]. Cheng et al. [41], Mandt and Tan [42] argued that if the session ephemeral secrets are exposed to an adversary $A$ accidentally, then some authentication mechanism must be incorporated in the session key distribution protocol such that this exposure should not compromise the resulting session key. According to the above discussions, we pointed out that Lin–Hwang’s
scheme cannot resist the known session-specific temporary information attack. For instance, in Lin and Hwang’s scheme, two ephemeral secrets $x$ and $y$ are selected by the client $C$ and server $S$ in each session respectively and compute the session key $SK = (g^x)^y = (g^y)^x = g^{xy}$. Now if these two ephemeral secrets $x$ and $y$ are compromised to $A$ by some means, then $A$ can easily compute the session key using $SK = (g^x)^y = g^{xy}$ or $SK = (g^y)^x = g^{yx}$. Hence, the Lin and Hwang’s scheme fails to prevent the known session-specific temporary information attack. Further, we explain this attack in Fig. 5.

5. Proposed ECC-based scheme

In this section, we proposed an elliptic curve cryptosystem based improved remote login scheme which provides the missing security provisions of Lin and Hwang’s scheme. The following notations are used throughout the proposed scheme (Table 2).

The proposed scheme consists of four phases—Registration phase, Password authentication phase, Password change phase, and Session key distribution phase. Now each of these phases is discussed below.

5.1. Registration phase

Initially, a client $A$ must register to the server $S$ with his own identity $ID_A$ and password-verifier $U_A$ and collects the server’s public key $U_S$, then server stores each legal client’s identity, password-verifier, and a status-bit in a write protected file as depicted in the Table 3, where the status-bit indicates the status of the client, i.e., when the client is logged-in to the server the status-bit is set to one, otherwise it is set to zero.

5.2. Password authentication phase

The password authentication protocol consists of following four steps:

**Step 1. Client → Server: $ID_A$, $E_K(IID_A, R_A, W_A)$.**

The client $A$, keys his identity $ID_A$ and the password $pu_A$ into the terminal. The client selects a random number $r_A$ from $[1, n - 1]$, computes $R_A = A \cdot U_S$ and $W_A = (r_A \cdot pu_A) \cdot G$. Then encrypts $(ID_A, R_A, W_A)$ using a symmetric key $K_x$ and sends it to the server. The encryption key $K_x$ is the $x$ coordinate of $K = pu_A \cdot U_S = pu_A \cdot d_S \cdot G = (K_x, K_y)$.  

Fig. 3. Impersonation attack in Lin–Hwang’s scheme.
Fig. 4. Many logged-in users’ attack in Lin–Hwang’s scheme.

Table 3
The verifier table.

<table>
<thead>
<tr>
<th>Identity</th>
<th>Password-verifier</th>
<th>Status-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_A</td>
<td>U_A = p w_A ⋅ G</td>
<td>0/1</td>
</tr>
<tr>
<td>ID_B</td>
<td>U_B = p w_B ⋅ G</td>
<td>0/1</td>
</tr>
<tr>
<td>ID_C</td>
<td>U_C = p w_C ⋅ G</td>
<td>0/1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Step 2. Server → Client: \((W_A + W_S), H(W_S)\).

The server computes the decryption key \(K_s\) by calculating \(K = d_S ⋅ U_A = p w_A ⋅ d_S ⋅ G = (K_s, K_s)\) and then decrypts \(E_{K_s}(ID_A, R_A, W_A)\) using \(K_s\). Subsequently, the server compares decrypted \(ID_A\) with received \(ID_A\) and \(\hat{e}(R_A, U_A)\) with \(\hat{e}(W_A, U_S)\). If all the conditions are satisfied, the server selects a random number \(r_S\) and computes \(W_S = r_S ⋅ U_S = r_S ⋅ d_S ⋅ G\). Then the server sends \((W_A + W_S)\) and \(H(W_S)\) to the client.

Step 3. Client → Server: \(ID_A, H(W_A, W_S)\).

The client retrieves \(W_S\) by subtracting \(W_A\) from \((W_A + W_S)\). If the hashed result of retrieved \(W_S\) is equal to the received \(H(W_S)\), then the client performs the hash operation \(H(W_A, W_S)\) and sends it to the server.


