A Secure Remote Mutual Authentication and Key Agreement without Smart Cards

1,2Han-Cheng Hsiang and 1Wei-Kuan Shih
1Department of Computer Science, National Tsing Hua University, Taiwan, Republic of China
2Department of Information Management, Vanung University, Taiwan, Republic of China

Abstract: This study proposes a new and secure scheme for remote mutual authentication without using the smart cards. The scheme may satisfy all of the essential security requirements. In the last couple of decades the Internet technology has advanced so rapidly. It leads to the spreading and penetration of the technology to the network services and applications. Remote user authentication is a very effective means to check the legality of a user. Among many schemes, password authentication has been commonly used. Also in many schemes proposed for the remote user authentication, smart card has been intensively used to store the secret information for authentication. However, the smart card and its reader are not always available everywhere and in anytime. With this scheme, the user can login to the remote server from anywhere and in anytime to access the secure service. This may be more practical and easy-to-use.

Key words: Password authentication, security attacks, information security, communications

INTRODUCTION

Remote user authentication scheme allows a server to check the legality of a remote user through open network. In addition, a smart card based remote mutual authentication scheme is very efficient to authenticate remote users (Juang, 2006; Wang et al., 2007). Lamport (1981) proposed the first well-known password based remote user authentication scheme rooted on verifier table, but this scheme was vulnerable to stolen verifier attack. Hwang et al. (1990) initially proposed a non-interactive password authentication scheme using smart cards, without storing verifier table in the server. Since then Tan and Zhu (1999), Yang and Shieh (1999), Hwang and Li (2000) and Sun (2000) have proposed new schemes to improve the efficiency and the security of remote authentication. Chien et al. (2002) proposed an efficient remote mutual authentication scheme using smart card allowing server and user to authenticate each other. However, Chien et al. (2002) scheme was vulnerable to the parallel session attack (Hsi, 2004). Later, Juang (2004) proposed another scheme preserving all the merits of Chien et al. (2002) scheme. Juang (2004) scheme is nonce based authentication and key agreement scheme. No synchronized clocks are required in the scheme. Besides, Juang (2004) scheme generates a session key for the user and server in their following communication.

Recently, Shieh and Wang (2006) pointed out the security flaws of Juang (2004) scheme and then proposed an improvement to remedy the security flaws. However, most of the user authentication schemes were designed using smart cards. In practice, card readers are not available everywhere. In particular, it is difficult to produce tamper-resistant smart cards. Research shows that the secret information kept in smart cards can be cracked by analyzing the dripped messages or the power consumption (Messerges et al., 2002). This study proposes a secure and efficient password authentication protocol without the use of smart cards. The proposed scheme fulfills the following requirements, which are regarded as important criteria for password authentication (Sun, 2000; Juang, 2004; Lee et al., 2008).

Requirement 1: Freely chosen password: The user can freely choose his password for easy memorization. This requirement is essential to make the password authentication protocol more user-friendly.

Requirement 2: Security: Security is the most important issue of any password authentication protocol. The password authentication scheme must be able to resist various kinds of attacks such that it can be applied in the real world.

Requirement 3: Session key agreement: The legal user and the server should be able to negotiate a session key to protect the transmitting messages after successful authorization.
Requirement 4: Mutual authentication: The authentication scheme allows the users and the remote server to authenticate each other. It is necessary to protect not only the server but also the legal user from malicious attacks.

This study presents a secure and efficient mutual authentication scheme without smart cards. The proposed scheme providing perfect forward secrecy, can withstand various malicious attacks.

BACKGROUND

Perfect forward secrecy: For evaluating a strong protocol, perfect forward secrecy is considered to be an important security issue. A protocol providing perfect forward secrecy means that even if one entity’s long-term secret key is compromised, it will never reveal any old short-term keys used before (Menezes et al., 1997; Sun and Yeh, 2006). For example, the well-known Diffie-Hellman key agreement scheme (Diffie and Hellman, 1976) can provide perfect forward secrecy.

