DYNAMIC PASSWORD BASED REMOTE USER AUTHENTICATION WITHOUT TIME STAMPING

Mohamed Hamdy Eldefrawy 1,*, Muhammad Khurram Khan 1 
and Khaled Alghathbar 1,2
1 Center of Excellence in Information Assurance
2 Information Systems Department
College of Computer and Information Sciences
King Saud University
P.O. Box 92144, Riyadh 11653, Saudi Arabia
*Corresponding author: meldefrawy@ksu.edu.sa; {m.khurram; k.alghathbar}@ksu.edu.sa

Received August 2011; revised December 2011

ABSTRACT. The rapid growth of electronic commerce and Internet-based applications demands strong privacy protection and robust system security, which are essential requirements for an authentication scheme or access control mechanism. The issue of remote user authentication has received much attention recently. Password authentication is one of the most convenient authentication mechanisms. Most of the presented remote user authentication schemes utilize time stamping in the authentication process, which requires accurate time synchronization. Because it would be very beneficial to overcome the insufficiencies of such schemes regarding the need for accurate time synchronization, in this paper, we present a new algorithm that uses two different types of hash functions, which come with a nested hashing chain and utilize the discrete logarithm problem in message exchange. We first review the existing solutions and emphasize on their shortcomings and drawbacks. We then propose a secure and reliable remote user authentication scheme without the need of time stamping which covers the vulnerabilities of existing solution. A detailed security analysis was also performed that covered types of attacks that could influence our scheme. The proposed scheme establishes a common session key that provides message confidentiality.

Keywords: Dynamic password, Nested hashing, Remote authentication, Smart card

1. Introduction. Since Lamport [1] proposed the first password-based remote authentication scheme to identify a legal user within an insecure communication environment, a series of relevant studies and authentication mechanisms has been investigated. However, most of the previously published schemes cannot achieve computation efficiency and system security in parallel. In password-based authentication schemes with smart cards, remote users are authenticated using their smart card as an identification token; the smart card takes an input password from a user, calculates unique parameters from the user-given password, creates a login message using these parameters, and then sends a login message to the server, which then checks the validity of the login request before allowing access to any services or resources. In this way, the administrative overhead of the server is greatly reduced and a remote user is only required to remember his password to log on. In addition to just creating and sending login messages, smart cards support mutual authentication, where a challenge-response communication between the card and server takes place so that each can verify the other’s identity. Mutual authentication is a vital necessity in most real-world applications, where one’s private information should not be released to anyone until mutual confidence is established. Thus, at this stage, we
are concerned with mutual authentication and secure session generation. From a security point of view, it is better to consider these topics in cooperation rather than disjointedly. A protocol providing authentication without key exchange is susceptible to an enemy who waits until the authentication is complete and then takes over one end of the communication line. Such an attack is not precluded by a key exchange that is independent of authentication. Key exchange should be linked to mutual authentication so that a party has assurances that an exchanged key (which might be used to facilitate privacy or integrity and thus keep authenticity alive) is in fact shared with the authenticated party, and not an impostor. For these reasons, it is essential to keep key exchange mutual authentication in mind in the design and analysis of authentication protocols.

In this paper, we propose a remote user authentication scheme without time-stamping. The proposed scheme uses dynamic password authentication that is valid only one time to provide improved protection. Our scheme provides a practical remote user authentication scheme while retaining all advantages of robust remote user authentication scheme without the need of time synchronization. Our scheme is featured as following: (1) It achieves mutual authentication and session key generation; (2) It prevents from the security attacks due to the disclosure of the secrets stored in the storage device; (3) No verification tables are required; (4) No time stamping is required and consequentially no time synchronization is required; (5) The public key of each communication party is authenticated without the need of key certificate; (6) The private key of each party cannot be disclosed by any other third party.

