An Enhanced Authenticated Key Agreement for Session Initiation Protocol

Mohammad Sabzinejad Farash¹, Mahmoud Ahmadian Attari²

¹ Department of Information and Communication Technology, Malek Ashtar University of Technology, Tehran, Iran. 
E-mail: sabzinejad@khu.ac.ir

² Faculty of Electrical and Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran. 
E-mail: mahmoud@eetd.kntu.ac.ir

crossref http://dx.doi.org/10.5755/j01.itc.42.4.2496

Abstract. In 2012, Xie proposed an authentication scheme based on Elliptic Curve Cryptography (ECC) for Session Initiation Protocol (SIP). However, this paper demonstrates that the Xie’s scheme is vulnerable to impersonation attack by which an active adversary can easily forge the server’s identity. Based on this attack, we also show that the Xie’s scheme is also defenseless to off-line password guessing attack. Therefore, we propose a more secure and efficient scheme, which does not only cover all the security flaws and weaknesses of related previous protocols, but also provides more functionalities. We also evaluate the proposed protocol by AVISPA (Automated Validation of Internet Security Protocols and Applications) tools and confirm its security attributes.

Keywords: Authenticated Key Agreement; Elliptic Curve; Impersonation Attack; Password Guessing Attack; Session Initiation Protocol; AVISPA tools.

1. Introduction

The session initiation protocol (SIP) is an application layer signalling protocol for creating, modifying, and terminating multimedia sessions among one or more participants. SIP was developed by the Internet Engineering Task Force (IETF) in 1996. With the widespread application of the Voice over IP (VoIP) in Internet [1] and mobility management [2–4], SIP has been receiving a lot of attention and the security of SIP is becoming increasingly important [5]. When a user wants to access a SIP service, he or she has to perform an authentication process from the remote server. Thus, authentication is one of the most important issues for SIP. Various authentication schemes, especially based on Elliptic Curve Cryptography (ECC), have been proposed to provide security for SIP for a decade [6–12].

In 2005, Yang et al. [13] indicated that the original SIP authentication scheme is vulnerable to off-line password guessing attack and server-spoofing attack. To overcome the attacks, Yang et al. proposed a modified scheme based on Diffie-Hellman key exchange protocol. However, Huang et al. [14] pointed out that the Yang et al.’s scheme may not be suitable for users with limited computational power and further proposed a new scheme. In [15], Jo et al. demonstrated that the schemes by Yang et al. and Huang et al. are both vulnerable to off-line password guessing attack.

Based on Yang et al.’s scheme, Durlanik and Sogukpinar [16] introduced an efficient authentication scheme for SIP by using Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol. Because of the adoption of elliptic curves, Durlanik and Sogukpinar’s scheme reduced the total execution time and the requirements for memory in comparison with Yang et al.’s scheme. However, Yoon and Yoo [17] indicated that Durlanik and Sogukpinar’s scheme still suffered from off-line password guessing and Denning-Sacco attacks, and projected an improved scheme to overcome the weaknesses. However, Liu and Koenig [18] demonstrated that Yoon and Yoo’s scheme still puts up with off-line password guessing and insider attacks.

In 2009, Tsai [19] proposed an efficient authentication protocol based on random nonce, in which one-way hash functions and exclusive-or operations were only utilized for computing all the communication messages. As a result, the computation cost was very low and it was suitable for low computation equipment. However, it was still
defenceless to off-line password guessing, Denning-Sacco and stolen-verifier attacks, furthermore, it did not provide any key agreement, known-key secrecy and perfect forward secrecy (PFS) [20–22]. To deal with the problems, Arshad and Ikram proposed an ECC-based authentication scheme [22]. But, Tang and Liu [23] demonstrated the vulnerability of Arshad and Ikram’s scheme to off-line password guessing attack and introduced an improved scheme to overcome the weakness.

