# Efficient and Flexible Password Authenticated Key Agreement for VoIP Session Initiation Protocol Using Smart Card

Liping Zhang, Shanyu Tang*, *Senior Member IEEE*, Zhihua Cai

*Secure Communication Institute, China University of Geosciences, Wuhan, 430074, China*

**Corresponding author: shanyu.tang@gmail.com, carolyn321@163.com,Tel/Fax:+862767848563*

**Short Title: Authenticated Key Agreement for Session Initiation Protocol**

## ABSTRACT

Providing a suitable key agreement protocol for session initiation protocol is crucial to protecting the communication among the users over the open channel. This paper presents an efficient and flexible password authenticated key agreement protocol for session initiation protocol associated with Voice over Internet Protocol (VoIP). The proposed protocol has many unique properties, such as session key agreement, mutual authentication, password updating function and the server not needing to maintain a password or verification table etc. In addition, our protocol is secure against the replay attack, the impersonation attack, the stolen-verifier attack, the man-in-middle attack, the Denning-Sacco attack, and the offline dictionary attack with or without the smart card.

**KEY WORDS:** key agreement; mutual authentication; session initiation protocol; elliptic curve

## 1. INTRODUCTION

Recent advances in Internet technology have enabled the development of Voice over Internet Protocol (VoIP). Compared with traditional Public Switched Telephone Networks (PSTNs), VoIP has many attractive merits such as low cost devices, deployment, operation, and maintenance etc. So VoIP is receiving much attention and becomes a strong competitor to traditional PSTNs. The designers of the VoIP communication systems mainly focus on a good level of quality of service (QoS) and do not pay enough attention on security problems. In a VoIP call, the voice packets are delivered and exposed to the unsecured public Internet. Therefore, VoIP calls are more likely to be threatened by attacks than conventional telephone calls. If VoIP tend to dominate the voice call market, a comparable level of QoS and network security should be provided.

Among many protocols used to handle sessions for VoIP, the Session Initial Protocol (SIP) is the widely used one, and the security of SIP is becoming increasingly important. The session initiation protocol was proposed for Internet Protocol (IP) based telephony by Internet Engineering Task Force Network Working Group [1]. SIP is an application layer control protocol for creating, modifying, and terminating multimedia sessions between participants [1]. As a request-response protocol, SIP authentication is inherited from HTTP Digest authentication, which makes it vulnerable to several types of security threats and attacks such as impersonation, eavesdropping, and message modification etc. An authentication key agreement is one of the most crucial technologies for achieving acceptable security level when SIP is used to protect the communications among the users. Confidentiality and authentication are two fundamental security services requirements for SIP. Therefore, mutual authentication and key agreement should be provided for secure communication between the users. Mutual authentication is needed in SIP connections to ensure that the call is establishing only between the legitimate users. To achieve secure communication, the shared session key generated through the key agreement process is used to encrypt/decrypt the voice packets so that only the intended recipient can decrypt and retrieve the valid messages.

The rest of this paper is organized as follows. Section 2 describes the related work. Some preliminaries are reviewed in Section 3. Section 4 presents our authenticated key agreement protocol. In Section 5, the security of our proposed protocol is discussed. The performance of the protocol is discussed in Section 6, and the paper is concluded in Section 7.

## 2. RELATED WORK

The original authentication protocol for SIP was based on hyper text transport protocol digest authentication [2], which was not strong enough for providing acceptable security level in practice. In 2005, Yang et al. (2005) [3] argued that the original SIP authentication protocol was vulnerable to the off-line password guessing attack and the server-spoofing attack. To strengthen the security, they proposed a secure SIP authentication scheme based on the Diffie-Hellman key exchange [4], in which security depended on the difficulty of Discrete Logarithm Problem. However, in the next year, Huang et al. [5] demonstrated that Yang et al.'s scheme could not resist off-line password-guessing attack and involved expensive exponential computation, so it was not suitable for devices with a low computational power. And then they proposed an efficient authentication scheme for session initiation protocol.  Later on, Jo et al. [6] pointed out that both the Yang et al.'s and Huang et al.'s authentication schemes were not secure against to the off-line password guessing attack. Following Yang et al.'s work, Durlanik et al. (2005) [7] suggested an efficient SIP authentication scheme by using Elliptic Curve Cryptography (ECC) in 2005. Compared with Yang et al.'s scheme, Durlanik et al.'s scheme reduced the total execution time and memory requirements; as the scheme was based on the elliptic curve cryptosystem, it could offer equivalent security as classical cryptosystems for much smaller key sizes. In 2009, Wu et al. (2009) [8] proposed an SIP authentication scheme based on ECC that provides provable security in the Canetti-Krawczyk security model [9]. They claimed that their scheme was secure against replay attacks, off-line password guessing attacks, man-in-the-middle attacks, and server spoofing attacks. Wu et al.'s scheme assumed that the communicating parties have shared a common secret beforehand between the IM Services Identity Module (ISIM) and the Authentication Center (AC). Although, compared with previous schemes, this pre-shared key scheme was more efficient and practice, the problem of distributing the shared secrets made this solution hard to scale. In 2010, Yoon et al.(2010) [10]  indicated that both Durlanik et al.'s and Wu et al.'s SIP authentication schemes were vulnerable to off-line password guessing attacks, Denning-Sacco attacks, and Stolen-verifier attacks. To improve security, they proposed an efficient authentication scheme for SIP based on ECC. However, Pu [11] and

Gokhroo et al. [12] argued that Yoon et al.'s scheme still suffered from both off-line password guessing attacks and replay attacks.

