Anonymous and Authentication Protocol for Multi-Server

Wen-Chung Kuo, Po-Wei Shih
Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, e-mail: simonkuo@yuntech.edu.tw

Yu-Chih Huang
Department of Information Management, Tainan University of Technology, Taiwan, R.O.C.

Lih-Chyau Wuu
Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology

Corresponding author: simonkuo@yuntech.edu.tw

In a multi-server environment, when a user wants to login to a different server to access services, he/she needs to register another user identity and password. Recently, the single sign-on authentication method has been proposed. The major characteristic of this method is that a user only needs to remember one identity and password which can login to different servers. This reduces user inconvenience and server resource usage. On the other hand, anonymity is an important issue. If a user’s identity is disclosed, an attacker can trace or masquerade as the user to login servers. Preventing disclosure of the user’s identity is very important. In this paper, we will propose an anonymous and single sign-on authentication scheme based on Lagrange interpolating polynomial for a multi-server environment. According to our security analysis, this proposed scheme maintains anonymity, provides mutual authentication and also resists many attacks such as lost smart card, insider attack and replay attack.

KEYWORDS: Anonymous, mutual authentication, memory consumes, Lagrange interpolating polynomial, replay attack.

Introduction

In recent years, people have used modern technologies such as e-mail, Twitter, Facebook, etc. to communicate to each other over the Internet. However, users should use different identities and passwords for different services. Naturally, it is difficult to remember many different identities and passwords. In
order to alleviate this problem, many authentication schemes based on different methodologies such as password [3, 4], smart card [2, 8, 9], or one-way hash function [7] have been proposed. In 2003, Lee et al. [3] proposed password authentication scheme (LKH-scheme). The major contribution of this scheme is that two points are used to generate one linear equation and then the characteristics of equation are used to authenticate the user.

The use of linear equations to replace traditional encryption can reduce computing cost and transmission quantity for authentication. Therefore, in 2010, Liaw et al. [4] proposed an efficient password authentication scheme based on a geometric approach for multi-server environments. Different from the above, their proposed authentication scheme uses a 2-dimensional plane and 3-dimensional space coordinates to achieve authentication. Although these schemes [3, 4] can achieve mutual authentication between user and server, they cannot provide user anonymity on a public network.

In order to achieve user anonymity, we propose an anonymous and single sign-on authentication scheme based on Lagrange interpolating polynomial for multi-server environments in this paper. In other words, a user only needs to remember one identity and password to be able to login to different servers. Each server does not save the user’s login information during authentication phase. This can reduce the usage of server resources. According to the security analysis, user anonymity is an important issue. If a user identity is disclosed, an attacker can trace or masquerade as the user to gain access to servers. Thus, we use linear equations to hide user identity and use the Lagrange interpolating polynomial to authenticate the user and server identities. Additionally, our scheme can address lost smart cards, resist replay attack and provide mutual authentication.

The rest of the paper is organized as follows: in Section 2, we review the Lagrange interpolating polynomial and Lee et al.’s scheme. Next, we propose an anonymous and single sign-on protocol for multi-server environments and then discuss security analysis in Sections 3 and 4, respectively. Finally, we draw some conclusions in Section 5.

### Related work review

#### The Lagrange interpolating polynomial

The Lagrange interpolating polynomial [10] is the polynomial $P(x)$ of degree $n \neq (n-1)$ that passes through $n$ points, and is given by Eq.(1),

$$P_j(x) = y_j \prod_{k=1, k \neq j}^{n} \frac{x-x_k}{x_j-x_k}$$

where is the Lagrange polynomial, and $x_i$ and $y_i$ are the $x$-value and $y$-value of point $(x_i, y_i)$, respectively. In our scheme, we use the Lagrange interpolating polynomial on modulus system.