Diffie-Hellman problem: Given a prime p, a generator g and two numbers g^l mod p and g^r mod p, one tries to find g^{l+r} mod p, it is believed infeasible to solve in polynomial time (Diffie and Hellman, 1976).

RELATED WORKS

Here, briefly reviews Juang’s and Shieh-Wang’s scheme. The notations are used throughout here.

U: The user
S: The server
ID: The identification of U
PW: The password of U
h(·): Secure one-way hash function
x: The secret key maintained by the server
E_k(m): The encryption function of the message m with the encryption key k
D_k(m): The decryption function of the message m with the decryption key k
SK: The session key shared between U_i and S for this protocol run
⊕: Exclusive-or operation
∥: String concatenation operation
→: A common channel

Review of Juang’s scheme: Juang’s (2004) scheme is based on the symmetric encryption. The scheme consists of two phases: the registration phase and the login and session key agreement phase.

Registration phase: Assume a user U_i submits his identity ID_i and password PW_i to the server over a secure channel for registration. The server computes \( V_i = h(\text{ID}_i, x) \), \( W_i = V_i \oplus PW_i \), and issues U_i a smart card containing \( W_i, \text{ID}_i \) and \( h(\cdot) \).

Login and session key agreement phase: When U_i wants to login to the server, he (including she) inserts his smart card into a card reader and inputs his identity ID_i and password PW_i.

Step 1: \( U_i \rightarrow S_i: E_{W_i}(s_{i}, N_{i}+1, N_s) \).
The smart card computes \( V_i = W_i \oplus PW_i \) then sends the message \( (N_i, \text{ID}_i, E_{W_i}(s_{i}, C_i)) \) to the server S, where, \( C_i = h(\text{ID}_i || N_i) \), \( E_{W_i}(\cdot) \) denotes a symmetric encryption algorithm using \( V_i \) as the secret key, \( N_i \) is a nonce and \( s_{i} \) is a random value chosen by the smart card to generate the session key SK.

Step 2: \( S \rightarrow U_i: E_{W_i}(s_{i}, N_{i}+1, N_s) \).
After receiving the message, S computes \( V_i = h(\text{ID}_i, x) \) and \( (s_{i}, C_i) = D_{W_i}(E_{W_i}(s_{i}, C_i)) \), where, \( D_{W_i}(\cdot) \) denotes the corresponding symmetric decryption algorithm of \( E_{W_i} \) using \( V_i \) as the secret key. After decryption, if \( C_i \) is not equal to \( h(\text{ID}_i, || N_i) \), or \( N_s \) is not fresh, the server rejects \( U_i \)'s request. Otherwise, the server sends the message \( E_{W_i}(s_{i}, N_{i}+1, N_s) \) to \( U_i \), where, \( N_i \) is a nonce and \( s_{i} \) is a random value chosen by the server to generate the session key SK.

Step 3: \( U_i \rightarrow S_i: E_{W_i}(N_i+1) \).
When \( U_i \) receives the message, the smart card decrypts and checks whether \( N_i+1 \) is in it. If yes, the smart card computes the session key \( SK = h(\text{ID}_i, || N_i) \), or \( N_s \) is not fresh, the server rejects \( U_i \)'s request. Otherwise, the server sends the message \( E_{W_i}(N_i+1) \) back to S.

Step 4: On receiving the last message, S computes \( D_{W_i}(E_{W_i}(N_i+1)) \) to check whether \( N_i+1 \) is in it. If \( N_i+1 \) is found, S and \( U_i \) have achieved mutual authentication and session key agreement.

Review of Shieh-Wang’s scheme: Shieh and Wang’s (2006) scheme consists of three phases: the registration phase, the login phase and the authentication and key agreement phase. The scheme works as follows:

Registration phase: Assume a user \( U_i \) submits his identity ID_i and password PW_i to the server over a secure channel for registration. If the request is accepted, the
server computes \( R_i = h(ID_i \oplus x) \oplus PW_i \) and issues \( U_i \) a smart card containing \( R_i \) and \( h(\cdot) \).

