Chapter 8

Password-based Remote User Mutual Authentication Scheme

Password based remote user authentication over unreliable network is a traditional method for accessing resources resides in a remote server. A variety of password authentication schemes for a number of applications, have been proposed in the open literature, however most of the schemes are found to be vulnerable to different kind of attacks. In this chapter, some security flaws in Lin and Hwang's password-based remote user mutual authentication scheme have been identified, and also similar types of attacks have been found in Peyravian and Zunic's, Hwang and Yeh's and Zhu et al.'s schemes. In remedy of these security drawbacks, an improved password-based remote user mutual authentication scheme based on elliptic curve cryptography is proposed. The security and efficiency analysis of the proposed scheme, and comparisons with other existing schemes demonstrate that our scheme performs well.

8.1 Introduction

The clients' authentication needs security for remote login while a client program tries to communicate with a server program over insecure networks like Internet. The identity and a secret password of a client are generally used for mutual authentication and access control, and if an efficient scheme is not followed, the password may be compromised during transmission. Also some systems in a hostile network need periodical change of the client's password for the protection of the valuable resources from an adversary, and until a secured password change protocol that allows the client to change his old password to a new one safely, the systems are not well protected. Based on public-key encryption, private-key encryption, hash function and their combinations, four basic approaches to design password-based authentication schemes are available, and in this chapter, we have presented a remote login scheme using private-key encryption technique.

As we know, the secured password authentication and update are two essential requirements for remote login over unreliable networks, and an ECC-based technique has been proposed here, which satisfies these two requirements and also provides additional security features that are not available in some schemes proposed so far. For instances, Peyravian and Zunic's scheme [61] 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 [62], Lee et al. [149], 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 [60], which has been analyzed in this chapter and certain security flaws have been identified. This chapter proposed an ECC-based password authentication and update schemes that removes the security flaws of Lin and Hwang's scheme and protects other attacks as well. As a proof of our claim, detail 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 for symmetric encryption, which requires lesser processing time than the time required in the public key encryption-based techniques.

The rest of the chapter is organized as follows. A brief review of the Lin and Hwang's scheme is given in Section 8.2. In Section 8.3, we discussed its weaknesses and a new scheme based on ECC has been proposed in Section 8.4. Security and efficiency analysis of the proposed scheme are given in Section 8.5 and 8.6, respectively, and Section 8.7 gives the concluding remarks.

8.2 Review of Lin and Hwang's Scheme

In this section, a brief description of the Lin and Hwang's scheme [60] that contains three parts: password authentication, password change and key distribution, where the following notations have been used (Table 8.1). Now the Lin and Hwang's scheme for password authentication, password change and distribution of secure session key are given below.

8.2.1 Password Authentication Protocol

It consists of the following steps:

Step 1. Client $\rightarrow$ Server: $id, \{r_C, pw\}_{K_S}$

Step 2. Server $\rightarrow$ Client: $r_C \oplus r_S, H(r_S)$

Step 3. Client $\rightarrow$ Server: $id, H(r_C, r_S)$

Step 4. Server $\rightarrow$ 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

Table 8.1: Notations are used in Lin-Hwang's scheme.

Notations	Description
id	Identity of the client, publicly known to all
pw	Secret password of a client
K_S	Public key of the server
$\{M\}K_S$	Encryption of the message M with the public key K_S
r_C, r_S	Random numbers chosen by the client and the server, respectively
q, g	A large prime number q, order of the group Z_q and g is the generator of Z_q^* where the Diffie-Hellman problem is considered to be hard
x, y	Random exponents chosen by the client and the server, respectively
$H(\cdot)$	Collision-resistant one-way secure hash function
$\oplus$	Bitwise XOR operator

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 $H(r_S)$ and sends back ($r_C \oplus r_S$, $H(r_S)$) to the client. After receiving ($r_C \oplus r_S$, $H(r_S)$) from the server, the client XORed r_C with $r_C \oplus r_S$ and retrieves r_S. The client compares if the hashed value of retrieved r_S and received $H(r_S)$, 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 own copies of r_S and r_C and compares it with received $H(r_C, r_S)$. If it is matched then the server sends a message 'Access granted' otherwise sends an error message 'Access Denied' to the client.

8.2.2 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: $r_C \oplus r_S, H(r_S)$

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.

If the client wants to change the old password pw to a new password new_pw, the client executes password change protocol. The password change protocol is almost similar to the password authentication protocol but, there is a minor difference in Step 3. In Step 3 of password change protocol, if the authentication token $H(r_C, r_S)$ is validated, the server computes $H(r_C + 1, r_S)$ and XORed it with $H(new\_pw) \oplus H(r_C + 1, r_S)$ to retrieve $H(new\_pw)$. Then the server replaces $H(pw)$ with $H(new\_pw)$, after validating the token $H(H(new\_pw), r_S)$.

8.2.3 Key Distribution Protocol

Let G_q is a finite cyclic group and g be the generator of order q, where q a large prime number. Let x, y are two elements of Z_q^* chosen by client and server, respectively and kept them secret for a session. The values of G_q, g, q and Z_q^* are made public. The key distribution protocol is given below.

Step 1. Client → Server: $id, \{g^x, pw\}_{K_S}$

Step 2. Server → Client: $g^x \oplus g^y, H(g^y)$

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

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

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

8.3 Weaknesses of Lin and Hwang's Scheme

The cryptanalysis of the Lin and Hwang's scheme [60] has been made in this section, and some of the common weaknesses are given below.