#### The password authentication scheme

In 2003, a password authentication scheme based on a geometric approach was proposed by Lee et al. [3]. The major contribution of the LKH-scheme is not having to remember different identities and passwords for various servers. There are four phases in the LKH-scheme: (1) the registration phase, (2) the login phase, (3) the authentication phase and (4) password key update. The notations and symbols used in the LKH-scheme are shown in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i$</td>
<td>The $i^{th}$ user</td>
</tr>
<tr>
<td>$\Gamma = {S1, S2, \ldots, Sm}$</td>
<td>A set of servers that $U_i$ would like to login to</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>Unique identity of $U_i$</td>
</tr>
<tr>
<td>$PW_i$</td>
<td>Password of $U_i$</td>
</tr>
<tr>
<td>$(x_j, y_j)$</td>
<td>The secret points of server $S_j$</td>
</tr>
<tr>
<td>$T$</td>
<td>Timestamp</td>
</tr>
<tr>
<td>$L$</td>
<td>Linear equation</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>XOR operation</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>One-way hash function</td>
</tr>
</tbody>
</table>

#### The registration phase

**Step LKH-R1.** $U_i \rightarrow$ Trusted Manager: $ID_i, PW_i$

In this step, $U_i$ sends his $ID_i$ and $PW_i$ to the trusted manager.

**Step LKH-R2.** Trusted Manager $\rightarrow U_i$: $\{ID_i, (C_{D_i}, D_{D_i})\}$
After receiving the message from $U$, the trusted manager calculates $(X, Y)$ and $(C, D)$ where $X = h(ID) \oplus x$, $Y = h(ID, y)$, $C = X \oplus PW$, and $D = Y \oplus PW$. Then, the trusted manager stores the message $(ID, (C, D))$ in a smart card and gives it to $U$.

The login phase

**Step LKH-L1.** $U \rightarrow Reader: PW$

In this step, $U$ inserts the smart card into a reader and enters $PW$ to obtain $(X, Y)$ by computing $X = C \oplus PW$ and $Y = D \oplus PW$. 

**Step LKH-L2.** Reader $\rightarrow U$: $\{A, B, R, T\}$

Smart card generates two random numbers $a$ and $b$, $U$ utilizes $(a, b)$ and $(X, Y)$ to generate $L: Y = f(x) = a \cdot x + b \pmod p$ and calculates $A = a \oplus X, B = b \oplus Y$, and $R = h(ID || a || b || T)$, where $T$ is the timestamp of the $U$.

**Step LKH-L3.** $U \rightarrow S$: $(ID, A, B, R, T)$

$U$ forwards $(ID, A, B, R, T)$ to $S$.

The authentication phase

**Step LKH-A1.** $U \rightarrow S$: $(ID, A, B, R, T)$

After receiving the message from $U$, $S$ checks $ID$ and verifies whether $|T - T| \neq \Delta T$, where $T$ is the current time on $S$, and $\Delta T$ is the expected time interval for transmission delay and clock offset error. If the time is within the expected range, $S$ uses secret points $(x, y)$ and $ID$ to obtain $(X, Y)$. Then, $S$ calculates $R = h(ID || a || b || T)$ and checks whether $R'$ is equal to $R$. If $R' = R$, then $L = f(x) = a \cdot x + b \pmod p$. This allows $S$ to check whether $(X, Y)$ is located on line $L$.

**Step LKH-A2.** $S \rightarrow U$: $(SID, R, T)$

If $U$ is authenticated, then $S$ calculates $R = h(SID, h(a), h(b), T)$ and forwards $(SID, R, T)$ to $U$.

**Step LKH-A3.** $U \rightarrow S$: $(SID, R, T)$

When $U$ receives the message from $S$, $U$ uses $SID$ and $T$ to calculate $R' = h(SID, h(a), h(b), T)$. Then, $U$ checks whether $R'$ is equal to $R$. If so, then $U$ and $S$ can achieve mutual authentication.

The password update phase

If $U$ wants to change his/her password, he/she can enter $PW$ into the smart card to obtain $(X, Y)$. Then, $U$ enters a new password $PW_{\text{new}}$ to calculate $C = X \oplus PW_{\text{new}}$ and $D = Y \oplus PW_{\text{new}}$. Finally, the new secret information $(ID, (C, D))$ is saved into the smart card. Afterwards, $U$ can access the system using his/her new password $PW_{\text{new}}$.

Lee et al.[3] uses a geometric approach to propose another encryption method to reduce computing cost and memory use. However, their method cannot provide user anonymity. We extend their contribution by incorporating anonymity in our scheme and propose an anonymous authentication protocol in the next section.