The server computes the hash value with own copies of \(W_S\) and \(W_A\) which is received from the client in step 2 and compares it with received \(H(W_A, W_S)\), to accept or denied the login request. If all of the conditions are satisfied then the server granted the client’s login request, otherwise rejects the client login request.

5.3. Password change phase

Step 1. Client → Server: \(ID_A, E_{K_s}(ID_A, R_A, W_A)\)

Step 2. Server → Client: \(W_A + W_S, H(W_S)\)

Step 3. Client → Server: \(ID_A, H(W_A, W_S), W_A + U'_A, H(W_S, U'_A)\)


If the client wants to change the old password \(p w_A\), to a new password \(p w'_A\), then the client computes the corresponding password-verifier \(U'_A = p w'_A ⋅ G\). In step 3, if the authentication token \(H(W_A, W_S)\) was authenticated, then server subtracted \(W_A\) from \(W_A + U'_A\), to extract the new password-verifier \(U'_A\). Now the server replaces \(U_A\) with \(U'_A\), if the hashed value of \((W_S, U'_A)\) and received \((H(W_S, U'_A)\) is same.
Fig. 5. Known session-specific temporary information attack in Lin–Hwang's scheme.

5.4. Session key distribution phase

Step 1. Client → Server: $ID_A, E_{EA}(ID_A, R_A, W_A)$
Step 2. Server → Client: $W_A + W_S, H(W_S)$
Step 3. Client → Server: $ID_A, H(W_A, W_S)$

In this protocol two random numbers $r_A$ and $r_S$ are chosen by the client and the server from $[1, n - 1]$. The client computes the final session key as $SK = (r_A \cdot pW_A) \cdot W_S = r_A \cdot r_S \cdot pW_A \cdot d_S \cdot G$ and the server computes $SK = (r_S \cdot d_S) \cdot W_A = r_A \cdot r_S \cdot pW_A \cdot d_S \cdot G$. Now, we illustrate the proposed ECC-based scheme in Fig. 6.

5.5. Correctness of the proposed scheme

All the proposed methods as given above followed a bilinear pairing that assures the correctness of the scheme. The proof of the bilinear pairing used, is given below. In order to proof $\hat{e}(R_A, U_A) = \hat{e}(W_A, U_S)$, we can rewrite

$\hat{e}(R_A, U_A) = \hat{e}(r_A \cdot d_s \cdot G, pw_A \cdot G) = \hat{e}(G, G)^{rA(pw_A)ds}$

$\hat{e}(W_A, U_S) = \hat{e}(r_A \cdot pw_A \cdot G, d_s \cdot G) = \hat{e}(G, G)^{rA(pw_A)ds}$

Therefore, $\hat{e}(R_A, U_A) = \hat{e}(W_A, U_S)$.

6. Security and efficiency analysis of the proposed scheme

In this section, the security analysis of the proposed scheme is given for the validation of our claim. Furthermore, the comparison of the proposed scheme with other related schemes is given for the performance study of our scheme.

6.1. Security analysis

The proposed scheme is free from all known cryptographic attacks and provides several security attributes as described below.
Fig. 6. The proposed ECC-based remote login scheme.
S1: Reply attack

Replay attack is an offensive action through which an adversary may impersonate the legitimate client by reusing the information obtained from a previous run protocol. In the proposed protocol, $W_A$ is encrypted by a secret symmetric key $K_s$ and it can be computed only by the server's and a legal client's secret. If the adversary wants to impersonate the legitimate client by replaying the older session message $(ID_A, E_{SK}(ID_A, R_A, W_A))$, but he cannot obtain $W_A$ and $W_S$, to know $W_A$ he has to compute the symmetric key $K = pw_A \cdot U_S = d_S \cdot U_A = (K_s, K_p)$, which is impossible, because the key can be computed from private key $d_S$ of server and password-verifier $U_A$ of the client or password $pw_A$ of the client and public key $U_S$ of the server. If the adversary replies with the wrong message $H(W'_A, W'_S)$ in step 3, but the server can detect it comparing with $H(W_A, W_S)$. So the proposed scheme can prevent this kind of replay attack.