In 2010, Yoon et al. [24] also proposed an authentication scheme based on ECC to deal with the problems in Tsai’s scheme [19]. In 2012, Xie [25] pointed out that Yoon et al.’s scheme still suffers from stolen-verifier and off-line password guessing attacks, and proposed a new security enhanced scheme for SIP to solve these problems. However, in this paper, we indicate that the Xie’s scheme is still vulnerable to impersonation attack, by which an active adversary can easily forge the identity of the server. Based on this attack, we also show that the Xie’s scheme still suffers from off-line password guessing attack. Then, we propose an improved scheme to enhance the security of the Xie’s scheme. Our improved scheme does not only maintain the merits and cover the demerits of the Xie’s scheme, but also meets all the requirements of such schemes. Our scheme also provides mutual authentication with key agreement. Moreover, our scheme provides a password change phase. Specifically, the users could renew their passwords anytime and anywhere. Finally, the security analysis is presented.

Typically, the theoretical analysis of cryptographic protocols is normally used to verify the security attributes in the design. However, it is not sufficient, and simulation tools must also be employed to verify all the security requirements of the protocol. AVISPA [26] is a strong simulation engine for automated security analysis of cryptographic protocols [32]. Therefore, we make use of the AVISPA tools to confirm the security attributes of the proposed protocol.

The rest of this paper is organized as follows. In Section 2, we review the Xie’s authenticated key agreement for session initiation protocol. In Section 3, we propose impersonation attack and off-line password guessing attack on the Xie’s scheme. An enhanced authentication scheme for SIP is proposed in Section 4. The proposed protocol is then analyzed for security by the use of theoretical analysis and AVISPA tools in Section 5. In Section 6, we make a comparison between our scheme and some related schemes. Finally, Section 7 concludes the paper.

2. A brief review of the Xie’s scheme

This section briefly reviews the Xie’s authentication scheme for SIP [25]. The Xie’s scheme consists of three phases: the setup phase, the registration phase and the authentication phase. The notations used in this paper are shown in Table 1.

2.1. System setup phase

In this phase, the server S sets the following system parameters: let \( q \) be a large prime number, \( E(GF_q) \) an elliptic curve group defined over a finite field \( GF_q \), \( P \) a generator of \( E(GF_q) \) of order \( q \), and \( h(\cdot) \) a cryptographic hash function. S also selects an integer \( K_s \in (1; q) \) as the long-live secret key, and computes the corresponding public key \( Q_s = K_sP \). At the end of this phase, S publishes all parameters except \( K_s \).

2.2. Registration phase

When U wants to register and become a new legal user, U and S execute the following steps over a secure channel:

R1: U sends password \( PW \) to S via a secure channel.

R2: S computes \( VPW = EK_s(PW) \) and stores \( VPW \) to the user account database (i.e., the registration table) corresponding to U’s information.

2.3. Authentication phase

If the legal user U wants to login into S, U and S perform the following steps:

A1: U \( \rightarrow \) S: REQUEST\{username, aP\}

U chooses a random integer \( a \in (1, q) \), computes and sends \( aP \) together with his or her username in a request message to S.

A2: S \( \rightarrow \) U: CHALLENGE\{realm, bP, \sigma, KP_x\}

Upon receiving the request message, S rst randomly chooses \( b, k \in (1, q) \) and computes \( bP, SK_x = baP, (k \cdot h(SK_x \parallel bP))P = (KP_x, KP_y) \) and \( \sigma = k - (h(SK_x \parallel bP) \parallel K_s) \pmod{q} \). Then, S sends the challenge message CHALLENGE\{realm, bp, \sigma, KP_x\} back to U.

A3: U \( \rightarrow \) S: RESPONSE\{username, realm, h(username)\parallel realm \parallel SK_u\parallel PW\}

Upon receiving the challenge message, U computes \( SK_u = abP, r = \sigma \cdot h(SK_u \parallel bP) \parallel Q_s = (r_x; r_y) \) and checks if \( r_x = KP_x \). If so, U computes \( h(username) \parallel realm \parallel SK_u \parallel PW) \) and sends RESPONSE\{username, realm, h(username)\parallel realm \parallel SK_u\parallel PW\} back to S. Otherwise, U rejects it.

A4: Upon receiving the response message, S computes \( PW = DK_x(VPW) \) and \( h(username) \parallel realm \parallel SK_u \parallel PW) \), and verifies if \( h(username) \parallel realm \parallel SK_u \parallel PW) = h(username) \parallel realm \parallel SK_u \parallel PW) \). If so, U is authenticated. Otherwise, S aborts the session.