1.1. Related work. In 1981, Lamport [1] proposed the first well-known remote password authentication scheme using smart cards. In Lamport’s scheme, the Authentication Server (AS) stores a password table at the server to check the validity of the login request made by the user. However, a high hash overhead and the need to reset passwords decrease the suitability and practical ability of Lamport’s scheme. In addition, Lamport’s scheme is vulnerable to a small $n$ attack [2]. Since then, many similar schemes [3,4] have been proposed. These all have a common feature: a verification password table should be securely stored in the AS. Actually, this property is a disadvantage from the security judgment. The AS will be partially or totally broken if an adversary steals, removes, or modifies the password table. Hwang and Li [5] pointed out that Lamport’s scheme secures from the risk of a modified password table and the cost of protecting and maintaining the password table. Further, they proposed a new remote user authentication scheme using smart cards. Hwang and Li’s [5] scheme does not maintain the password table at the server to check the validity of the login request. In addition, it can resist a message-replaying attack [6]. In the year 2000, Chan and Cheng [7] presented the security flaws of Hwang-Li’s scheme [5]. Afterwards Shen, Lin and Hwang [8] discussed a different type of attack on Hwang-Li’s scheme and also offered an enhanced scheme to solve the security problem of Hwang-Li’s scheme. Also, Chang and Hwang [9] clarified the practical problems of Chan-Cheng’s attack [7] on Hwang-Li’s scheme and Leung, Cheng, Fong, and Chen [10] mentioned that Shen, Lin and Hwang’s scheme [8] is still vulnerable to the attack proposed by Chan and Cheng [7]. Awasthi and Lal [11] also mentioned a different type of attack on Hwang-Li’s scheme and introduced a remote user authentication scheme. Awasthi and Lal [11] stated that their scheme provides forward secrecy to the AS. In the year 2004, Kumar [12] analyzed the practical drawbacks of Awasthi and Lal’s scheme [11]. In the same year, Lee et al. [13] raised a question about the correctness of Awasthi and Lal’s scheme [11]. Lee et al. [13] also proved that Awasthi and Lal’s scheme [11] is inaccurate and does not grant forward secrecy for the secret key of the authentication server, AS.
In 2004, Kumar [14] presented a new scheme to solve the security problems of Hwang-Li’s scheme [5]. Kumar’s scheme [14] removes the security vulnerabilities of Hwang-Li’s scheme [5], but does not provide a complete solution to solve all of the possible problems and withstand all possible attacks.

1.2. Organization. The remainder of this paper is organized as follows. Section 2 gives the notations. Section 3 reviews the other schemes. Section 4 shows some security flaws in previous schemes. Section 5 presents the proposed scheme. Section 6 illustrates the practical implementation of the proposed scheme. The security of the proposed scheme is analyzed in Section 7. Section 8 discusses the proposed scheme’s performance analysis and comparison. Finally, the conclusions are presented in Section 9.

2. Notations. The notations used throughout this paper are illustrated in the following table:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Denotes an identity of a remote user U&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>PW&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Denotes the corresponding password for a certain registered identity ID&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>SM&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Denotes the corresponding smart card of user U&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>T</td>
<td>Current time stamp</td>
</tr>
<tr>
<td>∆T</td>
<td>Valid time interval</td>
</tr>
<tr>
<td>AS</td>
<td>Denotes the authentication server</td>
</tr>
<tr>
<td>p</td>
<td>Large prime (usually at least 1024 bits)</td>
</tr>
<tr>
<td>q</td>
<td>Prime (typically of 160 bits) with q</td>
</tr>
<tr>
<td>G</td>
<td>G is a subgroup of Z&lt;sup&gt;*&lt;/sup&gt;&lt;sub&gt;p&lt;/sub&gt; and is often a subgroup of order q</td>
</tr>
<tr>
<td>g</td>
<td>Generator of G</td>
</tr>
<tr>
<td>h (·)</td>
<td>Denotes a cryptographic one way hash function</td>
</tr>
<tr>
<td>h&lt;sub&gt;A&lt;/sub&gt; (·)</td>
<td>Represents the first hash function</td>
</tr>
<tr>
<td>h&lt;sub&gt;B&lt;/sub&gt; (·)</td>
<td>Represents the second hash function</td>
</tr>
<tr>
<td>s&lt;sub&gt;int&lt;/sub&gt;</td>
<td>The OTP initial seed</td>
</tr>
<tr>
<td>s&lt;sub&gt;t&lt;/sub&gt;</td>
<td>The OTP seed number t for the t&lt;sup&gt;th&lt;/sup&gt; authentication (current seed)</td>
</tr>
<tr>
<td>OTP&lt;sub&gt;t&lt;/sub&gt;</td>
<td>The OTP number t for the t&lt;sup&gt;th&lt;/sup&gt; authentication</td>
</tr>
<tr>
<td>(x&lt;sub&gt;t&lt;/sub&gt;, y&lt;sub&gt;t&lt;/sub&gt;)</td>
<td>The nested hashing progress values for the t&lt;sup&gt;th&lt;/sup&gt; authentication</td>
</tr>
<tr>
<td>h&lt;sub&gt;B&lt;/sub&gt;&lt;sup&gt;y&lt;/sup&gt; (h&lt;sub&gt;A&lt;/sub&gt;&lt;sup&gt;x&lt;sub&gt;t&lt;/sub&gt;&lt;/sup&gt; (s&lt;sub&gt;t&lt;/sub&gt;))</td>
<td>Hashing the seed number t&lt;sup&gt;th&lt;/sup&gt; (s&lt;sub&gt;t&lt;/sub&gt;) by h&lt;sub&gt;A&lt;/sub&gt; (·) for x&lt;sub&gt;t&lt;/sub&gt; times followed by h&lt;sub&gt;B&lt;/sub&gt; (·) hashing for y&lt;sub&gt;t&lt;/sub&gt; times</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Review of Authentication Schemes. This section reviews three authentication schemes, which are composed of four parts: the registration phase, login phase, verification phase, and password change phase. This section will only focus on the first three phases, which will be mainly used in their security analyses.