8.3.1 Stolen-verifier Attack

The stolen-verifier attack, which is described in [63, 172], means that an outsider theft the password-verifier from the server's database and applies an off-line guessing attack on it to get client's exact password and hence, he can impersonate the legitimate client. In Lin and Hwang's scheme, the client C registers to the remote server S with $(id, H(pw))$ and S then stores the pair $(id, H(pw))$ in the database. An outsider $\mathscr{A}$ can successfully find out C's password pw by performing the following procedure.

Step 1. $\mathscr{A}$ steals $H(pw)$ from 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. $\mathscr{A}$ guess a password pw'', computes $H(pw'')$ and then compares the result with stolen $H(pw)$.

Step 3. If $H(pw'') = H(pw)$, then $pw'' = pw$, i.e., $\mathscr{A}$ correctly guesses C's password. Otherwise, $\mathscr{A}$ 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 $|\cdot|$ represents the cardinality of the set. It is known that a client generally chooses a weak password (low intensity value) for easy memorization, so the space $|PW|$ is not large enough. Therefore, the stolen-verifier attack is possible in Lin and Hwang's scheme.

The stolen-verifier attack in Lin and Hwang's scheme is also illustrated in Fig. 8.1.

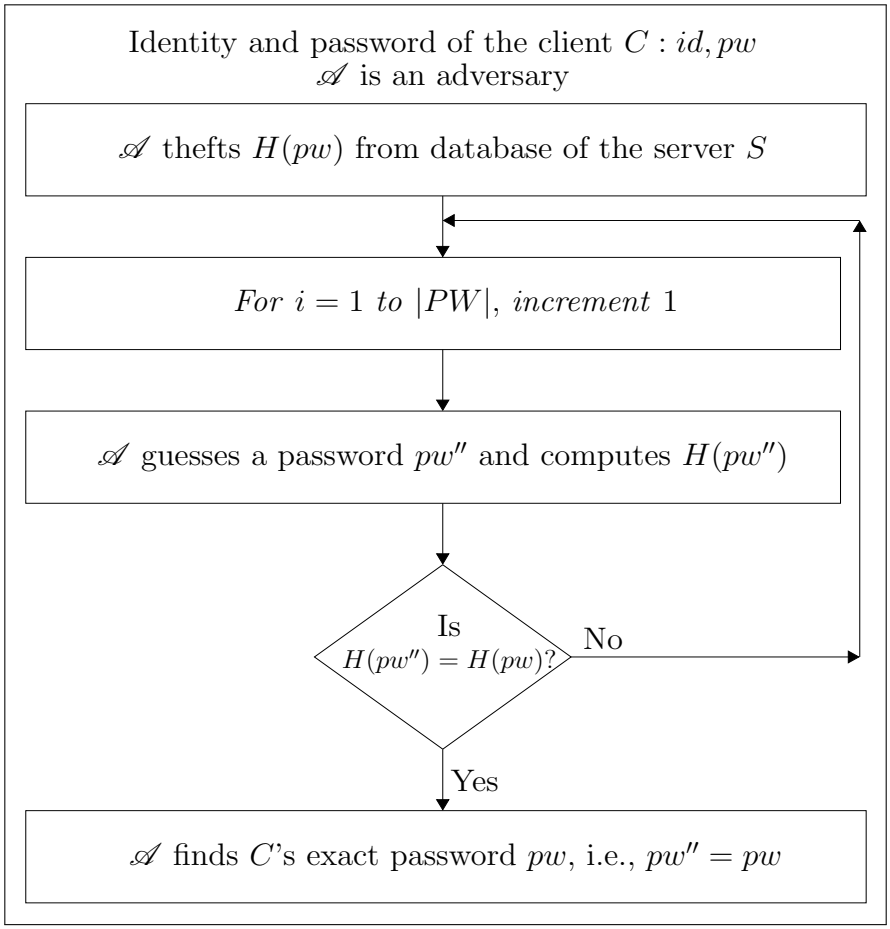

Figure 8.1: Stolen-verifier attack in Lin and Hwang's scheme.

8.3.2 Insider Attack

In insider attack as stated in [180, 215], a client C may register with a number of servers $S_1, S_2, \cdots, S_n$ using a common password pw and 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 $S_2, S_3, \cdots, S_n$ by using the same pw and id. In Lin and Hwang's scheme, initially the remote server stores the pair $(id, H(pw))$ of the client C in the database. Thus, the insider attack in Lin and 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 (Section 8.3.1).

Step 3. U_1 tries to use C's identity-password pair (id, pw), follows the password authentication protocol of Lin and Hwang's scheme and can easily login to the other remote servers $S_2, S_3, \cdots, S_n$.

The detailed description of this attack is given in the Fig. 8.2.