At the end of the execution of the protocol, the session key shared between U and S is set to \( SK = h(SK_u)\parallel P = h(SK_s)\parallel P) \).
I1: When the legal user follows users as a legal server. The proposed attack works as active adversary can easily introduce himself to the vulnerable to impersonation attack. We show that an Xie’s scheme

Table 1. The Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U$</td>
<td>a user</td>
</tr>
<tr>
<td>username</td>
<td>the identity of the user $U$</td>
</tr>
<tr>
<td>realm</td>
<td>client’s realm is used to prompt the username and password.</td>
</tr>
<tr>
<td>$PW$</td>
<td>the password of the user $U$</td>
</tr>
<tr>
<td>$V PW$</td>
<td>the password verifier of the user $U$</td>
</tr>
<tr>
<td>$S$</td>
<td>a remote server</td>
</tr>
<tr>
<td>$K_s$</td>
<td>the long-live secret key of the server</td>
</tr>
<tr>
<td>$Q_s$</td>
<td>the long-live public key of the server</td>
</tr>
<tr>
<td>$SK$</td>
<td>a session key</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>a strong cryptographic one-way hash function</td>
</tr>
<tr>
<td>$E_{K_s}(\cdot)$</td>
<td>a secure symmetric encryption algorithm under the secret key of the server</td>
</tr>
<tr>
<td>$D_{K_s}(\cdot)$</td>
<td>a secure symmetric decryption algorithm under the secret key of the server</td>
</tr>
<tr>
<td>$q$</td>
<td>a large prime number</td>
</tr>
<tr>
<td>$G_{F_q}$</td>
<td>a finite field with order $q$</td>
</tr>
<tr>
<td>$E(G_{F_q})$</td>
<td>an elliptic curve group defined over $G_{F_q}$</td>
</tr>
<tr>
<td>$P$</td>
<td>a generator of $E(G_{F_q})$ with order $q$</td>
</tr>
<tr>
<td>$|$</td>
<td>the string concatenation operation</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>the exclusive-or operation</td>
</tr>
</tbody>
</table>

3. Cryptanalysis of the Xie’s scheme

3.1. The proposed impersonation attack on the Xie’s scheme

In this section, we show that the Xie’s scheme is vulnerable to impersonation attack. We show that an active adversary can easily introduce himself to the users as a legal server. The proposed attack works as follows:

I1: When the legal user $U$ wants to login into the server $S$, sends the request message $\{\text{username, aP}\}$ to the $S$.

I2: An active adversary $\mathcal{A}$ may eavesdrop the communication flows between $U$ and $S$, intercept the request message $\{\text{username, aP}\}$, and do the following steps:

I2.1: Select a random number $b' \in (1, q)$ and compute $b'P$ and $SK'_s = b'aP$.

I2.2: Select another random number $\sigma' \in (1, q)$ as a signature and compute $s'(h(SK'_s || b'P))P + Q_s = (KP'_{s'}, KP'_{s'})$ (1)

I2.3: Send the challenge message $\{\text{realm, b'P, } \sigma', KP'_s\}$ back to the user $U$ on behalf of $S$.

I3: Upon receiving the challenge message, $U$ computes $SK_u = ab'P$ and $r = \sigma'(h(SK_u || b'P))P + Q_s = (r_x, r_y)$, (2)

and checks if $r_x = KP'_s$. The following statements indicate that the verification equation 2 holds:

\[
\begin{align*}
(r_x, r_y) &= s'(h(SK_u || b'P))P + Q_s \\
&= s'(h(b'P || b'P))P + Q_s \\
&= s'(h(SK'_s || b'P))P + Q_s \\
&= (KP'_{s}, KP'_{s})
\end{align*}
\]

Thus, $U$ would believe that the received message is generated by the legal server $S$. Then, $U$ computes $h(\text{username} || \text{realm} || SK_u || PW)$ and sends the response message $\{\text{username, realm, SK_u, PW}\}$ to $S$. Finally, $U$ computes the session key $SK = h(SK_u || P)$.

I4: The adversary intercepts the response message and computes the session key $SK = h(SK_u || P)$.