S2: Password guessing attack

The password guessing attack is an important issue in any password-based remote user authentication scheme. In practice, the client tries to use the weak password (low intensity) for easy memorization. The weak password can be easily guessed by the adversary and using that password an adversary may impersonate the legal client. In the proposed protocol, the server stores the password-verifier $U_A = pw_A \cdot G$ to a write protected file and the adversary cannot extract the password $pw_A$ from $U_A$ as he has to solve the ECDLP [30], thus the password guessing attack is infeasible to the proposed scheme.

S3: Impersonation attack

Assume that an adversary makes an effort to impersonate the server to exchange a session key with the legal client. The adversary intercepted the message $E_{SK}(ID_A, R_A, W_A)$ of previous run protocol. Now it is impossible for the adversary to figure out $W_A$ from the message $E_{SK}(ID_A, R_A, W_A)$, because $(ID_A, R_A, W_A)$ is encrypted by a symmetric secret key $K_s$, known to the client and the server. Then the adversary replies with the wrong message $(W'_A + W'_S, H(W'_S))$ as in step 3 to the client (here $W'_A$ and $W'_S$ are randomly chosen by the adversary). Upon receiving the message $(W'_A + W'_S, H(W'_S))$, the client compares $H(W'_A + W'_S - W_A)$ with $H(W'_S)$ and they are not same. Therefore, the client terminates the key distribution protocol. Accordingly the impersonation attack is infeasible in the proposed scheme.

S4: Denial of service attack

The server closes a login session if the number of login attempts of an account by an incorrect password exceeds a limit value. Even so, such a client’s account is still workable and later login requests will pass as long as correct password is provided. In Step 3 of the proposed password change protocol, suppose the adversary replaces $(ID_A, H(W_A, W_S)), W_A + U_A, H(W_A, U'_A)$ with $(ID_A, H(W_A, W_S), X, H(W_A, U'_A))$ and sends the later message to the server, where $X$ is any arbitrary random elliptic curve point. On receiving the message $(ID_A, H(W_A, W_S), X, H(W_A, U'_A))$, the server computes $X - W_A$ and $H(W_S, X - W_A)$, and compares the later value with received $H(W_A, U'_A)$. But they are different, and therefore, the server rejects the password change protocol with a failure message to the client. Therefore, the proposed protocol has the capability to detect the denial of service attack.

S5: Many logged-in users’ attack

The proposed protocol can withstand the many logged-in users’ attack. Let us assume that the password $pw_A$ and the login-id $ID_A$ of a legal client $A$, are leaked to more than one adversary. But in the proposed scheme only one adversary can login the remote server at the same time out of all who know the valid password $pw_A$ and login-id $ID_A$. When an adversary logged-in by using the valid password $pw_A$ and login-id $ID_A$, then the server sets the status-bit to one and meanwhile if other adversaries try to login the server at the same time with same password $pw_A$ and login-id $ID_A$, the server denies all the requests because the status-bit indicates still someone is logged in.

S6: Server spoofing attack

Server spoofing attack means, an adversary may try to masquerade as a server to know the client’s long-term secret. The long-term secret is the client’s password and server’s private key. The symmetric key $K = pw_A \cdot U_S = d_S \cdot U_A = (K_s, K_p)$ cannot be computed without server’s secret key $d_S$ or password $pw_A$ of the client $A$. In Step 1 of password authentication protocol, an adversary cannot obtain $W_A$ by decrypting $(ID_A, E_{SK}(ID_A, R_A, W_A))$ with a wrong key, then the adversary cannot get success in Step 2. By chance if the adversary knows the value of $W_A$, but from it password extraction is impossible due to difficulties of ECDLP. Therefore, an adversary cannot get success in the proposed protocol by server spoofing attack.