Nonce based SIP authentication scheme was proposed by Tsai et al. [13] in 2009. In this scheme, only one-way hash function and exclusive-or operations were used for mutual authentication and key agreement, so it reduced the computation costs. However, in [14], Yoon et al. showed that Tsai's scheme could not resist off-line password guessing attacks, Denning-Sacco attacks, and stolen-verifier attacks, and the scheme did not provide perfect forward secrecy. To overcome these weaknesses, Yoon et al. proposed a new scheme which not only could resist these attacks but also provided perfect forward secrecy. Later, Xie et al. [15] claimed that the Yoon et al.'s scheme was still vulnerable to stolen-verifier attacks and off-line password guessing attacks. Arshad et al. [16] also demonstrated that Tsai's scheme was vulnerable to off-line password guessing attacks and stolen verifier attacks. In addition, they found that Tsai's scheme did not provide known-key secrecy and perfect forward secrecy either. To improve the scheme, they proposed a revised authentication scheme based on ECC. Unfortunately, He et al. [17] argued that the Arshad et al.'s scheme still suffered from the off-line password-guessing attacks.

In most of the protocols mentioned above, the SIP server needs to store a password or verification table containing the passwords or the hashed passwords of all registered users for verification purposes, thereby making those schemes suffer from some attacks such as password guessing attacks, stolen-verifier attacks and server-spoofing attacks. In addition, since the password or verification tables are usually very large, maintaining the tables makes these solutions hard to scale up, and the reset password problem decreases its applicability for practical use.

In this paper, we propose an efficient and flexible password authenticated key agreement for session initiation by means of a smart card. The main merits of the proposed protocol include: (1) it does not maintain any password or verification table in the SIP server; (2) users can choose or change its own password freely; (3) both the user and the server can authenticate each other; (4) the user and the server can agree a session key; (5) it is secure against the replay attack, the impersonation attack, the stolen-verifier attack, the man-in-middle attack, and the Denning-Sacco attack; (6) even if the smart card was stolen, it still could resist the offline dictionary attack.

## 3. PRELIMINARIES

In this section, we introduce the basic concepts of the elliptic curve cryptosystem and the corresponding difficult problems associated with it. In an elliptic curve cryptosystem, the elliptic curve equation is defined as the form of $E_p(a,b): y^2 = x^3 + ax + b (\text{mod } p)$ over a prime finite field $F_p$, where $a, b \in F_p$ and $4a^3 + 27b^2 \neq 0 (\text{mod } p)$. Given an integer $t \in F_p^*$ and a point $P \in E_p(a,b)$, the scalar multiplication $tP$ over $E_p(a,b)$ can be computed as follows: $tP = P + P + ... + P$ (t times).

Definition 1. Given two points $P$ and $Q$ over $E_p(a,b)$, the elliptic curve discrete logarithm problem (ECDLP) is to find an integer $t \in F_p^*$ such that $Q = tP$.

Definition 2. Given three points $P$, $sP$ and $tP$ over $E_p(a,b)$ for $s, t \in F_p^*$, the computational Diffie-Hellman problem (CDHP) is to find the point $stP$ over $E_p(a,b)$.

Definition 3. Given two points $P$ and $Q = sP + tP$ over $E_p(a,b)$ for $s, t \in F_p^*$, the elliptic curve factorization problem (ECFP) is to find two points $sP$ and $tP$ over $E_p(a,b)$.

We assume that the three problems above are intractable. That is, there is no polynomial time algorithm that can solve these problems with non-negligible probability.

## 4. OUR PROPOSED SCHEME

In this section we describe our Authenticated Key Agreement (AKA) protocol. In our protocol, there are two entities, the user's smart card and the server. The proposed protocol consists of four phases: system setup phase, registration phase, authentication phase, and password changing phase. The procedure of the protocol is described in details as follows:

### *4.1. System setup phase*

*Step* $S1$: The server chooses an elliptic curve equation $E_p(a,b)$ with the order $n$, which is defined in Section 3.

*Step* $S2$: The server selects a base point $P$ with the order $n$ over $E_p(a,b)$, where $n$ is a large number of the security considerations. Then, the server chooses a random integer $s \in_R Z_p^*$ as a secret key and computes the public key $P_{pub} = sP$.