As can be clearly seen, the session key shared between $U$ and the adversary $\mathcal{A}$ is set to $SK = h(SK_u || P) = h(SK'_u || P) = h(ab'P || P)$. Thus, the adversary without knowing the password $PW$ and the server’s private key $K_s$ can easily impersonate the server $S$ and share a secret key with $U$.

3.2. The proposed off-line password guessing attack on the Xie’s scheme

As a result of the impersonation attack (see Section 3.1), off-line password guessing attack also can be applied to the Xie’s scheme by an active adversary. To do so, the adversary $\mathcal{A}$ applies an impersonation attack and obstructs the response message $\{\text{username, realm, h(\text{username} || \text{realm} || SK_u || PW)}\}$ at the end of the Step I3 in Section 3. After receiving the response message, $\mathcal{A}$ launches the off-line password guessing attack as follows:

G1: $\mathcal{A}$ selects a candidate password $PW'$ from the uniformly distributed dictionary of size $|D|$.

G2: As mentioned in the impersonation attack (see Section 3.1, Step I2.1), the session key $SK_u = SK'_s = ab'P$, is known for the adversary. Therefore, $\mathcal{A}$ can compute $h(\text{username} || \text{realm} || SK'_s || PW')$.

G3: $\mathcal{A}$ compares $h(\text{username} || \text{realm} || SK'_s || PW')$ with $h(\text{username} || \text{realm} || SK_u || PW')$. If they are equal, $\mathcal{A}$ guesses the right password of $U$. Otherwise, the adversary goes to the step G1 and does the next loop.

4. The proposed scheme for SIP

This section proposes an enhanced authentication scheme for session initiation protocol in order to overcome the above mentioned problems with the Xie’s scheme. The proposed protocol contains four phases: system setup phase, registration phase, login and authentication phase, and password change phase.

335
4.1. System setup phase

In the system setup phase, $S$ generates the following system parameters: an elliptic curve $E$ over a finite field $GF_q$, an additive group of points on the elliptic curve $E(GF_q)$, the generating point $P$ on $E(GF_q)$ of order $q$ and a secure hash function $h(\cdot)$. $S$ also selects an integer $K_S \in (1,q)$ as the long-live secret key, and computes $Q_S = K_SP$ as the corresponding public key. Finally, $S$ publishes the parameters $\{E(GF_q), P, q, h(\cdot), Q_S\}$.

4.2. Registration phase

Figure I shows the registration phase of our scheme. When a user wants to login into the remote server, he/she firstly should register to the remote server. In this phase, the user communicates with the server through a secure channel. The details of this phase are as follows.

R1: The user freely chooses his or her username and password $PW$, and interactively sends them to the server through a secure channel.

R2: The server computes $V PW = h(\text{username} \parallel K_S) \oplus h(\text{username} \parallel PW)$ and stores $\{\text{username}, V PW\}$ in its database.

4.3. Login and authentication phase

Figure I shows the login and authentication phase of our scheme. In this phase, the user communicates with the remote server through a public channel. When the user $U$ wants to login into the remote server, he or she performs the following steps to execute a session of the protocol:

A1: $U \rightarrow S$: REQUEST(username, $aP$)

$U$ chooses a random integer $a \in (1,q)$, computes and sends $aP$ in the request message $\text{REQUEST}(\text{username}, aP)$ to $S$.

A2: $S \rightarrow U$: CHALLENGE(realm, $bP$, $\sigma$)

Upon receiving the request message, $S$ first randomly chooses $b \in (1,q)$ and computes $bP$, $SK_u = baP$ and $\sigma = h(SK_u \parallel K_S aP \parallel bP \parallel aP)$. Then, $S$ sends the challenge message $\text{CHALLENGE}(\text{realm}, bP, \sigma)$ back to $U$.

A3: $U \rightarrow S$: RESPONSE(realm, $H$)

Upon receiving the challenge message, $U$ computes $SK_u = abP$ and checks if $\sigma = h(SK_u \parallel aQ_u \parallel bP \parallel aP)$. If so, $U$ computes $H = h(\text{realm} \parallel SK_u \parallel h(\text{username} \parallel PW))$ and sends $\text{RESPONSE}(\text{realm}, h(\text{realm} \parallel H))$ back to $S$ and computes the session key $SK = h(\text{username} \parallel SK_u \parallel aP \parallel bP)$. Other-wise, $U$ rejects it.