3.1. Wang-Li’s scheme. If user U<sub>i</sub> wants to log in to an AS [15], he or she must insert his or her smart card into a card reader and key in his or her identifier, ID<sub>i</sub>, and password, PW<sub>i</sub>. Then, the smart card generates a random number, r ∈ Z<sup>*</sup><sub>p</sub>; computes R<sub>i</sub> = h (ID<sub>i</sub> || x<sub>s</sub>) , X<sub>i</sub> = R<sub>i</sub> ⊕ h (ID<sub>i</sub> || PW<sub>i</sub>) , where x<sub>s</sub> is held by the authentication server. It computes t = g<sup>r</sup> mod p and V<sub>i</sub> = X<sub>i</sub> ⊕ h (ID<sub>i</sub> || PW<sub>i</sub>) . Then the smart card computes W<sub>i</sub> = h (V<sub>i</sub> ⊕ T) , where T is the current time stamp; computes s = h (t || W<sub>i</sub>) ; and sends a message, C<sub>1</sub> = { ID<sub>i</sub>, t, s, T }, to the AS. Upon receiving the authentication request message, C<sub>1</sub>, the remote system and smart card perform the following steps for mutual
authentication between the user and remote system. Let $T'$ be the time that the system receives $C_1$. The $AS$ compares $T$ and $T'$. If the difference between $T$ and $T'$ is within a valid time interval, $\Delta T$, $C_1$ is considered to be a valid message. The system computes $V'_i = h(ID_i || x_s)$, as well as $W'_i = h(V'_i \oplus T)$. The system compares $h(t || W'_i)$ with $s$. If they are equal, then the system validates the login call and proceeds to the next step; otherwise, it rejects the login request. The system picks a random number, $\bar{r} \in Z_q^*$, and computes the session key, $k = t^r \mod p$. The system acquires the current time-stamp, $T''$, and computes $w = h(t || W'')$, $u = g^r \mod p$, $v = h(u || w)$. The system sends back the message $C_2 = \{u, v, T''\}$ to $U_i$. Upon receiving the message $\{u, v, T''\}$, the smart card verifies the validity of the time interval between $T''$ and the current time-stamp, $T'''$, and then computes $W'' = h(V_i \oplus T'')$. If $v = h(u || w)$, the mutual authentication is complete. Then, $k = g^{wr} \mod p$ is used as the session key between the user, $U_i$, and the remote system. Figure 1 presents Wang-Li’s Scheme.

3.2. Wang-Liu-Xiao-Dan’s scheme. In the user registration phase, $U_i$ submits $ID_i$ to $AS$ via a secure channel. Then, $AS$ computes $N_i = h(PW_i) \oplus h(x) \oplus ID_i$, where $PW_i$ is chosen by $AS$, and personalizes a smart card, $SM_i$ stored in the parameters $\{h(.), N_i, y\}$. $AS$ sends $SM_i$ and $PW_i$ to $U_i$ via a secure channel. Figure 2 introduces the login and

User $U_i$  
Authentication Server ($AS$)

<table>
<thead>
<tr>
<th>Information held by $U_i$: $ID_i, PW_i$, Smart Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input $ID_i, PW_i$</td>
</tr>
<tr>
<td>Generate $r \in Z^*_q$</td>
</tr>
<tr>
<td>$u = g^r \mod p$</td>
</tr>
<tr>
<td>$V'_i = X_i \oplus h(ID_i \parallel PW_i)$</td>
</tr>
<tr>
<td>Generate $T$</td>
</tr>
<tr>
<td>$W'_i = h(V'_i \oplus T)$</td>
</tr>
<tr>
<td>$s = h(t \parallel W'_i)$</td>
</tr>
</tbody>
</table>