**Login phase:** When the user \( U_i \) wants to login to the server, he first inserts his smart card into a card reader then inputs his identity \( ID_i \) and password \( PW_i \). The smart card then performs the following steps to begin an access session:

**Step 1:** Compute \( a_i = R_i \oplus PW_i \).

**Step 2:** Acquire current time stamp \( T_w \) store \( T_a \) temporarily until the end of the session and compute \( MAC_w = h(T_w \| a_i) \).

**Step 3:** \( U_i \rightarrow S: ID_i, T_w, MAC_w \)
Send the message \( \{ID_i, T_w, MAC_w\} \) to the server and wait for response from the server. If no response is received in time or the response is incorrect, report login failure to the user and stop the session.

**Authentication and key agreement phase:** After receiving the message \( \{ID_i, T_w, MAC_w\} \) from \( U_i \), the server performs the following steps to assure the integrity of the message:

**Step 1:** Check the freshness of \( T_w \). If \( T_a \) has already appeared in a current executing session of user \( U_i \), reject \( U_i \)'s login request and stop the session. Otherwise, \( T_a \) is fresh.

**Step 2:** Compute \( a'_i = h(ID_i \oplus x) \), \( MAC'_w = h(T_w \| a'_i) \) and check whether, \( MAC'_w \) is equal to the received \( MAC_w \). If it is not, reject \( U_i \)'s login and stop the session.

**Step 3:** \( S \rightarrow U_i : T_w, MAC_w \)
Acquire the current time stamp \( T_w \). Store temporarily paired time stamps \( (T_w, T_a) \) and \( ID_i \) for freshness checking until the end of the session. Compute \( MAC'_i = h(T_a \| T_w \| a'_i) \) and session key \( SK = h((T_a \| T_w) \oplus a'_i) \). Then, send the message \( \{T_w, T_a, MAC'_i\} \) back to \( U_i \) and wait for response from \( U_i \). If no response is received in time or the response is incorrect, reject \( U_i \)'s login and stop the session.

**Step 4:** On receiving the message \( \{T_w, T_a, MAC'_i\} \) from the server, the smart card checks if the received \( T_a \) is equal to the stored \( T_a \) to assure the freshness of the received message. If it is not, report login failure to the user and stop the session.

**Step 5:** Compute \( MAC'_w = h(T_w \| a_i) \) and check whether it is equal to the received \( MAC_w \). If not, report login failure to the user and stop. Otherwise, conclude that the responding party is the real server.

**Step 6:** \( U_i \rightarrow S : T_w, MAC_w \)
Compute \( MAC''_w = h(T_w \| (a_i + 1)) \) and session key \( SK = h((T_a \| T_w) \oplus a_i) \), then send the message \( \{T_w, MAC''_w\} \) back to the server. Note that, in the message \( \{T_w, MAC''_w\} \), \( T_w \) is a response to the server.

**Step 7:** When the message \( \{T_w, MAC''_w\} \) from \( U_i \) is received, the server checks if the received \( T_w \) is equal to the stored \( T_w \). If it fails, reject \( U_i \)'s login request and stop the session.

**Step 8:** Compute \( MAC'''_w = h(T_w \| (a_i + 1)) \) and check whether it is equal to \( MAC''_w \). If it is not, reject \( U_i \)'s login request and stop the session. Otherwise, conclude that \( U_i \) is a legal user and permit the user \( U_i \)'s login.

At this moment, mutual authentication and session key agreement between \( U_i \) and the server are achieved. From now on, the user \( U_i \) and the server can use the session key \( SK \) in their further secure communication until the end of the access session.