A4: Upon receiving the response message, $S$ verifies if $h(\text{realm} \parallel SK_u \parallel \{V PW \odot h(\text{username} \parallel K_S)\}) = H$.

If so, $U$ is authenticated and $S$ computes the session key $SK = h(\text{username} \parallel SK_u \parallel aP \parallel bP)$. Otherwise, $S$ aborts.

Finally, the session key shared between $U$ and $S$ is set to

$$SK = h(\text{username} \parallel SK_u \parallel aP \parallel bP)$$

$$= h(\text{username} \parallel SK_u \parallel aP \parallel bP)$$

4.4. Password change phase

Figure 1 shows the password change phase of our scheme. The user $U$ can change the password freely in this phase. To do so, he/she firstly needs to execute the login and authentication phase with his/her username and old password $PW$. After receiving the successful authentication confirmation from the server and sharing the session key $SK$, the user $U$ inputs the new password $PW^*$ as follows:

C1. $U \rightarrow S$: $\{PWD, V\}$

The user $U$ computes $PWD = h(SK \parallel \text{username}) \oplus h(\text{username} \parallel PW^*)$ and $V = h(SK \parallel h(\text{username} \parallel PW^*))$, and sends them to the server.

C2. $S \rightarrow U$: $\{\text{Accept}, R_1\}$ or $\{\text{Reject}, R_2\}$

Upon receiving the message $PWD$ and $V$, the server computes $H_2 = PWD \oplus h(SK \parallel \text{username})$ and checks whether $V$ is equal to $h(SK \parallel H_2)$. If so, the server accepts the password change request, computes $R_1 = h(\text{Accept} \parallel \text{username} \parallel PWD \parallel V \parallel SK)$ and sends $\{\text{Accept}, R_1\}$ back to the user. Otherwise, they are not equal, the server rejects the password change request, computes $R_2 = h(\text{Reject} \parallel \text{username} \parallel PWD \parallel V \parallel SK)$ and sends $\{\text{Reject}, R_2\}$ back to the user. Finally, the server replaces $V PW$ with $PW^* = h(\text{username} \parallel K_S) \oplus H_2$.

It is obvious that the verification equation $h(\text{realm} \parallel SK_u \parallel \{PW^* \odot h(\text{username} \parallel K_S)\}) = H$ in Section 4.3, item A.4 is passed because

$$PW^* = h(\text{username} \parallel K_S) \oplus H_2$$

$$= h(\text{username} \parallel K_S) \oplus PW \oplus h(SK \parallel \text{username})$$

$$= h(\text{username} \parallel PW^*) \oplus h(SK \parallel \text{username})$$

and,$$

h(\text{realm} \parallel SK_u \parallel \{PW^* \odot h(\text{username} \parallel K_S)\})$$

$$= h(\text{realm} \parallel SK_u \parallel h(\text{username} \parallel PW^*))$$

$$= h(\text{realm} \parallel SK_u \parallel h(\text{username} \parallel PW^*)) = H$$
An Enhanced Authenticated Key Agreement for Session Initiation Protocol

<table>
<thead>
<tr>
<th>Registration phase</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>select username and PW</td>
<td>{username,PW}</td>
</tr>
<tr>
<td>compute V_PW = h(username</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Login and authentication phase</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>randomly choose a ( \in (1, q) )</td>
<td>{username, aP}</td>
</tr>
<tr>
<td>compute bP, SK_u = baP and</td>
<td>compute bP, SK_u = baP and \sigma = h(SK_u</td>
</tr>
<tr>
<td>{realm, bP, \sigma}</td>
<td>{realm, H}</td>
</tr>
<tr>
<td></td>
<td>and the session key SK = h(username</td>
</tr>
<tr>
<td></td>
<td>verify if h(realm</td>
</tr>
<tr>
<td></td>
<td>If so, compute the session key SK = h(username</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password change phase</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>randomly select a new password PW*</td>
<td>{PWD, V}</td>
</tr>
<tr>
<td>compute H_2 = h(username</td>
<td></td>
</tr>
<tr>
<td>PWD = h(SK</td>
<td></td>
</tr>
<tr>
<td>and V = h(SK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>compute R_1 = h(Accept</td>
</tr>
<tr>
<td></td>
<td>compute V_PW* = h(username</td>
</tr>
<tr>
<td></td>
<td>replace V_PW with V_PW*</td>
</tr>
<tr>
<td></td>
<td>otherwise, reject the password change request,</td>
</tr>
<tr>
<td></td>
<td>compute R_2 = h(Reject</td>
</tr>
<tr>
<td></td>
<td>{Accept/Reject, R_1/R_2}</td>
</tr>
</tbody>
</table>