$C_1 = \{ID_i, t, s, T\}$

Verify $ID_i$ and $T$

$V'_i = h(ID_i \parallel x_s)$

$W'_i = h(V'_i \oplus T)$

Verify $h(t \parallel W'_i) = s$

Generate $r \in Z_q^*$

$k = t^r \mod p$

Generate $T''$

$w = h(V'_i \oplus T'')$

$u = g^r \mod p$

$v = h(u \parallel w)$

$C_2 = \{u, v, T''\}$

Verify $T''$

$w' = h(V'_i \oplus T'')$

Verify $v \neq h(u \parallel w')$

$k = u^r \mod p$

Shared Session Key $k = g^{wr} \mod p$

**Figure 1.** Wang-Li’s user authentication scheme
verification phases of Wang-Liu-Xiao-Dan’s scheme [16]. Wang et al. stated that their scheme could make the security and secrecy of the password stronger by using a scheme where a password is chosen by the server and transmitted to the user to reduce the possibility of weak-password-selection.

![Figure 2. Wang-Liu-Xiao-Dan’s user authentication scheme](image)

3.3. Kim-Chung’s scheme. Kim-Chung’s scheme also has the merit that the server does not need to maintain a verification table [17]. They quarreled that their scheme can offer some useful features, including the early detection of an incorrect password, prevention of password leak, and so on. In the user registration phase, $U_i$ submits $ID_i, PW_i$, Smart Card

**Login Phase**

$CID_j = h(PW_j) \oplus h(N_j \oplus y \oplus T_j) \oplus ID_j$  \hspace{1cm} $ID_j, CID_j, N_j, T_j$

Check $T_j - T_i \leq \Delta T$

**Verification Phase**

$h(PW_j) = CID_j \oplus h(N_j \oplus y \oplus T_j) \oplus ID_j$

$ID_j' = N_j \oplus h(x) \oplus h(PW_j)$

Verify $ID_j = ID_j'$

$a' = h(h(PW_j) \oplus y \oplus T_j)\Delta T_j$

Check $T_j - T_i \leq \Delta T$

$a = h(h(PW_j) \oplus y \oplus T_j)$

Verify $a \neq a'$

![Figure 2. Wang-Liu-Xiao-Dan’s user authentication scheme](image)

4. Security Flaws in the Previous Authentication Schemes. In this section, we are going to show that the previously presented schemes are vulnerable to some attacks. For their security analyses, we use the same assumption as Kim-Chung’s scheme that all existing smart cards are vulnerable because the secret values stored in a smart card could be extracted by monitoring its power consumption.

4.1. Reflection attack. We consider the scenario of a reflection attack [18] on Wang-Li’s scheme. In the login phase, if attacker $U_a$ has intercepted and blocked a message transmitted, i.e., $C_1 = \{ID_i, t, s, T\}$, he or she can impersonate the remote system and send $C_2 = \{u, v, T''\}$ to $U_i$ of the authentication phase, where $u = t$, $v = s$, and $T'' = T$ is the current timestamp. Upon receiving the first item of the received message, i.e., $T''$, $U_i$ will compute $w' = h(h(V_i \oplus T''))$. $U_i$ will be fooled into believing that the attacker is the legal remote system. Because $U_i$ cannot actually authenticate the remote system’s
3.8 M. H. ELDEFRAWY, M. K. KHAN AND K. ALGHATHBAR

Figure 3. Kim-Chung’s user authentication scheme

Identity, Wang-Li’s authentication scheme fails to provide mutual authentication as the authors claim.

4.2. Masquerade attack. We next consider the scenario of a masquerade attack on Wang-Liu-Xiao-Dan’s scheme. The adversary can masquerade as any legal user, as follows. He (or she) just selects two random numbers, \( P \) and \( I \), as \( U_i \)’s password and falsified identification, and computes a new \( N_0 = h(P) \oplus h(x) \). After that, the adversary can forge a login request by \( U_i \) by computing

\[
C_0 = h\left( ID_i \oplus h(x) \right) \]

and sends a message, \( f(I; C_0; N_0; T_1) \), to \( S_j \).