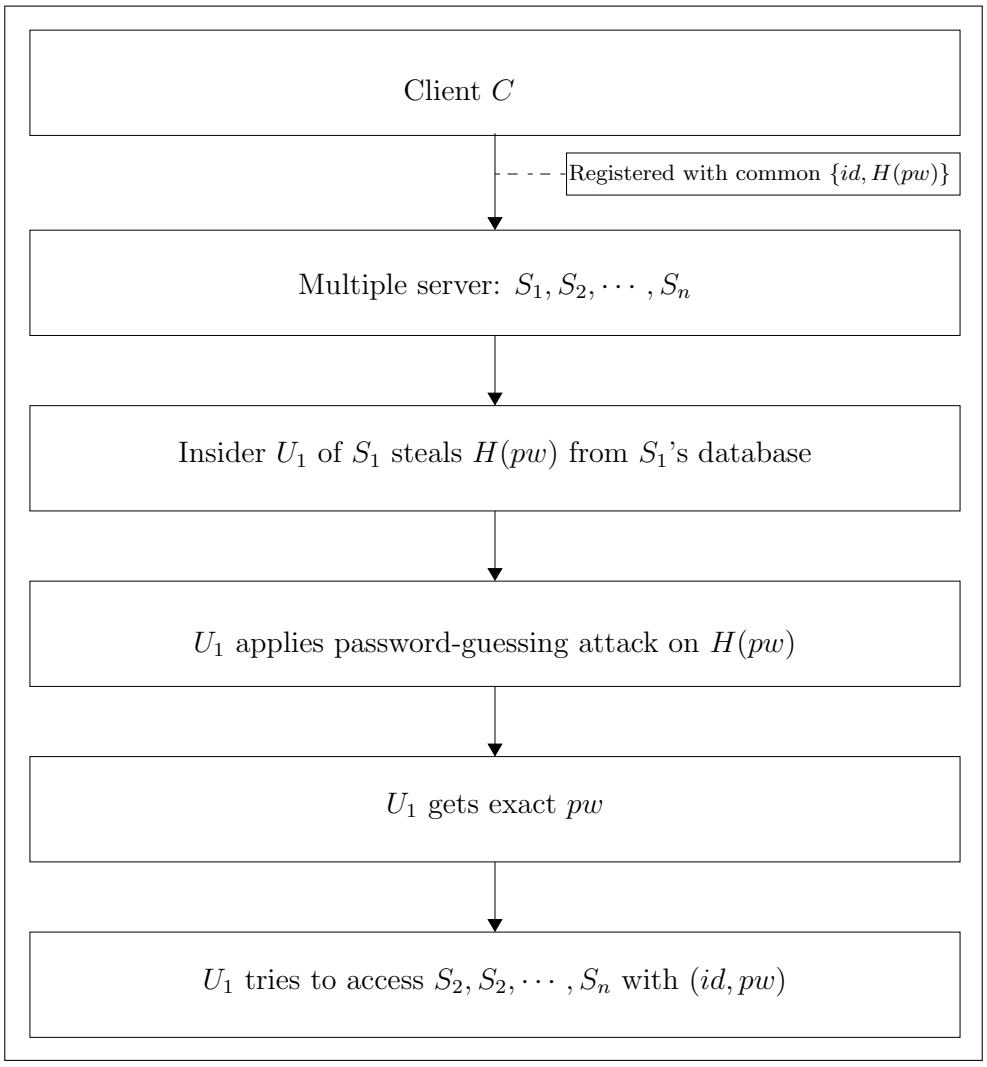

Figure 8.2: Insider attack in Lin and Hwang's scheme.

8.3.3 Impersonation Attack

According to the impersonation attack described in [62, 63], it is found that Lin and Hwang's scheme [60] 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. On receiving 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 $\mathscr{A}$ who may try to impersonate C to login with S as shown below:

1. $\mathscr{A}$ 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.
2. Then S decrypts $\{r_A, pw\}_{K_S}$ with his own private key, computes $H(pw)$ and compares it $H(pw)$ stored in the database. Since the computed $H(pw)$ equals 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 $\mathscr{A}$.
3. Then $\mathscr{A}$ 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 $\mathscr{A}$ computes the message $(id, H(r_A, r_S))$ and sends it back to S.
4. S computes $H(r_A, r_S)$ from his own r_S and received r_A and compares it with received $H(r_A, r_S)$. Since computed $H(r_A, r_S) =$ received $H(r_A, r_S)$, so the server S allows $\mathscr{A}$ to access C's account.

In addition, if the private key of S is leaked accidentally to $\mathscr{A}$, he can impersonate the client C after reveling C's password pw from the eavesdropped message $(id, \{r_A, pw\}_{K_S})$

sent by C to S during password authentication phase. Thus, Lin and Hwang's scheme fails to protect this kind of impersonation attack. For clarity, the details of this attack are given in Fig. 8.3.