The Proposed Scheme

This section proposes an anonymous authentication scheme for a multi-server environment. We assume that there are three entities in this scheme: registration center (RC), user (U), and server (S) in this scheme. The scheme includes the following four phases: (1) the registration phase; (2) the login phase; (3) the authentication phase and (4) the session key update phase. It is assumed that time is synchronous and information from smart card cannot be obtained through forced attacks. The notations and symbols used in the proposed scheme are shown in Table 2.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>User $i$</td>
</tr>
<tr>
<td>$S$</td>
<td>Server $j$</td>
</tr>
<tr>
<td>RC</td>
<td>Registration center</td>
</tr>
<tr>
<td>$CID_i$</td>
<td>Anonymous identity of $U_i$</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>Unique identity of $U_i$</td>
</tr>
<tr>
<td>$PW_i$</td>
<td>Password of $U_i$</td>
</tr>
<tr>
<td>$SID_i$</td>
<td>Unique identity of $S_i$</td>
</tr>
<tr>
<td>$(x_i, y_i)$, $(x_p, y_p)$</td>
<td>Registration center secret points</td>
</tr>
<tr>
<td>$T$</td>
<td>Timestamp</td>
</tr>
<tr>
<td>SK</td>
<td>Session key</td>
</tr>
<tr>
<td>$L$</td>
<td>Linear equation</td>
</tr>
<tr>
<td>$P(x)$</td>
<td>Lagrange polynomial</td>
</tr>
<tr>
<td>$n, p$</td>
<td>Two large primes $(n &gt; p)$</td>
</tr>
<tr>
<td>$N$</td>
<td>Random nonce</td>
</tr>
<tr>
<td>$h()$</td>
<td>One-way hash function</td>
</tr>
</tbody>
</table>
The registration phase

Preceding the registration phase, RC randomly chooses a 2D point \((x_i, y_i)\) and a Lagrange polynomial \(P(x)\): 
\[ y = ix^2 + jx + k \mod n \]
and then determines a point \((x_i, y_i)\) so that 
\[ y_i = ix^2 + jx + k \mod n. \]

The user registration subphase

The user registration interaction sequence is depicted in Fig.1. An explanation of various messages and steps in this subphase is provided below.

**Step RU1.** \(U_i \rightarrow RC: ID_i, PW_i\)

To register, \(U_i\) selects a password \(PW_i\) and then sends \(ID_i\) and \(PW_i\) to \(RC\) through a secure channel.

**Step RU2.** \(RC \rightarrow U_i: L_i: y = ax + b \mod p\) and \(y_{U_i}\).

After receiving a registration request from \(U_i\), RC uses \((x_i, y_i)\) and \((ID_i, PW_i)\) to generate linear equation \(L_i: y = ax + b \mod p\) and calculates hash value \(h(a/b)\). Then, the hash value \(h(a/b)\) is substituted into the Lagrange polynomial \(P(x)\): 
\[ y = ix^2 + jx + k \mod n \]
and calculates a hash value \(h(c/d)\). RC saves \(h(a/b)\) and \(y_{U_i}\) in its database. Finally, linear equation \(L_i\) and value \(y_{U_i}\) are embedded into the smart card and sent to \(U_i\) through a secure channel.

Figure 1
Interaction sequence of the user registration subphase

The server registration subphase

The server registration interaction sequence is depicted in Fig.2. An explanation of various messages and steps in this subphase is provided below.

**Step RS1.** \(S_j \rightarrow RC: SID_j, N_j\)

\(S_j\) selects a random nonce \(N_j\) and then sends \(SID_j\) and \(N_j\) to \(RC\) through a secure channel.

**Step RS2.** \(RC \rightarrow S_j: L_j: y = cx + d \mod p\) and \(y_{S_j}\).

After receiving the registration request from \(S_j\), RC uses \((x_j, y_j)\) and \((SID_j, N_j)\) to generate linear equation \(L_j: y = cx + d \mod p\) and calculates a hash value \(h(c/d)\). Then, RC substitutes \(h(c/d)\) into the Lagrange polynomial \(P(x)\): 
\[ y = ix^2 + jx + k \mod n \]
and enters his identity \((SID_j, N_j, h(c/d))\) and \(y_{S_j}\) in its database and sends \(L_j\) and \(y_{S_j}\) to \(S_j\) through a secure channel.

Figure 2
Interaction sequence of the server registration subphase

The login phase

When \(U_i\) wants to login to \(S_j\), he inserts his smart card and enters his identity \(ID_i\) and password \(PW_i\).