Figure 1. The proposed protocol
5. Security analysis

5.1. Theoretical analysis

Replay attack. Suppose an attacker \( \mathcal{A} \) intercepts \( \text{REQUEST}(\text{username}, aP) \) from \( U \) in step A1 and replays it to impersonate \( U \). However, \( \mathcal{A} \) cannot compute a correct session key \( SK = abP \) and deliver it to \( S \) in step A3 unless he/she can correctly guess the password \( PW \) and guess \( b \) from \( bP \) or \( a \) from \( aP \). When \( \mathcal{A} \) tries to guess a from \( aP \) or \( b \) from \( bP \), he/she will face the Elliptic Curve Discrete Logarithm Problem (ECDLP) which is untraceable. On the other hand, suppose \( \mathcal{A} \) intercepts \( \text{CHALLENGE}(realm, bP, s) \) from \( S \) in step A2 and replays it to impersonate \( S \). The replied message cannot pass the verification process \( \sigma = h(SK_u \| aQ_s \| bP \| aP) \), since \( a \) is a new nonce chosen by \( U \) in each session and the adversary has no control of it. Therefore, the proposed scheme can resist the replay attack.

Stolen-verifier attack. When attacker \( \mathcal{A} \) steals the secret key \( K_u \) of user \( U \) from the database of the server, \( \mathcal{A} \) cannot obtain the right password \( PW \) from \( VPW \) without knowing the secret key \( K_u \) of the server, which is a high entropy number and cannot be guessed by enumeration. Therefore, the proposed scheme is secure against stolen-verifier attack.

Denning-Sacco attack. Attacker \( \mathcal{A} \) may obtain the session key \( SK = h(\text{username} \| K_u) \Theta h(\text{username} \| PW) \) for some reasons, but he/she cannot obtain user’s secret password \( PW \) and server’s secret key \( K_u \) because he/she will face to obtain \( abP \) which is protected by a hash function.

Impersonation attack. An adversary \( \mathcal{A} \) cannot masquerade as server, because he/she cannot compute the signature \( \sigma = h(SK_u \| K_u aP \| bP \| aP) \) with-out knowing the server’s secret key \( K_u \). \( \mathcal{A} \) also cannot impersonate the user to authenticate with the server, because he/she cannot construct the message \( \text{RESPONSE}(\text{username}, \text{realm}, h(\text{realm} \| SK_u \| h(\text{username} \| PW))) \) without the knowledge of \( PW \). Therefore, the proposed scheme resists im-personation attack.

Password guessing attack. It is divided into online password guessing attack and off-line password guessing attack. Online password guessing attack can be preserved by limiting the login times. The exchanged messages between the user and the server, in the login phase (step A1) and verification phase (step A2), are independent of the user’s password; therefore the adversary cannot execute any off-line guessing attack on our scheme.

Man-in-the-middle attack. Password \( PW \) of \( U \) and the secret key \( K_u \) of \( S \) are used to prevent the man-in-middle attack. Therefore, the active adversary \( \mathcal{A} \) cannot intrude into the communication between \( S \) and \( U \) to intercept the exchanged data and inject false information.

Modification attack. An adversary \( \mathcal{A} \) cannot modify the communicated messages \( (\text{username}, aP) \) in step A1, \( (\text{realm}, bP, \sigma) \) in step A2 and \( (\text{username}, \text{realm}, h(\text{realm} \| SK_u \| h(\text{username} \| PW))) \) in step A3, because the user and the server detect them by verifying \( s \) and \( h(\text{realm} \| SK_u \| h(\text{username} \| PW)) \), respectively.