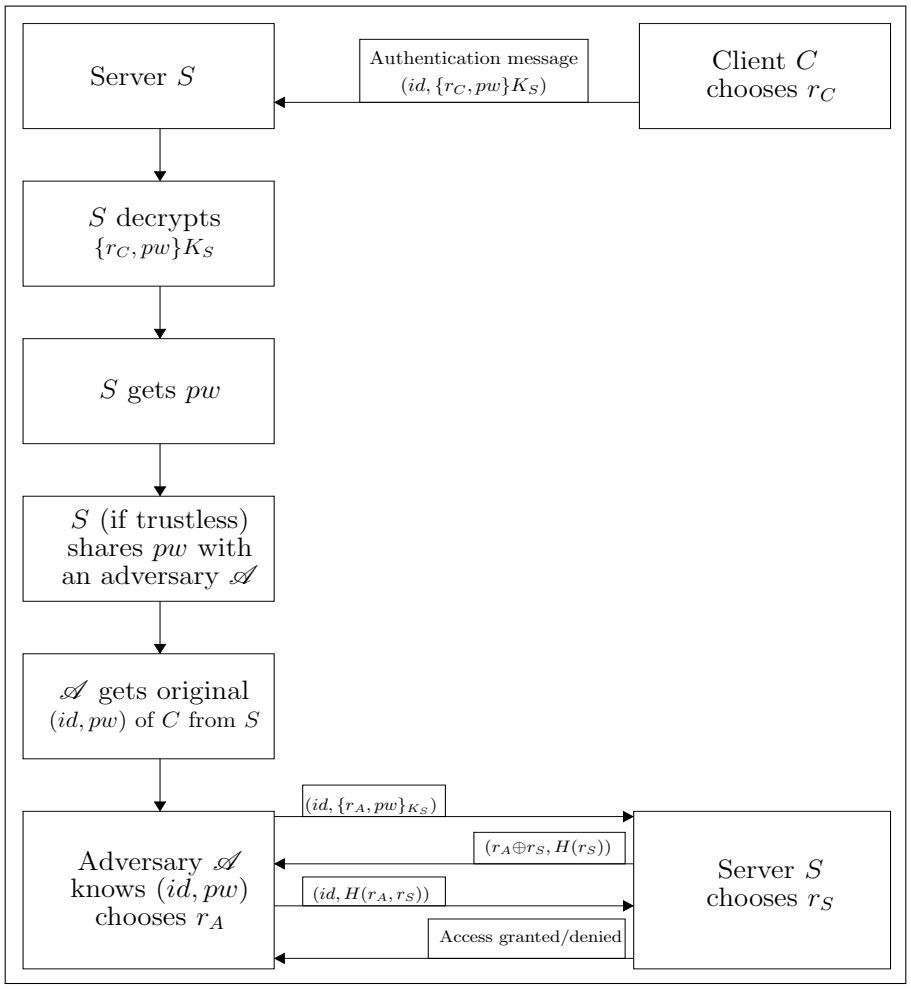

Figure 8.3: Impersonation attack in Lin and Hwang's scheme.

8.3.4 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 and Hwang's scheme, the remote server S stores identity, password-verifier pair $(id, H(pw))$ of the client C in the database. Assume that C's legitimate id and pw is accidentally exposed to many adversaries $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_m$,

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. $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_m$ choose random numbers $r_{A1}, r_{A2}, \cdots, r_{Am}$ and send the login requests $(id, \{r_{A1}, pw\}_{K_S})$, $(id, \{r_{A2}, pw\}_{K_S})$, $\cdots$, $(id, \{r_{Am}, pw\}_{K_S})$ to S concurrently.

Step 2. S decrypts all the messages $(id, \{r_{A1}, pw\}_{K_S})$, $(id, \{r_{A2}, pw\}_{K_S})$, $\cdots$, $(id, \{r_{Am}, pw\}_{K_S})$ and gets the same identity-password pair (id, pw). Thus, S allows all of $\mathscr{A}_1, \mathscr{A}_2, \cdots, \mathscr{A}_m$ to login and access C's account concurrently.

The many logged-in users' attack in Lin and Hwang's scheme is further illustrated using a flow chart as given in Fig. 8.4.

8.3.5 Known Session-specific Temporary Information Attack

The detailed explanation about the known session-specific temporary information attack (KSSTIA) is given in [94, 216, 217]. Cheng et al. [97], Mandt and Tan [218] argued that if the session ephemeral secrets are exposed to an adversary $\mathscr{A}$ 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 and 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 as $SK=(g^y)^x=(g^x)^y=g^{xy}$. Now if these two ephemeral secrets x and y are compromised to $\mathscr{A}$ by some means, then $\mathscr{A}$ can easily compute the session key using $SK = (g^x)^y$ or $SK = (g^y)^x$. Hence, the Lin and Hwang's scheme fails to prevent the known session-specific temporary information attack. Further, we explain this attack in Fig. 8.5.

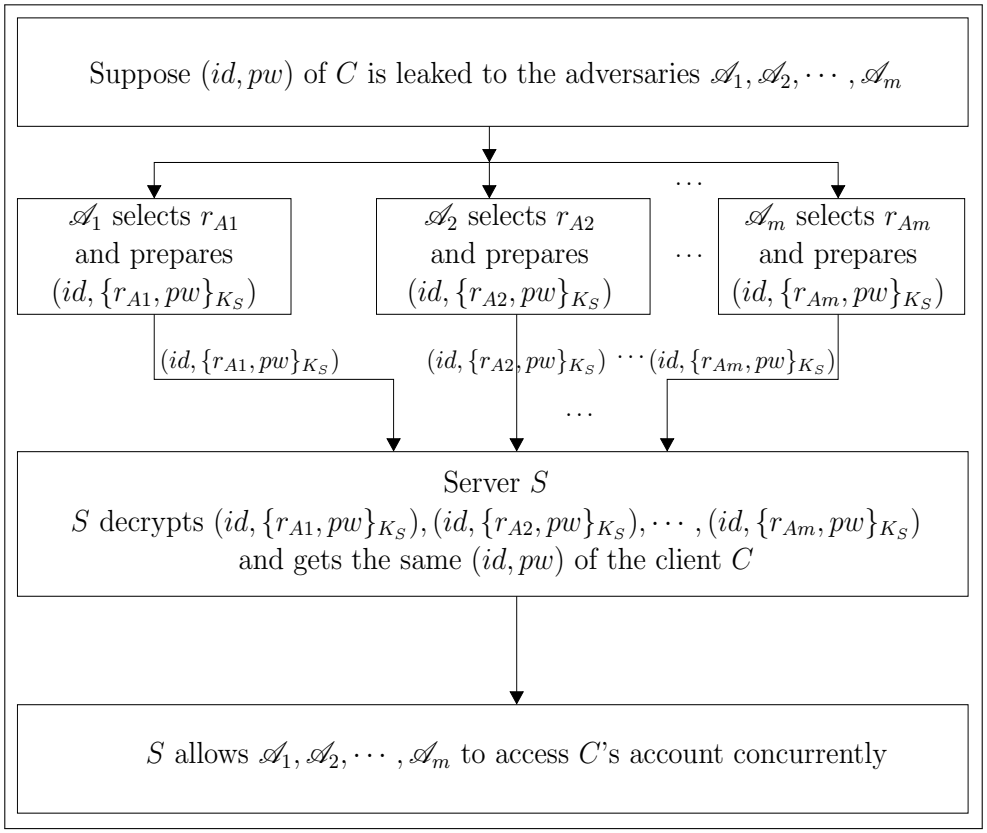

Figure 8.4: Many logged-in users' attack in Lin and Hwang's scheme.