**The Proposed Scheme**

Here, we propose a new remote mutual authentication and key agreement scheme without smart cards. The flowchart of the new protocol is shown in Fig. 1. Before describing the details of present protocol, we first list the notations as follows.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h(\cdot) )</td>
<td>A collision-resistant one-way hash function</td>
</tr>
<tr>
<td>( x )</td>
<td>The secret key maintained by the server</td>
</tr>
</tbody>
</table>

![Fig. 1: The flowchart of the proposed scheme](image)

335
p : A large prime number

\( g \) : A primitive element in GF(p)

\( E_k(m) \) : The encryption function of the message m with the encryption key k

\( D_k(m) \) : The decryption function of the message m with the decryption key k

\( SK \) : The session key shared between \( U_i \) and \( S \) for this protocol run

\( \oplus \) : Exclusive-or operation

\( || \) : String concatenation operation

\( \sim \) : A common channel

The proposed scheme consists of three phases: the registration phase, the login phase and the authentication and key agreement phase. The scheme works as follows:

**Registration phase:** Assume a user \( U_i \) selects a random number \( b, \) password \( PW \), and computes \( h(b, PW) \). He submits his identity \( ID_e \) and \( h(b, PW) \) to the server over a secure channel for registration. If the request is accepted, the server computes \( c_e = h(ID_e || x) \) and \( R_e = c_e \oplus h(b, PW) \). Then store \( R_e \) in S’s database and send an accepted message to \( U_i \) through a secure channel.

**Login phase:** When the user \( U_i \) wants to login to the server, he inputs his identity \( ID_e \), password \( PW \), and a large random integer \( d \approx p - 1 \). The user then performs the following steps to begin an access session:

**Step L1:** \( U_i \) computes \( a_i = h(b, PW) \) and \( E_{a_i}(g^i \mod p) \).

**Step L2:** Acquire current time stamp \( T_e \), store \( T_i \) temporarily until the end of the session and compute \( MAC_e = h(T_e\|a_i) \).

**Step L3:** \( U_i \rightarrow S: ID_e, T_e, MAC_e, E_{a_i}(g^i \mod p) \)

Send the message \( \{ID_e, T_e, MAC_e, E_{a_i}(g^i \mod p)\} \) to the server \( S \) and wait for response from the server. If no response is received in time or the response is incorrect, report login failure to the user and stop the session.

**Authentication and key agreement phase:** Upon receiving the message \( \{ID_e, T_e, MAC_e, E_{a_i}(g^i \mod p)\} \) from \( U_i \), the server \( S \) performs the following steps to assure the integrity of the message:

**Step V1:** Check the freshness of \( T_e \), If \( T_e \) has already appeared in a current executing session of user \( U_i \), reject \( U_i \)'s login request and stop the session. Otherwise, \( T_e \) is fresh.

**Step V2:** Compute \( a_{i}' = R \oplus h(ID_e||x) \), \( MAC_{i}' = h(T_e\|a_i) \) and check whether \( MAC_{i}' \) is equal to the received \( MAC_e \). If it is not, reject \( U_i \)'s login and stop the session.

**Step V3:** \( S \) chooses a large random integer \( e \approx p - 1 \) and computes \( g^i \mod p \). \( S \) retrieves \( g^i \mod p \) by computing \( D_k(E_{a_i}(g^i \mod p)) \). After all parameters are known, \( S \) computes session key \( SK = (g^i)^e \mod p \) for this scheme to operate.

**Step V4:** \( S \rightarrow U_i: E_{sk}(T_e), E_{h(PW)}(g^i \mod p, T_e) \)

Acquire the current time stamp \( T_e \). Store temporarily paired time stamps \( (T_e, T_i) \) and \( ID_e \) for freshness checking until the end of the session. \( S \) computes \( E_{sk}(T_e) \). Next, send the message including \( ID_e, E_{sk}(T_e), E_{h(PW)}(g^i \mod p, T_e) \) to \( U_i \). Then, send the message \( \{E_{sk}(T_e), E_{h(PW)}(g^i \mod p, T_e)\} \) back to \( U_i \) and wait for response from \( U_i \). If no response is received in time or the response is incorrect, reject \( U_i \)'s login and stop the session.

**Step V5:** \( U_i \rightarrow S: E_{sk}(h(T_e)) \).

On receiving the message \( \{E_{sk}(T_e), E_{h(PW)}(g^i \mod p, T_e)\} \) from the server \( S \), \( U_i \) computes \( D_{sk}(E_{sk}(h(T_e))) \) to retrieve \( h(T_e) \) mod p and \( T_e \). Then \( U_i \) computes the session key \( SK = (g^i)^e \mod p \) and \( T = E_{sk}(T_e) \) and then compare \( T \) with the received \( E_{sk}(T_e) \). If they are not equal, terminate this session. Otherwise, \( U_i \) computes \( E_{sk}(h(T_e)) \) and sends it to \( S \).