Then, the message can pass the server’s verification phase because the used timestamp, \( T_1 \), would successfully pass the check \( T_2 - T_1 \leq \Delta T \) and \( I \) would be matched with \( ID'_i = N'_i \oplus h(x) \oplus h(P) \). Thereby, \( S_j \) will accept the adversary’s login request.

4.3. Password guessing attack. We consider the scenario of a masquerade attack on Kim-Chung’s scheme. In the registration phase, the server, \( S_j \), stores \( \{K_1, K_2, R, h(\cdot)\} \) into the smart card, \( SM_i \), of \( U_i \). If an adversary has got the smart card, \( SM_i \), and extracted the secret values from it, he (or she) can easily figure out \( U_i \)’s password by performing an off-line password guessing attack as follows. The adversary picks a candidate password, \( P \), computes \( h(P) \) and \( C_0' = K_1 \oplus h(P) \), and checks whether \( C_0' \) is equal to \( K_2 \). A match indicates a correct guess of the password only if the used hash function is really collision free. However, if it is not, the attacker needs an additional verifier to check
the guessed password, which is one value from one of the previous login messages of $U_i$. However, it is not difficult to get the information because of using an insecure channel. An additional verification could be processed by computing $C_2' = K_2 \oplus h(P \oplus h(P))$ and checking whether $C_2'$ is equal to $C_2$ in the previous message. A match indicates the success of the offline password guessing. Then, the adversary could impersonate the user, $U_i$, of the smart card, $SM_i$.

5. Proposed Remote User Authentication Algorithm. In this section, we propose a new remote user authentication scheme that eliminates the security flaws described in the previous sections, as well as dispensing with time stamping. Figure 4 illustrates the proposed remote user authentication scheme. To resist such attacks, the proposed login and authentication phases are performed as follows.

![Proposed user authentication scheme](image-url)
We consider the registration stage to be established manually between \( U_i \) and \( AS \). The establishment of the algorithms and seeds requires manual intervention, e.g., \( U_i \) should go personally to the \( AS \) administrator to establish the system.

5.1. Registration phase. The user gets the two different hash functions established on his token plus two different seeds, \( h_A(\cdot) \), \( h_B(\cdot) \), \( s^\text{auth}_{2t-1} \), and \( s^\text{otp}_{2t-1} \), which are also installed on his token. Moreover, he chooses \( p, q \) and \( g \), where \( p \) is a large prime number with bit size 1024, \( q \) is a prime divisor of \((p - 1)\) with size 160-bit, and \( g \) is an element of order \( q \) in the Galois field, \( GF(p) \). The bit size of the output of \( h_A(\cdot) \) and \( h_B(\cdot) \) is \( |q| \). To ensure that the information is completely shared with the service provider, these two seeds are produced by the shared and unique parameters of the host and user.

5.2. Login and authentication phase. For the first time of the authentication process, after being prompted for his \( ID_i \) and \( PW_i \), the user sends the following vector, \( V^\text{user}_t = (x_{2t-1}, y_{2t-1}) \). The server generates \( t \), which are also installed \( h \) and \( g \). Upon receipt of \( V^\text{user}_t \), the server extracts \( x_{2t-1} = V^\text{user}_t \mod h_B(h_A(s^\text{auth}_{2t-1})) \) and checks \( g^{x_{2t-1}} = g^h \). Unless positive results are obtained, the authentication process fails. Otherwise, the server calculates \( V^\text{server}_t = (x_{2t}, y_{2t}) \) and sends them to \( U_i \), who checks \( g^{x_{2t}} = g^h \). Upon confirmation, both \( AS \) and \( U_i \) will calculate the shared session key, \( \kappa = g^{x_{2t-1}} \mod p \).

6. Numerical Illustration. In this section we need to present a practical and convincing model to address the practical use of our theoretical results. In the stage of registration we need to agree on the following:

\( p \) is a prime number of 1024 bit length:
\[ <113832011549867329696708834397458157323512665938227811759517696593165356147 \]

\( q \) is a prime number of typically 160 bits with \( |q| \):
\[ <7871177362971368137989833740146907250185681957> \]

\( g \) is a generator of \( Z_p^* \), \( g = k^{\frac{1}{p-1}} \mod p \) at \( k \) is any integer with \( 1 < k < p - 1 \), \( k = 2 \):
\[ <10316132990682464412076827632861088995920812582077313382959548880702180757 \]