**Step L1.** The smart card checks \((ID_i, PW_i)\) if the linear equation function \(L_i: y = ax + b \mod p\), \(a = a * ID_i + b \mod p\). If it is correct, then it generates a random point \((x_i, y_i)\) that is on the linear equation \(L_i\) and a random nonce \(N_i\) to calculate \(CID_i = h(a||b||N_i||T_i)\) and \(M_i = a + b + N_i\). Otherwise, the smart card rejects the request.

**Step L2.** \(U_i\) forwards \(\{CID_i, (x_i, y_i), M_i, T_i\}\) to \(S_j\), where \(T_i\) is the timestamp of the \(U_i\).

The authentication phase

The authentication phase interaction sequence is depicted in Fig.3. Following is an explanation of various messages and steps in this phase. When \(S_j\) receives the messages \(\{CID_i, (x_i, y_i), M_i, T_i\}\) from \(U_i\), the following steps are executed.

**Step A1.** \(S_j \rightarrow RC: \{CID_i, (x_i, y_i), M_i, T_i, SID_i, (x_j, y_j)\}\)

To prevent a replay attack, \(S_j\) ensures \(|T - T_i|\) is not greater than \(\Delta T\), where \(T\) is the current time on \(S_j\) and \(\Delta T\) is the expected time interval for transmission delay and clock offset error. If it is not correct, \(S_j\) ter-
minates the session. Otherwise, $S$ produces a random point $(x, y)$ on the linear equation $L_i: y = cx + d \mod p$, i.e., $y = cx + d \mod p$ and forwards $(CIDS_i (x, y), M_i, T, \text{SID}_i (x, y))$ to RC.

**Step A2.** RC $\rightarrow S$: \{V, $V_2, M_\beta (x_{RC}, y_{RC}), T_{RC}\}$

When RC receives message $(CIDS_i (x, y), M_i, T, \text{SID}_i (x, y))$ from $S$, RC performs the following steps.

1. RC checks whether $|T - T_0| \neq \Delta T$. If it is true, RC then uses SID to find the corresponding random nonce $N_j$ from the database.

2. RC utilizes the point $(x, y_j)$ and $(\text{SID}_i, N_i)$ to generate $L_i = cx + d \mod p$.

3. RC checks if $(x, y_j)$ exists in $L_i$. If it does not exist, then RC terminates the session. Otherwise, RC utilizes points $(x, y_j)$ and $(x, y_j)$ to recover $L_i: y = ax + b \mod p$ and calculate $N_i = a \oplus b \oplus M_i$ and $CIDS_i = h(a || b || N_i || T_0)$.

4. RC checks if $CIDS_i$ is equal to $CIDS$. If the values are equal, then RC uses $L_i$ and $L_{\beta}$ to calculate hash values $h(a || b)$ and $h(c || d)$ to recover the corresponding $y_1$ and $y_2$ values.

5. RC uses the three points $(h(a || b), y_1), (h(c || d), y_2)$ and $(x, y_j)$ to recover the Lagrange polynomial $\mathcal{P}(x) : y = cx^2 + fx + k \mod n$. If it cannot be recovered, then the request is rejected. Otherwise, RC utilizes $(h(a || b), y_1)$ and $(h(c || d), y_2)$ to generate the linear equation $L_{\beta}: y = ex + f \mod p$ and then selects a random point $(x_{RC}, y_{RC})$ on $L_{\beta}$.

6. RC computes $M_\beta = c \oplus d \oplus N_\beta, V_1 = h(e || f || N_\beta || T_{RC} || y_1)$ and $V_2 = h(e || f || N_\beta || T_{RC} || y_2)$. Finally, RC sends $(V_1, V_2, M_\beta (x_{RC}, y_{RC}), T_{RC})$ to $S$.

**Step A3.** $S_1 \rightarrow U$: \{V, $(x_{RC}, y_{RC}), T_{RC}\}$

After receiving the message $(V, (x_{RC}, y_{RC}), T_{RC})$ from RC, $S_1$ executes the following steps.
Sₐ uses \((h(c/d), y₁)\) and \((x₁, y₁)\) to generate \(L_{x₁}^*: y = ex + f \mod p \) when \(|T - T₁| \neq \Delta T\). 

Sₐ checks whether \(V₁^*\) is equal to \(V₁\). If so, then Sₐ calculates \(N₁ = e \oplus d \oplus M₁\). Sₐ calculates the session key \(SK = h(e||f||N₁)\) and forwards \((V₁, (x₁, y₁), T₁)\) to \(U₁\).