**Step V6:** \( S \rightarrow U_i: E_{sk}(T_e) \).

Upon receiving the message from \( U_i \), \( S \) computes \( D_{sk}(E_{sk}(h(T_e))) \) to retrieve \( Q = h(T_e) \). Then, \( S \) computes \( Q = h(T_e) \) by using \( T_e \) generated in Step V4 and compares it with \( Q \). If they are not equal, \( S \) terminates this session; otherwise, \( S \) computes \( E_{sk}(T_e) \) and sends the computation result to \( U_i \).

**Step V7:** After getting the transmitted message, \( U_i \) computes \( D_{sk}(E_{sk}(T_e)) \) and checks if \( T_e \) is in the decryption result for freshness checking. If it holds, the authentication is successful; otherwise, the connection is interrupted. After finishing mutual authentication, the user \( U_i \) and the remote server \( S \) can use the session key \( SK \) to encrypt/decrypt the secret information for the following communication.
SEAlERTY ANALYSIS

Provide perfect forward secrecy: Assuming the random values d and e are large, it is computationally infeasible for an adversary to find g^e mod p due to the Diffie-Hellman problem (Diffie and Hellman, 1976). In the proposed scheme, assume that even both the user U's password PW, and the server S's secret key x are all known by an attacker. Then the attacker still cannot decrypt E_{h(random)}(g^e mod p) to obtain g^e mod p and decrypt E_{h(random)}(g^e mod p, T_e) to obtain g^e mod p, since the attacker does not know PW, and b. Even if the attacker has known g^e mod p and g^e mod p, he cannot calculate g^x because the difficulty is similar to solve the Diffie-Hellman problem. So, the attacker does not have any opportunity to get the session key SK. Therefore, the session key is still secure. Hence, present scheme provides perfect forward secrecy with high security (Sun and Yeh, 2006).

Resist privileged insider's attack: In the registration phase, a user U selects a random number b, password PW, and then computes h(b⊕PW). The user submits ID, and h(b⊕PW) to the remote server S. If the privileged insider of S wants to use U's password to impersonate U, to login the other servers, the action will fail. Since U registers to S by presenting h(b⊕PW) instead of PW, the privileged insider of S can not directly obtain PW. Besides, as b is not revealed to S and h(·) is a collision-resistant one-way hash function, the privileged insider of S can not obtain PW by performing an off-line guessing attack on h(b⊕PW). Therefore, the proposed scheme can resist the privileged insider attack (Ku et al., 2005; Ku and Chen, 2004).

Resist the masquerade attack: If the adversary Eve has stolen U's authentication data R_e, which stored in S, Eve cannot masquerade as the legal user U, to login the remote server S, since, Eve cannot obtain h(b⊕PW) without knowing the knowledge of both PW and b. Hence, the adversary cannot forge a login message to pass S's authentication, the proposed scheme can resist the masquerade attacks.

Resist the stolen-verifier attack: In the proposed scheme, the user U's authentication data stored in S is R_e = h(ID)∥x ⊕ h(b⊕PW). Suppose that an attacker Eve has stolen the R_e, she can obtain h(b⊕PW) only if Eve has the information of h(ID)∥x), which implies she knows S's long-term secret key x. Since, h(b⊕PW) is hidden in R_e and the secret key x is under strict protection as assumed, Eve's obtaining h(b⊕PW) in this way is infeasible. In addition, if Eve is a legal user and has stolen her R_e, since h(·) is a collision-resistant one-way hash function, Eve's retrieving x is still computational infeasible. That is, the proposed scheme can resist the stolen-verifier attack.

Resist the server spoofing attack: Any attempt of an attack to impersonate as any remote server S to cheat U is infeasible, since he cannot construct the session key SK without the knowledge of PW, and b. Thus, the attacker cannot decrypt the transmitted messages from some legal user. After communicating with the masqueraded remote server, the legal user can detect immediately and terminates the session. Hence, the proposed scheme can protect the user from being cheated by the masqueraded remote server.