8.4 Proposed ECC-based Scheme

In this section, we proposed an ECC-based improve remote user mutual authentication scheme, which provides the missing security provisions of Lin and Hwang's scheme. The following notations given in Table 8.2 are used through out the proposed scheme. 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.

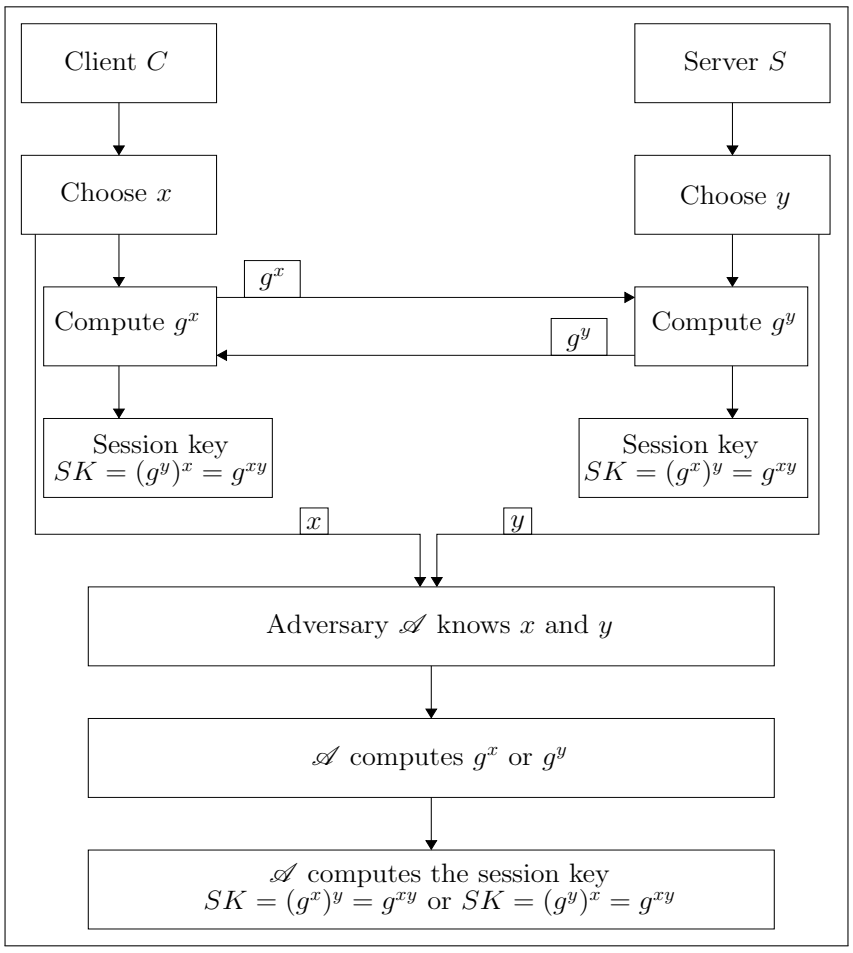

Figure 8.5: KSSTIA attack in Lin and Hwang's scheme.

8.4.1 Registration Phase

Initially, a client A must register in person to the server S with his own identity ID_A and password-verifier U_A and collects S's public key U_S, then S stores each legal client's *identity*, *password-verifier* and a *status-bit* in a written protected file as depicted in the Table 8.3, where the *status-bit* indicates the status of the client, i.e., when the client is logged-in to S then the *status-bit* is set to one, otherwise it is set to zero.

8.4.2 Password Authentication Phase

The password authentication protocol consists of following four steps:

Table 8.2: Notations are used in the proposed scheme.

Notations	Description
ID_A	Identity of the client A
pw_A	Secret password of the client A
d_S	Secret key of the server S
U_S	Public key of the server S, where $U_S = d_S P$
U_A	Password-verifier of the client A, where $U_A = pw_A P$
Kx	Secret key computed either using $K = pw_A U_S = pw_A d_S P = (Kx, Ky)$ or $K = d_S U_A = pw_A d_S P = (Kx, Ky)$
$E_{Kx}(\cdot)$	Symmetric encryption (e.g., AES) with Kx
P	Bases point of the elliptic curve group G_q (Section 3.1) of order n such that $nP = O$, where n is a large prime number
$H(\cdot)$	A collision-resistant one-way secure hash function (e.g., SHA-1)
r_A/r_S	Random numbers chosen by the client/server from [1, n-1] respectively
+/-	Elliptic curve point addition/subtraction

Table 8.3: Password-verifier table of the proposed scheme.

Identity	Password-verifier	Status-bit
ID_A	$U_A = pw_A P$	0/1
ID_B	$U_B = pw_B P$	0/1
ID_C	$U_C = pw_C P$	0/1
...	...	...

Step 1. Client $\rightarrow$ Server: $ID_A, E_{Kx}(ID_A, R_A, W_A)$.

The client A keys his identity ID_A and the password pw_A into the terminal. The client selects a random number r_A from $[1, n-1]$, computes $R_A = r_A U_S$

and $W_A = (r_A pw_A)P$. Then encrypts (ID_A, R_A, W_A) using the symmetric key Kx and sends it to the server. The encryption key Kx is the x coordinate of $K = pw_A U_S = pw_A d_S P = (Kx, Ky)$.

Step 2. Server → Client: $(W_A{+}W_S)$, $H(W_S)$.

The server computes the decryption key Kx by calculating $K = d_S U_A = pw_A d_S P = (Kx, Ky)$ and then decrypts $E_{Kx}(ID_A, R_A, W_A)$ using Kx. 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 $W_S = r_S U_S = r_S d_S P$. 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.