\( h_A(\cdot) \) and \( h_B(\cdot) \) are presented by SHA-1 and MD5 in order, also \( s^\text{auth} \) and \( s^\text{otp} \) take the following initial values \( <123456789> \) and \( <987654321> \). In the step of login and authentication, in the earliest authentication round \( t = 1 \), the remote user generates \( (x_1, y_1) \), randomly, to be \( <243> \) and \( <521> \), also calculates \( V^\text{user}_t = (x_1, y_1) \). Also, he calculates \( otp^\text{user}_t = h_B^{521}(h_A^{243}(s^\text{auth}_1)) \) and \( otp^\text{server}_t = h_B^{572}(h_A^{234}(s^\text{otp}_1)) \). As well as \( g^{otp^\text{server}_t} \mod p > 2772 \) and \( 753942923366301599427453758416891089758695572404574058570904877496067707784 \)
starts generating two uniformly distributed random values \( \text{exchanged vectors will be detected by the authentication partners.} \)

Seeds updating is going as the following: \( s_2^\text{auth} = h_A(\pi^\text{auth}) = <12577401518738800765610779722141167686351282942, s_2^\text{otp} = (h_A^{243}(s_1^\text{otp})) = <72708630625729893579429235264584363010387620287>. \)

In the verification phase, the user transfers \( g_{\text{auth}}^\text{user}, V_1^\text{user}, \text{calculated above}, \) to his remote server to let it calculate the challenges values \( (x_1, y_1) \) to obtain \( otp_1^\text{User} \) and consequently \( g_{\text{auth}}^\text{user} \mod p \). Upon the validity check of \( g_{\text{auth}}^\text{user} = g_{\text{auth}}^\text{user} \mod p \), the authentication server starts generating two uniformly distributed random values \( (x_2, y_2) = <359 \) and \( <434 \), and then calculates \( V_1^\text{server} = (x_2, y_2)^\pi h_A(h_A(s_2^\text{auth})) = <127823686367185312175428045796710836926 \), and \( otp_2^\text{Server} = h_B^{127}(h_A^{359}(s_2^\text{otp})) = <264796102732857081515780450562152243660, \) and consequently \( g_{\text{auth}}^\text{server} \mod p \) \( <6135864183216884915323832665638607898391217157371690431583319149581141963286595810062486015671862054771075863382013934993257516097688065985292686609691137626366710579983610584043275698169578638944341874002895706676000779485779903462608905454786588264011262925465202019673876605014727523122272135782456623>. \) Now, the authentication server responds to remote user with \( V_1^\text{server} \) and \( g_{\text{auth}}^\text{server} \). This allows the user to extract \( (x_2, y_2) \) to reach \( otp_2^\text{Server} \) and consequently \( g_{\text{auth}}^\text{server} \mod p \). Upon the validity check of \( g_{\text{auth}}^\text{server} = g_{\text{auth}}^\text{server} \mod p, \) both user and server calculate the shared session key of \( \pi_1 = g_{\text{auth}}^\text{user} \mod p = <68246021157114473074633395601168628754239511927565629586328788423888635874855294922563908116290321983673240735856942191446849274011333943615148742082289078472095723830962811134898147573027675925689125306804820221501896435112156113922788770235179496005722536462145804776935914478094419804233819174373>. \)

The second hash function \( h_B \) allowed us to go in the forward direction by protecting the \( h_A \) produced chain. Also, it is not admissible for \( (x_{2t-1}, y_{2t-1}) \) to be equal to zero.

7. Security Analysis. Naturally, the proposed scheme can resist an off-line guessing attack because it uses strong passwords with strong hash functions. Moreover, the gaining of unauthorized access by replaying reusable passwords is restricted by encoding passwords, which are used once. In this section, we will briefly give a security assessment of our proposed scheme.

7.1. Pre-play attack. Unless the challenge is protected, a type of “suppress-replay attack” becomes possible [2]. Consider that an intruder, \( E \), who is able to predict the next challenge, wishes to impersonate AS to U. \( E \) takes the AS role, by impersonating it to Ui. The challenge sent by Ui is memorized by E. Then, at some future time, \( E \) can impersonate \( Ui \) to AS, using this memorized response. Our proposal allows the challenges to be unpredictable uniformly distributed values of \( x_i \) and \( y_i \). If we suppose that \( x_i \) and \( y_i \) can take one value of forward \( m \) values, and the probability of successfully guessing a challenge will be the joint probability of \( x_i \) and \( y_i \), which is equal to \( 1/m^2 \). We can refer to this property as the ability to resist predictable attacks. The restriction of transferring password information in just one direction, from user to host, also increases the robustness against this type of attack. The two exchanged vectors between user and host are transferred in a cipher format.