Known-key security. In this attack, an adversary, who has some previous session keys, is willing to compute the next session keys. Assume that some previous session keys are known for the adversary \( \mathcal{A} \). It does not give any useful information to \( \mathcal{A} \) for computing the next session keys, because the short-term private keys \( a \) and \( b \) are changed in each session. Note that, \( \mathcal{A} \) cannot obtain \( (aP \) or \( bP \) because he/she will face the ECDLP. Therefore, the proposed protocol satisfies the known-key security.

Perfect forward secrecy. Perfect forward secrecy means that if long-term private keys of one or more entities are compromised, the secrecy of previous session keys established by the trusted entities is not affected. In the proposed protocol, the adversary who knows \( PW \) and \( K_u \) cannot determine the previous session keys because long-term private keys are not utilized for computing the session keys. In addition, the adversary cannot compute neither \( a \) nor \( b \) from \( s, aP, bP \) and \( K_u \) since he/she has to solve Elliptic Curve Diffie-Hellman Problem (ECDHP). Therefore, the proposed protocol satisfies the perfect forward secrecy.

5.2. Simulation results

In the last decade, we have witnessed the development of a large number of new techniques for the formal analysis of security protocols. Until now, many (semi-)automated security protocol analysis tools have been proposed (e.g., [26-28]). One of the tools that has seen the widest use is the AVISPA [26] which is a push-button tool for the automated
validation of Internet security-sensitive protocols and applications. It provides a modular and expressive formal language for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques.

The architecture of AVISPA is shown in Figure 2. The first step in using the tool is to present the analyzed protocol in a special language called High Level Protocol Specification Language (HLPSL). The HLPSL presentation of the protocol is translated into the lower level language called Intermediate Format (IF). This translation is performed by the translator called HLPSL2IF. This step is totally transparent to the user. IF presentation of the protocol is used as an input to the four different back-ends: On-the-fly Model-Checker (OFMC), CL-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree-Automata-based Protocol Analyzer (TA4SP). These back-ends perform the analysis and output the results in precisely defined output format stating whether there are problems in the protocol or not.

In order to evaluate the security of the proposed protocol by the AVISPA tools the protocol is coded in HLPSL. The HLPSL code of the proposed protocol is included in Appendix A. After execution of the code in AVISPA tool, the outputs of OFMC (Figure 3), CL–AtSe (Figure 4) and SATMC (Figure 5) back-ends were generated. According to the summary results, the proposed protocol is SAFE and there are no major attacks on it. Therefore, these results confirm the theoretical analysis in Section 5.1.

6. Security and performance comparison

In this section, we evaluate the performance and functionality of our proposed protocol and make comparisons with some related authenticated key agreement for session initiation protocols. Table 2 shows the main computation cost of our scheme. Table 3 shows the performance comparisons of our proposed protocol and some other related protocols.
We mainly consider the computations of login and authentication phase and session key agreement since these are the principal parts of an authentication protocol and should be implemented for each session. In Table 3, it is obvious that the computation cost of the proposed protocol is lesser than the Xie's scheme. However, it is worth several additional hash operations to achieve the security and functionality properties.

Table 4 lists the security comparisons among our proposed protocol and other related protocols. It demonstrates that our protocol has many excellent features and is more secure than other related protocols.

### Table 2. Computation cost of login and authentication phase

<table>
<thead>
<tr>
<th></th>
<th>User</th>
<th>Server</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of scalar multiplication</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>No. of hash function</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>No. of exclusive or</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 3. Comparison of computation costs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of exponentiation</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>No. of scalar multiplication</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>No. of point addition</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>No. of hash-to-point</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>No. of hash function</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>No. of exclusive or</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>No. of modular inverse</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>No. of symmetric key encryption</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Security</td>
<td>ECDLP</td>
<td>DLP</td>
<td>HASH</td>
<td>ECDLP</td>
<td>ECDLP</td>
<td>ECDLP</td>
<td>ECDLP</td>
<td>ECDLP</td>
</tr>
</tbody>
</table>