**Step A4.** When \(U₁\) receives \((V₁, (x₁, y₁), T₁)\), \(U₁\) checks whether \(|T - T₁| \neq \Delta T\). If they are not equal, \(U₁\) uses \((h(a/b), y₁)\) and \((x₁, y₁)\) to recover \(L_{x₁}^*: y = ex + f \mod p\) and check whether \(V₁^*\) is equal to \(V₁\). If they are equal, then \(U₁\) calculates the session key \(SK = h(e||f||N₁)\).

### The session key update phase

When the user wants to update the session key with the server, he/she can get a new session \(SK_{\text{new}}\) by using the below listed steps. The session key update interaction sequence is depicted in Fig.4.

**Step U1.** \(U₁ \rightarrow Sₐ: E_{SK}[(x₁, y₁), (x₁, y₁), N₁, h(e||f)]\)

\(U₁\) selects two points \((x₁, y₁)\) and \((x₂, y₂)\) on a plane where \((x₁, y₁)\) lies on the linear equation \(L₁\): \(y = ex + f \mod p\). Then, \(U₁\) calculates \(h(e||f)\) and selects a random nonce \(N₁\). Finally, \(U₁\) utilizes session key \(SK\) to encrypt the message \(\{(x₁, y₁), (x₁, y₁), N₁, h(e||f)\}\) and sends it to \(Sₐ\).

**Figure 4**

Interaction sequence of the update session key phase

**Step U2.** \(Sₐ \rightarrow U₁: E_{SK}[(x₁, y₁), N₁_{\text{new}}]\)

When \(Sₐ\) receives messages \(E_{SK}[(x₁, y₁), (x₁, y₁), N₁, h(e||f)]\) from \(U₁\), \(Sₐ\) executes the following steps.

1. \(Sₐ\) decrypts message \(E_{SK}[(x₁, y₁), (x₁, y₁), N₁, h(e||f)]\).
2. \(Sₐ\) uses \((h(c/d), y₁)\) and \((x₁, y₁)\) to recover \(L_{x₁}^*: y = e'x + f \mod p\).
3. \(Sₐ\) calculates \(temp = h(e||f)\) and checks whether \(temp\) is equal to \(h(e||f)\). If equal, \(Sₐ\) utilizes \((x₁, y₁)\) and \((x₂, y₂)\) to generate new linear equation \(L₂^*:\)

\[y = gx + h \mod p\]

\(and\) new random nonce \(N₁_{\text{new}}\).
4. \(Sₐ\) sends message \(E_{SK}[(x₁, y₁), N₁_{\text{new}}]\) to \(U₁\), where the point \((x₁, y₁)\) is on the linear equation \(L₂^*\).

Finally, \(Sₐ\) computes the new session key \(SK_{\text{new}} = h(g||h||N₁_{\text{new}}||N₁_{\text{new}}||SK)\).

**Step U3.** \(U₁ \rightarrow Sₐ: E_{SK_{\text{new}}} [N₁_{\text{new}}]\)

Similarly, \(U₁\) will perform the following steps when \(U₁\) receives the message \(E_{SK}[(x₁, y₁), N₁_{\text{new}}]\) from \(Sₐ\).

1. \(U₁\) decrypts the message and verifies if \(y₁ = g * x₁ + h \mod p\) for \((x₁, y₁)\).
2. If true, \(U₁\) generates a new session key \(SK_{\text{new}} = h(g||h||N₁_{\text{new}}||N₁_{\text{new}}||SK)\) and uses \(SK_{\text{new}}\) to encrypt \(N₁_{\text{new}}\) to send it to \(Sₐ\). Otherwise, the request is dropped.

**Step U4.** After \(Sₐ\) receives the message \(E_{SK_{\text{new}}} [N₁_{\text{new}}]\), \(Sₐ\) uses the new session key \(SK_{\text{new}}\) to recover \(N₁_{\text{new}}\) and checks if it is equal to the original \(N₁_{\text{new}}\). If it holds, the server updates to the new session key.

### Security analysis

In this section, we discuss the security of our scheme against various attacks and use the Burrows-Abadi-Needham Logic (a.k.a BAN Logic) [1] mechanism to prove that the session key can be correctly updated between the user and server in the session key update process.