S7: Perfect forward secrecy

Perfect forward secrecy means, if the private key of the server and the password of the client are compromised then the secrecy of previously established session keys should not be affected. Assume that the client’s password $pw_A$ and server’s secret key $d_S$ are known to an adversary. The adversary can compute $K = pw_A \cdot U_S = d_S \cdot U_A = (K_s, K_p)$ but he cannot derive the session key $SK = r_A \cdot r_S \cdot pw_A \cdot d_S \cdot G$. To know the session key $SK$, the adversary tries to compute it from the pair $(W_A, W_S) = (r_A \cdot pw_A \cdot G, r_S \cdot d_S \cdot G)$ directly, but it is impossible due to difficulties of computational Diffie–Hellman problem. In other words, if the current session key is leaked but from this disclosure the adversary is unable to draw all the past session keys, because the session key depends on random numbers $r_A$ and $r_S$. Hence the proposed protocol provides perfect forward security in the key distribution protocol.

S8: Insider attack

By stealing the password-verifier from the server’s verifier table, a privileged-insider of the server can access the other servers (where the client is registered with same identity and password) by making a valid login request. The proposed
scheme maintains a password-verifier table, which contains client’s identity ID, and password-verifier U = pA · G. Now it is computationally impossible to extract the password pA from the verifier U due to difficulties of ECDLP and hence the adversary who steals U cannot generate the symmetric key Ks. Thus the privileged-insider may not impersonate the legitimate client as he is unable to authenticate himself to the remote server without Ks and therefore, the insider attack is infeasible to the proposed scheme.

S9: Known session-specific temporary information attack

In our proposed scheme, after the successful password authentication, the client and the server computes the session key SK = rA · rS · pA · dS · G. Suppose that the ephemeral secrets rA and rS are exposed to an adversary. However, it is impossible derive the session key SK with the knowledge of rA and rS. Since the session key SK not only contains rA and rS but also contains the client’s password pA and server’s secret key dS. Therefore, to compute the session key the adversary have to know pA · dS · G. The computation of pA · dS · G from the pair (U, S) = (pA · G, dS · G) is equivalent to solve the CDHP, which is hard to solve by a polynomial time algorithm. Thus the known session-specific temporary information attack is not possible in the proposed scheme.

The Table 4 sums up certain cryptographic security attributes of the proposed scheme and some relevant schemes [2, 7, 9, 13], where it shows that our scheme prevents all related cryptographic attacks.

- **Impersonation attack in Zhu’s scheme**

  In Zhu’s scheme [13], the client makes the registration request to the remote server with the identity id and H(pw, s), where s is the salt generated by the client. Hence there is a possibility to compromise H(pw, s) with an outsider if the server is not trusted. The outsider who knows H(pw, s), performs the password authentication just by selecting a random number r′ w, a fresh timestamp T′ and then sends the authentication message (id, EKrc(r′ w · H(pw, s), T′)) to the remote server. The server then retrieves H(pw, s) after decrypting the message EKrc(r′ w · H(pw, s), T′), computes the hash value on H(pw, s) and compares with H(H(pw, s)) stored on server’s database. Therefore, the outsider successfully impersonates a legitimate client to login with remote server.

- **Many logged-in users’ attack in Peyravian–Zunic’s and Hwang–Yeh’s schemes**

  The Peyravian–Zunic’s [2] and Hwang–Yeh’s [7] schemes do not prevent the many logged-in users’ attack. In Peyravian–Zunic’s scheme [2], the server stores (id, H(id, pw)) in the database for a client having identity id and password pw. If client’s id and pw are leaked to multiple adversaries, then all who know id and pw follow the protected password transmission protocol of Peyravian–Zunic’s scheme and can access the client’s account concurrently to the same server in the same way as described in Section 4.4. Both the Hwang–Yeh’s scheme [7] and Lin–Hwang’s scheme [9] follow the same client registration and password authentication procedures and since, as shown in Section 4.4, the Lin–Hwang’s scheme cannot resist the many logged-in users’ attack, thus the Hwang–Yeh’s scheme is also vulnerable to this attack.

- **Insider attack in Peyravian–Zunic’s and Hwang–Yeh’s schemes**

  In practice, a client may register with a number of servers using same password pw and identity id. In Peyravian–Zunic’s scheme [2], the server maintains a record (id, H(id, pw)) against the client in a database. Now a privileged-insider of a remote server steals H(id, pw) and executes an off-line password guessing attack on it. Therefore, the privileged-insider gets exact password pw and accesses other servers where the client is registered as a legal client. So, the Peyravian–Zunic’s scheme is vulnerable to the insider attack. The Hwang–Yeh’s scheme [7] is also susceptible to the insider attack. In the scheme, the server stores (id, H(pw)), and a privileged-insider upon stealing H(pw), applies an off-line password guessing attack on H(pw) and can guess the exact password pw.