*Step* $S3$: The server chooses three secure one-way hash functions $h(\cdot):\{0,1\}^* \to \{0,1\}^k$, $h_1(\cdot):G \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^k$, and $h_2(\cdot):G \times G \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^k$, where $G$ is a cyclic addition group that is generated by $P$ over $E_p(a,b)$.

*Step* $S4$: The server keeps $s$ secret and publishes the public information $\{E_p(a,b), P, P_{pub}, h(\cdot), h_1(\cdot), h_2(\cdot)\}$.

## *4.2. Registration phase*

When user $U$ wants to register with the server, it performs the following steps with the server.

$Step\ R1$: The server verifies user $U$ through a secure identification protocol. If $U$ is eligible, then $U$ chooses its password $PW$ and a random integer $a \in_R Z_p^*$. Next, $U$ computes $h(PW \| a)$ and then sends $\{h(PW \| a), username\}$ to the server over a secure channel.

$U \to S : (h(PW \| a), username)$

$Step\ R2$: After the server receives the information from $U$ , it computes secret information $R = h(h(PW \| a) \| username)s^{-1}P$ .

$Step\ R3$: The server stores $R$ in the memory of a smart card and delivers this smart card to $U$ in a secure channel. Then the user keeps $PW$ and the smart card secretly for registration processes.

$Step\ R4$: After receiving the smart card, user $U$ will store $a$ in the smart card. Then the memory of the smart card contains $(R, a)$.

For each user, the registration phase performs once.

## *4.3. Authentication phase*

When user $U$ wishes to login to the server, it must inserts its smart card to a card reader and inputs its username and password $PW$ . Then the smart card and the server cooperate to perform the following steps as shown in Fig1.

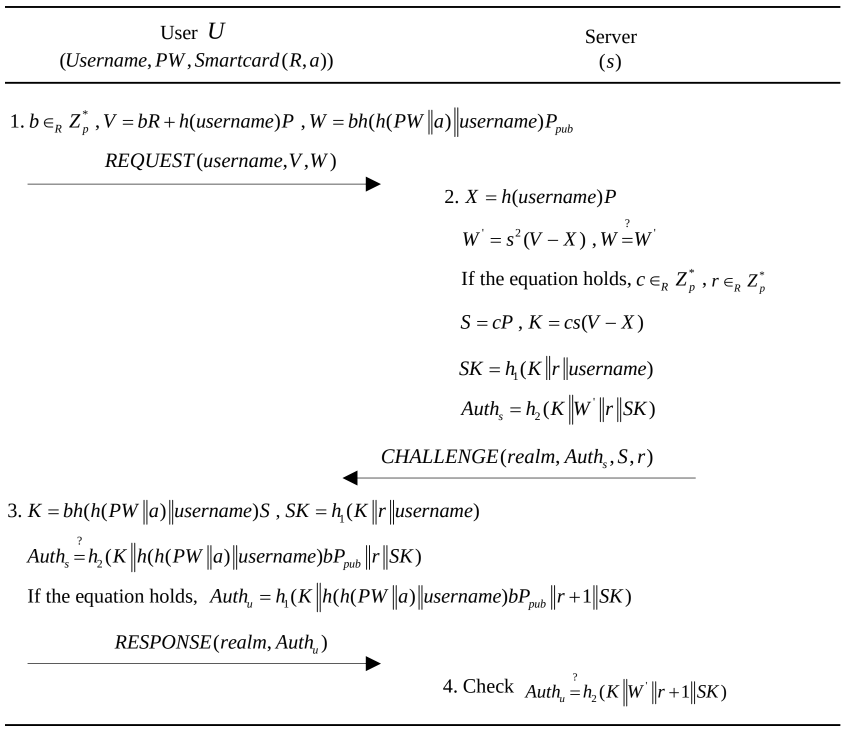

User $U$
$(Username, PW, Smartcard(R, a))$
Server
$(s)$
1. $b \in_R Z_p^*$ , $V = bR + h(username)P$ , $W = bh(h(PW \| a) \| username)P_{pub}$
$REQUEST(username, V, W)$
2. $X = h(username)P$
$W^{'} = s^2(V - X)$ , $W \overset{?}{=} W^{'}$
If the equation holds, $c \in_R Z_p^*$ , $r \in_R Z_p^*$
$S = cP$ , $K = cs(V - X)$
$SK = h_1(K \| r \| username)$
$Auth_s = h_2(K \| W^{'} \| r \| SK)$
$CHALLENGE(realm, Auth_s, S, r)$
3. $K = bh(h(PW \| a) \| username)S$ , $SK = h_1(K \| r \| username)$
$Auth_s \overset{?}{=} h_2(K \| h(h(PW \| a) \| username)bP_{pub} \| r \| SK)$
If the equation holds, $Auth_u = h_1(K \| h(h(PW \| a) \| username)bP_{pub} \| r+1 \| SK)$
$RESPONSE(realm, Auth_u)$
4. Check $Auth_u \overset{?}{=} h_2(K \| W^{'} \| r+1 \| SK)$

Fig.1. Authenticated key agreement phase

*Step* $A1$: User $U$ chooses a random integer $b \in_R Z_p^*$ , and computes $V = bR + h(username)P$ and $W = bh(h(PW \| a) \| username)P_{pub}$ . Next it sends a request message $REQUEST(username, V, W)$ to the server over a public channel.