Resist the replay attack: In the proposed protocol, if an attacker retransmits either the login message (ID, T_e, MAC_e, E_{h(random)}(g^e mod p)) in Step L3 of Login Phase or the response message (E_{sk} (T_e), E_{h(random)}(g^e mod p, T_e)) in Step V4 of Authentication and key agreement phase, the attacker cannot pass authentication successfully. Since the legality of the messages will be checked with the challenges d, e and T_e. On the contrary, only the legal partners know PW and b to bind the corresponding random numbers and nonce in the encrypted messages. Therefore, the proposed scheme can resist the replay attack.

Resist the man-in-the-middle attack: In the proposed scheme, the transmitted message is protected by h(b⊕PW) during each authentication. Attempt of the attack to produce the correct transmitted message without knowing h(b⊕PW) is impossible. Since S and U can check whether the transmitted messages during Steps L3 and V1 to V6 are forged or replaced, no attacker can change the transmitted messages. If any malicious attacker wants to mount this attack, either S or U will detect it. In addition, because the messages contain T_e, and T_e, the transmitted messages for different sessions are different. Consequently, the attacker cannot compute the correct messages or modify the messages which cannot be detected by user or server even if the attacker has collected all the messages in other sessions.

DISCUSSION

R1: Freely choose password: In registration phase of the proposed scheme, a new user can freely choose his password for registering to the remote server. Later, the user can use this pair of (ID, h(b⊕PW)) to pass the server's authentication procedure.
Table 1: Comparisons of various security attributes

<table>
<thead>
<tr>
<th>Security attribute</th>
<th>Our</th>
<th>Shieh-Wang</th>
<th>Jung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use smart card and reader</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Insider’s attack</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Securely change password</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Session key agreement</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No time synchronization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Server spoofing resistance</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Perfect forward secrecy</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

R2: Security: It is described in security analysis has some attack resistances, including the privileged insider’s attack, the masquerade attack, the stolen-verifier attack, the server spoofing attack, the replay attack, the man-in-the-middle attack and the perfect forward secrecy to confirm the security of the proposed scheme. With the security analysis described earlier, it is ensured that the proposed scheme is robust and secure.

R3: Mutual authentication: In this scheme, allows the user and the server to authenticate each other. As shown in the proposed scheme, $S$ and $U_i$ can authenticate each other in steps V6 and V7 of authentication and key agreement phase, respectively. Accordingly, mutual authentication is ensured in the proposed scheme.

R4: Session key agreement: As shown in authentication and key agreement phase, after finishing mutual authentication, the user $U_i$ and the remote server $S$ can compute the session key $SK = g^{sk} \mod p$. Later, they can use $SK$ to protect the transmitting messages in the following communication.

We will here show several functionality comparisons between the proposed scheme and related schemes in Table 1.

In Table 1, it shows that present schemes can achieve the essential requirements for a secure and efficient remote mutual authentication and key agreement scheme which is described in introduction. It is known that plenty of secret information can be stored in the smart card in advance to reduce the computation costs. Accordingly, the proposed scheme which does not adopt the smart cards to authenticate the user should spend slightly more computation cost than that of Juang’s and Shieh-Wang’s schemes (Juang, 2004; Shieh and Wang, 2006) for completing the remote authentication procedure. Nonetheless, because the card readers are not obtainable to all and everywhere, authentication protocols adopting the smart cards to authenticate the legitimacy of the user are not applicable to all.

CONCLUSIONS

Over the last decade, most of the proposed password authentication schemes are designed using smart cards, but these schemes do not suffice for users’ requirements. Because of the reader of smart cards is not always obtainable to everyone and everywhere. This study proposes a secure and efficient remote mutual authentication scheme without using the smart cards. We have demonstrated that the proposed scheme can satisfy all of the essential security requirements. After the analysis, it is concluded that the new method can be easily practical to the real world.

REFERENCES