**The authentication phase proof**

We use the BAN-logic to show that our scheme correctly updates between the user and the server. Let \(X\) and \(Y\) represent the range over statements. For the BAN-logic, the logical notations of the logic are given in Table 3.
Furthermore, we also define some logical postulates that we will use in the proofs as follows:

Table 3

The logical notation

<table>
<thead>
<tr>
<th>Items</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i \overset{SK}{\to} S_j$</td>
<td>$U_i$ and $S_j$ share a session key.</td>
</tr>
<tr>
<td>$U_i \models X$</td>
<td>$U_i$ believes a statement $X$.</td>
</tr>
<tr>
<td>$#(X)$</td>
<td>$X$ is fresh.</td>
</tr>
<tr>
<td>$U_i \Rightarrow X$</td>
<td>$U_i$ controls $X$.</td>
</tr>
<tr>
<td>$U_i \prec X$</td>
<td>$U_i$ receives $X$.</td>
</tr>
<tr>
<td>$U_i \models X$</td>
<td>$U_i$ sends $X$.</td>
</tr>
<tr>
<td>$[X, Y]_K$</td>
<td>$X$ and $Y$ are encrypted with the key $K$.</td>
</tr>
</tbody>
</table>

1. The Message-meaning rule

$$P \in [X]_K, P \models P \overset{SK}{\to} Q \Rightarrow P \models Q \models \neg X$$  \hspace{1cm} (2)

If principal $P$ believes that key $K$ is shared with principal $Q$ and $P$ receives message $X$ that is encrypted under $K$, then principal $P$ believes that principal $Q$ sent the statement $X$.

2. The Fresh concatenation rule

$$P \models \#(X) \Rightarrow P \models \#(X, Y)$$  \hspace{1cm} (3)

If principal $P$ believes the freshness of statement $X$, then principal $P$ believes the freshness of $(X, Y)$.

3. The Nonce-verification rule

$$P \models Q \models \neg X, P \models \#(X) \Rightarrow P \models Q \models X$$  \hspace{1cm} (4)

If principal $P$ believes that statement $X$ was not stated before and principal $Q$ sent the statement $X$, then principal $P$ believes that principal $Q$ believes statement $X$.

4. The Jurisdiction rule

$$P \models Q \models X, P \models Q \models \neg X \Rightarrow P \models X$$  \hspace{1cm} (5)

If principal $P$ believes that statement $X$ is under principal $Q$’s jurisdiction, then principal $P$ believes principal $Q$ on the validity of statement $X$.

Next, we show that our scheme should satisfy the following requirements:

- $G_1 : RC \models (x_i, y_i), (x_j, y_j)$  \hspace{1cm} (6)
- $G_2 : S_j \models (x_{RC}, y_{RC})$  \hspace{1cm} (7)
- $G_3 : U_i \models (x_{RC}, y_{RC})$  \hspace{1cm} (8)

Before we analyze the proposed protocol, we transform to the idealized form and identify the initial state of our scheme. The initial state is assumed to be the following:

- $A_1 : U_j \models U_i \overset{Y}{\to} RC$  \hspace{1cm} (9)
- $A_2 : U_j \models \#(T_{RC})$  \hspace{1cm} (10)
- $A_3 : U_j \models RC \Rightarrow (U_i \overset{Y}{\to} RC)$  \hspace{1cm} (11)
- $A_4 : S_j \models S_j \overset{Y}{\to} RC$  \hspace{1cm} (12)
- $A_5 : S_j \models \#(T_{RC})$  \hspace{1cm} (13)
- $A_6 : S_j \models RC \Rightarrow (S_j \overset{Y}{\to} RC)$  \hspace{1cm} (14)
- $A_7 : RC \models S_j \overset{L}{\to} RC$  \hspace{1cm} (15)
- $A_8 : RC \models U_i \overset{L}{\to} RC$  \hspace{1cm} (16)
- $A_9 : RC \models S_j \Rightarrow (S_j \overset{L}{\to} RC)$  \hspace{1cm} (17)
- $A_{10} : RC \models U_i \Rightarrow (U_i \overset{L}{\to} RC)$  \hspace{1cm} (18)
- $A_{11} : RC \models \#(T_i)$  \hspace{1cm} (19)

Now, we use the initial assumptions and the rules of the BAN-logic to analyze the idealized form of our scheme. The proofs are described as follows:

1. By $A_2, A_8$ and $RC \Rightarrow ((x_i, y_i), (x_j, y_j), T_1)$, we apply the message-meaning rule to derive

$$RC \models \#((x_i, y_i), (x_j, y_j))$$  \hspace{1cm} (20)

2. By $A_{11}$ and Eq.(20), we apply the fresh concatenation rule to derive
Resistance to replay attack

Since the transmitted messages $CID_o$, $V_1$ and $V_2$ include timestamps, the server and user can detect a replay attack directly. If an attacker re-submits the intercepted message, the attacker must choose a suitable time interval $\Delta T$ and modify these three messages. However, the attacker cannot modify them because he/she does not know $L$ and $L_{sec}$. Note that the replay attack is avoided while the system clock synchronization and transmission delay are accounted for.

Smart card stolen attack

This assumes that the smart card is lost or stolen. If the attacker wants to use this stolen smart card to login, the attacker must know the correct ID and PW to complete the login phase. Furthermore, the ID and PW cannot be recovered from the stolen smart card. Therefore, the lost smart card attack is prevented in our proposed scheme.

Resistance impersonation attack

In the authentication phase, an attacker may intercept message $\{CID_o (x, y), M_v, T_i\}$ and try to impersonate a legal user to pass authentication. However, the attacker cannot calculate $L_i: y= ax+ b \mod p$, because point $(x, y)$ is a secret held by the registration center. Therefore, the attacker cannot impersonate the legal user to pass authentication. On the other hand, if the attacker wants to masquerade as a server to spoof the user and intercept message $\{V_v, V_o, M_x (x_{RC}, y_{RC}), T_{RC}\}$, the attacker must modify $V_v$, but cannot do so since the attacker does not have $y_{RC}$. Additionally, the attacker cannot calculate $L_{sec}: y= ex+f \mod p$ and generate the session key $S_{sk}$. Thus, the attacker cannot masquerade as a server to spoof users.

Mutual authentication

In our scheme, $RC$ has the important role to authenticate user $U_i$ and server $S_i$. If the verification operation fails, this means that the credentials are incorrect and possibly the user $U_i$ or server $S_i$ is not legitimate. Therefore, mutual authentication will end. When the registration center sends message $\{V_v, V_o, M_x (x_{RC}, y_{RC}), T_{RC}\}$ to server $S_i$ and the server verifies $V_v$, $V_o$ as correct, the server can trust that user $U_i$ is legitimate since the user passed the registration center authentication. Conversely, server $S_i$ sends message $\{V_v, (x_{RC}, y_{RC}), T_{RC}\}$ to user $U_i$ and $U_i$ verifies if $V_v$ is correct. The user can

The security analysis

In this subsection, we discuss the resiliency of our proposed scheme against some common attacks such as replay attack, lost or stolen smart card attack, and impersonation attack, while providing mutual authentication and user anonymity.
trust that the server is legitimate since the authentication message cannot be forged.

**User anonymity**

In our scheme, the user uses $CID_i$ instead of $ID_i$ so the attacker cannot get the user’s real identity. In addition, we use a random nonce $N_i$ to calculate $CID_i$. This makes $CID_i$ unique every time. Thus, the attacker cannot trace the user’s identity. In the authentication phase, the registration center does not use $ID_i$ to search for the corresponding Lagrange polynomial value instead, it uses the hash value $h(a||b)$ to search. Therefore, the user does not send his/her identity over the public network. Hence, the attacker cannot trace the user’s real identity by intercepting a message transmitted between $U_i$ and $RC$.

The security properties of our scheme are compared with other authentication schemes [3, 4, 11]. The results are shown in Table 4. As can be seen from the Table 4, our scheme achieves user anonymity, mutual authentication and the session key security; it also prevents the lost smart card, replay and impersonation attacks.

### Table 4

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User anonymous</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resist the smart card stolen</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security of the session key</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevent the replay attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Prevent the impersonation attack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The performance evaluation

**Computational Load**

In Table 5, the computational load of our scheme is examined and compared with other authentication schemes [3, 4, 11]. The computation cost of a one-way function is denoted as $T_H$; $T_M$ is the computation cost of modular exponentiation; $T_R$ is the computation cost of a random nonce, select point and timestamp generation.

Note that we ignore the cost of XOR and //, because these operations require very little computation overhead. To be fair, we will not compare the session key update phase, because other schemes do not have this phase. As can be seen from the Table 5, our scheme has computation cost of 14 hashes, 7 instances of random nonce computing and 10 instances of modular exponentiation in the three phases of registration, login and authentication.