$U \to S : REQUEST(username, V, W)$

*Step* $A2$: After receiving the request message, the server computes $X = h(username)P$ and $W^{'} = s^2(V - X)$ . It then verifies whether the following equation holds $W^{'} \overset{?}{=} W$ . If the equation holds, it chooses two random integers $c \in_R Z_p^*$ and $r \in_R Z_p^*$ , computes $S = cP$ , $K = cs(V - X) = cbh(h(PW \| a) \| username)P$ ,

$SK = h_1(K \| r \| username)$ and $Auth_s = h_2(K \| W^{'} \| r \| SK)$ . Then it sends $CHALLENGE(realm, Auth_s, S, r)$ to user $U$ over a public channel.

$S \rightarrow U : CHALLENGE(realm, Auth_s, S, r)$

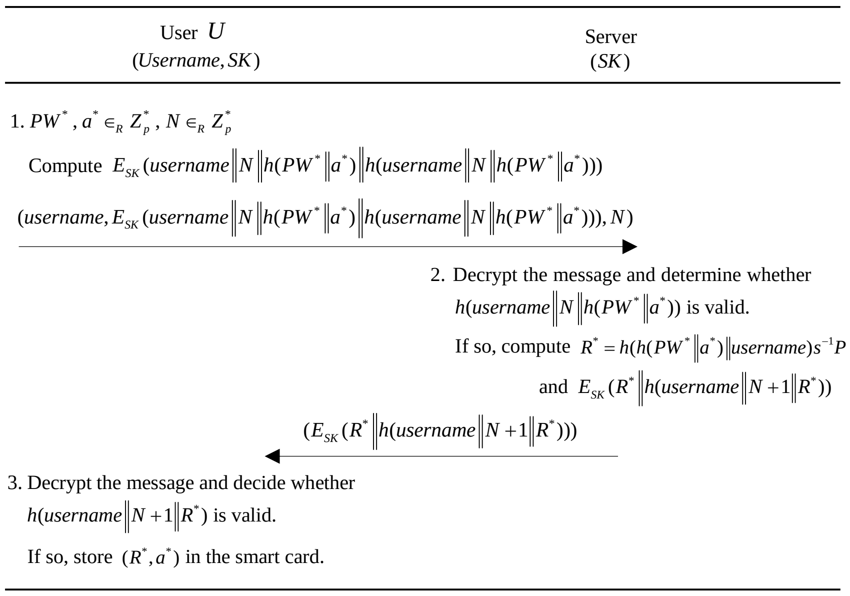
*Step* $A3$: Upon receiving the challenge message, $U$ computes $K = bh(h(PW \| a) \| username)S = bch(h(PW \| a) \| username)P$ and $SK = h_1(K \| r \| username)$ . Then it verifies whether the following equation holds $Auth_s \overset{?}{=} h_2(K \| h(h(PW \| a) \| username)bP_{pub} \| r \| SK)$ . If the equation holds, it computes $Auth_u = h_2(K \| h(h(PW \| a) \| username)bP_{pub} \| r+1 \| SK)$ and sends $RESPONSE(realm, Auth_u)$ to the server over a public channel. Otherwise, it deletes the received information and the protocol stops.

$U \rightarrow S : RESPONSE(realm, Auth_u)$

*Step* $A4$: After receiving the response message, the server verifies if $Auth_u \overset{?}{=} h_2(K \| W^{'} \| r+1 \| SK)$ . If the message is authenticated, the server sets $SK$ as the shared session key with user $U$ ; otherwise, it deletes the receiving information and the protocol stops.

## *4.4. Password changing phase*

When the user $U$ wants to update its password, it needs to agree on a session key $SK$ with the server via the authentication phase in advance. Figure 2 illustrates how the password changing phase works.



User $U$
$(Username, SK)$
Server
$(SK)$
1. $PW^*$ , $a^* \in_R Z_p^*$ , $N \in_R Z_p^*$
Compute $E_{SK}(username \| N \| h(PW^* \| a^*) \| h(username \| N \| h(PW^* \| a^*)))$
$(username, E_{SK}(username \| N \| h(PW^* \| a^*) \| h(username \| N \| h(PW^* \| a^*))), N)$
2. Decrypt the message and determine whether
$h(username \| N \| h(PW^* \| a^*))$ is valid.
If so, compute $R^* = h(h(PW^* \| a^*) \| username)s^{-1}P$
and $E_{SK}(R^* \| h(username \| N+1 \| R^*))$
$(E_{SK}(R^* \| h(username \| N+1 \| R^*)))$
3. Decrypt the message and decide whether
$h(username \| N+1 \| R^*)$ is valid.
If so, store $(R^*, a^*)$ in the smart card.