- **Known session-specific temporary information attack in Hwang–Yeh’s scheme**

  In Hwang–Yeh’s scheme [7], the client and the server, after the successful password authentication, compute the session key by applying some mutually agreed function to session ephemeral secrets rA and rS. If these two secrets rA and rS are leaked to an adversary by some means, then the resulting session key will be compromised. Therefore, the known session-specific temporary information attack is possible in Hwang–Yeh’s scheme.

6.2. Efficiency analysis

In this subsection, we summarize the following functional requirements which help to evaluate the efficiency of a remote user authentication scheme. Each of these constraints is a very crucial requirements for an efficient remote login scheme over the unreliable networks.

---

**Table 4**

Security comparisons of the proposed scheme with other remote login schemes.

<table>
<thead>
<tr>
<th>Security attributes/Schemes</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peyravian–Zunic [2]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>–</td>
<td>No</td>
<td>–</td>
</tr>
<tr>
<td>Hwang–Yeh [7]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Lin–Hwang [9]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Yes: Prevent the attack; No: Unable to prevent the attack; –: Not supported by the scheme; No [i] Proof. is given in [i].
Table 5
Functionality comparison of different remote login schemes with proposed scheme.

<table>
<thead>
<tr>
<th>Efficiency/Schemes</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
<th>E5</th>
<th>E6</th>
<th>E7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peyravian–Zunic [2]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Prevented</td>
<td>Not used</td>
<td>Smaller</td>
</tr>
<tr>
<td>Hwang–Yeh [7]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Prevented</td>
<td>Not used</td>
<td>Higher</td>
</tr>
<tr>
<td>Lin–Hwang [9]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Prevented</td>
<td>Not used</td>
<td>Higher</td>
</tr>
<tr>
<td>Zhu [13]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Not prevented</td>
<td>Used</td>
<td>Higher</td>
</tr>
<tr>
<td>Proposed</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Prevented</td>
<td>Not used</td>
<td>Smaller</td>
</tr>
</tbody>
</table>

Yes: Supported; No: Not supported.

E1: **Mutual authentication**
Client authentication may satisfy the necessary security requirements in the simple password-based remote login system. However, in many situations where the highly confidential data are exchanged between client and server, which means server authentication is also necessary to which client confidently communicates with the trusted server. Which indicates a mutual authentication is needed between client and server. The proposed scheme provides such requirements using three-way challenge–response handshake technique.

E2: **Choosing friendly password by the client**
Client can choose their password freely without any assistance from the remote server. In the proposed scheme an easy-to-remember (low intensity) password $pw$ is chosen by the client and the password-verifier $U = pwG$ is stored to the server’s database and from it $pw$ cannot be derived due to the difficulty of ECDLP.

E3: **Session key agreement**
The proposed scheme supports the session key agreement technique, which helps to establish common and secure session key between the client and the server in each session. With this session key, the client and the remote server can exchange highly confidential data between them securely.

E4: **Secure password change scheme**
A legitimate client can change their password after the registration phase. The proposed scheme has a secure (without DoS attack) password change scheme, i.e., the remote user can change/update their password any time.

E5: **Clock synchronization problem**
The problem of clock synchronization arises due to the time stamp used in a remote login system. Discard the timestamp to avoid this problem. The Zhu et al.’s [13] scheme used the timestamp to provide the security against the replay attack. However, the time stamp causes the clock synchronization problem, especially in the wide area network. It is better to include some self-verified mechanism in the protocol which can detect the replay attack. Accordingly, the proposed scheme and [2,7,9] passes up the usage of time stamp and prevents the clock synchronization problem by applying three-way challenge–response technique.