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

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's login request.

8.4.3 Password Change Phase

The password change phase of the proposed scheme is as follows:

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

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

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

Step 4. Server → Client: Password Change Granted/Denied.

If the client wants to change the old password pw_A, to a new password pw'_A, then the client computes the corresponding password-verifier $U'_A = pw'_A P$. 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. If the hashed value of (W_S, U'_A) and received $H(W_S, U'_A)$ is same, then the server replaces U_A with U'_A.

8.4.4 Session Key Distribution Phase

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

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

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

Step 4. Server → Client: Key distribution Granted/Denied.

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 $SK = (r_A pw_A)W_S = r_A r_S pw_A d_S P$ and the server computes $SK = (r_S d_S)W_A = r_A r_S pw_A d_S P$. Now, we illustrate the proposed ECC-based scheme in Fig. 8.6.

8.4.5 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

$\hat{e}(R_A, U_A) = \hat{e}(r_A d_S P, pw_A P) = \hat{e}(P, P)^{r_A pw_A d_S}$.

$\hat{e}(W_A, U_S) = \hat{e}(r_A pw_A P, d_S P) = \hat{e}(P, P)^{r_A pw_A d_S}$.

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

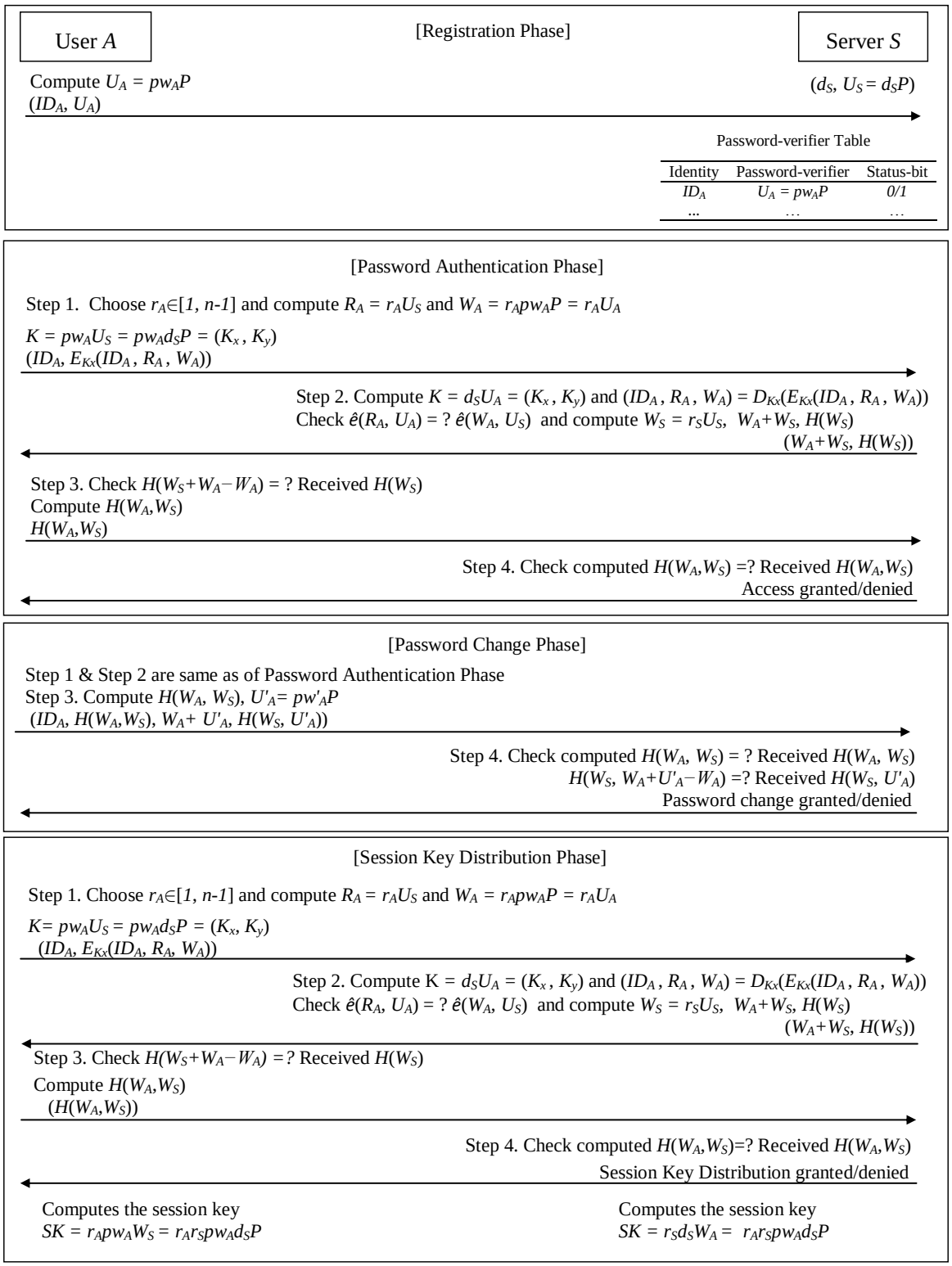

Figure 8.6: The proposed ECC-based remote login scheme

8.5 Security Analysis of the Proposed Scheme

In this section, the security analysis of the proposed scheme is given for the validation of our claim. The proposed scheme is free from all known cryptographic attacks and provides several security attributes as described below. Furthermore, the comparison of the proposed scheme with other related schemes is given for the performance study of the proposed scheme.