Fig.2. Password changing phase

*Step P*1: The user $U$ chooses its new password $PW^*$ and a random integer $a^* \in_R Z_p^*$. It then uses the session key $SK$ to encrypt the new password message $(username, h(PW^* \| a^*))$. Next it transmits username, $E_{SK}(username \| N \| h(PW^* \| a^*) \| h(username \| N \| h(PW^* \| a^*)))$ and $N$ to the server, where $N$ is a nonce for freshness checking.

$$U \rightarrow S : (username, E_{SK}(username \| N \| h(PW^* \| a^*) \| h(username \| N \| h(PW^* \| a^*))), N)$$

*Step P*2: Upon receiving the information, the server decrypts the message and then checks the validity of the authentication tag $h(username \| N \| h(PW^* \| a^*))$. If it is valid, the server computes the new secret information $R^* = h(h(PW^* \| a^*) \| username)s^{-1}P$ . It then sends encryption information $E_{SK}(R^* \| h(username \| N+1 \| R^*))$ to the user $U$ .

$$S \rightarrow U : (E_{SK}(R^* \| h(username \| N+1 \| R^*)))$$

$Step\ P3$: The user $U$ decrypts the received message and checks the validity of the authentication tag $h(username \| N+1 \| R^*)$. If it is valid, the user $U$ stores $(R^*, a^*)$ in its smart card.

## 5. SECURITY ANALYSIS

In this section, we discuss the security of our proposed protocol by analyzing some possible attacks, then evaluating the security of the protocol.

### *5.1. Replay attacks*

A replay attack is an offensive action in which an adversary impersonates or deceives another legitimate participant through the reuse of information obtained in a protocol. The following explains why the proposed protocol can resist replay attacks.

Suppose an adversary Alice intercepts the user $U's$ request message $REQUEST(username, V, W)$ and replays it to the server to impersonate the user $U$. However, Alice cannot construct a valid $V = bR + h(username)P$ without the knowledge of the secret key $s$. When Alice tries to guess the secret key $s$ from $V$ or $W$, she will face the ECDLP. Then the server will find the attack by checking whether $W' = s^2(V - X)$ and $W$ are equal.

On the other hand, suppose Alice intercepts $CHALLENGE(realm, Auth_s, S, r)$ from the server and replays it to impersonate the server. In order to pass the verification process of the user $U$, Alice needs to compute a valid $Auth_s$. When Alice tries to guess the correct password $PW$, the nonce $a$ and the random number $b$ from $V$ or $W$ to construct a valid $Auth_s$, she not only has to face the ECDLP but also needs to break the hash functions. If Alice cannot construct a valid $Auth_s$, the user $U$ will find out that $Auth_s$ is not equivalent to its computed $h_2(K \| h(h(PW \| a) \| username) bP_{pub} \| r \| SK)$. Then, the user $U$ will stop the protocol and not send $RESPONSE(realm, Auth_u)$ back to Alice.

Suppose that an adversary Alice impersonates $U$ and replays the $U's$ $RESPONSE$ message $RESPONSE(realm, Auth_u)$. For the same reason, if Alice cannot compute a valid $Auth_u$, the server will find out that $Auth_u$ is not equivalent to its computed $h_2(K \| W' \| r+1 \| SK)$. Then the server will delete $SK$ and stop the protocol. Therefore, the proposed protocol can resist the replay attacks.

## *5.2. Man-in-the-middle attacks*

The man-in-the-middle attack is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making the victims believe that they are talking directly to each other over a private connection, where in fact the entire conversation is controlled by the attacker.

Analysis shows the proposed protocol can resist the man-in-middle attacks. In the proposed protocol, the user $U$ and the server share a session key $SK$ only after mutual authentication between the user $U$ and the server. So, an adversary Alice cannot impersonate the user $U$ to establish a session key with the server unless she can pass the verification process of the server. If Alice tries to pass the verification, she has to face the ECDLP. On the other hand, for the same reason Alice cannot impersonate the server to share a session key with the user $U$. In addition, Alice neither can get the session key between the user $U$ and the server nor can intrude into the communication between the user $U$ and the server to intercept the exchanged data and inject false information. Thus, Alice cannot launch the man-in-middle attack to cheat either the user $U$ or the server.

## *5.3. Modification attacks*

A modification attack is an attempt by an adversary to modify information in an unauthorized manner.

Assuming that an adversary Alice intends to impersonate the user $U$ by sending $REQUEST(\ username, V^{'}, W^{'})$ to the server, $V^{'}, W^{'}$ are constructed by Alice. The server will find the attack by checking whether $W^{'} = s^2(V - X)$ and $W$ are equal, because Alice does not know the secret key $s$.