**Communication Load**

The communication load of our scheme was examined and compared with [11] and the results are shown in Table 6. The hash value length is assumed to be 128-bits, timestamp length is assumed to be 24-bits, the length of the each point value is assumed to be 16-bits, and each of the other elements are assumed to be 128-bits. The bits used for each interaction are added together and then divided by eight to obtain the number of bytes transferred for each interaction.

For example, from the authentication phase, there are four instances of message transmission which are:

$U_i \rightarrow S_j, S_j \rightarrow RC, RC \rightarrow S_j$ and $S_j \rightarrow U_i$. The results show the total communication load of our scheme is less than Xue’s scheme.
Table 5
Computational load

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Registration</th>
<th>Login</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration center</td>
<td>2TH</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User</td>
<td>0</td>
<td>TH + TR</td>
<td>TH</td>
</tr>
<tr>
<td>LKH-scheme[3]</td>
<td>0</td>
<td>TR</td>
<td>6TH + TR</td>
</tr>
<tr>
<td>Server</td>
<td>0</td>
<td>5TH + TR</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2TH</td>
<td>TH + TR</td>
<td>6TH + TR</td>
</tr>
<tr>
<td>Registration center</td>
<td>4TH</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User</td>
<td>0</td>
<td>3TH + TR</td>
<td>0</td>
</tr>
<tr>
<td>Server</td>
<td>0</td>
<td>4TH + TR</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>4TH</td>
<td>3TH + TR</td>
<td>4TH + TR</td>
</tr>
<tr>
<td>LYCH-scheme[4]</td>
<td>0</td>
<td>2TH</td>
<td>28TH + 4TR</td>
</tr>
<tr>
<td>Registration center</td>
<td>4TH</td>
<td>0</td>
<td>15TH + 2TR + 4TM</td>
</tr>
<tr>
<td>User</td>
<td>3TH + TR</td>
<td>2TH</td>
<td>7TH + 2TR + 6TH</td>
</tr>
<tr>
<td>Server</td>
<td>TR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>7TH + 2TR + 4TM</td>
<td>2TH</td>
<td>28TH + 4TR</td>
</tr>
<tr>
<td>Registration center</td>
<td>4TM + 2TH</td>
<td>0</td>
<td>3TH + TM</td>
</tr>
<tr>
<td>User</td>
<td>TR</td>
<td>TH + 3TR</td>
<td>3TH + TR + TM</td>
</tr>
<tr>
<td>Server</td>
<td>TR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>2TH + TR + 4TM</td>
<td>TH + 3TR</td>
<td>11TH + 3TR + 6TM</td>
</tr>
</tbody>
</table>

Table 6
Communication load for the authentication phase

<table>
<thead>
<tr>
<th>Protocols</th>
<th>(U_i \rightarrow S_j)</th>
<th>(S_j \rightarrow RC)</th>
<th>(RC \rightarrow S_j)</th>
<th>(S_j \rightarrow U_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our protocol</td>
<td>37</td>
<td>55</td>
<td>53</td>
<td>21</td>
</tr>
</tbody>
</table>

Conclusion

The proposed authentication scheme can achieve user anonymity and provide mutual authentication between server and users. In addition, this approach can resist various kinds of attacks. For a multi-server environment, our proposed scheme also provides single sign-on capabilities for users. Therefore, a user does not need to register on many different servers or remember he/she unique identity and password to login to participating servers. Furthermore, we use hash function and modular computing to replace the traditional encryption method which reduces computing cost.

References

2. Chuang, M. C., Chen, M. C. An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics. Expert
Summary / Santrauka

In a multi-server environment, when a user wants to login to a different server to access services, he/she needs to register another user identity and password. Recently, the single sign-on authentication method has been proposed. The major characteristic of this method is that a user only needs to remember one identity and password which can login to different servers. This reduces user inconvenience and server resource usage. On the other hand, anonymity is an important issue. If a user's identity is disclosed, an attacker can trace or masquerade as the user to login servers. Preventing disclosure of the user's identity is very important. In this paper, we will propose an anonymous and single sign-on authentication scheme based on Lagrange interpolating polynomial for a multi-server environment. According to our security analysis, this proposed scheme maintains anonymity, provides mutual authentication and also resists many attacks such as lost smart card, insider attack and replay attack.