E6: **Extra hardware device**
The scheme [13] prevents the off-line password guessing attack by employing the salting technique. To protect the salt an extra hardware device, named trusted platform module (TPM) [14] is used. The salt file is encrypted by the Root of Trust for Storage (RTS) or a storage key which is encrypted by the RTS, and stored on the hard disk of the client’s system. The usage of TPM puts an extra burden trouble on the client. However, the proposed scheme and [2,7,9] provides fully software-based solution; no extra hardware device is needed.

E7: **Bandwidth requirements**
The proposed scheme is implemented with ECC, and the symmetric key encryption/decryption technique is used, whereas other schemes [7,9,13] used the public key encryption/decryption. As the symmetric key encryption is faster and produces the cipher text of fewer bits than any public key encryption [30]. Thus the length of the transmitted message in Step 1 of the proposed scheme is smaller than others [7,9,13]. Consequently, the bandwidth requirement of the proposed scheme is smaller than Hwang–Yeh, Lin–Hwang and Zhu et al.’s scheme.

We provide a comparative study of different functional requirements of some existing schemes [2,7,9,13] with the proposed scheme, as shown in Table 5. Note that our scheme satisfies all above-mentioned requirements.

The key distribution protocol of Lin and Hwang’s scheme uses Diffie–Hellman key exchange algorithm [44] and since the random challenges $g^x$ and $g^y$ requires executing modular exponential (MEXP), which is an expensive operation and the Lin and Hwang’s scheme applies public key encryption/decryption technique. The public key encryption/decryption is slower operation compared with symmetric key operation and the computation cost of elliptic curve point multiplication is much less than that of modular exponentiation [45]. This is because 160-bit ECDLP and 1024-bit discrete logarithm problem (DLP) have the same security level [30]. Therefore, the Lin and Hwang’s scheme has high computation cost. The proposed protocol reduces the communication, computation and storage space costs as the ECC and symmetric key encryption/decryption are used. It is to be noted that the proposed scheme uses the general cryptographic hash function and instead the XOR operation, elliptic curve multiplication/addition (EPM/EAD) is used, which is quite slower than XOR operation, but instead public key encryption (having slower processing speed) the symmetric key encryption (faster) is used. Therefore, overall computation cost of the proposed scheme is lower than others [7,9,13]. A comparative study in terms of the different
Table 6

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Operations used</th>
<th>Encryption/decryption</th>
<th>Overall computation cost</th>
<th>ECC is used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peyravian–Zunic [2]</td>
<td>Hash, XOR</td>
<td>Not used</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Hwang–Yeh [7]</td>
<td>Hash, XOR</td>
<td>Public key</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Lin–Hwang [9]</td>
<td>Hash, XOR, MEXP</td>
<td>Public key</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Zhu [13]</td>
<td>Hash, XOR</td>
<td>Public key</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Proposed</td>
<td>Hash, EPM, EAD</td>
<td>Symmetric key</td>
<td>Low</td>
<td>Yes</td>
</tr>
</tbody>
</table>

operations and encryption/decryption used and overall computation cost in different schemes such as Peyravian–Zunic [2], Hwang–Yeh [7], Lin–Hwang [9], Zhu et al. [13] and the proposed scheme is given in Table 6. From the security analysis and efficiency discussion, it is obvious that the proposed scheme is efficient, secure and user friendly.

7. Conclusion

In this paper, an ECC-based secure and efficient scheme for password authentication and update used in remote login system is proposed. A protocol for distribution of session key among the client and server is also proposed. It is found that the proposed scheme improves the Lin and Hwang’s scheme and also removes the security flaws of Zhu et al.’s scheme like impersonation attack, clock synchronization problem, etc. The proposed scheme supports the generation of the symmetric key, which can be used for confidential exchange of messages using symmetric key encryption technique. The security analysis of the proposed scheme is given and confirms the protection against all related attacks.

Acknowledgments

The authors would like to thank the anonymous reviewers and editors who helped to improve this work. The first author, SK Hafizul Islam, is working as a full-time research scholar (Reg. No 2010DR0007) in the Department of Computer Science and Engineering, Indian School of Mines, Dhanbad, under the DST INSPIRE fellowship, Reg. No. IF10247, Department of Science and Technology, Govt. of India. Also, the authors express their gratitude to the ISEA Project, No. MIT(2)/2006-08/189/CSE, Ministry of Information Technology, Govt. of India.

References