8.5.1 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 Kx, 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 $(ID_A, E_{Kx}(ID_A, R_A, W_A))$ but, he cannot obtain W_A and W_S, to know W_A he has to compute the symmetric key $K = pw_A U_S = d_S U_A = (Kx, Ky)$, which is impossible, because the key can be computed from the 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, the server detects it comparing with $H(W_A, W_S)$. Thus, the proposed scheme can prevent this kind of replay attack efficiently.

8.5.2 Off-line Password Guessing Attack

The off-line password guessing attack is an important issue in any password-based remote user authentication scheme. In practice, the client tries to use weak password (low intensity) for easy memorization. The weak password can be easily guessed by the adversary and using that password he may impersonate the legal client. In the proposed protocol, the server stores the password-verifier $U_A = pw_A P$ to a written

protected file and the adversary cannot extract the password pw_A from U_A as he has to solve the ECDLP [12], thus the off-line password guessing attack is infeasible in our scheme.

8.5.3 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 $E_{Kx}(ID_A, R_A, W_A)$ of previous run protocol. Now it is impossible for the adversary to figure out W_A from the message $E_{Kx}(ID_A, R_A, W_A)$, because (ID_A, R_A, W_A) is encrypted by a symmetric secret key Kx, 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 $H(W_A^*)$ and they are not equal. Therefore, the client terminates the key distribution protocol. Accordingly the impersonation attack is infeasible in the proposed scheme.

8.5.4 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 on login requests will pass as long as correct password is provided. In Step 3 of the proposed password change protocol, suppose that the adversary replaces $(ID_A, H(W_A, W_S), W_A+U'_A, H(W_S, U'_A))$ with $(ID_A, H(W_A, W_S), X, H(W_S, U'_A))$ and sends the latter message to the server, where X is a random elliptic curve point. On receiving the message $(ID_A, H(W_A, W_S), X, H(W_S, U'_A))$, the server computes $X-W_A$ and $H(W_S,\ X-W_A)$, and compares the latter value with received $H(W_S, U'_A)$. 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 this kind of denial of service attack.

8.5.5 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 knows 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 identity-password pair (ID_A, pw_A), the server denies all the requests because the *status-bit* indicates still someone is logged-in.

8.5.6 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 $K = pw_A U_S = d_S U_A = (Kx, Ky)$ 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 $(ID_A, E_{Kx}(ID_A, R_A, W_A))$ 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.

8.5.7 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 $K = pw_A U_S = d_S U_A = (Kx, Ky)$ and thus figure out W_A and W_S from the messages $(ID_A, E_{Kx}(ID_A, R_A, W_A))$ and $(W_A + W_S, H(W_S))$ but, he cannot derive the session key $SK = r_A r_S pw_A d_S P$. To know the session key SK, the adversary tries to compute it from the pair $(W_A, W_S) = (r_A pw_A P, r_S d_S P)$ directly but, it is impossible due to difficulties of CDHP. 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.

8.5.8 Insider Attack

By stealing the password-verifier from the server's verifier table, a privileged-insider of the server can access 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_A and password-verifier $U_A = pw_A P$. Now it is computationally impossible to extract the password pw_A from the verifier U_A due to difficulties of ECDLP and hence the adversary who steals U_A, cannot generate the symmetric key Kx. Thus, the privileged-insider may not impersonate the legitimate client as he is unable to authenticate himself to the remote server without Kx and therefore, the insider attack is infeasible to the proposed scheme.

8.5.9 Known Session-specific Temporary Information Attack

In our proposed scheme, after successful password authentication, the client and the server computes the session key $SK = r_A r_S pw_A d_S P$. Suppose that the ephemeral secrets r_A and r_S are exposed to an adversary. However, it is impossible to derive the session key SK with the knowledge of r_A and r_S. Since the session key SK not only

contains r_A and r_S but, also contains client's password pw_A and server's secret key d_S. Therefore, to compute the session key, the adversary has to know $pw_A d_S P$. The computation of $pw_A d_S P$ from the pair $(U_A, U_S) = (pw_A P, d_S P)$ is equivalent to solve the CDHP, which is hard to solve by a polynomial-time bounded algorithm. Thus, the known session-specific temporary information attack is not possible in our scheme.

Now we discussed the vulnerabilities of Peyravian and Zunic's scheme [61], Hwang and Yeh's scheme [62] and Zhu et al.'s scheme [63] against some common attacks as shown below:

- **Impersonation Attack in Zhu et al.'s Scheme:** In Zhu et al.'s scheme [63], 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'', a fresh timestamp T'' and then sends the authentication message $(id, E_{Ks}(r'', H(pw, s), T'')$ to the remote server. The server then retrieves $H(pw, s)$ after decrypting the message $E_{Ks}(r'', H(pw, s), T'')$, 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 and Zunic's Scheme and Hwang and Yeh's Scheme:** Peyravian and Zunic's scheme [61], and Hwang and Yeh's scheme [62] do not prevent the many logged-in users' attack. In Peyravian and Zunic's scheme, 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 and Zunic's scheme and can access the client's account concurrently to the same server in the same way as described in Section 8.3.4. Both the Hwang and

Yeh's scheme [62], and Lin and Hwang's scheme [60] follow the same client registration and password authentication procedures and since, as shown in Section 8.3.4, Lin and Hwang's scheme cannot resist the many logged-in users' attack, thus the Hwang and Yah's scheme is also vulnerable to this attack.

- **Insider Attack in Peyravian and Zunic's Scheme and Hwang and Yeh's Scheme:** In practice, a client may register with a number of servers using same password pw and identity id. In Peyravian and Zunic's scheme [61], 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, Peyravian and Zunic's scheme is vulnerable to the insider attack. Hwang and Yeh's scheme [62] 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 and Yeh's Scheme:** In Hwang and Yah's scheme [62], the client and the server, after successful password authentication, compute the session key by applying some mutually agreed function to session ephemeral secrets r_C and r_S. If these two secrets r_C and r_S 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 and Yah's scheme.