7.2. Non-repudiation attack. To prevent a non-repudiation attack, the user AS have to update \( s^\text{auth} \) and \( s^\text{otp} \) twice per session. Hence, the user session will be used as the host’s next verifier, and vice versa, so that any unauthorized modification of the exchanged vectors will be detected by the authentication partners.
7.3. **Forgery attack.** To mount a forgery attack on the proposed scheme, an adversary must generate an authentication message corresponding to the given $g^{otp_{2i-1}}$, $V^\text{user}_i$ vectors. Because the adversary does not know $s^{auth}$ and $s^{otp}$, he cannot correctly update the seeds nor correctly produce session OTPs that will be accepted by the communicating parties. Hence, the proposed scheme can resist the forgery attack.

7.4. **Insider attack.** If an AS insider tries to impersonate the user to access other hosts using the shared Session Key, $\kappa = g^{otp_{2i-1}} \text{mod} p$ between them, s/he will not be able to do so because the cooperation of the OTPs’ seed fabrication between this user and the different hosts is strong. Furthermore, as the OTP production, using two different types of strong hashes, $h_A(\cdot)$ and $h_B(\cdot)$, is strong, the host insider cannot derive those OTPs by performing an off-line guessing attack on what he has received.

7.5. **Small challenge attack.** Attacks based on sending small challenges by intruders who impersonate the communication host only affect the backward hash chains’ OTPs. Our scheme uses forward hashing techniques, which eliminates this type of attack completely.

7.6. **Mutual authentication.** The AS is authenticated to $U_i$ by checking the equality of $g^{otp_{2i-1}} = g^{otp_{2i-1}}$ and the user is also authenticated to the host by verifying $g^{otp_{2i-1}} = g^{otp_{2i-1}}$. This confirms the occurrence of the two-way authentication.

7.7. **Known key security.** The protocol provides known-key security. Hence, each run of the protocol between two principal $U_i$ and AS should produce a unique session key, which depends on $otp_{2i}$ and $otp_{2i-1}$. Even if an adversary has learned some other session keys, he cannot predict a new $otp_{2i}$ nor a new $otp_{2i-1}$. Therefore, the protocol still achieves its goal in the face of the adversary.

7.8. **Small subgroup attack.** One way to avoid a small subgroup attack [19] is to make $G$ a $\mathbb{Z}_q$ subgroup of a prime order, $q$. This is done by choosing $g$, a subgroup generator, to have a prime order $q$. In this case, the only proper subgroup of $G$ consists of a single identity element. Moreover, the provided key confirmation property has the responsibility to notify the principals of this type of attack. Hence, the shared secret will be equal to $\kappa^w \neq \kappa$, at $w = (p - 1)/r$, where $r$ is a small factor of $(p - 1)$.

7.9. **Implicit key authentication.** An authentication protocol is said to provide implicit key authentication (of $U_i$ to AS) if the AS is assured that no other entity aside from a specifically identified $U_i$ can learn the value of a particular secret key. Any modification of the exchanged messaged cannot be recovered by an intruder. Hence, the random challenges values are only known to the challenger, and $s^{auth}$ and $s^{otp}$ are only known by the communicating parties.

7.10. **Key confirmation.** The assurance of the rightful participants in a key-establishment protocol is that the intended recipient of the shared key actually possesses the shared key. The two checks of $g^{otp_{2i-1}} = g^{otp_{2i}}$ and $g^{otp_{2i-1}} = g^{otp_{2i-1}}$ have the responsibility of providing this key confirmation.

7.11. **Explicit key authentication.** If both implicit key authentication and key confirmation (of $U_i$ to AS) are provided, the key establishment is said to provide explicit key authentication (EKA) (of $U_i$ to AS). A key agreement that provides explicit key authentication to both principals is called authenticated key agreement with key confirmation (AKC) protocol.
7.12. **Key control.** Neither of the principles must be able to force the key to be any chosen value; otherwise one party could force the use of an old key. One potential benefit is that each principle does not have to rely on any other party to generate appropriate keys. As long as neither party is malicious, it can often be guaranteed that the session key is sufficiently random inputs by the utilization of random and uniformly distributed values \((x, y)\). A related benefit is that the principals can often be sure that the session key is fresh by ensuring that their own input is fresh.