If an adversary Alice tries to impersonate the server and sends $CHALLENGE(realm, Auth_s^{'}, c^{'}P, r^{'})$ to the user $U$, where $c^{'}, r^{'}$ are chosen by Alice and $Auth_s^{'}$ is constructed by Alice. But the $CHALLENGE$ message cannot go through the verification process of the user $U$ as the password $PW$, nonce $a$ and random number $b$ are not known.

Supposing that an adversary Alice wishes to impersonate the user $U$ and sends $RESPONSE(realm, Auth_u^{'})$ to the server, where $Auth_u^{'}$ is computed by Alice. However, the server will find the modification by checking $Auth_u \overset{?}{=} h_2(K \| W^{'} \| r+1 \| SK)$. Therefore, the proposed protocol can resist the modification attacks.

## *5.4. Denning-Sacco attacks*

The Denning-Sacco attack occurs when an attacker compromises an old session key and tries to find a long-term private key (e.g., user password or server private key) or other session keys.

In the proposed protocol, the session key is $SK = h_1(K \| r \| username) = h_1(cbh(h(PW \| a) \| username)P \quad \| r \| username)$ . Supposing an adversary Alice obtains the session key $SK$ . Alice cannot obtain the $U's$ password from $SK$ and other intercepted messages, because Alice not only has to face the ECDLP but also needs to break the hash functions. Therefore, the proposed protocol can resist Denning-Sacco attacks.

### *5.5. Stolen-verifier attacks*

The stolen-verifier attack means an adversary who steals the password-verifier from the server can use it directly to masquerade as a legitimate user in a user authentication process.

For example, if an adversary wants to get the valuable information through stolen the verification table stored in the server, she or he cannot implement the stolen-verifier attack successfully, since no password or verification table stored in the server database. So the protocol can resist the stolen-verifier attacks.

### *5.6. Offline dictionary attacks without the smart card*

The offline dictionary attack without the smart card is defined as the process in which attackers attempt to determine whether each of their guessed passwords is correct or not via the intercepted messages transmitted between the user and the server.

Assuming that an adversary Alice intends to carry out the offline dictionary attack, she obtains the $REQUEST$ message $REQUEST(username, V, W)$ through eavesdropping the communication between the user $U$ and the server. To obtain the $PW$, Alice needs to extract $h(h(PW\|a)\|username)$ from $V = bR + h(username)P$ or $W = bh(h(PW\|a)\|username)P_{pub}$, which is equivalent to solving an instance of elliptic curve discrete logarithm problem. So it is unlikely for Alice to do the offline dictionary attack by using the $REQUEST$ message. Additionally, the adversary Alice cannot derive $PW$ from the information $Auth_s$ or $Auth_u$, because the entropy of $K$, $a$, $r$ and $SK$ are all very large. Therefore, the offline dictionary attacks without the smart card is invalid in the proposed protocol.

### *5.7. Offline dictionary attacks with the smart card*

The offline dictionary attack with the smart card is defined as the process in which attackers attempt to determine whether each of their guessed passwords is correct or not via the information stored in the smart card of the user and the intercepted messages transmitted between the user and the server.

Assuming that an adversary Alice obtains the secret information $(R, a)$ stored in the smart card of the user $U$ and intercepts the $REQUEST$ message, the $CHALLENGE$ message and $RESPONSE$ message transmitted between the user $U$ and the server. Compared with the offline dictionary attack without the smart card, the addition information known by Alice in this attack is $(R, a)$. However, Alice cannot extract $h(h(PW\|a)\|username)$ from $R$ and then check whether each of their guessed passwords is correct or not via $h(h(PW\|a)\|username)$. Because computing

$h(h(PW \| a) \| username)$ from $R$ is equivalent to solving an instance of elliptic curve discrete logarithm problem. Furthermore, for the same reason, Alice cannot obtain $h(h(PW \| a) \| username)$ from $Auth_u$ and $Auth_s$. Therefore, the offline dictionary attack with the smart card also is invalid in the proposed protocol.

## *5.8. Session key security*

Session key security means that at the end of the key exchange, the session key is not known by anyone but only the two communicating parties.

In the proposed protocol, the session key $SK = h_1(K \| r \| username) = h_1(cbh(h(PW \| a) \| username\ )P \| r \| username)$ is not known by anyone but only the user $U$ and the server since $K = cbh(h(PW \| a) \| username)P$ cannot be constructed correctly by the adversary Alice without the knowledge of $(b, a, PW)$ or $(s, c)$. None of this session key $SK = h_1(K \| r \| username)$ is known to anybody but the user $U$ and the server. Therefore, the proposed protocol provides session key security.

## *5.9. Known-key security*

Known-key security means that each run of an authentication and key agreement protocol between two communicating parties should produce unique secret keys (session keys).