### Table 4. Comparison of security attributes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reply attack</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>Man-in-the-middle attack</td>
<td>Secure</td>
<td>Secure</td>
<td>Insecure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Insecure</td>
<td>Secure</td>
</tr>
<tr>
<td>Impersonation attack</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Secure</td>
<td>Insecure</td>
<td>Secure</td>
<td>Insecure</td>
<td>Secure</td>
</tr>
<tr>
<td>Password guessing attack</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Secure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Secure</td>
</tr>
<tr>
<td>Denning-Sacco attack</td>
<td>Insecure</td>
<td>N/A</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>Stolen-verifier attack</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td>Mutual authentication</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
</tr>
<tr>
<td>Session key security</td>
<td>Provided</td>
<td>N/A</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
</tr>
<tr>
<td>Known key secrecy</td>
<td>Provided</td>
<td>N/A</td>
<td>Not provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
</tr>
<tr>
<td>Perfect forward secrecy</td>
<td>Provided</td>
<td>N/A</td>
<td>Not provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
<td>Provided</td>
</tr>
</tbody>
</table>

N/A: Not Applicable or Not Available

### 7. Conclusions

In this paper, we briefly reviewed the Xie's authenticated key agreement protocol session initiation protocol. We demonstrated that the Xie's scheme is vulnerable to the impersonation attack in which an active adversary with-out knowing the users' password and the server's private key can easily impersonate the server to the users and share secret keys with them. As a result of the impersonation attack, we pointed out that the Xie's scheme also suffers from the off-line password guessing attack. The main aw of the Xie's scheme is due to the signature scheme used by the server which is forgeable. To overcome the security weaknesses, we proposed an improved scheme. In comparison to the related schemes, the proposed scheme not only is secure against well-known crypto-graphical attacks such as guessing attacks, replay attacks, but also provides mutual authentication, perfect forward secrecy and secure password change.
References


Received September 2012.
Appendix A. HLPSL code of the proposed protocol

role client(
  A,S : agent,
  SND,RCV : channel (dy),
  H : hash_func,
  P, Qs : public_key)
played by A
def=
  local State : nat,
  PW : symmetric_key,
  Kas, Ra, Sigma, SKu, Ta, Ts, F : text,
  Username, Realm : message
const sec_kas1, sec_sk1, sec_ra, sec_pw : protocol_lid
init State :=0
transition
1. State =0/ \RCV(start)=| >
  State' :=1
  \Ra' :=new()
  \snd(\Ra'.Username)
  \witness (A, S, na, Ra')
  \secret(Ra', sec_ra, A)
2. State=1
  \RCV(Realm.{{Ra}.Rs}_P.H(Username.PW)) = | >
  State' :=2
  \F' :=H(Realm.{{Ra}.Rs}_P.H(Username.PW))
  \snd(Realm.F')
  \secret(Kas', sec_kas1, {A,S})
  \request (A, S, na, Ra)
  \request (A, S, na, PW)
end role

role server(
  S,A : agent,
  SND,RCV : channel(dy),
  H : hash_func,
  P, Qs : public_key)
played by S
def=
  local State : nat,
  PW : symmetric_key,
  Ra, Rs, Sigma, SKs, Ta, Ts, F, Kas : text,
  Username, Realm : message
const sec_kas2, sec_rs, sec_pw1 : protocol_lid
init State :=0
transition
1. State =0/ \RCV({Ra'}_P.Username)=| >
  State' :=1
  \Ra' :=new()
  \snd({Ra'}_P.Username)
  \witness (S, A, na, Ra')
  \secret(Ra', sec_ra, S)
  \secret(PW, sec_pw, S)
  \request (A, S, na, Ra)
end role

role session(
  A,S : agent,
  H : hash_func,
  P, Qs : public_key)
def=
  local
  SA, RA, SS, RS : channel (dy)
composition
client (A, S, SA, RA, H, P, Qs)
  \server (S, A, SS, RS, H, P, Qs)
end role

role environment( )
def=
  const
  na, ns : protocolid,
  a, s, i : agent,
  h : hash_func,
  p, qs, qi : public_key
intruder_knowledge={a, s, h, p, qs, qi}
composition
session (a, s, h, p, qs)
  \session (a, i, h, p, qi)
  \session (i, s, h, qs, qi)
end role

goal
secrecy_of
  sec_kas1, sec_kas2, sec_ra, sec_rs, sec_pw, sec_pw1
authentication_on na
authentication_on ns
end goal