7.13. **Key freshness.** The derived session key is fresh, as opposed to the reuse of old keying material. This is to be done by maintaining the randomization of the generated \(otp\) and consequently the generated session key \(\kappa = g^{otp_{2t-1} \cdot otp_{2t}} \mod p\).

8. **Performance Analysis.** To compare the performances of our scheme and the other schemes we should consider the computational cost for each one, by counting the number of Hash and Exponential operations for the \(t\)th login with respect to the chain algorithm from the user side, \(U_i\). The hash computation costs of the proposed scheme in the registration, login, and authentication phases are summarized in Table 1.

| Table 1. Comparison of our scheme and its related schemes in terms of computational costs |
|-----------------------------------------------|-----------------------------------------------|
| **Proposed Scheme**                           | **Kim-Chung’s Scheme**                         |
| \(U_i\)                                      | \(AS\)                                       | **Login Phase**                             | **Verification Phase** | **Communication Costs** | **Time Stamping** |
| \(\left(\frac{2 + x_{2t-1}}{y_{2t-1}}\right) + 1Xor + 1Exp\) Hash | \(-\)                                        | \(4Xor + 3Hash\)                            | \(-\)                   | \(-\)                   | NOT-required     |
| \(\left(\frac{2 + x_{2t}}{y_{2t}}\right) + 1Xor + 1Exp\) Hash | \(\left(\frac{4 + x_{2t-1}}{y_{2t-1} + y_{2t}}\right) + 2Xor + 2Exp\) Hash | \(3Xor + 1Hash\)                            | \(9Xor + 3Hash\)       | \(-\)                   | Required         |
| \(160 + 1024\) bits                         | \(160 + 1024\) bits                         |                                              |                         |                         |                   |
| **Wang-Liu-Dan’s Scheme**                    | **Wang-Li’s Scheme**                         |
| \(U_i\)                                      | \(AS\)                                       | **Login Phase**                             | **Verification Phase** | **Communication Costs** | **Time Stamping** |
| \(4Xor + 2Hash\)                            | \(-\)                                        | \(1Exp + 3Hash + 1Xor\)                     | \(-\)                   | \(-\)                   | Required         |
| \(2Xor + 2Hash\)                            | \(8Xor + 3Hash\)                             | \(1Exp + 2Hash\)                           | \(2Exp + 5Hash\)       | \(-\)                   | Required         |
| \(\left(\frac{160}{1024}\right)\) bits     | \(\left(\frac{160}{1024}\right)\) bits     |                                              |                         |                         |                   |

Note that \(\text{Exp}\) denotes Exponentiation operations, \((\oplus)\) stands for an operation of exclusive-or, and Hash means Cryptographic hash operations. It is clear that our scheme’s computation cost Hashes of the login and verification phases are increased by the random and uniformly distributed values \((x, y)\). This is the payment for removing time stamping and time synchronization. An authentication mechanism that uses time-stamping [14] requires time synchronization capability for a certain reference between the \(U_i\) and \(AS\), which is not easy to apply in many applications. All of the algorithms that have previously
been discussed need accurate time synchronization to achieve a reliable time-stamping in the login and the authentication process. However, our proposed algorithm does not need any time synchronization to prevent the clock un-synchronization problem. Although our proposed protocol additionally needs \((x, y)\) hash function operations to complete the whole authentication procedure, our scheme is efficient. This is because only lightweight operation modules such as a one-way hash function and exclusive-or operation are required. In Borst et al., [20] the authors showed that there are ways to implement a sufficiently fast hash function on a smart card to perform authentication and other access control mechanisms.

9. Conclusions. Identifying remote users as legal or illegal is a key issue in network security. This study proposed a remote user authentication scheme using two-nested-one-way hash functions and the discrete logarithm problem. The proposed authentication scheme is different from other such schemes in avoiding the need for time stamping and consequently time synchronization. Because our proposed algorithm mainly uses the one-way hash function in the login and authentication processes, thus the problems associated with the cost of computation can be avoided. A detailed security analysis was also performed that covered many types of attacks that could influence our scheme. The proposed scheme not only provides mutual authentication between the user and server, but also establishes a common session key to provide message confidentiality. Future research will apply the proposed scheme to more complex user authentication environments with smart cards such as a multi-server environment with a single user in which clients can be remotely authenticated once and access multiple servers at different locations as many times as they want.

REFERENCES