In the proposed protocol, the server and the user $U$ randomly and independently generate the random number $c$ and $b$ separately, the session key $SK = h_1(cbh(h(PW \| a) \| username)P \| r \| username)$ of each session is not connected with the session keys of any other sessions. Knowing a session key $SK = h_1(cbh(h(PW \| a) \| username)P \| r \| username)$ and the random values $c$ and $b$ is not enough for computing the other session keys $SK' = h_1(c'b'h(h(PW \| a) \| username)P \| r' \| username)$, because in each session a fresh session key is generated depending on $c'b'h(h(PW \| a) \| username)P$, and this secret differs in every session. Therefore, the proposed protocol provides the known-key security.

### *5.10. 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 honest entities is not affected.

In the proposed protocol, suppose that the user's password $PW$ and the server's secret key $s$ are compromised. The adversary Alice cannot obtain the session key $SK$ for the past sessions. Because Alice still faces the ECDLP to compute the $SK = h_1(cbh(h(PW \| a) \| username)P \| r \| username)$ when she tries to extract the value $c$ from $S = cP$. Therefore, the proposed protocol satisfies the property of perfect forward secrecy.

### *5.11. Mutual authentication*

Mutual authentication means that both the user $U$ and the server are authenticated with each other within the same protocol.

In the proposed protocol, the server and the user can authenticate each other by checking $Auth_u$ and $Auth_s$, respectively. Therefore, the proposed protocol can provide mutual authentication.

### *5.12. Security chosen and update password*

In the proposed protocol, the legitimate user with the smart card can freely choose her or his favorite password in the registration phase. It will make users easy to remember their own passwords. The proposed protocol also provides an update password phase for users to change their password freely. Any other person, even having stolen or lost the smart card, cannot change or update the password without knowing the current session key $SK$ sharing between the user $U$ and the server.

## 6. COMPLEXITY ANALYSIS

In this section, we summarize the functionality of the proposed protocol and compare the proposed protocol with Xie et al.'s protocol. In Xie et al.'s protocol, the server needs to store a password table of all registered users for verification. In the proposed protocol, the password is embedded in $h(PW \| a)$. After receiving $\{h(PW \| a), username\}$ in the registration phase, the server computes $R = h(h(PW \| a) \| username)s^{-1}P$ and stores it in the memory of a smart card, and then delivers the smart card to the user $U$ via a secure channel. During the registration process, the server does not need to store a password table. In addition, the proposed protocol provides a securely update password phase for users to change their password freely and can resist stolen smart card attacks. As shown in Table 1, the proposed protocol can provide more unique properties such as no password or verifier table and password update freely, which were not considered in Xie et al.'s protocol. These new features are very important in implementing a practical and universal authenticated key agreement for session initiation protocol.

As the protocol of Xie et al. is currently the most secure and efficient one in the literatures, we compare the proposed protocol and Xie et al.'s protocol in terms of computational costs. First, we define some notations as follows.

(1) $T_{ecsm}$ the time for executing a scalar multiplication operation of elliptic curve.

(2) $T_{ecpa}$ the time for executing a point addition operation of elliptic curve.

(3) $T_h$ the time for executing a one-way hash function.

(4) $T_{inv}$ the time for executing a modular inversion operation.

(5) $T_{ske}$ the time for executing a symmetric key encryption operation.

(6) $T_{skd}$ the time for executing a symmetric key decryption operation.

In the registration phase, the proposed protocol requires one hash operation on the user side, one scalar multiplication of elliptic curve and one modular inversion operation on the server side. In the authentication phase, the user takes four scalar multiplication operations to compute $bR, h(username)P, bh(h(PW\|a)\|username)P_{pub}$ and $bh(h(PW\|a)\|username)S$ ; one point addition operation to obtain $V = bR + h(username)P$ ; and six one-way hash function operations to compute $h(username), h(PW\|a), h(h(PW\|a)\|username), Auth_s, Auth_u$ and $SK$ . The server takes four scalar multiplication operations to get $h(username)P, s^2(V-X), S$ and $K$ ; one point addition operation to compute $V-X$ ; and three one-way hash function operations to obtain $SK, Auth_s$ and $Auth_u$ . In the password changing phase, the user takes three one-way hash function operations to compute

$h(PW^{*}\|a^{*}), h(username\|N\|h(PW^{*}\|a^{*})$ and $h(username\|N+1\|R^{*})$ ; one symmetric key encryption operation and one symmetric key decryption operation. The server takes one scalar multiplication operation and one modular inversion operation to compute $R^{*}$ ; three one-way hash function operations to compute $h(username\|N\|h(PW^{*}\|a^{*})), h(h(PW^{*}\|a^{*})\|username)$ and $h(username\|N+1\|R^{*})$ ; and one symmetric key encryption operation and one symmetric key decryption operation.

Table 2 shows that our protocol costs more computational overhead compared with Xie et al.'s protocol. This is because the proposed protocol does not maintain any password or verification table on the server and provide securely update password phase for users to change their password freely, which requires more operations to achieve the unique properties of the protocol and then resist all possible attacks of an authenticated key agreement protocol. For example, in our protocol, an adversary cannot carry out a stolen-verifier attack, since no password or verification table stored in the server. Therefore, this computational increase is indispensable for constructing a reliable and trustworthy authenticated key agreement for Session Initiation Protocol used by VoIP.