The Table 8.4 sums up certain cryptographic security attributes of the proposed scheme and some relevant schemes [60, 61, 62, 63], where it shows that the proposed scheme prevents all related cryptographic attacks.

Table 8.4: Security comparisons of the proposed scheme with others.

Schemes	8.5.1	8.5.2	8.5.3	8.5.4	8.5.5	8.5.6	8.5.7	8.5.8	8.5.9
Peyravian & Zunic [61]	Yes	No [149]	No [62]	No [219]	No	No [62]	-	No	-
Hwang & Yeh [62]	No [152]	No [63]	No [63]	No [60]	No	Yes	No [60]	No	No
Lin & Hwang [60]	Yes	No	No	Yes	No	Yes	Yes	No	No
Zhu et al. [63]	Yes	Yes	No	Yes	Yes	Yes	-	Yes	-
Proposed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Yes: Prevent the attack; **No:** Unable to prevent the attack; - : Not supported by the scheme; **No [i]:** Proof. is given in Ref. [i].									

8.6 Efficiency Analysis of the Proposed Scheme

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 very crucial requirements for an efficient remote login scheme.

8.6.1 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. That is, it indicates a mutual authentication is needed between client and server. The proposed scheme provides such requirements using three-way challengeresponse handshake technique.

8.6.2 Choosing Friendly Password by 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_A is chosen by the client and the password-verifier $U_A = pw_A P$ is stored in the server's database and from it pw_A cannot be derived due to the difficulty of ECDLP.

8.6.3 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.

8.6.4 Secure Password Change

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 safely.

8.6.5 Clock Synchronization Problem

The problem of clock synchronization arises due to the timestamp used in a remote login system. Discard the timestamp to avoid this problem. Zhu et al.'s scheme [63] used the timestamp to provide security against the replay attack. However, the timestamp 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 the schemes devised in [60, 61, 62] passes up the usage of timestamp and prevents the clock synchronization problem by applying three-way challengeresponse handshake technique.

8.6.6 Extra Hardware Device

The scheme [63] 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) [156] 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 in the hard disk of the client's system. The usage of TPM puts an extra burden on the client. However, the proposed scheme and the schemes designed in [60, 61, 62] provide fully software-based solution; no extra hardware device is needed.

8.6.7 Bandwidth Requirements

The proposed scheme is implemented with ECC, and the symmetric key encryption/decryption technique is used, whereas other schemes [60, 62, 63] 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 technique [12]. Thus, the length of the transmitted message in Step 1 of the proposed scheme is smaller than others [60, 62, 63]. Consequently, the bandwidth requirement of the proposed scheme is smaller than Hwang and Yeh's, Lin and Hwang's and Zhu et al.'s schemes.

We provide a comparative study of different functional requirements of some existing schemes [60, 61, 62, 63] with the proposed scheme, as shown in Table 8.5. 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 [86] and since the random challenges g^x and g^y require executing modular exponential (MEXP), which is an expensive operation and Lin and Hwang's scheme applies public key encryption/decryption technique. The public key encryption/decryption is slower operation compared with symmetric key encryption/decryption operation and the computation cost of elliptic curve scalar point multiplication is much less than that of modular exponentiation [194]. This is because 160-bit ECDLP and

Table 8.5: Functionality comparisons of different remote login schemes and ours.

Efficiency/ Schemes	8.6.1	8.6.2	8.6.3	8.6.4	8.6.5	8.6.6	8.6.7
Peyravian & Zunic [61]	Yes	Yes	No	Yes	Prevented	Not used	Smaller
Hwang & Yeh [62]	Yes	Yes	Yes	Yes	Prevented	Not used	Higher
Lin & Hwang [60]	Yes	Yes	Yes	Yes	Prevented	Not used	Higher
Zhu et al. [63]	Yes	Yes	No	Yes	Not Prevented	Used	Higher
Proposed	Yes	Yes	Yes	Yes	Prevented	Not used	Smaller
Yes: Supported; **No:** Not supported.							

1024-bit discrete logarithm problem (DLP) have the same security level [12]. 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 scalar point multiplication/elliptic curve point addition (ECMP/ECPA) 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 [60, 62, 63]. A comparative study in terms of the different operations and encryption/decryption used and overall computation cost in different schemes such as Peyravian and Zunic [61], Hwang and Yeh [62], Lin and Hwang [60], Zhu et al. [63] and the proposed scheme is given in Table 8.6. From the security analysis and efficiency discussion, it is obvious that the proposed scheme is efficient, secure and user friendly.

Table 8.6: List of different operations and encryption/decryption is used and the overall computation cost of different schemes.

Schemes	Operations used	Encryption/ Decryption	Overall computation cost	ECC is used
Peyravian & Zunic [61]	Hash, XOR	Not used	Low	No
Hwang & Yeh [62]	Hash, XOR	Public key	High	No
Lin & Hwang [60]	Hash, XOR, MEXP	Public key	High	No
Zhu [63]	Hash, XOR	Public key	High	No
Proposed	Hash, ECMP, ECPA	Symmetric key	Low	Yes

8.7 Chapter Summery

In this chapter, an ECC-based secure and efficient scheme for password authentication and update used in remote login system is proposed. A protocol for generation of session key among the client and server is also proposed. It is found that the our scheme improves 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 that can be used for confidential exchange of messages using symmetric key encryption technique. The security analysis of our scheme is given and confirms the protection against all attacks.

The next chapter i.e., Chapter 9 designed a dynamic identity-based remote login scheme that offers remote user mutual authentication, session key agreement, leaked key revocation, etc.