## 7. CONCULSION

This paper has proposed an efficient and flexible password authenticated key agreement protocol for SIP where the user and the server can achieve mutual authentication and key agreement by using password and the smart card. In comparison with other related protocols, the proposed protocol not only provides many unique characters, such as mutual authentication, session key agreement, password updating freely and the server not needing to maintain a password or verification table etc, but also can withstand the replay attack, the impersonation attack, the stolen-verifier attack, the man-in-middle attack, the Denning-Sacco attack, and the offline dictionary attack with or without the smart card. Especially, the proposed protocol does not require any password table for verification, which makes this solution easy to scale up and enhances its applicability for practical use.

## ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China [Grant numbers 61272469, 61075063].

## REFERENCES

1. Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Sparks R, Handley M, Schooler E. SIP: session initiation protocol. RFC 3261, June 2002.

2. Franks J, Hallam-Baker PM, Hostetler JL, Lawrence SD, Leach PJ, Luotonen A, Stewart LC. HTTP authentication: basic and digest access authentication. Internet RFC2617, June 1999.

3. Yang C, Wang R, Liu W. Secure authentication scheme for session initiation protocol. Computers& security 2005; 24: 381-386.

4. W. Diffie, M. Hellman. New directions in cryptology. *IEEE Transaction on Information Theory* 1976; 22:644-654.

5. Huang H, Wei W, Brown G. A new efficient authentication scheme for session initiation protocol. *Proceedings of JCIS 06*, 2006.

6. Jo H, Lee Y, Kim M, Kim S, Won D. Off-line Password-Guessing Attack to Yang's and Huang's Authentication Schemes for Session Initiation Ptorocol. *Proceedings of INC, IMS and IDC*, 2009; 618-621.

7. A.Durlanik, I. Sogukpinar. SIP authentication scheme using ECDH. *World Enformatika Society Transaction on Engineering Computing and Technology* 2005; 8:350-353.

8. L.Wu, Y. Zhang, F. Wang A new provably secure authentication and key agreement protocol for SIP using ECC. *Computer Standards & Interfaces* 2009; 31:286-291.

9. Canetti, R., Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. *Proceedings of EUROCRYPT 2001*, 2001;453-474.

10. EJ Yoon, KY Yoo, et al..A secure and efficient SIP authentication scheme for converged VoIP networks. *Computer Communications* 2010; 33:1674-1681.

11. Q. Pu. Weaknesses of SIP authentication scheme for converged VoIP networks. http://eprint.iacr.org/2010/464.

12. Gokhroo M.K., Jaidhar C.D., Tomar A.S. Cryptanalysis of SIP Secure and Efficient Authentication Scheme. *Proceedings of ICCSN 2011* 2011; 308-310.

13. Jia Lun Tsai. Efficient Nonce-based authentication scheme for session initiation protocol. *International Journal of Network Security* 2009; 9:12-16.

14. Yoon E, Shin Y, Jeon I, Yoo K. Robust mutual authentication with a key agreement scheme for the session initiation protocol. *IETE Technical Review* 2010; 27:203-213.

15. Qi Xie. A new authenticated key agreement for session initiation protocol. *International Journal of Communication systems* 2012; 25:47-54.

16. Arshad R, Ikram N. Elliptic curve cryptography based mutual authentication scheme for session initation protocol. *Multimedia Tools and Applications* 2011; DOI: 10.1007/s11042-011-0787-0.

17. Debiao He, Jianhua Chen and Yitao Chen. A secure mutual authentication scheme for session initiation protocol using elliptic curve cryptography. *Security and Communication Networks* 2012; DOI:10.1002/sec.506.

Table 1. The functionality comparisons between our protocol and Xie et al.'s protocol

| | Xie et al.'s protocol | Our protocol |
|---|---|---|
| No password or verifier table | No | Yes |
| Password update function | No | Yes |
| Secure to Denning-Sacco attacks | Yes | Yes |
| Secure to password guessing attacks | Yes | Yes |
| Secure to stolen smart cards | N/A | Yes |
| Session key agreement | Yes | Yes |
| Secure mutual authentication | Yes | Yes |
| Perfect forward secrecy | Yes | Yes |

N/A: Not Applicable or Not Available

Table 2. Computational comparisons between our protocol and Xie et al.'s protocol

| | Xie et al.'s protocol | Our protocol |
|---|---|---|
| Registration phase | $1T_{ske}$ | $1T_{ecsm} + 1T_{h} + 1T_{inv}$ |
| Authentication phase | $6T_{ecsm} + 6T_{h} + 1T_{ske} + 1T_{skd} + 1T_{ecpa} + 1T_{inv}$ | $8T_{ecsm} + 2T_{ecpa} + 9T_{h}$ |
| Password change phase | | $2T_{ske} + 2T_{skd} + 6T_{h} + 1T_{ecsm} + 1T_{inv}$ |